Exploration of Energy Efficient Manycore Networks-on-chip Architectures

Submitted in
fulfillment of the requirements for the degree of
Doctor of Philosophy

by

Sonal Yadav
ID No. 2013RCP9006

Under the Supervision of

Prof. Manoj Singh Gaur
Director, IIT Jammu

Prof. Vijay Laxmi
Professor, MNIT Jaipur

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
MALAVIYA NATIONAL INSTITUTE OF TECHNOLOGY JAIPUR

August 2019

CERTIFICATE

This is to certify that the thesis entitled **"Exploration of Energy Efficient Manycore Networks-on-Chip Architectures"** being submitted by **Sonal Yadav (2013RCP9006)** is a bonafide research work carried out under my supervision and guidance in fulfillment of the requirement for the award of the degree of **Doctor of Philosophy** in the Department of **Department of Computer Science and Engineering**, Malaviya National Institute of Technology, Jaipur, India. The matter embodied in this thesis is original and has not been submitted to any other University or Institute for the award of any other degree.

Place: Jaipur
Date:

Prof. Manoj Singh Gaur
Director, IIT Jammu

Prof. Vijay Laxmi
Professor
Department of Computer Science and Engineering
MNIT Jaipur

DECLARATION

I, **Sonal Yadav**, declare that this thesis titled, **"Exploration of Energy Efficient Manycore Networks-on-Chip Architectures"** and the work presented in it, are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this university.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this university or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself, jointly with others, I have made clear exactly what was done by others and what I have con-tributed myself.

Date:

Sonal Yadav
(2013RCP9006)

ABSTRACT

A sustained performance while designing high-performance single-processor has become infeasible. The computational density of single but more powerful complex processor does not scale power with higher clock speed. Computational gain with conventional single processor had ended about a decade ago. Then computer paradigm shifted from a complex single-processor to multi-core processor. The continuous advancements in CMOS technology have grown the fabrication of a large number of cores within the same die. Cores in multi-core or many-core processor can compute faster, although the interconnect delay limits overall performance of these cores. **N**etworks-on-**C**hip (NoC) is popularly used on-chip interconnect with the growing number of cores. It is designed to provide high performance, and low power inter-core communication to/fro shared caches and off-chip memory. Modern processor designs feature one of the NoC architectures, **S**ingle physical network (S-NoC), or **M**ultiple physical networks (Multi-NoCs).

Our dissertation explores multi-NoCs for the better power and energy efficiency of general-purpose manycore processors. Multi-NoCs are known for multiple NoC networks which facilitate efficient separation of different on-chip messages towards the multiple traffic flows. Each traffic flow is fully dedicated to its NoC network. Physical resources and physical link width of single NoC network split into multiple physical networks to construct multiple simple, independent and parallel NoC networks. These multiple networks can separate different message classes at logical (virtual channel) and physical levels. As each nanometer technology significantly increases the contribution of static power in total processor power, we have proposed a static power efficient custom-made 2-network-NoC architecture. The proposed multi-NoC also expands the design space to improve energy efficiency through the placement of network selector. As a message request demultiplexed on any one NoC among the multiple NoCs, we have rechristened network selector as network demultiplexer. In traditional multi-NoCs, network interface segments messages into flits and transmit them to multi-NoCs using network demultiplexer. We place network demultiplexer along with routing unit and switch allocator of the router to explore the optimal placement of network demultiplexer. Other than placement, existing issues and challenges for provisioning of traffic distribution of multi-NoCs are identified through an extensive survey on traditional static traffic distribution. We have proposed a case study for different combinations of static traffic distribution to analyze its impact on power and energy efficiency of multi-NoCs. We realize that static message distribution is not suitable while running different applications on general-purpose computers. Runtime variation in the number of messages varies runtime criticality of messages. These limitations overcome through our proposed runtime adaptive traffic distribution. We devise message criticality of NoC traffic using message dependency of caches through fine grain analysis of cache state space. Our proposal captures the channel utilization and redirects the traffic towards the underutilized channel of NoC networks. Towards the end, our experiments show the effectiveness of proposed solutions for current and future NoC architectures.

DEDICATIONS

This thesis
is dedicated to
my supervisors, husband, parents, and grandparents.

ACKNOWLEDGEMENTS

My PhD years have been, and am sure will remain, one of the best phases of my life. I have been really lucky to have had the chance to work with and be mentored by Prof Manoj Singh Gaur and Prof. Vijay Laxmi, my research advisor. Prof. Manoj Singh Gaur is a pleasant personality. His honesty and dedication towards work always inspired me. Thank you for always ready to hear the problems and fill the brevity to face out the situations. A big thanks for providing a positive environment to develop me as a good person and bring out my strengths by always giving challenging and difficult tasks. I am privileged to work with him.

My co-supervisor, Vijay Laxmi is one of the fascinating women I have met in my life. Her enthusiasm, excellent insights, strong technical expertise, and systematic problem-solving approach will stay with me throughout my professional career. I think she struck a wonderful balance between being a highly indulgent advisor and yet ensuring that I was making progress and not getting stuck in a hush. I admire her immense energy, optimism and positive attitude towards everything in life. Thank you Prof. Vijay Laxmi for being my supervisor!

Throughout my PhD years not only I acquire technical knowledge but also learn to work ethically. I honored to interact with some of the most distinguished researchers in our field. A special mention of Prof. Hemangee K. Kapoor who have played an important role in shaping my PhD work, and always supports me during my ups and down of research and always ready to hear me. I thank her for insightful feedback and comments.

I would also like to acknowledge my DREC committee members Dr. S. J. Nanda for always motivating and providing valuable feedback. Also, my gratitude to other DREC members Prof. Lava Bhargava, Dr. E. S. Pilli, Dr. Girdhari Singh for your support throughout my PhD. Also, thanks to academic staff Mr. Jasbeer at the MNIT office, for helping me during PhD administrative interactions.

Special thanks to Vinod P. for always motivating and inspiring me for hard work. Thank you to my current and former PhD mates - Smita, Rimpy, Gaurav, Ramesh B. Battula, Shweta, Vineeta, Ashish, and Anil - for spending good time discussions on tea/coffee and the great lunch and dinner outings. Thanks, Rashi and Bharat for always arranging VLSI Lab keys while working in late evenings and supporting me while I was learning Verilog. Thanks, Sharwan, Ankur, Pranjal for supporting me while working late night in the Department. I have had a special rapport with each one of you personally, and I cherish that bond. A shout out to some of my close friends who have shared the school and undergraduate experience with me, and have been a source of mutual support.

Special thanks to Mr. Sanjay Bhasker for providing higher configuration servers for running benchmarks during my experiments, Dr. E. S. Pilli for always resolving the network bugs during holidays while I was accessing synopsis design compiler.

My parents' support has been enormous through my PhD years. Thank you, Papa, for supporting me always. Thank you, mumma, you have been a constant source of guidance throughout my life, and this thesis is a result of your blessings. The biggest thanks to teach me so many moral values, you're the first teacher of my life. Papa and Mumma, words cannot do any justice to how much you have done for me; I will just say I am proud to have followed your footsteps and here to another copy of you. And I don't have enough words to thank you Mom for your patience throughout my studies and always supporting me. The last but not least thanks to my husband Nilesh, I couldn't complete my PhD without your unconditional support and love. Thank you for your patience.

I would also like to acknowledge UK-India Education and Research Initiative (UKIERI) grant No. IND/CONT/E/11-12/78 through the project ”HiPER NIRGAM” supporting MNIT Jaipur, India, and University of Southampton, UK, and Indo-Spain DST project DST/INT/Spain/P35/11/1 and Spanish Ministerio de Economay Compatitividad (MINECO) under grant PRI-PIBIN-2011-0989 for providing funding support to this research.

Once again thank you all for making my journey wonderful and have had the unique opportunity of spending quality time at MNIT Jaipur.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

CMP	Chip Multiprocessor
ISA	**I**nstruction **S**et **A**rchitecture
MESI	**M**odified, **E**xclusive, **S**hared, and **I**nvalidation
MNN	Multi-Network-NoC
MPN	Multi-Planar-NoC
MRN	Multi-Router-NoC
MSN	Multi-Switch-NoC
NI	**N**etwork **I**nterface
PARSEC	**P**rinceton **A**pplication **R**epository for **S**hared m**E**mory **C**omputers
VC	Virtual Channels
VN	Virtual Network

Chapter 1

Introduction

1.1 Evolution of Manycore Processors and Present Challenges

Modern computers shape after a radical reform from the 1930s to till date. The first generation of computers evolved with the invention of vacuum tubes for electronic circuits. In 1947, the invention of the transistor was a historic invention by Shockley and his colleagues at Bell Labs. The inventors were awarded Noble Prize as this marvelous invention began the second generation of computers. A real explosion in computer history was the invention of **I**ntegrated **C**ircuits (ICs) by Jack Kilby in 1960s. This was the third generation of computers that had transistors fabricated on a chip and founded the base for current processor architecture.

In the 1980s, the microprocessor was fabricated as a single **V**ery **L**arge **S**cale **I**ntegration (VLSI) chip and founded the fourth generation of computers. The IBM System/360 with Intel microprocessor in 1981 was the first commercially available **P**ersonal **C**omputer (PC) [5]. Continued advancements in transistor technology resulted in **U**ltra **L**arge **S**cale **I**ntegration (ULSI) era, and it became feasible to fabricate billions of transistors on a chip. These advancements reduced the required space in a computer and significantly increased computational speed. One of the factors affecting performance was delay of processor memory interface. For keeping cost down, memory is much slower than CPU speed. Caches emerged as a solution to address this problem. At architectural level, caches were invented to speed up the memory accesses and consequently computations. Caches were faster memories placed between CPU and main memory to reduce the access time for faster computation of instructions/data. A good amount of cache capacity and multiple levels of cache hierarchy significantly speed up the computation. A higher power dissipation became the limiting factor for the processing speed of a single processor. As a result, research shifted to lightweight multiple cores processor. Evolution of different characteristics with each generation of computers is shown in Fig 1.1.

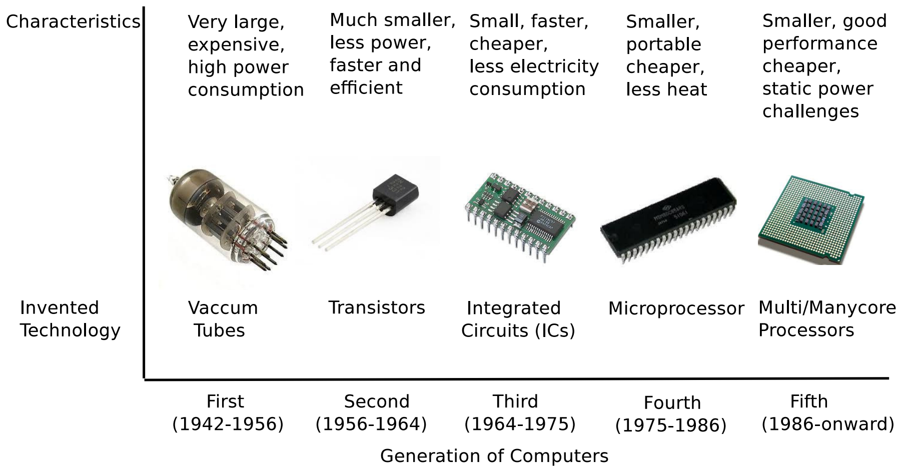

Characteristics
Very large, expensive, high power consumption
Much smaller, less power, faster and efficient
Small, faster, cheaper, less electricity consumption
Smaller, portable cheaper, less heat
Smaller, good performance cheaper, static power challenges
Invented Technology
Vaccum Tubes
Transistors
Integrated Circuits (ICs)
Microprocessor
Multi/Manycore Processors
First (1942-1956)
Second (1956-1964)
Third (1964-1975)
Fourth (1975-1986)
Fifth (1986-onward)
Generation of Computers

FIGURE 1.1: Generations of computers: invented technology and characteristics from first(1942) to fifth (till date) generations of computers.

For seeking ever-faster chips, multicore designs transformed to manycore processors. Another paradigm shift was from **C**omplex **I**nstruction **S**et **C**omputers (CISC) to **R**educed **I**nstruction **S**et **C**omputers (RISC). Doubling the number of cores does not necessarily result in doubling the performance. Fig 1.2[1] shows the trade-off between performance and the number of cores since 1970.

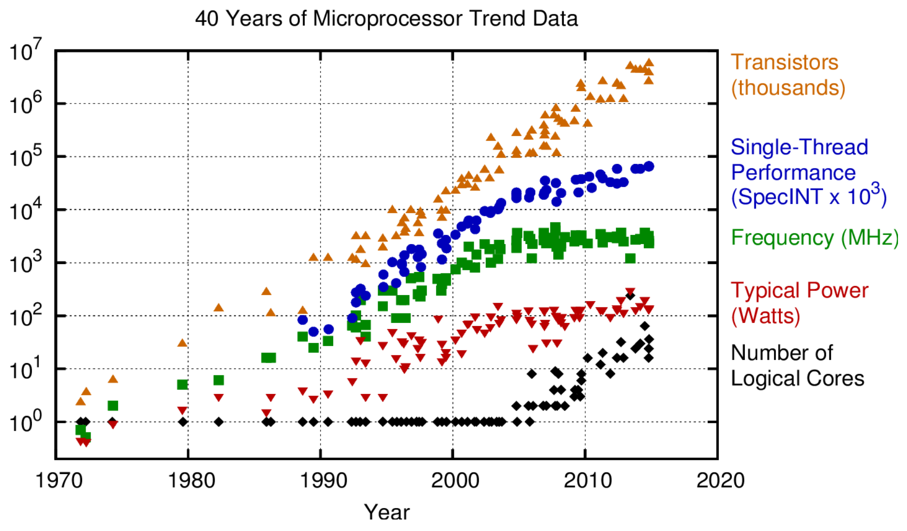


FIGURE 1.2: Impact of semiconductor scaling on processor design metrics such as the number of transistors, thread performance, frequency, power, and the number of cores since the 1970s to till date.

[1]Original data up to the year 2010 collected and plotted by M. Horowitz, F. Labonte, O. Shacham, K. Olukotun, L. Hammond, and C. Batten. New plot and data collected for 2010-2015 by K.Rupp.

To improve the performance, hardware industry is trying to address the challenges of other performance limiting factors like caches, offchip memory, and interconnect communication delays that limit the data/instruction access time and consequently, processor computation speed.

With the advancement in both computing architectures and processor technology, many-core architectures are begin designed with going to have hundreds of cores into a single chip. By increasing the number of **P**rocessing **E**lements (PEs) in the chip, there is a need for efficient and scalable communication infrastructure. The communication delay and power consumption of traditional interconnections have become the major bottleneck. The networks-on-chip can address many of the on-chip communication issues such as performance limitations of bus interconnects and integration of a large number of PEs on a chip [147]. Modern networks-on-chip uses customised architectures for effective communication techniques to meet modern processor design objectives. As can be seen from Fig 1.3[2], the performance gain of the processor is almost flattening now.

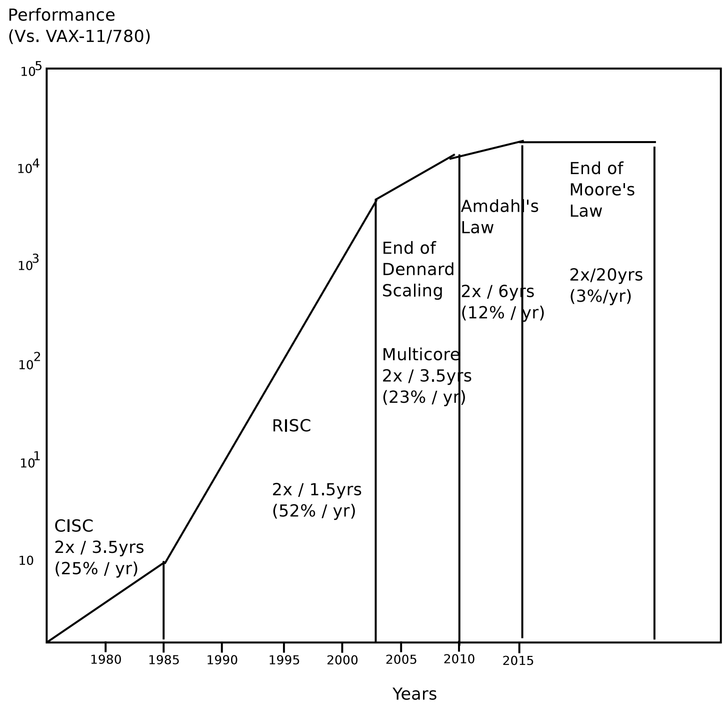


FIGURE 1.3: Growth in processor performance with different computer eras such as CISC, RISC, Dennard scaling, Amdahl's law, and Moore's law since the late 1970s

[2]Original data up to 2010, source: Computer Architecture-A Quantitative Approach John L. Hennessy and David A. Patterson.

1.2 Networks-on-Chip (NoC)

Manycore processors have two primary on-chip components: core and uncore. Core components are also known as processing elements. These elements are made up of engines which perform computations through vector units. On the other hand, uncore constitutes accelerators, caches, memory controllers, and interconnects. **N**etworks-on-**C**hip (NoC) has emerged as a feasible and scalable interconnect to connect an ever-increasing number of varied on-chip cores and uncore components. It transports the messages between the core-uncore component(s). Computation speed of cores is significantly dependent on the efficiency of NoC. Providing a scalable, high performance, and energy efficient NoC is a key research area for manycore processors.

1.2.1 Contemporary Computing Challenges

Processors evolved in different computing directions with the advent of the Internet. NoCs are popularly used for all computing domains, and its challenges vary with various computing domains of processors as discussed below:

- **Embedded Computers** are designed for a specific purpose. A microprocessor integrates with a device to control its operations. These computers are installed in smart appliances, telecommunication, and automobiles to perform special operations. Modern large embedded systems are considering **M**ulti-**P**rocessor **S**ystem-**o**n-**C**hip (MPSoCs) architectures [26]. These processors are designed to fulfill the unique requirements of embedded applications. They use multiple CPUs along with other hardware subsystems to implement a system. The growing complexity of MPSoC requires NoC communication infrastructure to attain more component integration in less space along with modularity and scalability that make it preferable to buses and point to point communication for large systems. As embedded systems run specific applications, it is convenient to customise NoC for specific traffic patterns that help to attain the objective of more performance, functionality, and flexibility for less cost, lower power consumption and smaller footprint.

- **General Purpose Computers** are designed to meet the need of a variety of applications. They can be programmed to perform the same functions as that of a special purpose computer to solve a particular scientific and engineering problems. Desktop, notebooks, tablets, and smartphones are underlying personal computers that use **C**hip **M**ulti-**P**rocessor (CMP) to **S**ystem-**o**n-**C**hip (SoC) processor architectures [27, 28]. These computers run a vast variety of applications. It is hard to design NoC that works efficiently with a variety of traffic patterns, especially with the constraint that these devices may not always be placed in an air-conditioned environment. Heat dissipation is, therefore, a major challenge. In such a case, NoCs have more emphasis on good power-performance trade-off and energy efficiency.

- **Cloud, Server, and Enterprise Systems** are accessed by personal computer for an individual computing needs that are accessed through Internet facility. Cloud service providers operate as a utility, and these services are charged on a usage basis. Enterprise systems are large computers which are shared by a potentially large number of users. Most of the algorithms are parallel as they are designed for high performance and low energy execution of the application on manycore processors. NoCs for these processors have an emphasis on performance efficiency as these systems provide services to clients [29–31]. Presence of air conditioning makes other metrics secondary issues for these systems.

- **Supercomputers and Grid Computers** are the most expensive, and physically the largest category of computers that normally offer the highest performance. They are used for the highly demanding computations like weather forecasting, scientific work, and engineering design and simulation. Grid computers provide a more cost-effective alternative. They combine a large number of personal computers and disk storage units in a physically distributed high-speed network, called a grid. As performance is a major concern for these computers, NoC infrastructure is primarily designed to achieve performance efficiency to speed up the processors [32].

The wide use of general purpose processors motivates us to address NoC design objectives for these processors.

1.2.2 NoC Objectives

The following NoC objectives are important in meeting the design objective of general purpose processors:

- **Scalability.** NoC provides communication infrastructures for manycore architectures. As network size increases, communication in one machine cycle does not remain possible. Message delivery cycles increase for cores placed at the far ends of the chip. Ensuring NoC scalability in respect of consistent growth in the number of cores that can sustain increasing parallel workloads remains a challenge. So the researcher prefers only those architectural designs and methods that do not impose additional overheads on increasing network size.
- **Limited Power Budget.** Modern power budget of the chip does not permit to switch on all the transistors at the same time despite the abundance of transistors. With each successive process generation, the percentage of a chip that can switch at full frequency drops exponentially due to power constraints. This utilization wall is known as dark silicon, which is under-clocked or not used silicon all the time [76]. The power dissipation raises the core temperature, which affects the performance and reliability of a chip. Power dissipation of NoC approaches roughly 19-35% of total chip power in modern manycore processor designs [76]. Due to power budget constraints, power efficient NoC design is a big challenge for the manycore processors.
- **Energy Efficiency.** Modern range of supporting clock frequency and growing number of on-chip cores set much tighter power budgets for all system components according to the International Technology Roadmap for Semiconductors (ITRS) and Semiconductor Industry Association (SIA) roadmaps [149]. Device scaling depicts benefit in computation and storage components. However communication energy does not scale down. So, minimization of communication-energy is a growing concern in modern technologies.
- **Heterogeneous Applications.** Heterogeneous nature of applications implies different computing requirements. Some applications have a large memory footprint that regularly generates cache misses. These applications are categorized as communication bound workloads. Performance of these applications is closely related to the efficiency of NoC.

Whereas, processor bound applications run with smaller memory footprint meeting overall performance metrics (such as throughput, delay, etc.) in a mix of both memory and processor bound heterogeneous applications is another challenge for NoC architects.

- **Selection of Performance Metrics.** Selection of performance metrics is important for evaluating different NoC designs. In previous research, a major shortcoming was classifying NoC performance in application-agnostic metrics such as message transmission latency and memory bandwidth instead of processor performance metrics such as execution time and throughput [78, 79].

- **Chip Area.** Die area directly impacts the manufacturing cost. The designing cost of manycore chips has been growing alarmingly. For example, as the chip area increases the yield[3] reduces. A good NoC design must acquire a minimal area for targeted performance.

- **Co-Design of On-chip Memory Component and Networks-on-Chip.** Performance of on-chip shared caches is closely coupled with NoC architecture. Any cache miss necessitates a data transfer through NoC. As a result, NoC interconnects primarily decides the performance of memory hierarchy. Exploring a combined design space for on-chip caches, memory controllers, and NoC is another important dimension needed to be explored for a good NoC design.

1.3 Thesis Overview

This section discusses thesis motivation and research plan at hardware and software levels. This is followed by a discussion on thesis contributions to achieve our measurable objectives. By the end of the section, we elaborate organisation of the rest of the thesis chapters.

1.3.1 Thesis Motivation

In the future many-core era, the **N**etworks-on-**C**hip (NoC) power consumption is expected to increase because of the availability of more computation resources. Many of these would have

[3]Non-defective chips or number of good dies

to be switched off while inactive to keep power consumption and chip temperature low. However, the NoC infrastructure must be kept alive to serve shared caches and memory accesses. A recent study shows that the proportion of NoC power consumption becomes appreciable in comparison to computation counterpart. A 32 core chip at 45*nm* substantially raises NoC power ($\sim$ 42%) among the remaining on-chip active resources (cores, shared caches, memory controllers, PCIe controllers) [33]. Therefore, low power becomes the primary objective of modern NoC designs.

In total NoC power budget, static power comprises of more than 74% of the total NoC power at high network utilisation on 22*nm* technology [34]. The proportion of the static power further increases on low network utilisation. This figure is expected to increase in future technology generations. Reducing static power consumption is another design objective of NoC interface between processor and memory cores.

Multiple **N**etworks-on-**C**hip (Multi-NoCs) offer independent parallel data flows through more than one NoC interconnects. Freedom of different hardware customisations along with flexibility in fine-grain traffic distribution makes multi-NoCs more power and energy efficient as compared to bandwidth equivalent single-NoC [38]. Many multi-NoCs customisations are made to address the modern research challenges of dark silicon [94, 122], hardware accelerators [109], reconfiguration [90, 91], fault tolerance [57], security [47], and power gating [46, 93]. Real processor implementation of multi-NoCs is MIT's RAW [68] and TRIPS [69] research prototypes. In commercial chip product series, Tilera [70] and Adapteva Epiphany [71] processors have implemented multi-NoCs.

Multi-NoCs are primarily implemented for application-specific processors e.g., SoC[4], MPSoC[5], and FPGAs[6]. General-purpose processors have not been much explored for multi-NoCs implementations. CMPs[7] run a wide variety of applications with unpredictable heterogeneous

[4]**S**ystem-on-**C**hip
[5]**M**ulti-**P**rocessor **S**ystem-on-**C**hip
[6]**F**ield-**P**rogrammable **G**ate **A**rrays
[7]**C**hip **M**ulti-**P**rocessor

runtime traffic variations[8]. For these processors, it is difficult to design a static power efficient customised multi-NoC that should dynamically adapt to the traffic distribution according to runtime variations of fine-grain messages from computation bound to communication and memory-bound applications. In this thesis, our objective is to design an energy efficient multi-NoC suited for a general purpose computer

1) exploration of static power efficiency through possible customisation in multiple NoC at the hardware level.
2) able to deal with a mix of memory-bound, communication-intensive, and CPU-bound workloads in an adaptive manner.

1.3.2 Research Plan

Conventional multi-NoC architectures are shown in Fig 1.4. There are two distinct categories shown as 'customised multi-NoC version1 (v1)' and 'customised multi-NoC version2 (v2)'. Both the architecture versions are derived through customisation of single-NoC. Architecture v1 consists of two NoC planes, each plane having its NI (**N**etwork **I**nterface), routers and links connecting routers, connected to the core/tile.

Whereas architecture v2 shares routers in the two NoC networks. Here only NI and links are duplicated. The figure shows multi-NoC architectures where requisite resources are duplicated. But this can be extended to wherein resources are quadrupled or replicated to the necessary depth. Both the architectures partition the traffic through multiple networks, albeit in different ways. The v1 cannot provide runtime adaptivity to traffic. Once traffic is separated at NI and injected into one of the networks, it cannot be diverted/jumped to another network as there is no link between routers of two NoC planes. Whereas v2 can move/change (or interchange) the traffic between the networks at runtime (at any point of time). So it is more flexible in terms of traffic distribution. The NoC challenges like congestion, traffic hotspot, faults can be better handled with v2. Therefore, we keep v2 as our baseline multi-NoC design.

[8]Traffic has control and data types of messages. These messages further divided into few more subclasses. The volume of these messages vary from application to application and reflects at runtime in terms of wide variations and impact the performance of NoC.

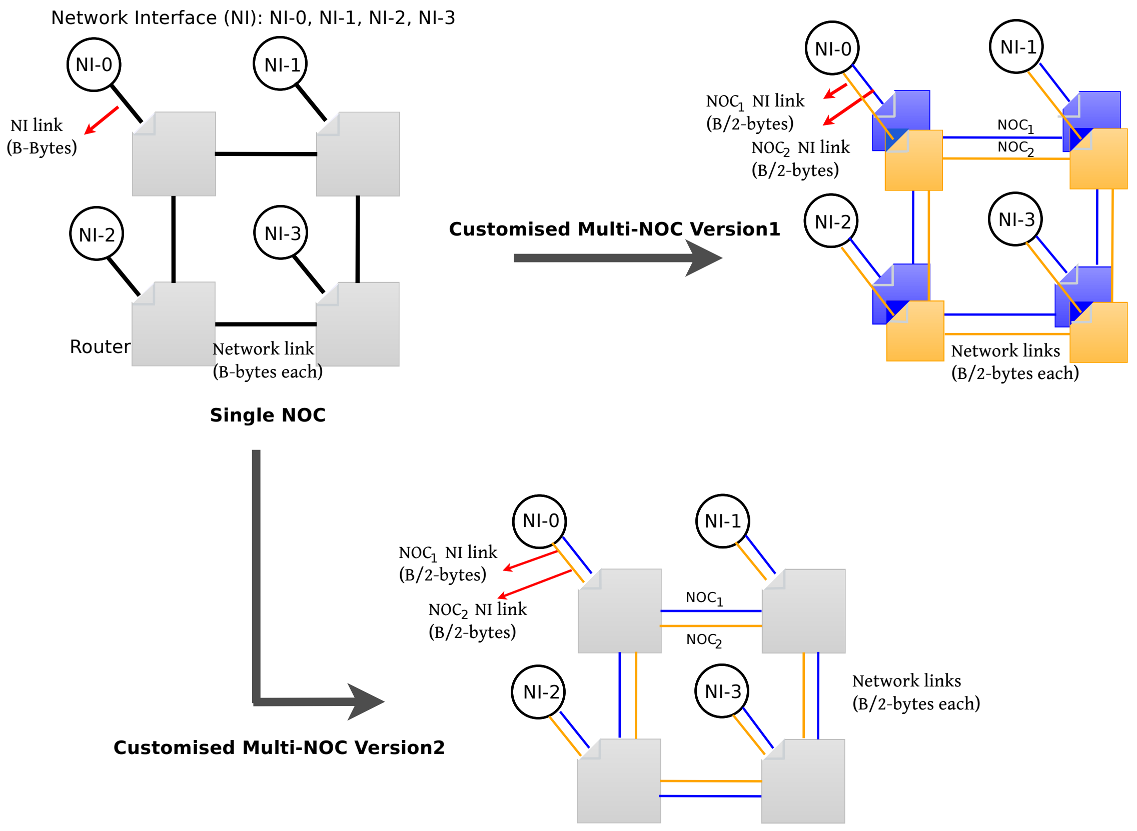

Network Interface (NI): NI-0, NI-1, NI-2, NI-3
NI-0
NI-1
NI-2
NI-3
NI link (B-Bytes)
Router
Network link (B-bytes each)
Single NOC
Customised Multi-NOC Version1
NOC$_1$ NI link (B/2-bytes)
NOC$_2$ NI link (B/2-bytes)
NOC$_1$
NOC$_2$
Network links (B/2-bytes each)
Customised Multi-NOC Version2

FIGURE 1.4: Customisation of multiple NoCs from single-NoC. This figure illustrates two architectural variations of 2-NoC network. (v1) customised multi-NoC version1 wherein resources of single-NoC are entirely partitioned and form two NoCs in different planes; and, (v2) customised multi-NoC version2 wherein router is shared though internal resources of router and physical links are partitioned between both the NoCs.

Fig 1.5, shaded part, presents our research plan, and the leaf nodes summarise our action plan to meet thesis objectives. In our case, NoC is an electrical interconnect as optical, and wireless alternatives are not commercially available. As Multi-NoCs favor a number of simple, independent and parallel data flows, we use multi-NoC rather than single-NoC as our architectural choice.

By replication at various levels of the network hierarchy, different types of multiple-network configurations can be achieved. Such networks can be classified under multi-plane-NoC and multi-network-NoC.

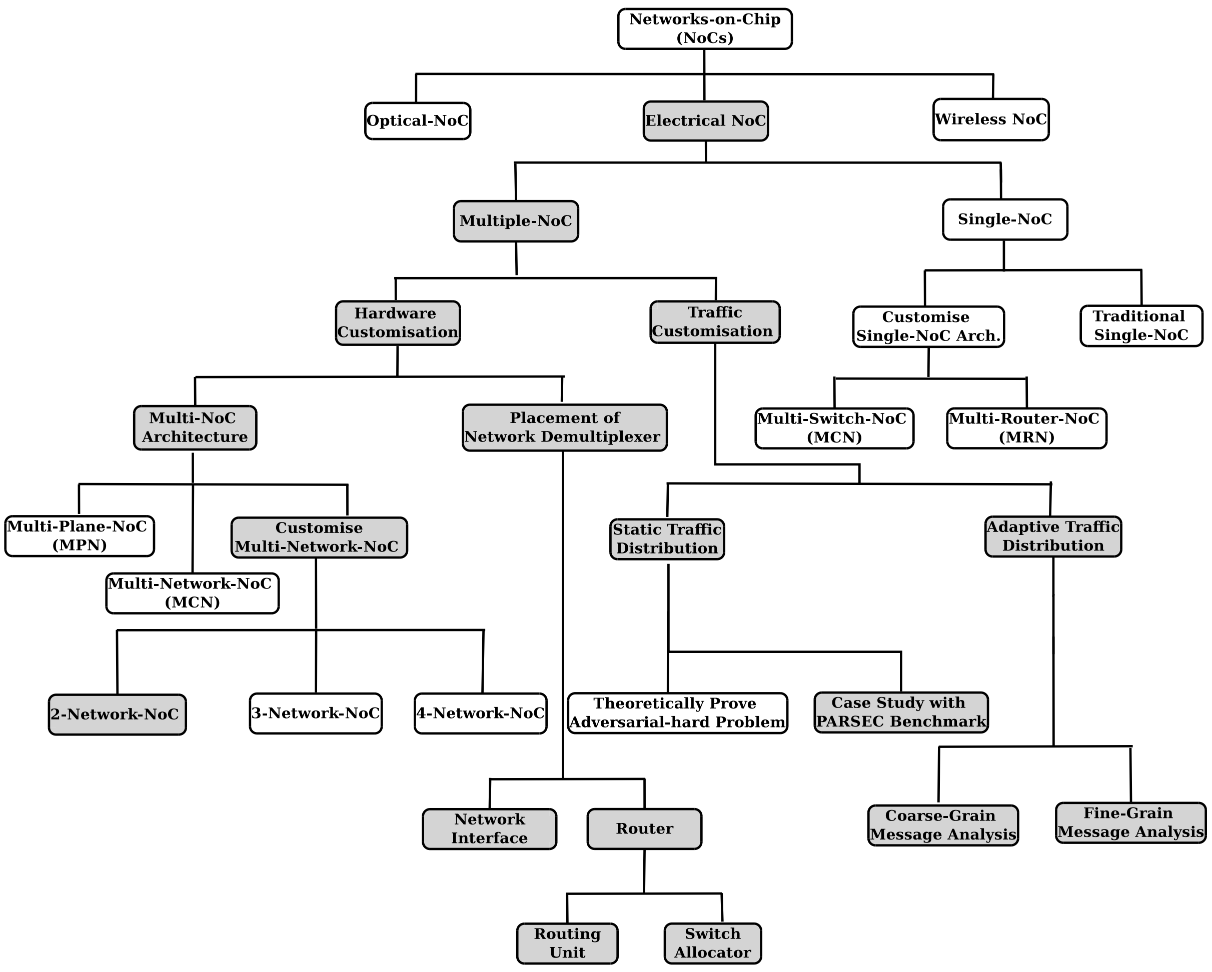


FIGURE 1.5: In our research plan, we target both hardware customisation and traffic customisation for multiple-NoC. In a 2-network-NoC, we explore placement of network demultiplexer for selection of the one of the NoC network from the energy efficiency viewpoint. We also explore how traffic distribution across these two networks affect NoC performance.

Different multiple NoCs as presented in Fig 1.6 [9,10] can be categorized as

1) MPN (Multi-Planar-NoC): Complete NoC including routers and links are replicated and can be viewed as a stack of NoCs, each NoC in a different plane of the stack.

2) MNN (Multi-Network-NoC): All links including NoC as well as NI links are replicated.

[9]The symbols N, E, S, and W represents the directions North, East, South, and West in mesh topology wherein routers are connected with other routers of the topology. Whereas diagonal symbol C represents the direction of router connection to 'Local Core' through NI.

[10]For the sake of simplicity in representation, we assume single NI that is connected to the router through a single NI link (C). In actual network, modern processors use more than one NI links corresponding to their NIs. Multiple levels of on-chip cache hierarchy cause different respective cache controllers and one directory controller. Each one is required individual NI to forward traffic through NoC.

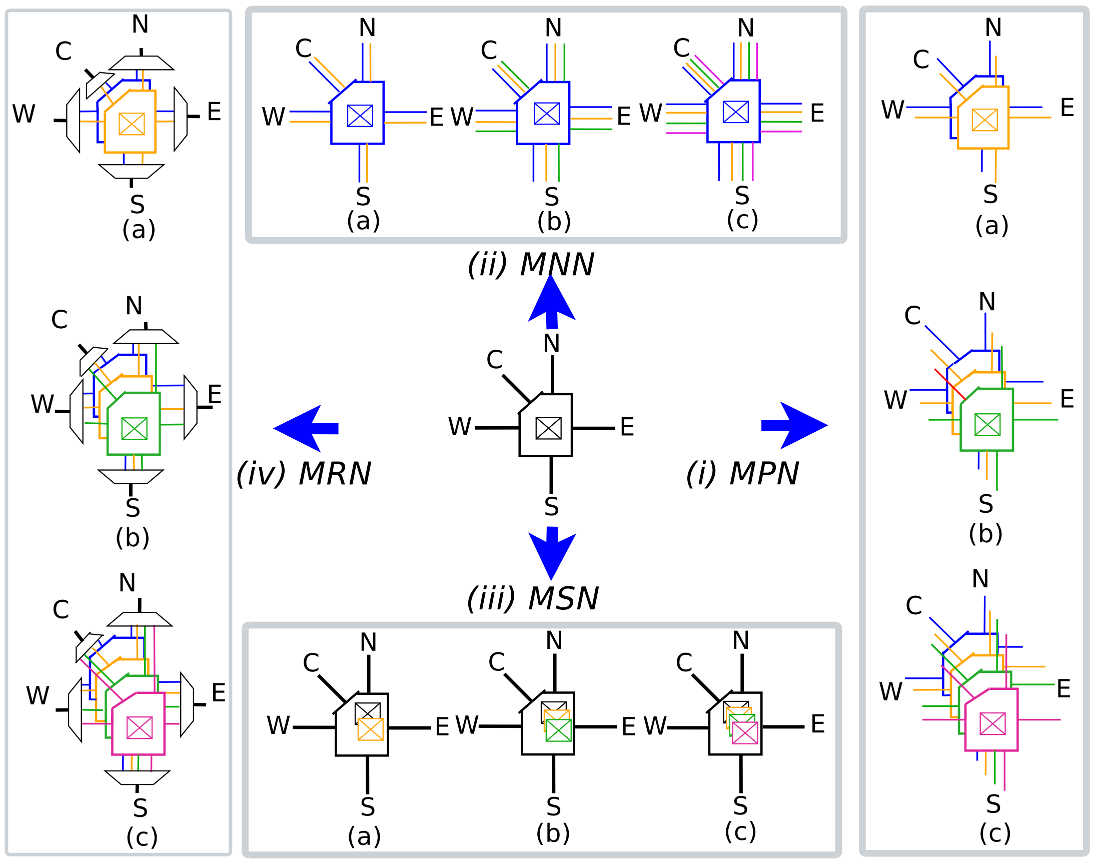


FIGURE 1.6: The different router[9] architecture derived from single-NoC router (center), (i) **M**ulti-**P**lane-**N**oC (MPN): a) Bi-planar-NoC, b) Triple-planar-NoC c) Quad-planar-NoC, (ii) **M**ulti-**N**etwork-**N**oC (MNN): a) Dual-Network-NoC, b) Triple-Network-NoC c) Quad-Network-NoC, (iii) **M**ulti-**S**witch-**N**oC (MSN): a) Dual-Switch-NoC, b) Triple-Switch-NoC c) Quad-Switch-NoC, (iv) **M**ulti-**R**outer-**N**oC (MRN): a) Dual-Router-NoC, b) Triple-Router-NoC c) Quad-Router-NoC

3) MSN (Multi-Switch-NoC): Only switching hardware within router (for example, crossbar switch) is replicated.

4) MRN (Multi-Router-NoC): Only routers are replicated, they are connected through single links in the network.

Depending upon the degree of replication, networks can be dual (only duplication of the resources), triples (three units instead of one), and quad (each resource is replicated four times). For the sake of simplicity in hardware implementation, we limit[11,12] our implementation up to dual-network NoC.

[11]We have compared dual-NoC hardware implementation with quad-NoC in Chapter 2 (Page 50).

[12]In brief, in Chapter 5 (Page 116), we have discussed the impact on hardware implementation with an increasing number of NoC networks.

1.3.3 Thesis Objectives

The thesis motivation discussed in Subsection 1.3.1 identifies currently unexplored issues of NoC power efficiency and traffic distribution of general purpose chip multiprocessor. The objective of our thesis is to primarily explore NoC efficiency for CMPs at the hardware and software levels using multiple NoC networks.

1) **Architectural Customisations at Hardware Level**

   Our thesis investigates static power efficiency through architectural customisations at the hardware level. The following questions need to be answered in the perspective of different goals and commonplace use of dual-network NoCs for manycore general-purpose processors:

   - Is it possible to make dual-network NoCs more static power efficient over traditional and state-of-the-art multi-NoCs through customisation of these architectures?
   - Can we place the of network selection hardware unit somewhere other than the conventional network interface placement in customised dual-network NoC architectures? Is there any impact of placement on static power and energy efficiency?

2) **Traffic Distribution at Software Level**

   NoC performance is significantly affected by traffic distribution as efficient distribution balances traffic across the routers and links of the network. In the case of dual-network NoC, the conventional traffic distribution methods need to be re-explored to address modern challenges. Therefore, exploring a new traffic distribution techniques is our another thesis objective. We shall explore the following questions in the thesis at software level:

   - What is the impact of retrospective static traffic distribution variation on dual-network NoCs? Is there any impact on performance and energy efficiency? If so, how do we capture such impact/limitations with real benchmarks of general-purpose applications?
   - Is there any method to improve the shortcomings of static traffic distribution? Can we improve the underutilization of dual-network NoC networks at runtime?

- How do we make dual-network NoC adaptive to runtime traffic variations of applications on general-purpose processors? How do we analyse NoC traffic at the fine-grain level to devise criticality/urgency of messages? How do we decide fine-grain messages distribution to utilize underutilized NoC network?

1.3.4 Thesis Contributions

In our thesis, we have explored static power and energy efficiency at hardware and software levels. We have customised dual-network NoCs for static power efficiency that is further explored for energy efficient placement of NoC network selector. On exploring such custom implementations at hardware levels, we accompany it with software level energy efficient traffic distributions. For CMPs, exploration of traffic distribution is equally important as hardware customisations. Through these proposals, we primarily target power, energy, and area efficiency without any significant impact on latency, execution time, and throughput of NoC. We achieve these measurable objectives in Chapter 4, 5, 6, and summarize these contributions in Chapter 7. Our novel contributions are as fol lows:

1) **Customised Dual-Network NoC Architecture to Attain Power Efficiency.** In multiple NoCs, networks carry parallel traffic flows through external NI links and internal network links. We customise[13,14,15] multi-channel NoCs through keeping a single NI link. Only router-to-router links are replicated as shown in Fig 1.7. Such customisation reduces the static power without impacting traffic distribution flexibility of multiple NoCs. Backpressure mechanism automatically fills the buffers of both NoC since core and NI are faster in message processing than NoC. Thus, single NI links shall not affect traffic injection rate, and duplication of NoC links speed up communications through the availability of two parallel links at each port. The static power advantage comes from half of the external NI links that reduce the number of I/O ports of the router, the number of **V**irtual **N**etworks (VNs) and **V**irtual **C**hannels (VCs), routing logic and control logic overhead, size of the crossbar, etc.

[13] A. Sharma, Y. Gupta, S. Yadav, M. S. Gaur, L. Bhargava, V. Laxmi, "A Power, Thermal and Reliability-Aware Network-on-Chip," in Proc. of IEEE-iNIS, **IEEE**, Bhopal, Dec. 2017.

[14] S. Yadav, V. Laxmi, M. S. Gaur, "C^2-DLM: Cache coherence aware dual link mesh for on-chip interconnect," in Proc. of 19th Int. Symp. on VLSI Design and Test (VDAT), **IEEE**, pp.1-2, 26-29 June 2015.

[15] S. Yadav, V. Laxmi, and M. S. Gaur, "A Power Efficient Dual Link Mesh NOC Architecture to Support Nonuniform Traffic Arbitration at Routing Logic," in Proc. of the 29th Int. Conf. on VLSI Design (VLSID), **IEEE**, Kolkata, pp. 69-74, Jan. 4-8, 2016.

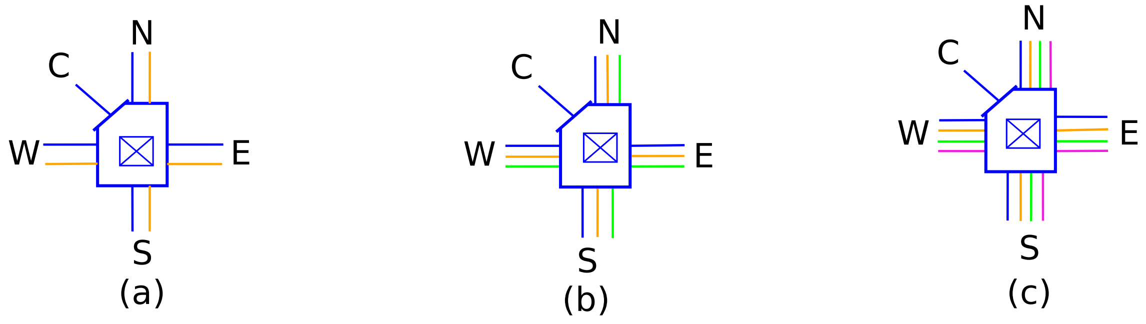


FIGURE 1.7: Proposed customisation in multi-network-NoC architecture (a) proposed 2-network-NoC (b) 3-network-NoC (c) 4-network-NoC

In the proposed architecture, dual networks originate from the router using network links, so we have placed network selection hardware unit along with the routing unit of the router rather than conventional network interface placement. This customisation is scalable with network size and results in following improvements–

- Through synthesis, we achieve 62% and 58% improvement in static power, and 30% and 25% area efficiency with proposed 2-network-NoC over conventional dual-network-NoC and single-NoC respectively at 32*nm* technology.
- Experiment on PARSEC[16] benchmarks finds 45% efficiency in total router power at (65*nm*, 1*GHz*). The power efficiency approaches 58% as technology shrink to 32*nm* at 1*GHz* frequency. In contrast, the power efficiency limits to 40% as frequency increases to 2.5*GHz* at 65*nm* technology.

2) **Target 2-Network NoC Architecture.** The placement[17] exploration of the network selection hardware unit has not been reported prior to our work in multi-NoC. As a flit is demultiplexed to any of the multiple NoC networks through network selection hardware unit, we named it as **N**etwork **D**emultiplexer (`Net-Demux`). The conventional multi-NoCs design places `Net-Demux` at the network interface, whereas our proposed 2-network-NoC places `Net-Demux` along with routing unit of the router. The network interface places `Net-Demux` in the datapath whereas at the router the placement is possible at the control unit. The placement at the control unit of the router reduces area and power overhead. In a router, the critical path delay is dominated by the crossbar. The placement of `Net-Demux`

[16]**P**rinceton **A**pplication **R**epository for **S**hared m**E**mory **C**omputers

[17]S. Yadav, V. Laxmi, H. K. Kapoor, M. S. Gaur, and M. Zwolinski, "A Power Efficient Crossbar Arbitration in Multi-NoC for Multicast and Broadcast Traffic," in Proc. of IEEE-iSES'18, **IEEE**, Hyderabad, IN, Dec. 17-19, 2018. (Best Paper Award)

at the switch allocator[18] (schedules the crossbar connections between input and output ports for flit traversal) is more efficient as compared to the routing unit placement. The experimental results show–

- Through synthesis, we achieve switch allocator placement is 21% and 41% static power efficient over network interface placement of dual-Network-NoC and single-NoC, respectively. This is 29% improvement over routing unit placement.
- Experiments with PARSEC benchmark show that switch allocator, routing unit, and network interface placements are 46%, 40% and 30% energy efficient over single-NoC on PARSEC benchmarks.
- The placement at switch allocator and routing unit are respectively 33% and 26% more energy efficient over conventional network interface placement.
- The placement at switch allocator improves the critical path delay by 33% over conventional NI placement.

3) **Case Study on Static Message Distribution Problem of CMPs and Proposed Runtime Adaptive Fine Grained Message Distribution for Improved Runtime Utilisation of 2-Network-NoC.** Most of the multi-NoCs uses static communication infrastructure. Static traffic distribution is the simplest offline message distribution method. We have conducted and present a case study[19] for different combinations of static traffic distribution to find its impact on performance and energy efficiency of multi-NoCs. We observe that it is hard to find the best solution from all the feasible choices. Efficient static message distribution can be determined for a single benchmark, but the same distribution policy is not necessarily the best solution for the rest of the applications. We have following conclusions with the case study using general-purpose applications of PARSEC benchmark–

   - Communication intensive benchmarks like swaptions (in power), x264 (in energy), and rtview (in throughput) show significant $\approx 5\times$ variations.

[18]S. Yadav, V. Laxmi, M. S. Gaur, and H. K. Kapoor "Late Breaking Results: Improving Static Power Efficiency via Placement of Network Demultiplexer over Control Plane of Router in Multi-NoCs," in Proc. of Int. Conf. Design Automation Conference (DAC), **ACM/EDAC/IEEE**, Las Vegas, NV, US, June 2-6 2019.

[19]S. Yadav, V. Laxmi, M. S. Gaur, and H. K. Kapoor, "Adaptive On-the-Fly redistribution of Traffic with Multi-NoC for Fair Distribution of Traffic in Chip Multiprocessor," in Journal of System Architecture, Elsevier, February 2019. (Communicated)

To alleviate these deficiencies, we also propose runtime adaptive message distribution[20] for customised multi-NoCs. The proposed adaptive method considers the runtime dynamics and balances the traffic between multiple NoC network. Adaptivity utilises the flexibility of multiple NoC architecture and allows messages to change the NoC networks at runtime on underutilisation of NoC networks. We compare the benefits of adaptive approach with two state-of-the-art traffic distribution as follows.

- Runtime adaptive traffic distribution attains energy saving up to 36% and 14% compared to the single-NoC and static traffic distribution respectively for communication-intensive PARSEC benchmarks suite.
- The link utilisation improves 16% and 21% over static message distribution and single-NoC.

We have implemented customised multiple NoC in GARNET[21] that is integrated with Gem5 simulator. The results are evaluated using PARSEC benchmarks in full-system simulation. The area and power of hardware implementation are estimated through synthesis using Synopsis Design Compiler.

1.4 Thesis Organization

Chapter 2 presents a detailed review on multiple networks-on-chip. We have classified and compared various multiple NoC architectures from the perspective of result metrics, processor types, their design objectives, and addressed NoC problems.

Chapter 3 compared the traffic distribution of dual NoC with single-NoC. We have briefly discussed our experimental setup and simulation tools. A brief introduction is presented on essentials for designing multiple networks-on-chip.

Chapter 4 describes in detail the proposed 2-network-NoC architecture. In this design, the network selection hardware unit is needed to place along with the routing unit of the router. The efficiency of proposed NoC architecture is compared with single-NoC architecture.

[20] S. Yadav, V. Laxmi, M. S. Gaur, and H. K. Kapoor, “Comprehensive State Space Model of Caches with respect to On-Chip Interconnect Traffic,” in Proc. of Int. Conf. on (RIEECE), **IEEE**, Bhubaneswar, IN, July 2018. (in press)

[21] A detailed cycle-accurate network-on-chip model inside Gem5 full-system simulator.

Chapter 5 covers how the placement of network selection hardware unit impacts power, performance, and energy efficiency of multi-NoCs micro-architecture. It also shows that the proposed routing unit and switch allocator placements are more energy efficient over network interface placements.

Chapter 6 describes the proposed case study for static traffic distribution on 2-network-NoC. This case study shows the limitations of static traffic distribution and its impact on power, performance, and energy efficiency of NoC. We propose a runtime adaptive traffic distribution to overcome the limitations of static traffic distribution. Router adapts itself with the runtime variations of messages for different traffic classes across various applications of general purpose computing.

Chapter 7 concludes the thesis and discuss future research directions.

Chapter 2

Multiple-NoC Overview

In this chapter, we have presented a detailed literature review on multiple networks-on-chip. These architectures are inspired from earlier proposed multiple interconnects such as multiple buses, multiple rings, and modern hybrid wireless interconnects. Few additional customised architectures are also surveyed wherein additional signal lines are inserted to carry forward some messages. All these customised architectures were proposed for achieving different power-performance tradeoff within the constraint of their architectural limitations.

Using multiple networks-on-chip, various NoC research problems are resolved. We have classified their literature according to targeted processor types, performance metrics, various architectural customizations to meet the specific objective of application areas. In addition to hardware level customisations, we have also reviewed the conventional and modern traffic distribution methodologies used by various multiple networks-on-chip architectures.

2.1 Introduction

One way to improve performance is to use multiple hardware resources in parallel. Replicating hardware, however, multiplies the cost and power of the circuit. So it is not a practical approach to gain performance. Alternatively, rather than doubling the hardware resources slice the resources to form parallel lightweight interfaces for parallel operations. Slicing of hardware resources was initially implemented for **A**rithmetic **L**ogic **U**nit (ALU) to improve the performance through multiple parallel computations. Bit-sliced ALUs process k-bit data that are divided into m slices of $\frac{k}{m}$ bits. For example, a 64-bit data can be sliced into four ALUs, and each computes ($\frac{64}{4}$)=16-bits. The example of these processors is **G**eneral **P**urpose **G**raphical **P**rocessing **U**nit (GPGPU). The bit-slice approach simplifies the circuit structure and reduces the hardware cost.

Recent work on NoCs has focused on utilizing replication to increase throughput without any latency penalty for multithreaded and shared workloads. The duplication of the network can double throughput while using the spare area of tile after mapping of the basic router. Plane replication is not a new idea, the mad postman network [8] used four independent planes to forward packets in a 2D mesh, but it has not been explored before due to its silicon cost.

2.2 Multiple Networks-on-Chip

Multiplanes are going to play an important part in modern and future NoCs as they are able to utilize the unused area in tiled architectures, exhibit low power dissipation rates and could simplify the critical path by reducing/removing virtual channel arbitration [148].

Table 2.1: Different Processors with Multiple NoCs

Parameters	Configuration
CMP	C. Gomez et al. [127], 2008; A. K. Abousamra et al. [106], 2012; M. Lodde et al. [130], 2012; J. S. Miguel et al. [8], 2015; Z. Li et al. [109], 2016
MPSoC	F. Gilabert et al. [39], 2010; H. Bokhari et al. [123], 2014; V. Akhlaghi et al. [110], 2015
SoC	S. Noh et al. [119], 2006; J. Wu et al. [124], 2016; Y. J. Yoon et al. [131], 2017, J. Wu et al. [93], 2017
FPGA	B. Sethuraman et al. [128], 2005; P. Ezhumalai et al. [121], 2010

2.2.1 Multiple NoCs for Various Processors

The work on multiple networks-on-chip has been reported for various processor types as listed in Table 2.1. The processors are designed specifically to the applications they run. Therefore, the architecture of multiple networks-on-chip is customized according to the workloads carried by the network.

CMPs. Chip multiprocessors integrate tens or even hundreds of processor cores onto the same die. It constitutes an alternative to traditional monolithic designs with better levels of performance, scalability, and performance/energy ratio [52]. Higher clock frequencies and the increasing transistor density have necessitated power dissipation and temperature as critical design issues in current and future architectures.

In CMP systems, all the processor cores and other support components (memory components, accelerators, memory controllers) are interconnected using a high-speed on-chip network. Examples include prototypes such as the 48-core Single-chip Cloud Computer developed by Intel and real products such as the Tilera's 100-core TILE-Gx100 processor.

MPSoCs. Multi-Processor Systems-on-Chip (MPSoCs) represents a much more resource-constrained domain. Area and power-efficient implementations are of paramount importance to fulfill the requirements of embedded system platforms like those for multimedia, broadband, and networking applications. Operating frequencies are typically lower. In this domain, resource overprovisioning to meet predefined performance constraints is not affordable, therefore VCs are an attractive solution to maximize link utilization. MPSoCs are most sensitive to their area and power overheads.

SoCs. System-on-Chip features an increasing number of processing elements (PE) and other components. The entire system, which is required to run a device, is integrated into a single chip that is known as SoC. The leading manufacturers like Qualcomm, Samsung, Apple, and MediaTek are manufacturing SoCs for smartphones and tablets. The fundamental challenge of using NoCs is that systems often have very different constraints than other processors. The primary objective of NoC is to lowering the thermal and power issues of SoC and provide just enough performance to accommodate specific tasks. Therefore, thrift is a virtue in designing NoC while targeting power and area reduction.

FPGA. NoCs were introduced into the FPGA domain mainly to simplify tile-based reconfiguration and its effective communication architecture. Research in [5] [6] address the capabilities of FPGAs to support NoC based multi-processor applications. Hilton et al. [4] incorporate flexibility into their design for FPGA based circuit-switched NoCs.

2.2.2 A note on terminology and quality metrics

Partitioned/sliced/multiple/layers/multi/subnetworks are the different terms used by researchers to describe communication infrastructures that simultaneously employ more than one NoC networks.

2.3 Multiple/Parallel Networks Architectures

In this section, we review how multiple networks evolved from multiple interconnects to multiple NoCs to address various research challenges.

2.3.1 Multiple Ring Architectures

In the 90s, multi-ring networks were explored in a lot of research work. Few of them are discussed here. The ring interconnection network, as shown in Fig 2.1 (a), has many attractive properties. One important property is that each processor in a ring requires a fixed number of links (only two) irrespective of the size of the network that makes ring interconnection network truly scalable. Such systems have simpler wiring and are therefore relatively inexpensive to build. These networks have one serious drawback of inefficient interprocessor communication between processors that are not neighbors. Therefore, broadcasting at interconnection level is a problem. The MultiRing network, as shown in Fig 2.1 (b) provides an efficient and general interprocessor communication and broadcasting mechanism at the interconnection level, unlike the simple ring network.

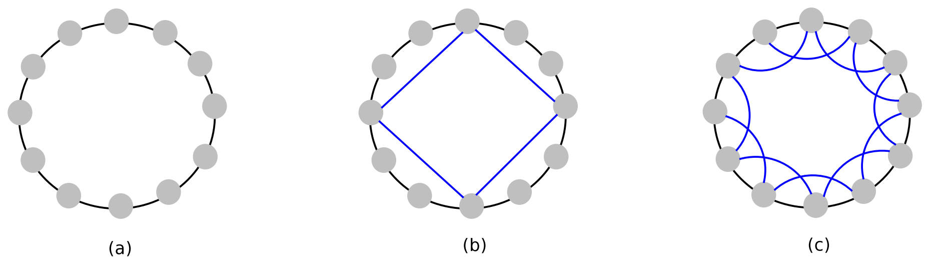


FIGURE 2.1: Outer and Inner Rings

In 2002, M. Junginger et al. [99] proposed that data communication can be improved by applying overlay networks wherein a topology consists of multiple rings, backup links and dual mode links as shown in Fig 2.1 (c). Backup links are activated when the system detects the broken links that were connected to the failed node. The activated backup link is converted to a regular link. The proposed multi-ring improves performance and scalability of peer group communication.

In 1997, H. R. Arabnia [103] proposed a reconfigurable multi-ring network (RMRN), as shown in Fig 2.2 (a). It is designed for more complex parallel algorithms. The RMRN is a viable architecture for image processing and computer vision problems. The network was designed to address the problem of stereo correspondence, known as the stereocorrelation operation. Stereocorrelation is one of the most computationally intensive imaging tasks required in many applications, including remote sensing, geographic information systems, and robot vision.

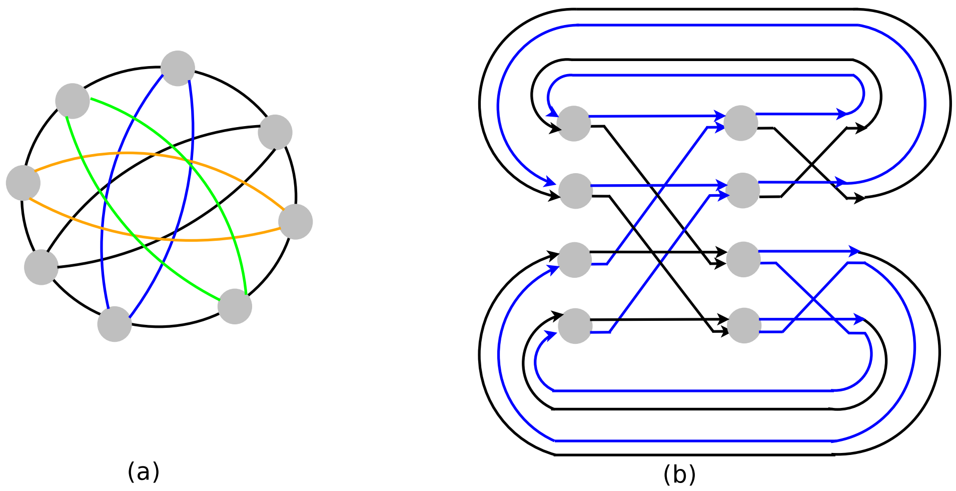


FIGURE 2.2: (a) Reconfigurable Multi-ring Network (b) Multi-stage Ring Networks

In 1999, D. Yoo et al. [100] presented a new multiple ring network for multiprocessors, called the Multistage Ring Network(MRN), as shown in Fig 2.2 (b). The MRN has a 2-level hierarchy of rings, and its interconnection of global rings forms a type of the multistage network.

In 1998, A. S. T. Lee et al. [104] proposed a heuristic for designing a set of stacked rings given an arbitrary set of node-to-node demands. A stacked ring set, as shown in Fig 2.3, is one where several rings are routed around the same topology but each ring may adopt different sizes with the overall aim to reduce the total node count. A random search scheme is employed to find feasible routing and wavelength assignment for connections on individual rings.

In 2006, M. A. Wani et al. [105] published a parallel algorithm for polygon approximation targeted at reconfigurable multi-ring hardware, as shown in Fig 2.4 (a).

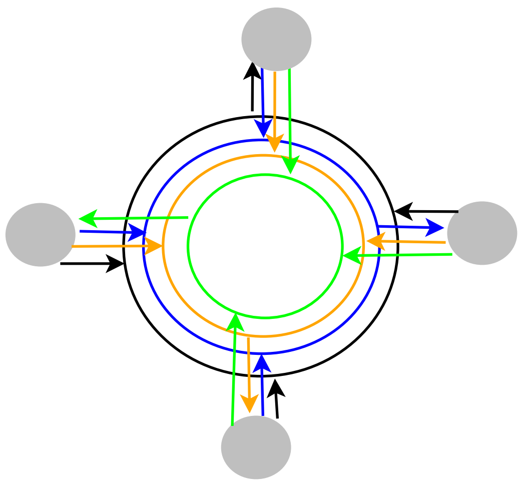


FIGURE 2.3: WDM Stacked Rings

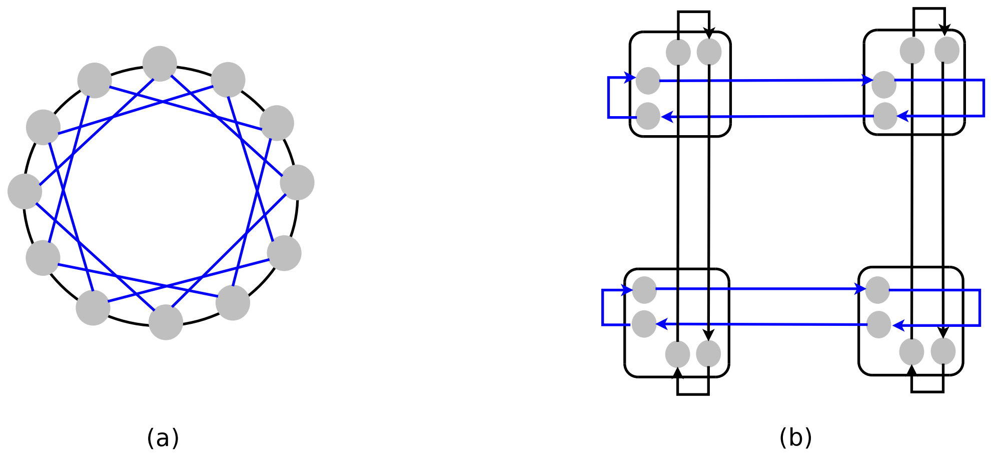


FIGURE 2.4: (a) Reconfigurable multi-ring for polygon approximation (b) Ring-based routers composing TornadoNoC wherein a box denotes a ring-based router, and a filled circle denotes a ring. An inter-ring hardware interface at each router provides the communication between horizontal and vertical rings of the mesh.

In 2013, J. Lee et al. [97] proposed a TornadoNoC having multiple ring-based router microarchitecture, as shown in Fig 2.1(b). The proposed architecture is a lightweight and scalable on-chip network architecture for the manycore processors. To prevent livelocks and deadlocks, a sequence numbering scheme and a dynamic ring inflation technique are proposed, and their correctness is formally proven. The primary objective of TornadoNoC is to achieve substantial gains in (a) scalability to manycore systems and (b) the area/power footprint, as compared to current state-of-the-art router implementations. The new router is demonstrated to provide better scalability to hundreds of cores than an ideal single-cycle wormhole implementation and other scalability-enhanced low-cost routers. Extensive simulations using both synthetic traffic patterns and real applications running in a full-system simulator corroborate the efficacy of the proposed design. Finally, hardware synthesis analysis using commercial 65nm standard-cell libraries indicates that the area and power budgets of the new router are reduced by up to 53% and 58%, respectively, as compared to existing state-of-the-art low-cost routers.

During the 90s a number of multi-ring architectures have been proposed. During that generation, the design was focused on replication of hardware resources to form multiple networks. In the 20th century, area and power became a big issue with the advent of manycore processors.

2.3.2 Multiple Interconnects

Flores et al. [52] propose heterogeneous interconnects for energy-efficient message management in CMPs. Continuous improvements in integration scale have made major microprocessor vendors moving to designs that integrate several processing cores on the same chip as the interconnection network significantly impacts overall performance and energy consumption [142].

Authors propose a heterogeneous interconnect to efficiently manage messages for performance and energy efficiency. Such proposal consists of two approaches. The first is Reply Partitioning, a technique that splits replies with data into a short Partial Reply message that carries a subblock of the cache line that includes the word requested by the processor plus an Ordinary Reply with the full cache line. This technique allows all messages used to ensure coherence between the L_1 caches of a CMP to be classified into two groups: critical and short, and noncritical and long. The second approach is the use of a heterogeneous interconnection network composed of low-latency wires for critical messages and low-energy wires for noncritical ones.

Detailed simulations of 8 and 16-core CMPs obtain average savings of 7% in execution time and 70% in the Energy-Delay squared Product (ED^2P) metric of the interconnect over previous works (from 24 to 30 percent average (ED^2P) improvement for the full CMP).

2.3.3 Additional networks with Networks-on-chip

In 2012, M. Lodde et al. [130] design a fast and simple gather control network (GCN) as shown in Fig 2.5 along with the NoC that collects all the acknowledgement messages of the L1 caches. The main NoC is still used to transport requests, responses (including messages with data) and coherence requests. However, all the acknowledgement messaging associated with coherence

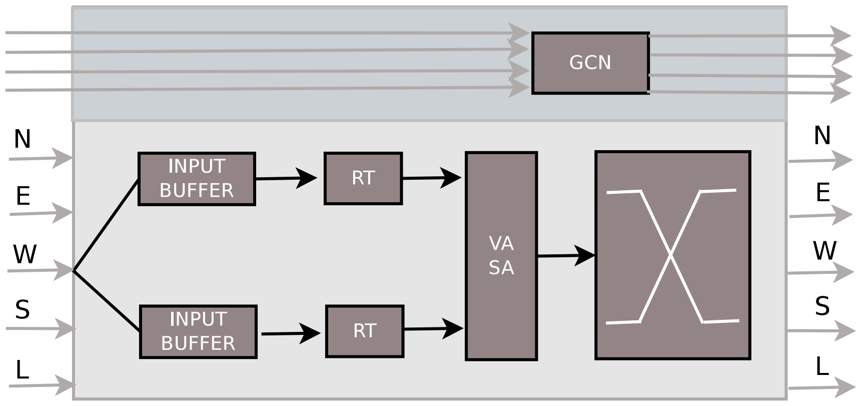


FIGURE 2.5: Additional Networks [130]

requests is separated from the NoC and sent through the gather network. Experimental results demonstrate on a 16-tile system with the control network that execution time improves up to 17%, with an average improvement of about 7.5%. The control network has a negligible impact on the area when compared to the switches.

2.3.4 Multiple Routers for Networks-on-Chip

NoCs can consume a considerable share of chip power. Moreover, diverse applications are executed in these multicores, where each application imposes different communication requirements on the NoC. To realize a NoC which is Energy and Delay efficient, in 2015, H. Bokhari et al. [129] propose Malleable NoC as shown in Fig 2.6 that is composed of multiple **V**oltage **F**requency (VF) optimized routers instead of single router per node in traditional NoCs. At runtime, depending on application profile, they select one of VF optimized routers to form con-

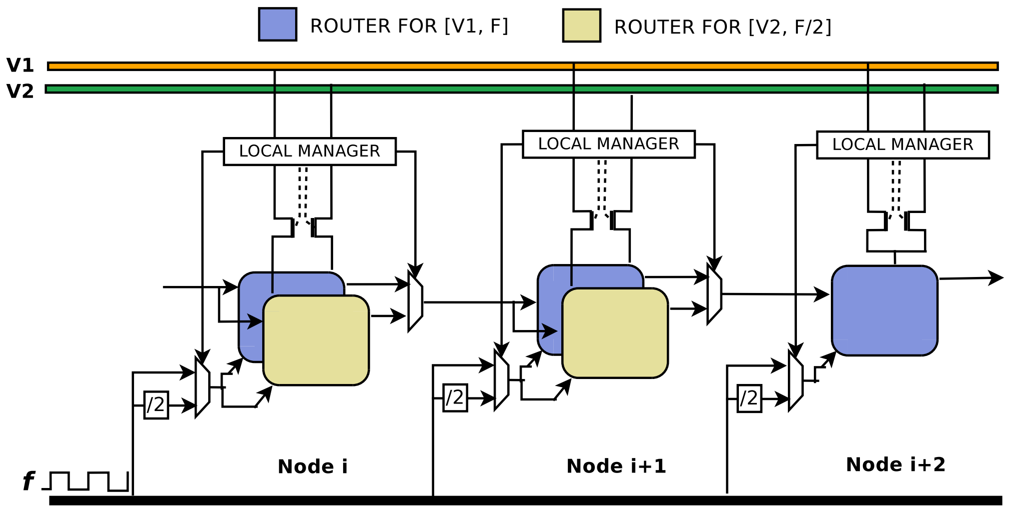


FIGURE 2.6: Multiple Routers [129]

stantly changing energy efficient NoC fabric. It enables utilizing one router from each node to create an adaptable NoC at runtime. They use the multi-Vt optimization technique to design routers for different VFs. Low Vt cells can switch faster but consume high energy, whereas high Vt cell consumes low energy but switch at a much lower frequency. Multi-Vt optimisation is a technique to improve the power efficiency of circuits wherein low Vt cells are inserted on circuit logic paths with negative slack to meet the latency constraint. And, high Vt cells replace normal cells on circuit paths with positive slack to save energy.

A variety of multi-program benchmarks executing on Malleable NoC exhibit a reduction in Energy Delay Product (EDP) as much as up to 46% for widely differing workloads.

These schemes adopt many switching voltage regulators to scale the voltage. However, these switching voltage regulators are expensive for chip area. At the same time, it is necessary to add some special buffers to connect the routers in different clock domains.

2.3.5 Hybrid Wireless Networks-on-Chip Architecture

Heterogeneous System Architectures (HSA), in Fig 2.7, integrate cores of different architectures (CPU, GPU, etc.) on single chip that are gaining significance for many high-performance applications. NoCs in HSA are monopolized by high volume GPU traffic, penalizing CPU bound application performance. In addition, building efficient interfaces between systems of different specifications while achieving optimal performance is a demanding task. Homogeneous NoCs are widely used for many core systems, though, they fall short in meeting these communication requirements. To achieve high-performance interconnection in HSA, S. H. Gade et al. [96], in 2017, propose HyWin topology using mm-wave wireless links. The proposed topology implements sandboxed heterogeneous sub-networks, each designed to match the needs of a processing subsystem, which are then interconnected at second level using the wireless network. The sandboxed sub-networks avoid conflict of network requirements while providing optimal performance for their respective subsystems. The long-range wireless links provide low latency and low energy inter-subsystem network to provide easy access to memory controllers, lower level caches across the entire system. By implementing proposed topology,

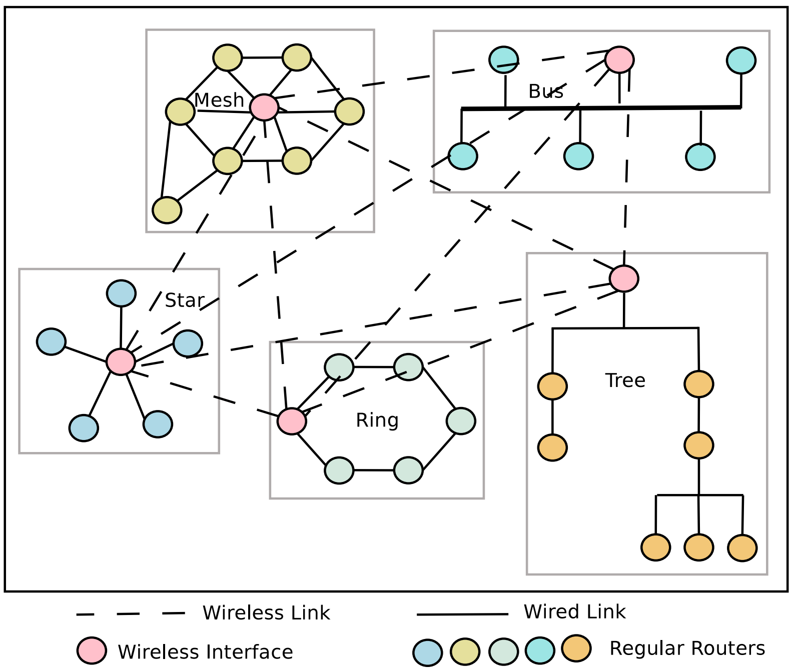


FIGURE 2.7: Heterogeneous subnetworks interconnected by mm-wave wireless interfaces [96]

authors improve application performance by 29% and reduce latency by 50%, while reducing energy consumption by 64.5% and area by 17.39% as compared to baseline mesh[1] [96].

2.3.6 Shortcut Paths Through Long-Range Links along with Regular NoC

Similar to multiple NoCs, Teimouri et al. [126] divide the n-bit wide network resources in a router, such as links, buffers, and a crossbar, into two parallel $\frac{n}{2}$-bit sub-networks to support reconfigurable shortcut paths. These networks, along with regular networks, are useful for application specific processor where network traffic communication is known a priori, and shortcut paths through long-range links are inserted accordingly. A good volume of throughput is achieved, albeit on additional hardware cost and power.

2.4 Multiple NoC Implementations

The work on multiple networks-on-chip can be broadly classified between physical implementation and academic proposals by researchers.

[1]In baseline mesh topology, the CPU tiles (single CPU tile consists of CPU core associated with L_1 and L_2 caches), GPU tiles (single GPU tile has four GPU cores and L_2 cache), memory controllers tiles and tiles of the last level shared caches are arranged in a grid layout, and each tile is connected with a router.

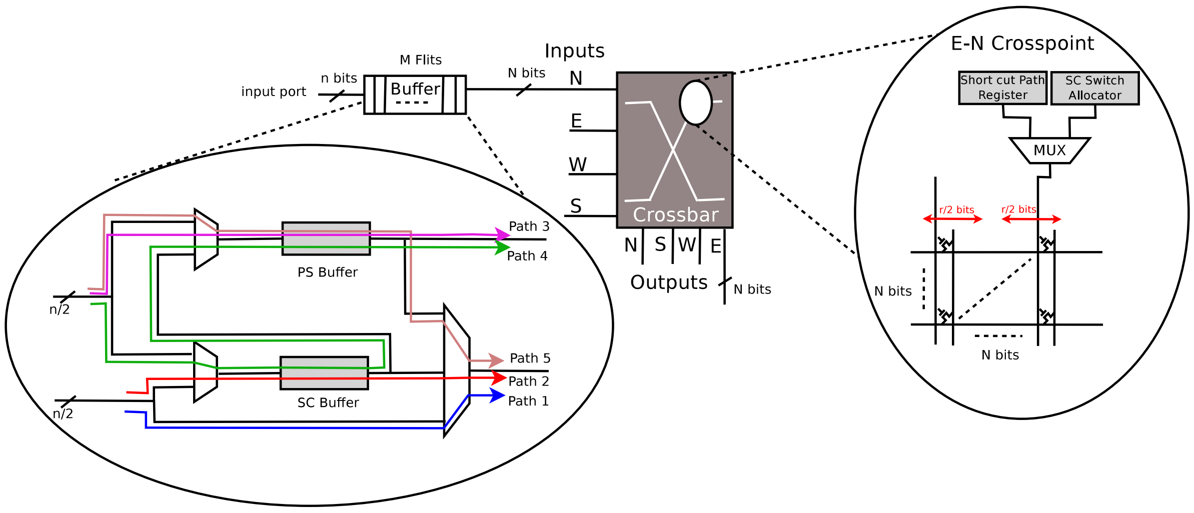


FIGURE 2.8: Router architecture with reconfigurable shortcut paths [126]

2.4.1 Physically Implemented Multiple Networks-on-Chip

Table 4.1 summarizes list of multiple NoCs designed and physically implemented in real. In the following paragraphs, details of each are discussed briefly.

TABLE 2.2: Multiple NoCs Based Real IC Designs

Architecture	Year
Mad Postman Routing Chip (MP1)	Computer Systems Research Group [120], 1992
The RAW Microprocessor	M. Taylor et al. [68], 2002; J. S. Kim et al. [133], 2003, M. Taylor et al. [134], 2004
On-Chip Interconnection Networks of the TRIPS Chip	P. Gratz et al. [69], 2007
On-Chip Interconnection Architecture of the Tile Processor[a]	D. Wentzlaff et al. [35, 70], 2007
Adapteva Epiphany[b]	A. Varghese et al. [71], 2011

[a]TILE-Gx processor series continue bringing new products based on TILE-Gx.
[b]Adapteva Epiphany continues in the market with new series of manycore processors.

1) **Mad-Postman Routing Chip (MP1)**

In 1992, Computer Systems Research Group at the University of Surrey [120] fabricated Mad-Postman Routing Chip (MP1) prototype that is the first step in the development of fine-grain scalable parallel computers. The four routers are configured into grid or torus networks.

The MP1 design takes the advantage of mad-postman flow control and adaptive routing to handle the congestion of the network. Both features together promise a significant reduction in latency to design a fast network chip for parallel computers.

It is is not an optimal implementation.

2) **Raw Microprocessor**

In 2002, another multi-NoC chip prototype was designed and known as MIT's RAW [68] architecture. It uses two static networks for operands and other two dynamic networks for remaining traffic.

Performance, energy efficiency, and cost-effectiveness are gained through the design of multi-NoC for RAW architecture. For scalar codes with a moderate degree of instruction-level parallelism, C and Fortran compiler of RAW is effective at exploiting parallelism by automatically partitioning the program graph, placing the operations, and programming the routes for the static router. The speedup is attained from 6× to 11× versus a single tile on Specfp applications for a 16-tile Raw processor and 9× to 19× for 32 tiles.

Applications with a minimal (two- or three-way) amount of instruction-level parallelism generally do not benefit much from running on Raw. This is because inter-tile latency is great enough that it is cheaper to compute locally than to distribute the computation to a neighbor tile.

3) **TRIPS**

Subsequently, in 2007, the TRIPS (**T**era-op, **R**eliable, **I**ntelligently adaptive **P**rocessing **S**ystem) prototype chip contains two data networks: the OCN (**O**n-**C**hip **N**etwork) and the OPN (**OP**erand Network). The OCN attaches two processor cores to a single second-level cache, various IP blocks, and I/O units. Their primary NoC requirement was low latency, implying a single-cycle-per-hop router. Another requirement was network-addressing flexibility to use shared cache in multiple modes. So OCN network is dedicated to the communication of memory traffic. Another OPN network transmits operands. Each OPN message consists of a control and data phit, where a phit is the amount of data processed by the OPN router in a cycle. The OPN supports different physical wires for the control and data phit. Each OPN message consists of one flit, split into a control phit and a data phit.

The OPN router has five inputs and five outputs. Each input has two four-entry-deep FIFO buffers. One for control phits and one for data phits. The local input has no FIFO buffer that eliminates a cycle from the insertion of packets into the network. The OCN packet's control and data phits have separate 5×5 crossbars.

This system consisting of two types of NoCs delivers high bandwidth and low latency to achieve system performance goals.

In both the OPN and OCN routers, the input FIFO buffers dominate the routers, consuming 75 percent of the router area. The OCN routers are more than twice the size of the OPN routers because of the extra buffering required for virtual channels. Therefore, the networks' overall area and power consumption were higher because of the wide input FIFO buffers required by the wide channel widths.

4) **Tilera**

In 2007, Tilera [35, 70] manufactured a Tile processor architecture with multi-NoC named as iMesh on-chip interconnect. It connects the multicore processor tiles with five 2D mesh networks, each specialized for different use. These networks isolate communication among different sub-systems. Memory traffic is separated from user-specified on-chip traffic. **U**ser **D**ynamic **N**etwork (UDN) and **ST**atic **N**etwork (STN) are used for operands and messages, **I/O** **D**ynamic **N**etwork (IDN) is used for IO and system traffic, **M**emory **D**ynamic **N**etwork (MDN) and **T**ile **D**ynamic **N**etwork (TDN) are used for the memory and cache coherence for attaining high on-chip communication bandwidth.

5) **Adapteva Epiphany**

In 2011, Adapteva [2] Epiphany [71], [50] launched a chip with three independent NoC mesh networks that carry different traffics such as rMesh for on-chip read, cMesh for on-chip write, xMesh for off-chip write.

This was claimed to be the world's first low-power chip design.

A typically 64 cores Adapteva chip consumes 70 $GFLOPS/W$ power, 8.1 mm^2 area and 88 $GFLOPS/S$ performance.

[2]Recently, Epiphany-V chip is arrived in the market, and Adapteva has joined the kilo-core club with 1024 chips. Also, announces 'The Brain' project with 5,00,000 core on a chip at 3nm.

2.4.2 Academically Proposed Multiple Networks-on-Chip

Different proposals to design multiple networks-on-chip by researchers across the globe are summarized in Table 2.3. The common factor in all such proposals is of creating multiple networks for an efficient flow of traffic. Researchers have evaluated their work using different performance metrics. Table 2.4 presents proposed multiple NoC architecture along with performance metric it improves. The work is classified as per the major gain in evaluation metric as multiple objectives are achieved with some designs. Different works achieve specific NoC performance metric in the targeted application domain by employing different customization as discussed in the following paragraphs.

TABLE 2.3: Multiple Networks-on-Chip strategies proposed academically in different works.

Classification	**References**
Express cube topologies	B. Grot et al. [115]
Multiple Subnetworks[a]	R. Das et al. [Catnap [46], 2013], J. Wu et al. [Chameleon [95], 2015]; Lian S et al. [BoDNoC [139], 2017]; J. S. Miguel et al. [8], 2015; V. Akhlaghi et al. [110], 2015
Concentration and Channel Slicing	P. Kumar et al. [112], 2009
Customized multiple NoCs Architecture	T. C. Xu et al. [108]
Multimode on-chip interconnect	H. Bokhari et al. [Supernet [117], 2015]
Multiplanes Networks-on-Chip[a]	Noh et al. [119]; A. K. Abousamra et al. [106], 2012; J. Wu et al. [93], 2017
Multi-Switch Approach	F. Gilabert et al. [39], 2010
Hybrid two layer router architecture	J. Wu et al. [124], 2016
Physical channel replication	Carara et al. [118], 2007
Runahead NoC as an accelerator with baseline NoC	Z. Li et al. [109], 2016
Hybrid buffered and bufferless NoC[b]	J. Fang et al. [98], 2018
Multi-Clock On-Chip Networks	P. Ezhumalai et al. [121], 2010
DarkNoC Layers	H. Bokhari et al. [123], 2014
Direction-Sliced Subnetworks	F. Wang et al. [125], 2015
Hybrid multiple Networks-on-Chip[a]	J. Wu et al. [92], 2017
Parallel Sub-links	C. Gomez et al. [127], 2008

[a]The work is also reported in the year 2017.
[b]The work proposed in the 2018.

1) **Multiple Subnetworks**

R. Das et al. [46] propose Catnap architecture, which consists of synergistic subnet selection and power-gating policies. Unlike a single-network, a multiple-network design is more amenable to power gating as its subnetworks (subnets) can be power gated without compromising the connectivity of the network. Catnap maximizes the number of consecutive idle cycles in a router while avoiding performance loss due to overloading a subnet.

They evaluate a 256-core processor with a concentrated mesh topology. The average network power of a power-gating optimized multiple-network design with four subnets could be 44% lower than a bandwidth equivalent single-network design for an average performance cost of about 5%.

The limitations of Catnap design and its suitability vary for different processor configurations. They argued that multi-NoC design is attractive for large processors with high network bandwidth requirements. As the number of cores reduces, the network bandwidth requirement reduces, and fewer subnets are necessary to satisfy the processor's per-core bandwidth. The opportunities for power gating is proportional to the number of subnets. Benefits of these policies are lower for smaller processors. They use concentrated topology because concentration is a simple and effective way to reduce network latency and power. These policies are effective for a 64 cores processor, benefits of a multi-NoC design with power gating are higher as the number of cores increases.

TABLE 2.4: Gain in result metrics with different multiple NoC architectures

Parameters	**Configuration**
Power[a]	R. Das et al. [46], J. Wu et al. [Chameleon [95], 2015], M. Buckler et al. [116]; H. Lu et al. [90], 2015; H. Lu et al. [91], 2018
Energy	V. Akhlaghi [110], P. Kumar et al. [112]; A. K. Abousamra et al [106], 2012; T. C. Xu et al. [108], J. S. Miguel et al [8], 2015
Execution Time	J. Lee et al. [111]
Latency	J. Lee et al. [111], V. Akhlaghi [110], T. C. Xu et al. [108]
Area	J. Lee et al. [111], P. Kumar et al. [112]
Bandwidth	P. Ezhumalai et al. [121], 2010

[a]The work of 2018 target the gain in power.

J. Wu et al. [95] propose Chameleon, a novel heterogeneous Multi-NoC design. Chameleon employs a fine-grained power gating algorithm which exploits power saving opportunities at different levels of granularity simultaneously. Thus, they propose a performance-aware traffic allocation policy for multiple NoCs. This proposal delivers an average of 3.39% higher performance than Catnap, the best contemporary alternative. It consumes an average of 17.16% less power than Catnap [46]. However, the coarse-granularity power-gating is inefficient for energy-proportional design.

Lian S et al. [139] propose a multi-granularity memory system that provides multiple access granularity for the applications with various spatial localities. In the multi-granularity access pattern, the one-size-bandwidth NoC design cannot utilize the bandwidth efficiently. They propose a novel NoC design, called BoDNoC, which can merge multiple narrow subnets to provide various bandwidths. Experimental results show that BoDNoC can improve the throughput by 23.5% and reduce energy consumption by 37.2% in comparison with one-size-bandwidth NoC design. However, the results are evaluated only with synthetic traffic. The approach is not validated with real traffic.

J. S. Miguel et al. [8] model a theoretical, ideal **N**on**C**ritical **N**etworks-**o**n-**C**hip (NoC-NoC) where every word is fetched such that its fetch latency is equal to its access latency. Caches fetch data in bulk (blocks of multiple words). Depending on the application's memory access patterns, some words are needed right away (critical) while other data are fetched too soon (non-critical). So NoCNoC addresses this issue and ensures that all the data shall be delivered just at the right time, no earlier than needed. In NoCNoC, the baseline NoC is divided into multiple subnetworks wherein each subnetwork is operating at a different frequency and voltage. The subnetworks split the baseline 128-bit channel width. Each subnetwork is assigned to fetch a subset of words that share a common access latency. The subnetwork's frequency is then configured such that the fetch latency is equal to the access latency. A brute-force search is performed with all possible subnetwork configurations to find the one that yields the lowest energy. They employ **D**ynamic **V**oltage-**F**requency **S**caling (DVFS) to slow down the non-critical subnetwork. This allows the system to save energy on words predicted to be non-critical since they can tolerate a higher fetch latency.

The criticality-aware NoCNoC achieves up to 60.5% energy savings with no loss in performance. Furthermore, 62.3% of energy is wasted in fetching data that is not used by the application.

Authors also perform a limit study of the impact of data criticality in NoC design and limit study to estimate the amount of energy wasted in using traditional, criticality-oblivious NoC designs. They have used six subnetworks in the design of multiple NoC. A large number of subnetworks increase the control logic overhead.

J. Wu et al. [93] propose Multi-NoC (Multiple Network-on-Chip) that behaves well in power gating for reducing leakage power, which constitutes a significant fraction of NoC power. They propose CRA a novel Multi-NoC design with distinct routing algorithms for different subnets. Integrated with a congestion-aware power gating and packet scheduling policy. The CRA can achieve low power without degrading performance at varying network utilization.

The experimental results with proposed multi-NoC show that CRA consumes an average of 15.58% less power than Catnap, the state of the art power efficient Multi-NoC design, and the EDP (Energy-Delay Product) is 8.59% lower than Catnap on average.

A proposal by V. Akhlaghi et al. [110] consists of two network layers where one layer is dedicated to the packets transmitted to near destinations, and the other layer is used for the packets transmitted to far destinations. The actual physical channel width connecting the cores is divided between the two layers. The locality is defined based on the number of hops between the nodes. The relative significance of the two types of communications determines the optimum ratio for the channel width division. Their result showed that for the link width of 128 bits and local communications of more than 30%, the proposed network architecture outperformed the conventional one, on average, by 56%, 60%, and 70%, in terms of the average network latency, energy consumption, and EDP, respectively.

2) **Channel Slicing**

Channel slicing is used together with concentration by P. Kumar et al. [112]. Concentration and slicing are well-known techniques in interconnection networks that have been adopted in different off-chip networks such as the Cray X1 [113] and the Cray Black-Widow network [114]. To achieve trade-off between complexity and performance, hybrid implementations of concentration and channel slicing can be used to reduce the cost of NoC while maintaining requisite performance. NoC with the concentrated network has wider channels that may result in poor channel utilization because of some small packets in the on-chip network. For example, while cache lines can exceed 512–1024 bits in width, request packets, control packets, or coherency messages are much narrower.

P. Kumar et al. [112] show how the poor channel utilization introduced by concentration can be mitigated by channel slicing in NoC. Because concentration shares the channels between multiple nodes that lead to performance degradation. The use of channel slicing increases utilization of the wiring resources and improve performance. They propose virtual concentration wherein concentration and channel slicing are combined to reduce the cost of the network further. Their proposed virtual concentration save area and energy by 69% and 32% compared to baseline mesh and 88% and 35% over baseline concentrated mesh.

J. Lee et al. [111] propose a novel router micro-architecture, i.e., Sharded Router that employs fine-grained bandwidth "sharding" (i.e., partitioning) and stealing in order to mitigate the elevation in the zero- load latency caused by slicing in four physically separated networks. Consequently, the zero-load latency of the Sharded Router becomes identical to that of a conventional router, whereas its throughput is markedly improved by fully utilizing all available bandwidth. Detailed experiments using a full-system simulation framework indicate that the proposed router reduces the average network latency by up to 19% and the execution time of real multi-threaded workloads by up to 43%. Finally, hardware synthesis analysis verifies the modest area overhead of the Sharded Router over a conventional design.

3) **Multimode On-Chip Interconnect**

H. Bokhari et al. [117] illustrate how dark silicon can be exploited to realize a multilayered NoC where each layer is optimized for a particular voltage-frequency (V–f) level using the multi-Vt optimizations. This architecture is named as SuperNet. When the chip is switched on, **F**abric **M**anager (FM) by default initializes the SuperNet in the normal mode, i.e., one of NoCs powered with a pre-defined operating value for voltage-frequency tuple. Depending on various factors such as power budget, reliability requirement, etc., SuperNet can be activated in one of the available modes; each mode reflects operating at a certain V–f pair.

4) **Multiplanes Networks-on-Chip**

Few works on multiplane NoCs compare the possible alternatives of VC-less multiplanes, a single plane with more virtual channels and multiplanes with less virtual channels.

Noh et al. [119] propose a multiple-plane router with modified switch allocator, which can provide better latency and performance in NoC environment. By adding more crossbar switches in router architecture, the complexity of switch allocator is increased. Increase in complexity is offset by better performance. They also compare the single plane router with more number of VCs and double-plane router. The experiment results indicate the increase of a number of planes is better than an increase in the number of virtual channels in one port from the performance and complexity point of views.

Y. J. Yoon et al. [57] compared multiplane architectures with a fixed wire and buffer constraint, and showed multiplanes are more efficient than VCs when the input buffer storage is limited; besides, they provided higher per-watt efficiency for regular traffic patterns.

A. K. Abousamra et al. [106] propose deja vu switching for multiplane NoC wherein control and data planes are separate. They present the argument that instead of having one interconnect plane to serve all traffic, power can be saved if the NoC splits into two planes. A fast plane through circuit switching dedicated to the critical messages and a slower by reducing the voltage and frequency, more power-efficient plane assigned only to the non-critical messages. This split can be beneficial to save energy only if system performance is not significantly degraded by the slower plane. They explore the need for timely delivery of the non-critical messages. They analyze the constraints that govern how slow the power-efficient plane can operate without negatively impacting system performance. They evaluate design through simulations of 16 and 64 core CMPs. They achieve an average NoC energy savings of 43% and 53%, respectively.

5) **Multi-Switch Approach**

F. Gilabert et al. [39] explored different architectures for deterministic wormhole routing:

a traditional VC design, the multi-switch approach in which a VC-less switch is replicated as many times as virtual channels, and the multi-network approach which corresponds to the multiplane design with both link width and buffer constraints. They argue that the multi-switch approach provides better performance than an equivalent multi-network with only a small area overhead.

6) **Physical channel replication**

   Carara et al. [118] discuss trade-offs on using circuit and packet switching, arguing in favor of the former with a fixed packet size. Next, it proposes and justifies the replacement of virtual channels by replicated channels, based on the abundance of wires expected in current and future deep sub-micron technologies.

7) **Runahead NoC as an accelerator with baseline NoC**

   In Runahead NoC configuration [109], the author has used a single Baseline64 network, which carries 100% of the injected packets, along with the proposed Runahead network that carries latency-sensitive packets (i.e., single-flit packets and critical words). As delivery is not guaranteed in the Runahead network, duplicate injection of latency-sensitive packets into both networks is required. The total channel width, in this case, is 18 bytes (8 bytes for the Regular network and 10 bytes for the Runahead network).

   Two extra bytes to the Runahead network channel width are accounted for any additional metadata for supporting critical word forwarding. This does not give Runahead network any performance advantage since all packets are single-flit; in fact, it incurs a power and area disadvantage.

8) **Hybrid buffered and bufferless NoC**

   J. Fang et al. [98] proposed a hybrid NoC with a dedicated bufferless NoC and a buffered NoC, as well as an application-aware mechanism to choose optimally efficient NoC. They proposed a new metric NC (Network-Computation) ratio to classify big data load in latency-sensitive and non-latency-sensitive. NC ratio refers to the ratio of the average length of the network stages to the average length of the computing stages during the measurement period. It reflects the network feature of different applications in several stages as the characteristics of the application vary at different stages. They use NC ratio=2 as the cut-off point to classify the request into two types. They examined both the 64-thread and 128-thread system, and their proposed mechanism shows significant improvements in system performance.

However, heterogeneous NoCs lack flexibility, as there is no fairness in resource distribution. This may cause hotspots of traffic and may result in congestion. Also, routers may generate heat when more traffic activity exists in the networks. As a consequence, several issues like more number of faults, thermal issues, and reliability affect the system. Therefore, heterogeneous NoC is sometimes not a suitable architectural choice.

9) **Hybrid multiple Networks-on-Chip**

J. Wu et al. [124] propose a hybrid Multi-NoC design called HM-Mesh that adopts a hybrid CMesh and Mesh architecture and leverages CMesh network for power efficiency at low network utilization. HM-Mesh is able to adaptively schedule packets to different subnets according to the network load, and smartly perform power gating to achieve good energy efficiency. Each computational node is connected to a corresponding router in the Mesh subnet. A central controller is connected to the control network. The controller in the node keeps track of the buffer occupancy of the corresponding router in the CMesh subnet, and if congestion is detected, it signals the central controller, which will broadcast a signal to all controllers to turn on the Mesh subnet. When the central controller detects there is no packet in the Mesh subnet for a number of consecutive cycles, it broadcasts a signal to all controllers to turn off the Mesh subnet. The power gating process is periodically performed by the central controller.

The experimental results show that HM-Mesh delivers an average of 4.87% higher performance than Catnap, the state of the art power efficient Multi-NoC design. More importantly, HM-Mesh consumes an average of 29.2% less power than that of Catnap [46].

J. Wu et al. [92] argue that the core count grows rapidly, therefore, NoC consumes an increasing fraction of the modern processors/SoCs power. It is thus very important to design energy-efficient NoC architecture. They propose Chameleon, a novel heterogeneous Multi-NoC design. Chameleon employs a fine-grained power gating algorithm which exploits power saving opportunities at different granularity levels simultaneously. Integrated with a congestion-aware traffic allocation policy, Chameleon can achieve both high performance and low power at varying network utilization.

10) **Multi-Clock On-Chip Networks**

P. Ezhumalai et al. [121] design and implement an NoC framework for FPGAs, Multi-Clock On-Chip Network for Reconfigurable Systems (MoCReS). They propose a novel microarchitecture for a hybrid two-layer router that supports both packet-switched communications, across its local and directional ports as well as time multiplexed circuit-switched communications among the multiple IP cores directly connected to it.

They have placed and route VHDL models of the advanced router architecture show an average improvement of 20.4% in NoC bandwidth (maximum of 24% compared to a traditional NoC). Authors develop a library of network components that can parameterize the hybrid router model using a different number of ports, channel width, and depth of buffers.

11) **DarkNoC Layers**

H. Bokhari et al. [123] propose l number of network layers, where a network layer consists of m routers. At a given time, only one of the network layers is illuminated (activated), while the rest of the network layers remain dark (deactivated). When a network layer is illuminated, all of its routers are active, and thus, at a given time, only m routers are active. Each network layer is optimized at design-time to operate in a certain Voltage-Frequency (VF) range. That is, multi-Vt circuit optimization of CAD tools is used to optimize all the routers of a network layer for a particular VF range. All the layers in a region are managed by a hardware-based darkNoC Layer Manager (dLM). The function of the dLM is to switch between network layers when directed by the system-level DVFS manager.

However, physically different routers, leveraging the extra transistors available due to dark silicon.

12) **Parallel and Partitioned Networks**

B. Grot et al. [115] propose the MECS topology describes their topology using S=2 sliced topology as they evaluate MECSx2. They propose the Multidrop Express Channels (MECS) topology and discuss an implementation based on two parallel and partitioned networks (MECS-X2).

T. C. Xu et al. [108] propose a heterogeneous network that is designed to separate the message classes. A unicast/multicast capable router processes multicast invalidation messages and other unicast/multicast control messages, whereas another unicast only router processes unicast/multicast data messages.

Their proposed multiple NoC design is compared with the homogeneous baseline network as well as two other network designs. Experimental results show that the average network latency and energy-delay product of the proposed design have improved 24.4% and 10.2% compared with the baseline network.

13) **Direction-Sliced Subnetworks**

F. Wang et al. [125] propose the direction-slicing scheme into the 2D torus and mesh network, respectively. For the direction-slicing in the 2D-torus network, all X+ channels (West In and East Out), Y– channels (South In and North Out), and Local channels in each router are split into ever-on slices and form the ever on subnet to guarantee full connectivity. While remaining X– channels (East In and West Out) and Y+ channels (North In and South Out) are the gated slices. Contrary, the 2D torus shows the different direction-slicing, considering both symmetrical topology and simple routing design. All X+ channels in even rows, X– channels in odd rows, Y– channels in even columns, Y+ channels in odd columns, and local channels are utilized to constitute the ever-on subnet, and other remaining channels which belong to gated slices can be power-gated to pursue the leakage power-saving.

14) **Parallel Sub-links**

C. Gomez et al. [127] analyze the option of distributing the wires among several parallel links connecting the same two switches. This technique is known as Space Division Multiplexing (SDM). The number of parallel sub-links and their width are two key parameters that are studied together with the relationship with the mean packet size. They propose that SDM is a technique to take into account in on-chip networks. It allows to highly increase the network accepted traffic at the expense of a small latency increase or even no increase. Moreover, in some networks, it allows reducing the network hardware, providing similar performance results, which results in a reduction in the consumption of area and power. They divide the wires into several parallel links connecting to the same two routers to improve the network throughput while improving area occupation and power dissipation.

B. Sethuraman et al. [128] propose a competitive NoC architecture in FPGAs. The area occupied by the network should be kept to a minimum. This ensures that the maximum area can be utilized by the logic while maintaining the performance of the router network. Reducing area also reduces power consumption. They propose implementation of a parallel router which can support five simultaneous routing requests at the same time with an area overhead of only 352 Xilinx Virtex-II Pro FPGA slices (2.57% of XC2VP30).

2.5 Application Areas of Multiple NoCs

The research on multiple networks-on-chip has targeted various research domains, as listed in Table 2.5.

TABLE 2.5: Application Areas of Multiple NoCs

Parameters	**Configuration**
Congestion	Y. J. Yoon et al. [57], 2013
Fault Tolerance	Y. J. Yoon et al. [57], 2013
Network Reconfiguration[a]	H. Lu et al. [90], 2015; H. Lu et al. [91], 2018
Security	J. Sepúlveda et al. [47], 2015
Dark Silicon[b]	S. Hesham et al. [94], 2017; M. Shafique et al. [122], 2017
Power Gating[b]	R. Das et al. [46], 2013; J. Wu et al. [93], 2017
Big Data[a]	J. Fang et al. [98], 2018
Accelerator	Z. Li et al. [109], 2016

[a]The targeted application area of multiple NoC in the year 2018.
[b]The multiple NoC target the problem in the year 2017.

2.5.1 Congestion

The congestion control is one of the most challenging issues when designing a high-throughput low-latency NoC. The incoming traffic often exceeds the outgoing bandwidth resulting in queuing delay, packets loss, and blocking of new connections. The proposed solutions suggest to increase the size of the buffers, but this causes an area overhead. Limiting buffers size degrades the NoC performance in terms of latency and throughput. Y. J. Yoon et al. [57] propose the utilization of multiple NoCs for path adaptivity, i.e., the ability of a router to dynamically adapt the path of a packet according to the current status of the network. When the router detects that an output port is congested, it dynamically changes the routing policy so that following packets can avoid the blocked channel. Path adaptivity relies on the extensive use of virtual channels because they can be used to implement refined solutions for deadlock avoidance or recovery.

2.5.2 Fault Tolerance

The tolerance to temporary and permanent faults in the NoC components is an issue of growing importance for complex SoCs. The virtual channel renaming is a technique to support fault tolerance across different virtual channels [132]. With VC renaming, the number of VCs recognized by the routers and NIs can be larger than the number of physical queues used to implement them. This solution is particularly effective when VCs are used to partition incompatible traffic patterns for deadlock avoidance, but it is limited to handle faults that occur in VC components. Instead, Y. J. Yoon et al. [57] propose that multiple NoCs offer additional fault-tolerance options: e.g., the network interfaces can avoid routing packets to a plane with faulty routers and/or links by choosing a different plane. Since all planes are isolated from one another, multiple NoCs can tolerate faults caused by links, crossbars, and switch allocators.

2.5.3 Network Reconfiguration

The temporal and spatial heterogeneity of on-chip traffic cannot adapt to existing NoC power consumption as per its traffic intensity and hence lead to suboptimal power efficiency. Existing approaches either resort to over-provisioned NoC design for the spatial distribution of traffic or coarse-grained power gating that only serves traffic temporal variation. H. Lu et al. [90, 91] propose ShuttleNoC for boosting on-chip communication efficiency by enabling localized power adaptation. ShuttleNoC architecture permits packets shuttling between multiple subnetworks to achieve localized power adaptation.

It could achieve optimal power efficiency with up to 23.5% power savings and 22.3% performance boost in comparison with traditional heterogeneity-agnostic NoC designs.

2.5.4 Security

Ensuring security in Multi-processors Systems-on-Chip (MPSoCs) leads to several design challenges due to their intrinsic complexity. Distributed and collaborative MPSoC computing forces the application to span across the computing resources. As a result, sensitive information is exchanged among the different computation components through the Networks-on-Chip (NoCs). Security zones can be built for wrapping the sensitive IPs. However, these zones may not always be physically close. J. Sepúlveda et al. [47] propose an architecture for creating NoC-based disrupted security zones. They propose a reconfigurable security architecture for disrupted protection zone in NoC-based MPSoCs. One network (Data) transmits the data among the IPs of the MPSoC, while other (Service) uses for security services (Different flit widths and buffer depth). A group-wise key agreement technique at NoC level is employed to create secure and dynamic communication channels among the hardware components. They efficiently manage dynamic security domains while presenting a low impact on the performance and cost of MPSoCs.

2.5.5 Dark Silicon

S. Hesham et al. [94] explores the use of Circuit-Switched (CS) NoCs as a low complexity energy-efficient solution for future platforms in the Dark-Silicon (Si) era. They present a thorough analysis for circuit-switched NoCs from hardware synthesis perspective at different threshold voltage (V_{th}) and supply levels. The proposed NoC is implemented as a mixed V_{th} double layered design, where each layer is optimized for a frequency level at a different supply voltage. Layer switching is explored on a per-router-port granularity level; links are either activated on the fast or the slow layer based on the speed requirements, while the other layer is kept dark. The proposed NoC architecture show energy savings up to 34% compared to conventional single-layer single supply CS-NoCs.

M. Shafique et al. [122] propose that dark silicon can be exploited to realize a multilayered NoC where each layer is optimized for a particular voltage-frequency (V–f) level using the multi-Vt optimizations. Depending upon the network load, an appropriate layer is selected, while other NoC layers are kept dark. To support efficient runtime operation, a lightweight NoC layer switching and flushing strategy and corresponding architectural support are proposed in these works.

2.5.6 Power Gating

As the development of processors/SoCs, NoC consumes an increasing fraction of the modern processors/SoCs power. Thus, designing energy-efficient NoC architecture is imperative. Multi-NoC (Multiple Network-on-Chip) behaves well in power gating for reducing leakage power, which constitutes a significant fraction of NoC power. J. Wu et al. [93] propose Combined Routing Algorithms (CRA), a novel multi-NoC design with distinct routing algorithms for different subnets. Integrated with a congestion-aware power gating and packet scheduling policy, CRA is able to achieve low power without degrading performance at varying network utilization. The experimental results show that CRA consumes an average of 15.58% less power than Catnap, the state of the art power efficient multi-NoC design, and the EDP[3] is 8.59% lower than Catnap on average. However, power gating schemes introduce significant wake-up delay because of the loss of network connectivity.

2.5.7 Big Data

Big data demands application-aware network-on-chip framework. IoT and cloud computing requires exascale computing systems with high performance and low power consumption to process massive amounts of data. J. Fang et al. [98] propose a hybrid NoC framework, combining buffered and bufferless NoCs to make the NoC framework aware of applications' performance demands. An optimized congestion control scheme is also devised to satisfy the requirement in energy efficiency and the fairness of big data applications. They use trace-driven simulation to model big data applications. Compared with the classical buffered NoC, the proposed hybrid NoC can significantly improve the performance of mixed applications by 17% on average and 24% at the most, decrease the power consumption by 38% and improve the fairness by 13.3%. But the heterogeneous NoCs have lack of flexibility.[4]

[3]Energy-Delay Product

[4]As explained in Subsection 2.4.2, architecture 8) hybrid buffered and bufferless NoC (Page 38).

2.5.8 Accelerator

With increasing core counts and higher memory demands from applications, it is imperative that networks-on-chip (NoCs) provide low-latency, power-efficient communication. Conventional NoCs tend to be over-provisioned for worst-case bandwidth demands leading to ineffective use of network resources and significant power inefficiency; average channel utilization is typically less than 5% in real-world applications. In terms of performance, low-latency techniques often introduce power and area overheads and incur significant complexity in the router microarchitecture. Z. Li et al. [109] propose that both low latency and power efficiency are possible by relaxing the constraint of lossless communication. This is inspired by internetworking where best effort delivery is commonplace. They propose the Runahead NoC, a lightweight, lossy network that provides single-cycle hops. Allowing for lossy delivery enables an extremely simple bufferless router microarchitecture that performs routing and arbitration within the same cycle as link traversal. The Runahead NoC operates either as a power-saver that is integrated into an existing conventional NoC to improve power efficiency or as an accelerator that is added on top to provide ultra-low latency communication for selected packets. Runahead NoC reduces power consumption by 1.81x as a power-saver and improves runtime and packet latency by 1.08x and 1.66x as an accelerator. However, bufferless router designs have low throughput.

2.6 Remarks on Multiple-NoC Architecture Customisations

Multiple interconnects, multiple-NoC architectures, and their targeted application domain have been discussed in Section 2.3 to Section 2.5. The proposed implementations are customised to achieve the desired result metric of a specific application domain. The variety of customisation shows the flexibility of these multiple NoC/interconnect architectures. It indicates these architectures are full of scope for different customisations to solve modern and near future NoC research problems.

These architectures can be customised for

1) Different processor designs such as for CMPs, SoCs, MPSoCs, FPGA, etc.

2) Homogeneous as well as heterogeneous architectures.

3) Exploiting spatial and temporal locality of traffic and parallelism through multiple NoC.
4) Any NoC result metric such as power, area, bandwidth, latency, execution time, and energy improvement through customisations.

In our work, we employ the customization for homogeneous multi-NoC networks for general purpose processors as the homogeneous networks are easily scaled to the same die area [74].

2.7 Traffic Distribution in Multiple Networks-on-Chip

Networks-on-chip is a communication substrate between the cores. NoC not only serves the request between source to destination cores but also maintain the consistency between shared caches and offchip memory. Therefore on-chip traffic is directly decided by spatial and temporal locality of data. These factors affect the delay in addressing a cache miss. On a cache miss, core searches for the data in other shared caches, and in offchip memory if a search is unsuccessful. Single-NoC has single physical channels and cannot support the distribution of messages across multiple networks as in multiple NoC. As the traffic has multiple message classes as per the cache coherence protocol, it is interesting to explore how messages distribution across multiple NoCs affect network efficiency. Reported work in this employ either static or dynamic distribution strategies. A brief overview is as follows:

2.7.1 Static Traffic distribution

In previous work [8, 36, 57, 74] of multi-NoCs, the traffic is statically distributed on multiple NoC networks as follows.

- Control messages are assigned to one NoC network, and the other NoC network is for data messages [36].
- Uniform distribution of messages on both NoC networks [74].
- One to one mapping between message classes and physical channels can be used to assign messages on multi-NoCs [8, 57].

In other static traffic distribution approaches for multiple NoCs, A. K. Mishra et al. [79] propose a heterogeneous dual NoC network wherein one NoC network is designed for bandwidth-sensitive applications and another for latency-sensitive applications. However, these designs do not allow the different classes' packets through the same network, which makes a waste of bandwidth and throughput.

M. Buckler et al. [116] propose power reduction by differentiating among different kinds of traffic (such as one-to-many, many-to-one or request/response) and optimizing for each type.

None of the multiple NoCs architecture has explored static traffic distribution in detail, especially where the workload is a mix of computation intensive and memory intensive applications and its impact on power-performance efficiency of these networks.

2.7.2 Adaptive Single-NoC Traffic Distribution and Other System Level Approaches

In single-NoC, the dynamic traffic distribution was limited to VC exploration. Z. Fang et al. [58] and C. A. Nicopoulos et al. [64] propose the runtime distribution of messages. They not only prioritize VCs for the defined class of critical messages but also dynamically allocate VCs, and buffer resources according to network traffic conditions.

Likewise, the VC controller and VC allocation modules are proposed by M. Lai et al. [65] that is modified to introduce congestion aware dynamically-allocated VCs architecture. The statically allocated VC structure lacks flexibility on various traffic conditions that cause low buffer utilisation. If the router is configured with deep VCs, head-of-line blocking will lead to low throughput. Inversely, shallow VCs have distributed packets over a large number of routers. In a low traffic rate, the packet transfer is interrupted by many contentions and increase the latency. Therefore, M. Lai et al. proposed the use of shared buffers among VCs whose structure varies with traffic condition. In low traffic rate, VC depth is extended to reduce packet latency while in high traffic rate, the number of VCs increases to avoid congestion situations to improve throughput. In the proposed approach, all the buffers are shared by linked lists, and each VC is associated with a list for the single packet from different sizes according to traffic conditions.

Other than single-NoC dynamic traffic distribution, few research works are done on dynamically adaptive schemes to exploit parallel execution for the processor and adaptive cache-line granularity for memory architectures

Initially, processor and memory architectures have used different runtime adaptive methods at the system level. During the execution of a processor, it can switch the execution from in-order to out-of-order or vice-versa, and it can also vary the pipeline depth throughout program phases to better support different levels of **I**nstruction-**L**evel **P**arallelism (ILP) or **T**ask-**L**evel **P**arallelism (TLP). The examples of these architectures are MorphCore, CoreFusion, Composable Lightweight Processors, and ARM big.LITTLE architecture [18]. The idea to dynamically adapt resources according to program phase behaviour has been exploited for several microarchitectural units, such as the issue queue, reorder buffer, caches, and branch predictors that improve the flow in instruction pipeline [141]. These techniques rely on the fact that conventional microprocessors are designed to maximize performance for a wide range of applications. However, a program rarely fully utilizes every microarchitectural resource to achieve high performance and reducing the sizes of those underutilized microarchitectural resources may turn into a penalty of energy reduction and performance impacts.

Subsequently, caches have different dynamically adaptive proposed schemes. Caches can dynamically select properties between inclusive and exclusive for the **L**ast-**L**evel **C**ache (LLC) depending on application characteristics. The cache line granularity can be adjusted at runtime according to the spatial locality in the workload, or partition the LLC adaptively between cores depending on how beneficial additional cache size would be for a particular core. All of these techniques take advantage of runtime variations in application behavior and adapt the microarchitecture to extract better performance or power efficiency.

Runtime variation in working set sizes, spatial locality, and ILP and TLP also change communication characteristics and application's memory access and data sharing behavior. Similar effects of phase behavior and underutilization of hardware resources are seen in NoC traffic with the advent of manycore architectures, although these behaviors have largely gone unexplored prior to our work. Traditional NoCs with rigid structures are unable to accommodate such changes properly. Changing minor architecture will lead to radical changes in the system's communication demands. Though, it can be handled through runtime adaptive NoCs.

We have explored adaptive traffic distribution for multiple NoCs architecture as these architectures facilitate multiple physical networks for traffic distribution. The implementation challenges of adaptive approaches are different for multiple NoCs as compared to processor, memory, and single-NoC.

2.8 Architecture of Network Selection Hardware Unit and Its Placement

None of the works reviewed here have explored the architecture of the network selection unit and the impact of its placement on power-performance efficiency of the multiple NoC networks. Few works have discussed memory controller placement, thermal-aware IP core placement, and energy-aware task mapping.

D. Abts et al. [86] were the first who explored the impact of placement of **M**emory **C**ontrollers (MCs). In a given system, a number of cores are more abundant than that of MCs. They investigate the on-chip design space to find the optimal number of the memory controller and their placement relative to mesh and torus topologies, different routing algorithms, and workloads. The placement of MCs can reduce contention, hot spots, and lower the variance in latency for memory-intensive applications.

Likewise, W. Hung et al. [87] employ a genetic algorithm to minimize thermal hot spots through optimized IP placement. While application-specific researchers J. Hu et al. [88] and K. Srinivasan et al. [89] find the optimal mapping of tasks to cores, and the optimal static routes between these cores to achieve bandwidth and latency requirements. The communication patterns are known a priori and can be specifically targeted based on communication graphs and also use optimal algorithms for energy minimization.

2.9 Design Space of Multiple NoC Networks

Different proposals on multiple NoCs discussed under Section 2.4 have used a different number of multiple NoC networks. The choices are based on different objective and application areas of their work. Other than these works, A. Ejaz et al. [43] have explored multiple NoC architecture itself by comparing the power efficiency of quad-network-NoC with dual-network-NoC by doing synthesis.

Other than the power overheads, an increasing number of NoC networks add up new control tasks to the critical path that raises network delay and hence leads to a higher critical path delay [39, 57]. Therefore, dual NoCs are suggested by A. Ejaz et al. [43] over quad NoCs. Architecture of dual-network-NoC is more power efficient over both types of quad channel NoCs as compared in Table 2.6.

TABLE 2.6: Synthesis results for different design choices of multi-NoCs (These results are based on A. Ejaz et al. [43] for a single router)

NoC→	**Single NoC**	**Dual NoC**		**Quad NoC**	
Size→ / **Met.**[a]**↓**		Dual-network	Bi-planar	Quad-network	Quad-planar
	$1 \times 1 \times 128$[b]	**$1 \times 2 \times 64$**	**$2 \times 1 \times 64$**	**$1 \times 4 \times 32$**	**$4 \times 1 \times 32$**
Power (mW)	30.1	32.6	56.1	36.12	62.2
Clock (GHz)	2.1	1.52	3.2	1	2.1

[a]Met- Metric

[b] The first parameter indicates a number of planes in multiple NoC. The middle parameter is used to specify the number of NoC networks and the last parameter specify the bandwidth of each NoC network.

These results were obtained on default wire load model. Each type of multiple NoC is corresponding to architectures, as demonstrated in Fig 2.9. The size of NoC specifies three dimensions of the network i.e., the number of planes, the number of NoC networks, and bandwidth of each NoC network, respectively.

Dual-network-NoC ($1 \times 2 \times 64$) has approximately same power efficiency as of single-NoC. Though, it is the most power efficient as compared to bi-planar-NoC ($2 \times 1 \times 64$) and quad NoC networks. These power results are obtained by Ejaz et al. when the total number of physical resources (physical links, link-width, buffers) are kept constant across these NoC architectures. Though, the power efficiency of dual NoCs decreases with an increase in the number of networks, as observed in Table 2.6. This is because of the increase in hardware resources (NI including wires in case of dual-network) and routers + NI in case of bi-planar networks shall increase power consumption. Therefore, Our selection of dual-network-NoC as a baseline multiple NoC architecture is implied through this study.

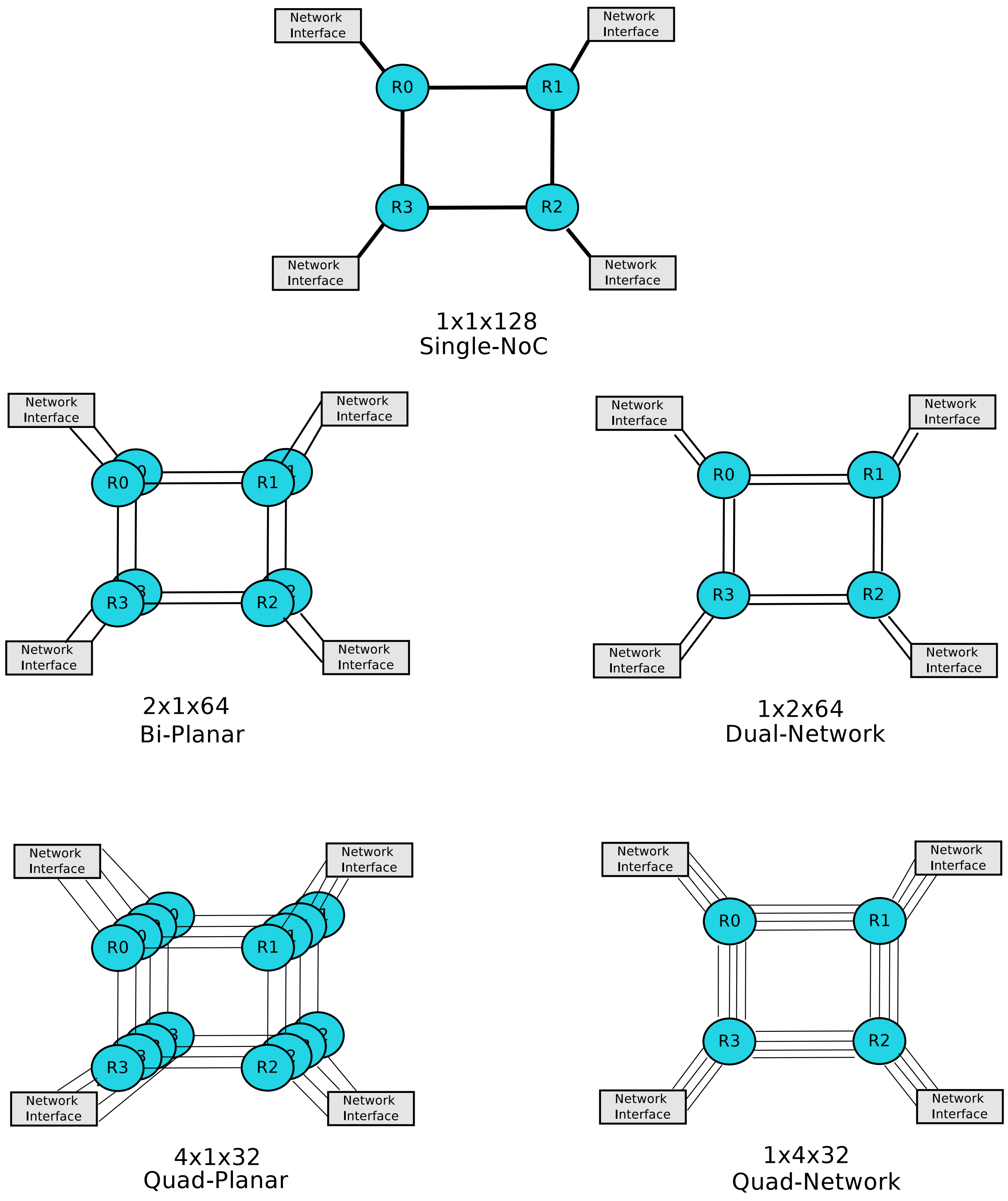


FIGURE 2.9: Different types of multiple NoC networks up to quad networks as compared in Table 2.6, according to A. Ejaz et al. [43].

2.10 Concluding Remarks

In this chapter, we have discussed a literature review on commercially and academically implemented multi-NoCs architectures. In all the discussed architecture, the multi-NoC itself power is comparable with single-NoC architecture. All the advantage in result metric is achieved by applying different implementation techniques through customizing each network for specific tasks. Existing approaches in multiple NoCs improve network efficiency (power or execution time or both) by

1) using different network for specific tasks.
2) designing different networks with different power gating and using one network at a time.
3) using VF scaling for different networks.

Except for a couple of researchers, none of the work enlights multi-NoC architecture itself. None of the work has explored possible customizations in multi-NoC architectures to make architecture itself power and energy efficient. None of the work has explored the architecture and placement of network selection hardware unit that is the primary and essential integration of hardware unit in multi-NoC.

The traditional static methods of traffic distribution also remain unexplored experimentally. None of the work to the best of our knowledge reported the impact of static distribution on multi-NoC result metric. Hypothetically, the messages are distributed as

1) separate networks dedicated to control/request and data/response.
2) a number of networks dedicated to each message class.
3) uniform distribution of messages between the networks.

None of the work has explored dynamically adaptive traffic distribution wherein a traffic class can change the network on underutilization of the network except the recent proposal

1) H. Lu et al. [91], 2018 propose a reconfiguration hardware unit that is introduced between the physical links of the multiple NoCs. At runtime, if the router faces congestion, then another network is powered on, and a portion of traffic is redirected through another network.

Though, such proposal has the additional hardware cost of reconfiguration unit that causes additional power penalty. Whereas, in our proposal, we keep the runtime adaptive traffic distribution mechanism simple to avoid any additional cost.

We also made our choice to study up to two networks with dual-network-NoC architecture that is employed by following observations/arguments:

1) Dual-network-NoC is power efficient as compared to bi-planar-NoC and quad NoCs [43].
2) With up to dual NoC, no additional area overhead as there is enough space between tiles of the chip to accommodate two networks while the die area keeps constant [74].
3) With dual NoC, control logic overhead is lower than the more number of NoC networks that can be considered as negligible.
4) Dual-network-NoC has more flexibility to explore adaptivity as traffic can easily change the network with these architectures as compared to bi-planar-NoC.

Thus, dual-network-NoC uses dual physical networks on the same chip and has the flexibility to efficiently separate traffic classes of cache coherence protocol to improve the performance [36, 38, 43, 57].

In the next chapter, we shall do traffic analysis on dual-network-NoC and compare it with respect to single-NoC. Our experimental methodology and tools set up shall be introduced in the next chapter. In short, all the discussion on dual-network-NoC traffic analysis and experimental methodology lay the foundations for the work presented in rest of the chapters.

Chapter 3

Preliminary Studies

The main objective of this chapter is to identify parameters for NoC simulation framework for evaluation of our proposals in later chapters. Towards this end, we investigate communication traffic patterns across NoC for different types of applications working under different workload conditions. This study forms the basis of our work on message distribution in later Chapter 4, Chapter 5, and Chapter 6. There we show that NoC performance can be improved by dividing messages across a dual-network-NoC by reserving a given NoC network for a subset of message classes. For our simulation, we use Gem5 with Garnet NoC interconnect.

3.1 Introduction

In the previous chapter, we have discussed various multiple interconnects and NoC architectures. We have chosen traditional multiple-NoC architectures [38, 43, 57, 131] for further exploration. As we have discussed in Chapter 1 (Page 9) dual-network-NoC has more traffic distribution flexibility, and also it has a power-efficient hardware implementation as compared in (refer to Page 50, Chapter 2) with dual-plane and designs with more number of NoC networks. In this thesis, we limit exploration to dual networks. In this chapter, we present experimental methodology and explore NoC traffic distribution between single-NoC and dual-network-NoC. NoC traffic is one important factor that significantly impacts the power consumption. In following paragraphs, we shall be using term power efficiency to denote less power consumption. System A is said to be more power efficient than system B if A's power consumption is less than that of B.

The selection of NoC traffic for performance analysis is extremely important as it leads to inferences about applicability of the methodology of the architecture. Traditional synthetic traffic patterns with their regular stochastic models fail to capture the dynamics of real, multi-threaded CMP workloads in terms of variations in source/destination patterns and injection rate fluctuations and are therefore ineffective in predicting real-world NoC performance [18]. To ensure more realistic NoC behaviour, most recent NoC-related research uses real application traffic from benchmark suites, such as the PARSEC (**P**rinceton **A**pplication **R**epository for **S**hared-m**E**mory **C**omputers) [44]. PARSEC has a diverse mix of applications making this benchmark suitable for devising and evaluating message distribution schemes for more efficient communication protocols.

We compare the traffic distribution of dual NoCs with single-NoC. Through experiments with real traffic, we analyze the impact of benchmark parameter and hardware configuration on performance and power efficiency of the network. Such analysis is utilized for experimental setup and simulation parameter selection for our work presented in subsequent chapters. Thus, in this chapter, we have covered all the important aspects of traffic distribution for dual-network-NoC and their experiments that lay the foundation for the next chapters.

In summary, our key contributions are:

- Setup for our experimental methodology with various tools integration and simulation mechanism with real traffic.
- Study of NoC performance with real traffic and its input workloads.
- Comparison of dual-network-NoC performance vis-a-vis traffic with respect to single-NoC using different benchmark parameters and NoC hardware configurations to select/-fix/prepare our experimental framework/parameters. So that the proposed architectures in next chapter use these foundations/basis for their incremental customize contributions.
- Categorizing the messages of cache coherence protocol into different classes and study the impact of the distribution of these classes through NoC on its performance. This lays the foundation of message distribution strategies presented in Chapter 4 and 6.

This preliminary study helps us identify NoC architectural parameter to be used in subsequent chapters.

3.2 Experimental Methodology

For these preliminary studies, we have used Gem5 [41] that is a most extensively used open source simulator. It facilitates the execution of benchmarks for evaluating the efficiency of manycore processors. Gem5 simulator is an integration of M5 and GEMS simulators [41]. M5 supports CPU models, **I**nstruction **S**et **A**rchitecture (ISAs), input/output devices, infrastructure whereas GEMS supports interconnect models including cache coherence protocols. Gem5 is suited for evaluating the performance of manycore processor architectures. For our thesis, we employ other tools with Gem5. This is depicted in Fig 3.1.

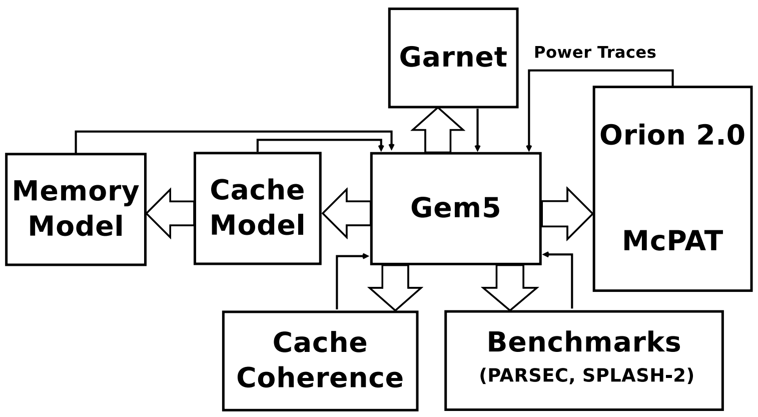


FIGURE 3.1: Schematic representation of our simulation framework

Networks-on-Chip. Garnet is a detailed interconnection network model that supports bus, crossbar, point-to-point interconnects to mesh and torus networks-on-chip topologies. Mesh is popular among all topologies due to its simplicity and scalability up to hundreds of cores. Garnet simulates a detailed router micro-architecture model, as discussed in Appendix C (Page 185) and requisite components of on-chip networks. Hardware parameters of these components[1] can be configured with different values in Garnet during the simulation. For the implementation of dual-network-NoC, we needed to make significant changes in the Garnet code. These changes were required at mesh topology, NI, and link slicing as well as router micro-architecture code.

Cache and Memory. For cache and memory models, Gem5 includes a flexible Ruby infrastructure and a domain specific language, SLICC is used for specifying cache coherence protocols. Using Ruby, a developer can expressively define cache hierarchies and coherence protocols, including those expected in emerging heterogeneous processors.

[1]Virtual channels, buffer size, link-width, etc., we have discussed these components in Subsection 3.5.2.3 (Chapter 3, Page 74), Subsection 3.5.2.2 (Chapter 3, Page 73), and Section 4.3 (Chapter 4, Page 83).

Power. Gem5 is integrated with ORION 2.0 simulator for NoC power and area estimation. McPAT (Multicore Power, Area, and Timing) is also an integrated power, area, and timing modeling framework for multithreaded, multicore, and manycore architectures. It models power, area, and timing simultaneously and consistently and supports comprehensive early stage design space exploration for multicore and manycore processor.

Benchmarks. We have integrated Gem5 with PARSEC benchmark for real-time analysis of our proposals. The full-system simulation runs the parallel section of benchmarks, and it is our most important performance metric. The simulation consists of three phases in chronological order.

- Warm up phase: Initially, all caches are empty or filled with non-relevant data. Any application at the onset of execution shall generate lots of cache miss. This phase is not used for analysis of the performance of communication infrastructure or cache coherence protocol [57] as this is a transient state.
- **R**egion **o**f **I**nterest (RoI): This phase is relatively steady-state, and the cache miss is likely to be an outcome of techniques to implement coherence for given communication infrastructure.
- Cleanup phase: This phase is encountered when an application has almost run or completed. This is used by operating system for garbage collection.

All results and discussion pertain to statistics generated over RoI phase for only the parallelized versions of each workload. The thesis thus uses full-system simulations instead of trace-driven ones. However, limitations of the Gem5 [41] simulator restrict us from running simulations for more than 64-cores.

3.2.1 Processor and Memory

In Gem5, the processor hardware configuration parameters are listed in Table 3.1. For networks-on-chip, gem5 is integrated with a detailed cycle-accurate GARNET [40] model. It provides support for modeling of packet-switched NoC with pipelined routers following either wormhole [73] or virtual channel flow controls (Appendix C). Gem5 also facilitates execution of benchmarks in **F**ull **S**ystem (FS) that allows real device drivers and operating systems to be

Table 3.1: Processor Simulation Parameters

Parameters	**Configuration**
Cores	16/64 cores @ 2 GHz, ALPHA-21264 ISA[a], Out-of-Order (OoO)[b], six pipeline stages: Fetch-Decode-Rename-Issue-Execute-Writeback, Issue 4 instructions/cycle, reorder up to 80 instructions on the fly
Networks-on-Chip	GARNET[c]
Power	ORION[d] 2.0, 32nm Technology, 1 Volt
Benchmark	PARSEC[e] (medium workload, 16 threads)

[a]ISA-Instruction Set Architecture.

[b]Out-of-Order (OoO)- execution allows instructions-level parallelism and improves processor performance. Out-of-order execution makes sure that some other instruction which is independent of the missed one is executed so that the CPU does not waste clock cycles by living in idle state.

[c]GARNET-A cycle accuRate iNtErconnection neTwork simulator.

[d]ORION-A pOwer-perfoRmance IntercOnnection Network simulator.

[e]PARSEC-Princeton Application Repository for Shared-mEmory Computers.

run, not just user-level programs. It models bare hardware and privileged instructions. Applications can execute both user-level and kernel-level instructions. To experiment with Gem5 in full system simulation, a Linux 2.6.27 kernel image is booted with ALPHA instruction set architecture in detailed out–of–order CPU mode.

Table 3.2: Memory parameters for on-chip cache and off-chip memory

Parameters	**Configuration**
Caches	Split into L_0 (instruction) and L_0 (data), each 64KB per core, private, 4-way set associative, 64-B block size, 3-cycle latency, L_1 Caches: 2MB, shared, 8-way set associative, 64-byte block size, 20-cycle bank latency, 20 MSHRs[a] L_2 Caches, 16MB, shared, 16-way set associative, 64-byte block size, 32-cycle bank latency, 32 MSHRs, cache line size 64-B
Cache Coherence	multicast (Directory) miss is solved in three hops
Main Memory	ddr3_1600_x64, 8 memory controllers, data transfer rate 12.8 GB/S, 80 cycles access latency, block size 64 B

[a]MSHR-Miss Status Holding Register.

The cache and memory parameters[2] in Gem5 are listed in Table 3.2. For good support of caches, three levels[3] of cache hierarchy (Appendix A) are used with MESI cache coherence protocols

[2]They are used with ruby memory model.

[3]Hierarchy of cache levels plays a major role in faster memory access compared to direct main memory access [21]. Intel's Nehalem, Haswell-E, Core i7 processor and AMD's K10 (Barcelona) are the examples of processor that have used three-levels of cache hierarchy [22].

(Appendix B). The three levels of cache hierarchy are used wherein L_0 and L_1 are private, and L_2 is shared and distributed between the cores. L_0 cache is split into data and instruction caches. The size of L_0 (L_1) cache is 64KB (2MB) with 2-way (8-way) set associative and L_2 cache is 16KB with a 16-way set associative. The hit latencies for the caches are 3, 20 and 32 cycles respectively. The Gem5 uses 512 MB size of memory with ruby memory model that has 80 cycles of hit latency. The Gem5 is integrated with ORION 2.0 [61] at 32nm technology to estimate the power consumption of NoC.

3.2.2 Networks-on-Chip

Networks-on-Chip are formed with links and routers. We assume that flit traverse physical link in a single cycle. Our router model uses the four-stage pipeline for running medium to high workloads. The details of NoC hardware configuration for the router, physical links, flit width, routing, and network topology are listed in Table 3.3 for single-NoC and Dual-network-NoC. Different stages traversed in every single cycle of the router are:

- **B**uffer **W**rite (BW) and **R**oute **C**omputation (RC)
- **V**irtual **C**hannel **A**llocation (VA)
- **S**witch **A**llocation (SA)
- **S**witch **T**raversal (ST)

Our choice of employing a four-stage pipelined router is governed by the following observations.

- Two-stage and three-stage pipelined router designs are suited only for low traffic loads. Under high workloads, speculation cost offsets any advantages gained by the reduction in pipeline length and may lead to larger critical path delay [63].
- Likewise, a single-cycle router is suitable only when (a) input buffer is not full, (b) flits to be transmitted next are available in the buffer, (c) and no conflict for input and output ports for existing flits. Practically, such ideal conditions are difficult to achieve.

Thus, four stage pipelined router model is used for our work to avoid any performance penalty due to the pipeline structure with medium and high workload benchmarks.

3.3 NoC Traffic Analysis

We have analyzed NoC traffic through full system simulation of benchmarks for a more accurate evaluation of the NoC system as compared to trace based and synthetic traffic based simulation. Full system simulation for all applications in the benchmark suite takes a long time. In full system simulation, the hardware of computer system is simulated at the level of the details such that the complete software stacks from real systems can run on the simulator. We have chosen **P**rinceton **A**pplication **R**epository for **S**hared m**E**mory **C**omputers (PARSEC) [55] benchmark as it is suitable and most commonly [136] used for studies of chip multiprocessors as compared to other similar benchmarks such as SPEC [138] and SPLASH-2 [44].

PARSEC is better suited among all the benchmarks for the evaluation of manycore processors since it consists of applications requiring small, medium, and high workloads. PARSEC repository continues to be maintained and updated by the developers for sustaining compatibility with emerging workloads. It includes emerging applications in the fields of pattern recognition, data mining, and system applications that mimic large multi-threaded programs commonly used in the industry (e.g., Intel) as listed in Table 3.5.

The benchmark suite covers a wide range of parallelization models, synchronization primitives, working set sizes, data locality, communication-to-computation ratios, and off-chip traffic. The

TABLE 3.3: NoC Configuration in Garnet

Parameters	**Configuration**
Network router	four-stage pipelined, virtual channel flow control, three VN[a] for multicast MESI[b] directory protocol, four VCs[c] per port for single-NoC, two VCs per port for dual-network-NoC, five flits buffer depth, 8×8 and 16×16 crossbar for single-NoC and dual-network-NoC respectively
NoC Networks	Single-NoC/Dual-Network-NoC (single-plane)
Physical link/flit width	16-B for single-NoC and 8-B for dual-network-NoC, link latency 1 cycle, for single-NoC control flit is 8-B and data flit is 16-B, for dual-network-NoC both control and data flits are 8-B
Routing	Dimension Order XY
Network Topology	8x8 mesh, each node has a router and core

[a]VN-Virtual Network.
[b]MESI-**M**odified-**E**xclusive-**S**hared-**I**nvalidation.
[c]VCs-Virtual Channels.

benchmark also provides analysis of the traffic in the context of cache memory, on-chip communication, and off-chip memory of manycore processor. Therefore, it is popularly used benchmark for modern processors actual performance evaluation. Its input data are large enough with a high degree of concurrency/multithreading suitable for evaluating the performance of manycore processor. It focuses on emerging workloads since more powerful processors will be available in the near future.

As the number of processor cores increases, applications, too, grow in terms of the size of working datasets, degree of concurrency, inter and intra-thread communication to accommodate user demands. To ensure efficient resource utilization of manycore processors, efficient traffic distribution is needed to utilize NoC network effectively and to make the execution of cores faster. In manycore architectures, communication is the bottleneck, and improving its performance shall improve processing efficiency also. The traffic distribution can be improved through an efficient classification of NoC traffic.

TABLE 3.4: Different Characteristics of Applications

Characteristics[a,b]	**Type1**	**Type2**	**Type3**	**Type4**
Data Sharing (DS)	Low (Lo)	High (H)		
Exchange (E)	Low (Lo)	Medium (M)	High (H)	
Granularity (G)	Coarse (C)	Medium (M)	Fine (F)	
Data Pattern (DP)	Parallel (P)	Pipelined (PL)	Unstructured (Un)	
Working Set (WS)	Small (S)	Medium (M)	Large (L)	Unbounded (U)

[a]The listed characteristics define the traffic patterns of PARSEC applications.
[b]These characteristics are used in Table 3.5.

Analysis of application behavior is essential to identify the requisite size, quantity, and multiplicity of network resources like buffers, virtual channels, and physical links. We have run the PARSEC benchmark using full system simulation for experiments throughout the thesis as it represents a wide range of traffic patterns for manycore processor. State of the art application's data exhibits variability in respect to granularity, sharing, size of working data sets, as shown in Table 3.4. The different characteristics [137] of application are as follows:

1) Data Sharing: It defines all synchronization primitives used inside the application of benchmark suite like *locks, barriers*, and *wait conditions*. The total count of these primitives categorizes application under low or high data sharing.

TABLE 3.5: The PARSEC Benchmarks of various application domains with working data set and their detailed functionality

Application	**Domain**	**Working Data Set**	**Characteristics**[a] **(DS**[b]**/E**[c]**/G**[d]**/DP**[e]**)**	**Functionality**
Blackscholes	Financial Analysis	in_16.txt (1KB), in_64k.txt (4.1MB)	Lo/Lo/C/P	Intel RMS benchmark that calculates prices for portfolio with a Blackscholes Partial Differential Equation
Bodytrack	Computer Vision	SequenceB_1 (2.5MB),SequenceB_3 (9.9MB)	H/M/M/P	Tracks a 3D pose of a human body through multiple cameras that includes video surveillance & character animation
Canneal	Engineering	100.nets (2.2KB), 400000.nets(14.3MB)	H/H/F/Un	Uses cache aware Swapped Atomically (SA) to minimize the routing cost of a chip design
Dedup	Enterprise Storage	medias.dat (10.6MB), mediam.dat (32.2MB)	H/H/M/PL	Deduplication is a method to generate backup storage system similar to real–world
Ferret	Similarity Search	queriess (65.9KB), corelm(18.4MB) & corels(4.6MB)	H/H/M/PL	Content based image similarity search of feature rich data set, i.e., audio, image, video, 3D shapes
Fluidanimate	Animation	in_15k.fluid (545kB), in_300k.fluid (11MB)	Lo/M/F/P	Smoothed Particle Hydrodynamics (SPH) method to simulate an incompressible fluid
Freqmine	Data Mining	T10I4D100k_1k.dat (40.3KB), kosarak500k.dat (16.2MB)	H/M/M/P	Frequent Pattern (FP) growth method that uses different data mining techniques
Streamcluster	Data Mining	none	Lo/M/M/P	A large amount of data is organized under real conditions, i.e., intrusion detection system
Swaptions	Financial Analysis	none	Lo/Lo/C/P	Employs Monte Carlo (MC) simulation to compute portfolio prices to asset interest variation for risk management
Vips	Media Processing	barbados_256 × 288.v (295.5KB), bigben_2662 × 5500.v (295.5KB)	Lo/M/C/P	VASARAI Image Processing Systems (VIPS) uses transformation & convolution
X264	Media Processing	eledream_64 × 36_3.y4m (10.4KB), eledream_640 × 360_3.y4m (11.1KB)	H/H/C/PL	The image of the Elephant Dream movie is used to create the input videos for the X264.
Rtview	Computer Graphics	bunny.obj (2.5 MB), happy_buddha.obj (42.4 MB)	H/M/M/P	Models 3D scans of real physical objects. It is used for collision & visibility detection.

[a]Refer Table 3.4 for characteristics details.
[b]DS-Data Sharing
[c]E-Exchange
[d]G-Granularity
[e]DP-Data Pattern

2) Exchange: The communication volume of traffic between threads can be *low, medium,* and *high*. According to working data set, efficient data exchange for some PARSEC applications through shared caches is constrained by cache capacity.

3) Granularity: The granularity of input data can be defined as *coarse, medium,* and *fine*. In *fine-grained* data, *coarse-grained* or *medium-grained* data chunk up into *fine-grained* segments. For example, at the *coarse-grain*, the traffic can be divided into `on-chip cache` and `NoC`, and `offchip memory` traffic types. Whereas in *medium-grain*[4], these types are further divided into subclasses such as `control, request control, response control, writeback control, response data,` and `writeback data`. Finally, *fine-grain*[5] traffic is the cache messages such as `load, store, acknowledgements,` etc., *fine-grain* messages are present at the last level of granularity as no further segmentation is possible. Likewise, PARSEC benchmark suite uses three possible granularity of input data during computation according to the type of job/function performed by different applications.

4) Data Pattern: PARSEC application data pattern can be categorized under *parallel, pipelined,* and *unstructured*. The parallelism is dependent on the proportions of the parallelizable application code. Different parallelization models are used in the PARSEC benchmark suite. The amount of parallelism is also dependent on the working set size and the cache block size used for an application. Whereas, *pipelined* data patterns use a complex heterogeneous parallelization model wherein specialized threads execute different functions with different characteristics at the same time. Applications which employ a pipeline have dedicated thread pools for each parallelized pipeline stage. *Unstructured* data patterns do not have fixed types of runtime behavior. For some cycles, these patterns show sequential (or serialized) behavior, for another few cycles, they may exhibit parallelized or pipelined behavior.

5) Working Set: It defines working data set of benchmark application. PARSEC put it under the category of *testing (test, simdev), simsmall, simmedium, simlarge,* and *unbounded (native)* input data. For these working sets, the volume of data increases in ascending order. The full system simulation time also increases with the volume of data (from one day to 20 days for small to large working data set).

PARSEC applications fall into the category of streaming, general purpose, mobile, server, scientific, engineering, and graphical applications as listed in Table 3.5. Different characteristics

[4]Medium granularity of traffic is discussed in Section 3.6 (Page 77).

[5]Fine-grain granularity of traffic shall be discussed in Chapter 6 (Page 136).

of applications are one of the cause for different volumes and types of traffic generated inside the network [137].

Recent manycore processors are using shared distributed caches to keep a coherent and consistent view of memory to all cores. Details on communication of distributed shared caches for manycore processor architecture can be found in Appendix-B. Cache coherence protocol coordinates spatially distributed caches and ensures up-to-date multiple cached copies of data. Distributed memory hierarchy consists of a memory bank, three levels of cache hierarchy to reduce the overall access time to write (read) data to (from) processor cores from (to) cache/memory. Execution of any application is limited by the time to access data from memory. On-chip cache structure reduces this access time.

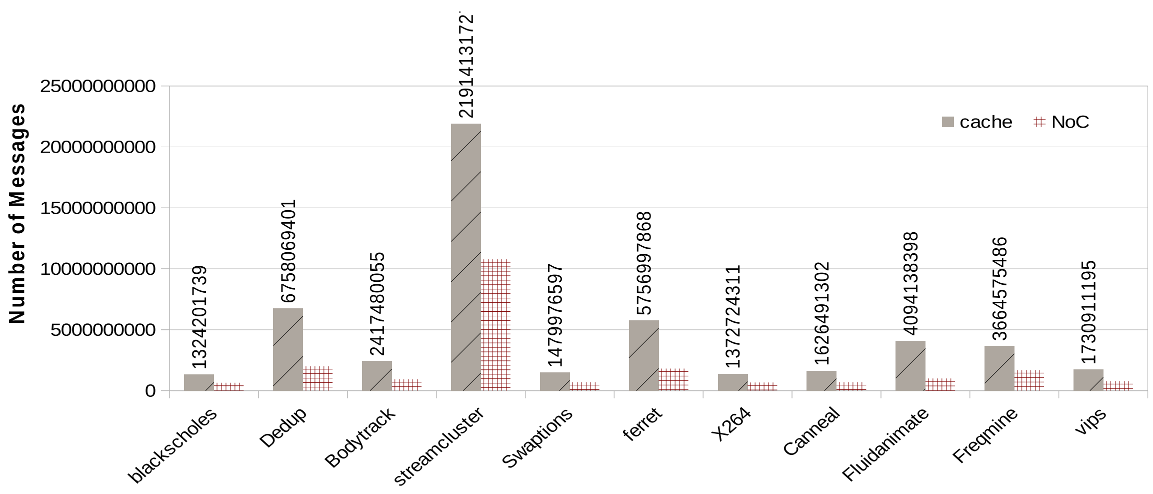


Figure 3.2: Cache and NoC traffic (64 processor cores traffic with PARSEC benchmark).

A scalable and efficient NoC is required to handle the volume of traffic generated to/fro cache(s) during updating shared caches, in a cache coherence protocol. The number of transmitted flits through NoC is dependent on cache coherence traffic propagated in the network. Fig 3.2 and Fig 3.3 show the NoC traffic, cache traffic and memory traffic for different applications of PARSEC (refer for details on application to Table 3.5).

Fig 3.2 and Fig 3.3 depict overall communication traffic for cache and memory. It needs to be seen if it is possible to parallelize part of this traffic for improving performance. NoCs support

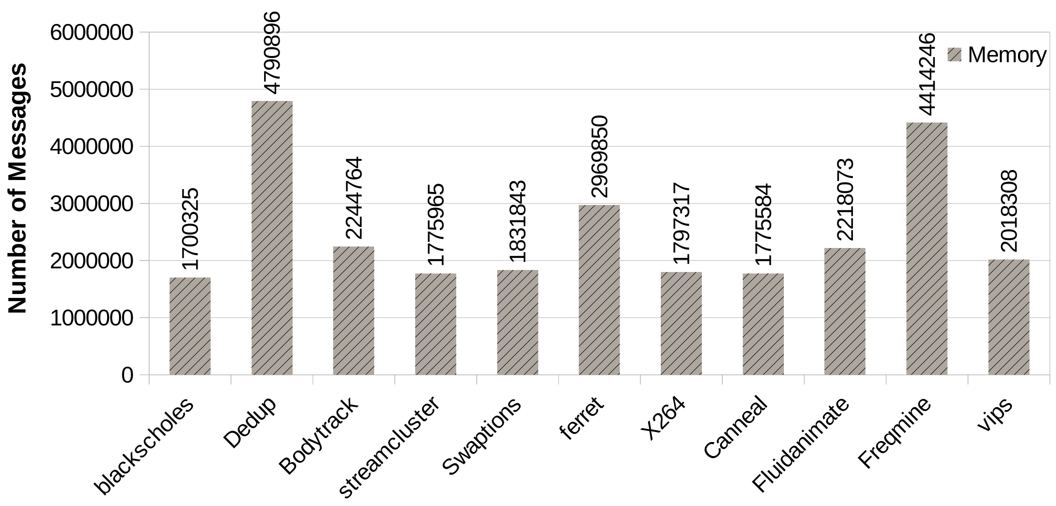


FIGURE 3.3: Memory traffic (64 processor cores traffic with PARSEC benchmark).

delivery of messages in parallel but identifying parallelism requires to investigate what messages are non-overlapping in a given cache coherence protocol. This traffic classification is important to quantify NoC traffic as the system performance can be improved by faster communication among the cores. Impact of traffic distribution on the performance of computation bound, communication bound, and memory bound applications is also explored. PARSEC benchmark consists of all three categories-computation bound, communication bound and memory bound of applications.

3.4 NoC Deadlock and Head-of-Line(HoL) Blocking

Modern NoC architecture uses buffers, virtual channels, and virtual networks to avoid traffic deadlock and head-of-line blocking problem as discussed in the following subsections:

3.4.1 Buffers

A significant evolution primarily starts with the introduction of FIFO queue (buffers) in single-NoC router microarchitecture as shown in Fig 3.4(a). The buffers are used to temporarily store incoming flits before these can be forwarded to destinations. Traffic can be broadly categorized into request and response messages. Sometimes, single buffer architecture is congested when the network is flooded with request messages.

Cores which initiate request messages wait for responses. Both types of messages use the same network resources. Responses may not make progress in case of congestion and may result in a deadlock. Since requesting cores are waiting for responses, and response cannot be sent/forwarded due to unavailability of resources (buffer space) at the destined core(s). For

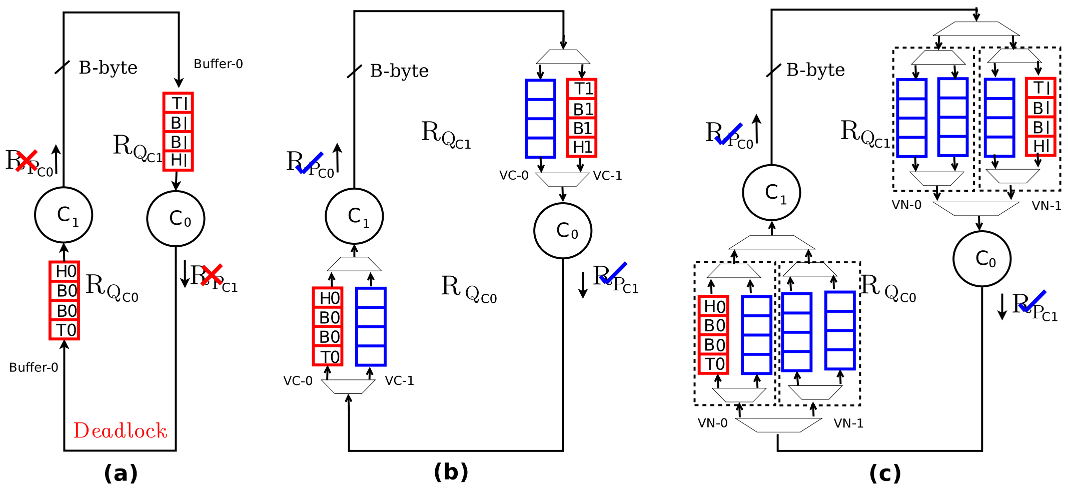


FIGURE 3.4: NoC hardware provisions to avoid traffic deadlock and head-of-line blocking (a) single-NoC single-buffer (b) single-NoC with virtual channels (c) single-NoC with virtual networks. Where H, B, T represent Head, Body, and Tail flits respectively. The single buffer and single VC terms are used interchangeably.

example, in Fig 3.4 (a), the core C_1 is waiting for the reply ($R_{P_{C1}}$) from core C_0 which cannot send the reply ($R_{P_{C1}}$) as the buffer of core C_1 is full with request messages ($R_{Q_{C0}}$). Similarly, core C_0 is waiting for reply ($R_{P_{C0}}$). Thus, deadlock may occur if a resource dependency exists between different classes of messages. One possible solution to deal with deadlock is the addition of virtual channel, discussed in the next subsection, as shown in Fig 3.4 (b).

3.4.2 Virtual Channels

Buffer associated with each physical link is partitioned into several small queues/buffer storage to form virtual channels rather than a single deep queue. The virtual channels associated with each physical link are allocated independently but compete with each other for the physical bandwidth. They decouple buffer resources from transmission resources. This decoupling allows active messages to pass blocked messages by using a different VC. This improves the utilization of network bandwidth that would otherwise be left idle. Virtual channels are introduced in NoC for resolving **H**ead-of-**L**ine (HOL) blocking problem [73]. For example, the dual buffers (virtual channels) as in Fig 3.4(b) can resolve the head-of-line blocking by facilitating more than one virtual channels for inter-core communication. Thus, they can increase network throughput by utilizing idle buffer space.

3.4.3 Virtual Networks

Virtual Networks (VN) are introduced in NoC to avoid deadlock between messages because of the message dependency in cache coherence protocol [73]. They also reduce the queuing delay of coherence messages by separating control (request) and data (response) traffic. All the message classes use the same single physical network although, the request and response traffic classes are separated at VN level and transmitted via different virtual networks to avoid the protocol level deadlock. Performance due to logical separation is, however, limited by a single physical channel between routers. The request and response messages are also subdivided between the different class of messages with the advancement of cache coherence protocols. The class of request message can be mapped to VN-0, and the class of response message can be assigned to VN-1 as shown in Fig 3.4 (c). Thus, virtual networks help to avoid deadlock due to message dependency. The traffic distribution with multiple virtual networks is significantly limited in a single-NoC network as physical links are not replicated.

3.5 Traffic Analysis: Dual-Network vs Single NoC

The limitation of the single physical NoC as discussed in previous section can be resolved with dual-network-NoC that comprises of two physical NoC. These architectures can help in deadlock prevention as well as do efficient traffic distribution.

Manycore general-purpose processors employ variety of workloads including pipelined multi-threaded workloads as discussed in previous Section 3.3. These workloads have a large number of concurrent transactions for distributed shared caches. This results in increase of the communication traffic volume across the network. Both single-NoC and dual-network-NoC rely on **V**irtual **N**etworks (VNs) for parallel transmission of different class/categories of cache coherence messages. These networks logically separate message classes to avoid deadlock in coherence protocols. Buffers are placed in these networks to hold the flits of a packet over a channel. Instead of a single buffer, multiple buffers are used corresponding to each channel which are known as **V**irtual **C**hannels (VCs). Though VCs increase critical path delay,

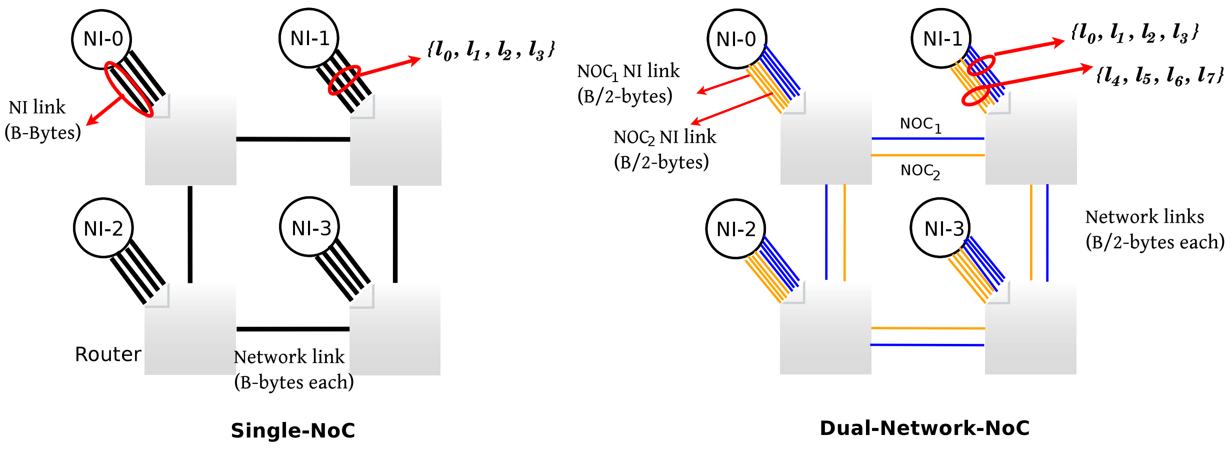


FIGURE 3.5: There are four NI links (three level of cache hierarchy and one for directory controller) between core and router. In dual-network NoC, these links are replicated. Also, router-router links are replicated. These are shown as red and blue respectively.

power dissipation and area, they simplify network operation and address HoL (Head-of-Line) blocking problem.

Dual-network-NoC not only partitions the physical link width but also splits the respective number of VCs between dual network though they keep the same number of virtual networks. Dual networks speed up the communication of network traffic. Rather than initiating few parallel transmission with complex VC allocation logic, dual-network-NoC favor the dual networks with simple, independent and parallel data flows. The physical link width of single-NoC is divided by two as there are two networks in dual-network-NoC. Total physical link-width remains same for both single-NoC and dual-network-NoC. This ensures a fair comparison of the two architectures.

3.5.1 Benchmark Parameters and Hardware Configuration Impact on NoC Traffic

We analyze throughput variations between single-NoC and 2-network NoC for all the applications of PARSEC benchmark in full system simulations. In following subsections, we present how throughput varies with workloads, number of threads, number of cores, NoC buffer size, NoC virtual channels, and instruction set architectures to analyze the impact of these parameters on NoC traffic. Throughput is computed as the number of bytes received per unit of time, picoseconds, i.e., 10^{-12}s in our case. In all our results presented in this chapter, both single-NoC and dual-network-NoC consists of 16 and 64 processors.

3.5.1.1 Workloads Choices

PARSEC allows an application to be run with three different types of workload – Low, Medium, and High. To see the impact of workload variation, we have experimented with two extreme cases of **L**ow (L) and **H**igh (H) workload conditions. For most applications save for

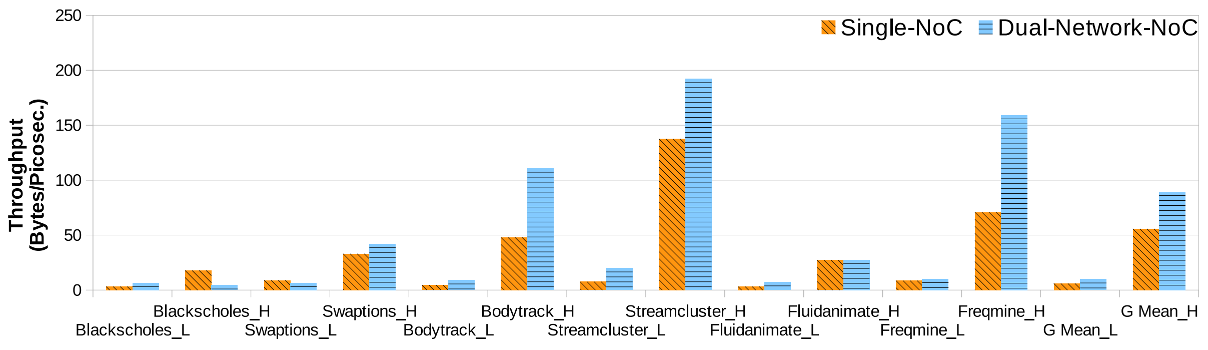


FIGURE 3.6: Throughput comparison for Low (L) and High (H) workloads with PARSEC benchmark. Results are evaluated with 16 core processor.

computation-bound ones, increase in workload is likely to increase traffic to/from caches. From Fig 3.6, we observe that dual-network-NoC exhibits an overall 1.6x more throughput as compared to single-NoC for high workloads. Applications like *bodytrack*, *streamcluster*, and *freqmine* have improved throughput since they have more number of read and write operations. We observe that the number of messages that pass through NoC has increased in high workload

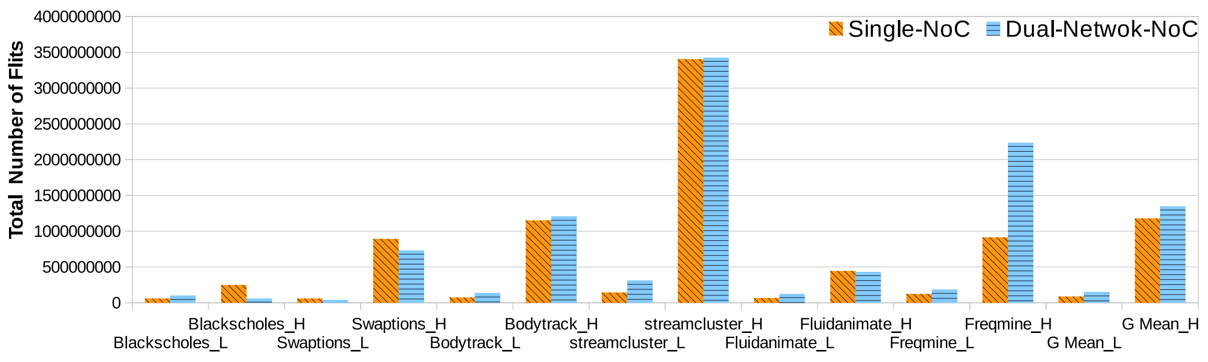


FIGURE 3.7: Number of flits variation from Low (L) to High (H) workloads for different PARSEC benchmarks. Results are evaluated with 16 core processor.

scenario as shown in Fig 3.7, and throughput is also increased as compared to low workload. For low workloads, performance of dual-network-NoC is comparable with that of single-NoC.

This is expected as message density of cache coherence protocols is not high enough to require parallel transfer.

3.5.1.2 Number of Threads

As we increase the number of threads from 8 to 64, we observe that throughput is improved for both single-NoC and dual-network-NoC for most of the applications, i.e., *blackscholes*, *swaptions*, *x264*, *dedup*, *fluidanimate*, and *ferret*. In few cases for 64 threads, dual-network-NoC does not perform better than single-NoC.

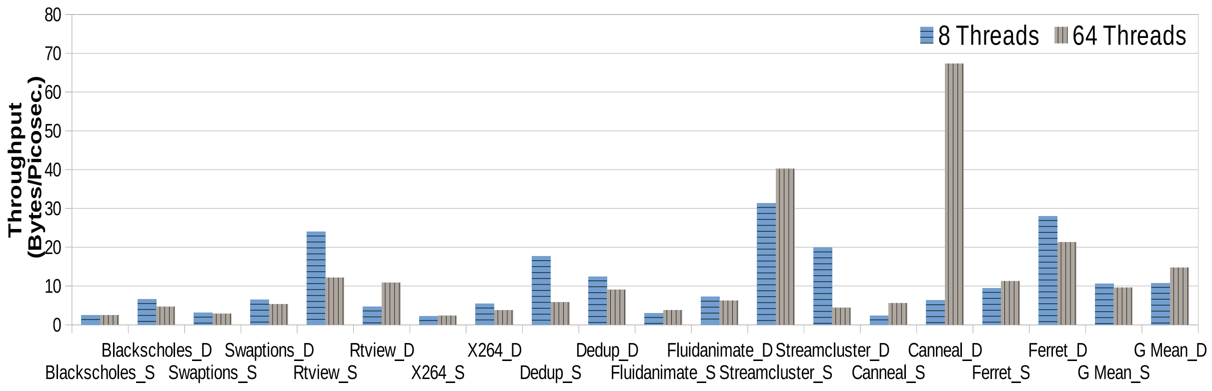


FIGURE 3.8: Throughput when number of threads are 8 and 64 with PARSEC benchmark. In x-axis, S and D with benchmark indicate Single-NoC (S) and Dual-Network NoC (D). Results are evaluated with 64 core processor.

On increasing number of threads, all the message types do not increase in the same proportion when compared to those for 8 threads. The message distribution for dual-network-NoC as was assigned for the case of 8 threads may not remain effective. Few applications, i.e., *rtview*, *dedup* and *streamcluster* do not perform well. This is due to increase in synchronization overhead with increase in number of threads for these applications. Therefore, throughput is decreased. Contrary, application like *canneal* exhibits significant improvement in throughput because of inherent parallization as compared to other applications, and hence on average 1.5x throughput gain is achieved with dual-network-NoC although no gain is observed with single-NoC.

3.5.2 Selection of NoC Hardware Configuration

In this section, we investigate performance in respect of variations in number of core, size of buffer and number of VCs.

3.5.2.1 Number of Cores

In Fig. 3.9, throughput of 16 core and 64 core processors are compared. As the number of cores increases, throughput gain is expected. The dual-network-NoC with 64 core exhibits 1.6x more throughput as compared to 16 core processor. There is an improvement in all applications since multiple threads are easily mapped to 64 cores. PARSEC benchmark applications are best mapped according to number of cores [49]. Increasing number of cores shall, however, not ben-

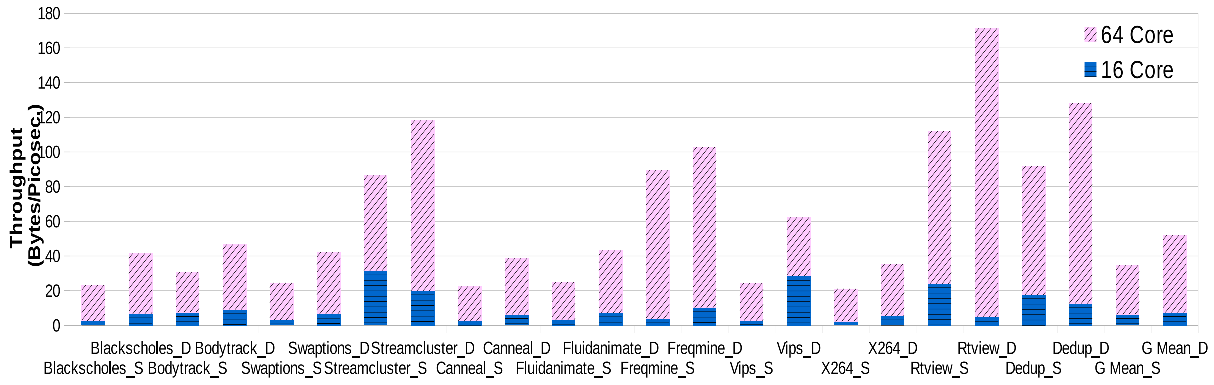


FIGURE 3.9: Throughput comparison between 16 and 64 core processor with PARSEC benchmark. In x-axis, S and D with benchmark indicate Single-NoC (S) and Dual-Network NoC (D).

efit an application if it does not have high enough degree of parallelism to exploit availability of more cores. For applications like *dedup*, *freqmine*, and *streamcluster* significant improvement in throughput is observed since they have more parallel support for multithreading. With 64 cores processor, we observe minimum gain with *vips* whereas we get best throughput improvement of 23x in *rtview* for 2-network. From Fig 3.9, we also infer that throughput improves for 64 cores single-NoCs compared to 16 cores processor, but this improvement is less as compared to dual-network-NoC.

3.5.2.2 NoC Buffer Size

The buffers are used to hold the flits awaiting for channel bandwidth. A single buffer can hold multiple flits. To accommodate a single flit, the provisioned space for buffer should be equivalent to the size of flit. So the total buffer size should be at least the product of flit size and the maximum number of flits it can store. The selection of buffer size significantly impacts the efficiency of the network. The total amount of buffering space required is the product of the number of virtual channels multiplied by the individual virtual-channel depth. By virtual channel depth, we mean, buffer/queue size that is the capacity of a buffer to accommodate a number of flits. The buffers contribute a lot to area and power consumption of NoC [19]. The variation in throughput with respect to buffer size is shown in Fig 3.10. In our experiments, we

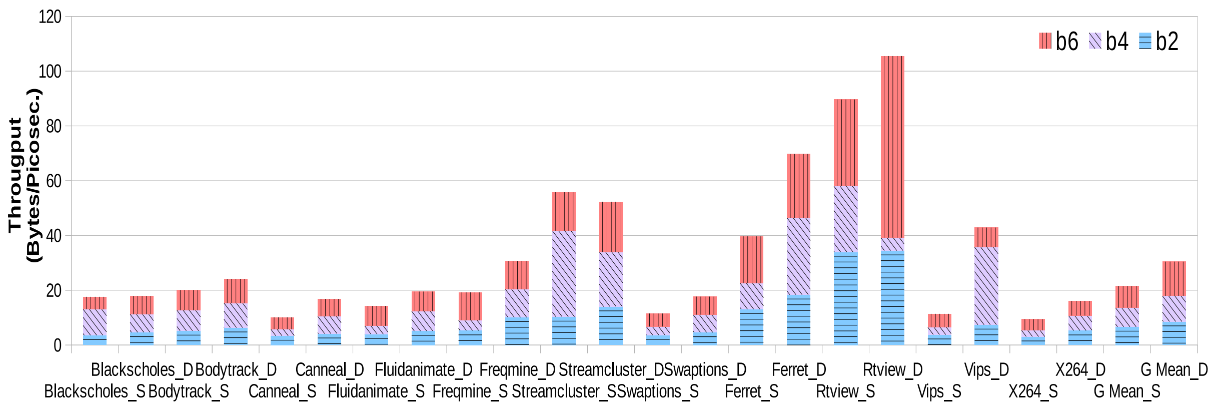


FIGURE 3.10: Throughput when buffer size 2/4/6 in a 16 core processor using PARSEC benchmark. In x-axis, S and D with benchmark indicate Single-NoC (S) and Dual-Network-NoC (D).

denote buffer size as number of flits it can store. We have experimented with buffer sizes of 2, 4, 6 to identify the suitable buffer size that supports the overall benchmark suite performance for both single-NoC and dual-network-NoC. In all three cases of buffer size (2, 4 and 6), dual-network-NoC outperforms single-NoC. Fig 3.10 demonstrates that, overall, dual-network-NoC gives 1.7x throughput compared to single-NoC when buffer size is 6. Some applications such as *blackscholes* and *streamcluster* on single-NoC and applications on dual-network-NoC, i.e., *bodytrack, fluidanimate* and *ferret* exhibit better performance with buffer size = 4 as compared to buffer size = 6. This is due to the fact that critical messages are delayed by waiting in the queue while these messages should be prioritized in delivery. So queuing latency increases with buffer size = 6. Contrary, too small buffer size = 2 does not accommodate a good volume of flits on the NoC network that results in drop in the performance. Other applications such as *canneal*, *x264*, and specific on single-NoC, i.e., *bodytrack, rtview, vips*, and on dual-network-NoC, i.e., *blackscholes, fluidanimate, swaption* exhibit negligible improvement with increase in buffer size. Except for *rtview*, performance with buffer size = 2 is much less as compared to other buffer sizes. Any application that generates intermittent traffic shall remain almost immune to buffer sizes as flits density is not too high to warrant storing them in buffer. For communication bound applications, buffer size matters and filled buffers are responsible for the performance bottleneck. Buffer sizes cannot be too large owing to the area and power overhead they incur.

3.5.2.3 NoC Virtual Channels

Virtual Channels allow the upstream router to use a second free lane (a VC with buffer space available) when a first packet is blocked in downstream router [20]. Thus, virtual channels are used to resolve head-of-line blocking and for improving throughput of the network. The comparison between throughput when a number of virtual channels are 2/4/6 is shown in Fig 3.11. We observe that overall throughput is best with 6 virtual channels, i.e., 1.9x for dual-network-

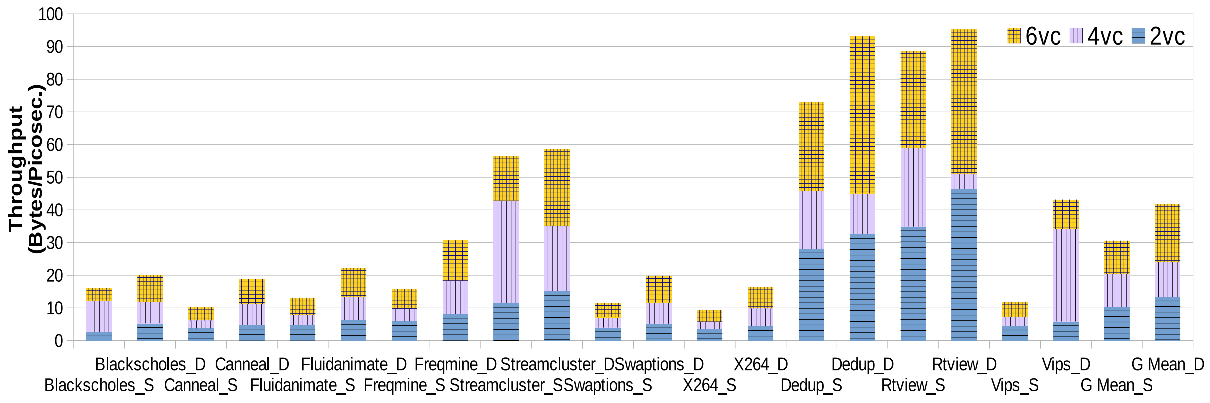


FIGURE 3.11: Throughput when number of virtual channels are 2/4/6 in a 16 core processor using PARSEC benchmark. In x-axis, S and D with benchmark indicate Single-NoC (S) and Dual-Network NoC (D).

NoC compared to single-NoC. For most applications, throughput steadily rises with increase in number of virtual channels. Although some applications such as *rtview* (2 VC, dual-network-NoC), *vips* (4 VC, dual-network-NoC), and *streamcluster* (4 VC, single-NoC) exhibit better performance with lower and middle number of VCs. Application *dedup* on a single-NoC does not show any improvement with increase in number of virtual channels. Whereas some applications such as *x264* and specific on single-NoC, i.e., *canneal*, *fluidanimate*, *freqmine*, *swaption*, *dedup*, *vips* do not show any appreciable increase in performance when virtual channels are increased from 4 to 6. Increase in hardware resource can improve the performance only of the application has inherent requirement for these resources as well as capability to exploit this availability.

3.5.2.4 Instruction Set Architecture

Towards the end, we have compared two different instruction set architectures ALPHA and x86 as shown in Fig 3.12. A real-time simulation of PARSEC shows the difference in results for these instruction set architectures. The throughput gain is more with ALPHA compared to x86.

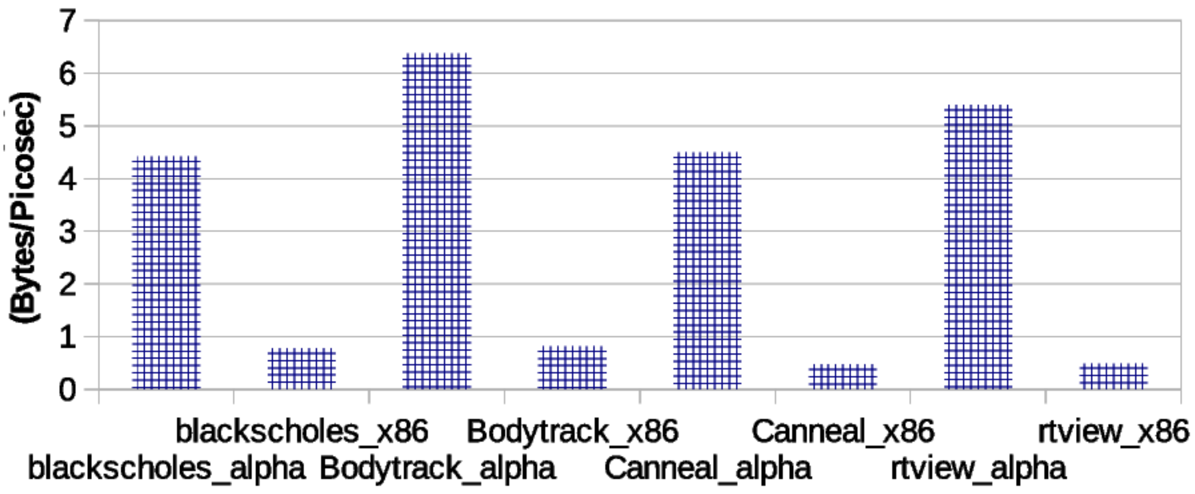


FIGURE 3.12: Throughput comparison between ALPHA and x86 processor architectures.

3.5.3 Discussions

Inferences from our experiments as discussed earlier are as follows. In the following discussion, we relate these inferences to the choices for our simulation framework as used in subsequent chapters.

- NoC traffic is higher for ALPHA instruction set because ALPHA based on RISC and x86 uses CISC architecture [155]. As a result, we have selected ALPHA over x86 as evaluation with higher NoC traffic shall be a better validation of our proposals.

- Ideally, high workload generates more traffic and is likely to be more reliable for validating NoC results. But executing applications at high workload requires a considerable amount of time. As a trade-off, we have selected medium workload to get time bound results.

TABLE 3.6: Low/Middle/High range specification for cache parameters.

Characteristics[a]	**Low**[b]	**Middle**	**High**[c]
Cache Size-L_0 (I/D)[d]	8/16 KB	32/64 KB	128/256 KB
Cache Size-L_1	256/512 KB	2/4 MB	8/16 MB
Cache Size-L_2	4/8 KB	16/32 MB	64/128 MB
Cache Line Size	8/16	32/64	128/256
Set Associativity-L_0 (I/D)	1	2/4	8/16
Set Associativity-L_1	2/4	8/16	32/64
Set Associativity-L_2	4/8	16/32	64/128
Cache Hierarchy Levels	one	two/three	four

[a]The low, middle, and high range specified as per the study of few works of literature. Though slight variations are possible in the specified range of these specifications.

[b]The range mentioned in the table or lower values.

[c]The range specified in the table or above values.

[d]I-Instruction Cache and D-Data Cache

- The variation in cache size, cache line size, selection of set associativity, and its number of cache hierarchy levels per core also impact the volume of NoC traffic. Traffic analysis with these cache patterns/parameters has already been reported in PARSEC benchmarks literature review [136]. On the basis of this review, we have selected parameter values that lie in middle range, i.e., neither too low nor too high. The selection of low values of these parameters generates a lot of traffic due to a higher number of cache misses. Though this is aligned with our requirement of high volume of NoC traffic, these smaller values of cache parameters are not realistic approach. When we look at the options of too large value of these parameters. Again, this is also not realistic parameters due to power-performance and area trade-off constraints. These extreme choices significantly increase the cost. Therefore, we have selected the middle value of these parameters as listed in Table 3.6 which generates a good volume of NoC traffic for correct evaluations of NoC designs.

- Selection of the number of threads has maximum limit dependent on the parallelization supported by the application and number of cores. So we have selected 16 threads which are supported by all applications in benchmark suite and also good for the number of 16 and 64 processor cores simulations.

- Buffer size also should be carefully selected as it may significantly impact the throughput, and queuing delays under high workloads. Larger size may increase the queuing delays and smaller size accommodate less number of available flits, and thus it may impact the throughput. So we have selected buffer size = 4 which is neither too low nor too high.
- The number of virtual networks help to resolve the protocol level message dependency. For MESI (**M**odified, **E**xclusive, **S**hared, and **I**nvalidation) cache coherence protocol [135], we need atleast three virtual networks to resolve protocol level dependency of messages. In Appendix B, we explain this protocol and justify the need for three VNs.
- Several small queues/buffers are used to form virtual channels. Buffers are significant contributor in NoC area and power, so usually a large number of VCs are not used in NoC networks. As virtual channels decouple the buffers to increase throughput by allowing active messages to pass blocked messages, they cannot be limited to a too small number. By taking care of power-performance trade-off, we have employed four virtual channels for single-NoC. The number of virtual channels are two for dual-network-NoC as we partition the VCs between 2-networks to keeps the same number of total VCs between both 2-network and single-NoC.

When we create dual-network-NoC, we keep cache parameters, number of cores, workload, number of threads, number of virtual networks[6] and size of the buffer keep the same as used in single-NoC. The only varied parameters are the number of virtual channels, link width, and flit width. Total number of virtual channels, total link width and total flit width remain same across dual-network and single NoCs

3.6 Mid-level Granularity of Message Classes

Flits that traverse NoC networks belong to different message classes. The number of message classes depends on the cache coherence protocol. We have used MESI directory protocol

[6]The number of virtual networks is dependent on the cache coherence protocol. The minimum number of required virtual networks is decided by the protocol designer. For MESI directory protocol, atleast three VNs are essentially required irrespective of NoC networks to avoid protocol level deadlock.

wherein MESI (**M**odified, **E**xclusive, **S**hared, and **I**nvalidation) cache states maintain consistency with memory data through multicast invalidations to shared copy of data hold by other caches [135] This protocol requires three hops in the critical path before the requested data block is obtained. Details of the protocol are presented in Appendix A, which also details the message classes for MESI. We have selected multicast for our experiments as the power consumption overhead due to repeated injections of the same type of messages is less with multicast over broadcast protocols [83]. The invalidation messages are considerably high in the broadcasting protocol that significantly increases the unnecessary NoC traffic and power consumption of the network.

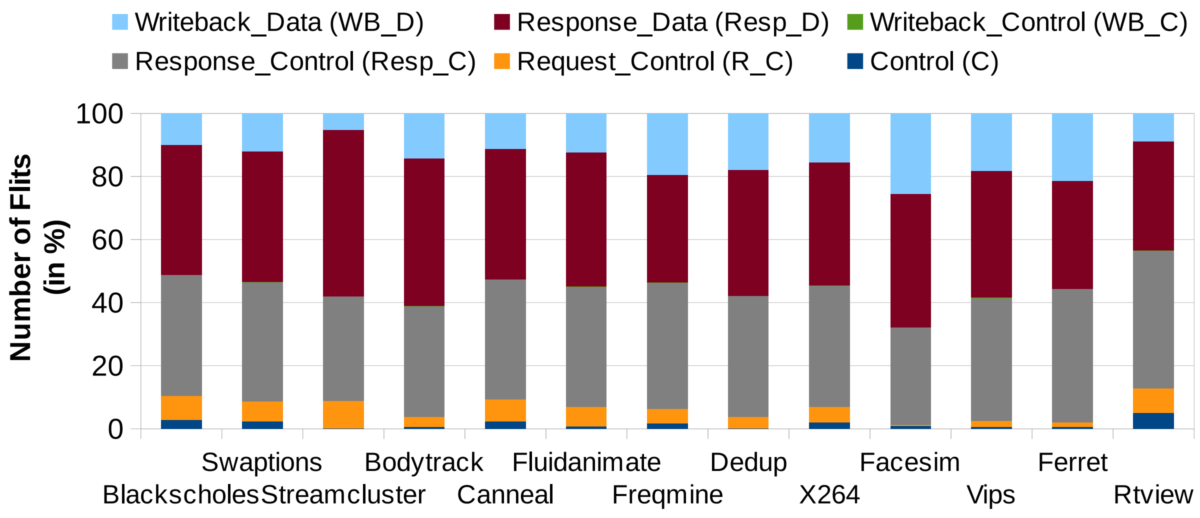


FIGURE 3.13: A comparison of inter and intra benchmarks PARSEC suite, percentage-wise distribution of the number of flits belongs to mid-level of granularity of message classes for MESI protocol.

All the volume of NoC traffic with MESI can be put into the following major classes:

1) Control (C) -This class of messages belong to MESI protocols only. The cache state invalidation and upgradation events are classified under control messages. These events are generated when the data block is modified. It invalidates the rest of the older copies of the data block.

2) Request_Control (R_C) - It initiates data block replacement events at shared caches only.

3) Response_Control (*Resp_C*) - The acknowledgment messages are generated by memory and cache in the response of memory writeback and response data. The cache state unblock event messages also be classified into this category.

4) Writeback_Control (*WB_C*) - It initiates data block replacement event at private cache when cache consists of the unique/modified copy of data.

5) Response_Data (*Resp_D*) - If the data block is a unique copy of the data, this event of messages changes the state of the cache to modified state. When the cache requires the memory data, the response data messages initiate the request to memory.

6) Writeback_Data (*WB_D*) - It initiates data block replacement event at the cache and writes the data to the memory.

The count of the different class of messages varies in the PARSEC benchmark, as can be seen in Fig. 3.13. Delay sensitive messages such as `control`, `request control`, and `writeback control` are scarce in quantity but have a larger impact on performance. `Control` initiates invalidation for caches, `request control` messages unblock the data from blocking states and `writeback control` initiates the replacement of cache block for memory. Although such messages are less in number but critical from a performance point of view.

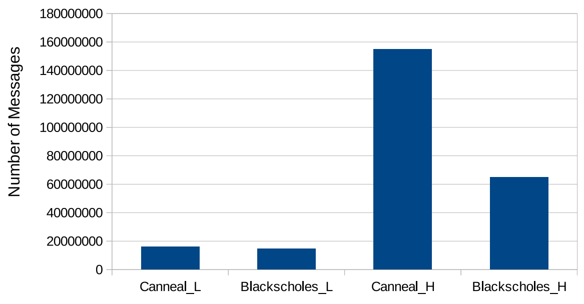


FIGURE 3.14: Message count of NoC traffic with Low(L)(*Canneal_L*, *Blackscholes_L*) and High(H) (*Canneal_H*, *Blackscholes_H*) workload of benchmarks.

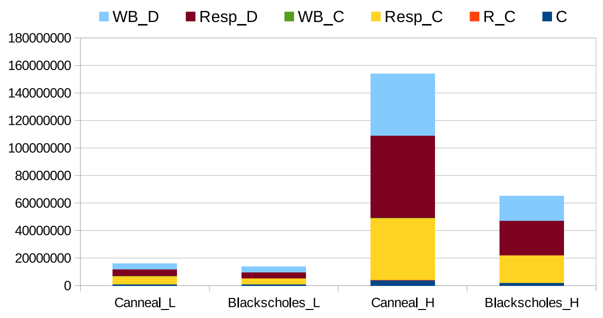


FIGURE 3.15: Middle-grain message classes distribution for benchmarks while Low(L)(*Canneal_L*, *Blackscholes_L*) and High(H) (*Canneal_H*, *Blackscholes_H*) variations observed in the total number of messages.

Fig 3.14 shows the message count of NoC traffic with Low(L) and High(H) workload for *blackscholes* and *canneal* benchmarks. Here, we observe the smallest and largest variations in the total number of messages. Fig 3.15 shows the message distribution at mid-level granularity (we shall be referring to this as middle grain).

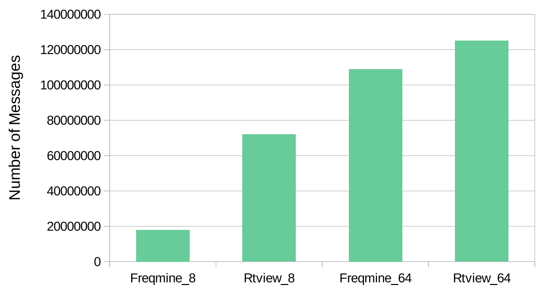


FIGURE 3.16: Number of messages variation when the number of threads varies from 8 to 64 during throughput analysis.

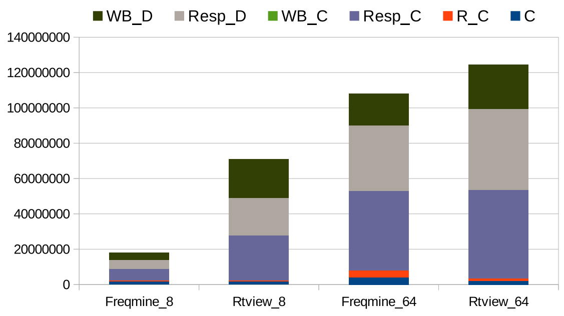


FIGURE 3.17: Middle-grain message class analysis when the number of threads varies from 8 to 64 during throughput analysis.

Likewise, Fig 3.16 shows the number of messages variation when the processor core varies from 8 threads to 64 threads during throughput analysis of *freqmine* and *rtview* benchmarks. As the maximum hike observed with *freqmine* and minimum with *rtview* benchmarks. Therefore, we have selected these two benchmarks for our middle-grain analysis. A significant hike is observed across all message classes whereas, for *rtview* except `control` and `writeback data`, we observe the increase in rest of the message classes, as shown in Fig 3.17.

Middle-grain message class variation between ALPHA and x86 ISA with respect to a number of messages received during throughput analysis across different benchmarks is shown in Fig 3.18. We observe x86 ISA shows less number of messages received as compared to ALPHA during throughput analysis. We observe significant variation in `response control` and `response data` messages as compared to the rest of the message classes, as shown in Fig 3.19. Also, a significant variation observed for `request control` only for the *bodytrack* benchmark. As *bodytrack* belongs to computer vision application, its traffic patterns are different than other applications.

We shall use middle-grain analysis in subsequent Chapter 4 and Chapter 5 to define static traffic distribution for our proposed customized NoC architectures. Further, the middle-grain analysis shall be extended up to fine-grain analysis in Chapter 6 up to the last level of granularity for message criticality analysis. The adaptive message distribution hardware unit shall shall utilise such study by for efficient traffic distribution on our proposed custom-made NoC architecture.

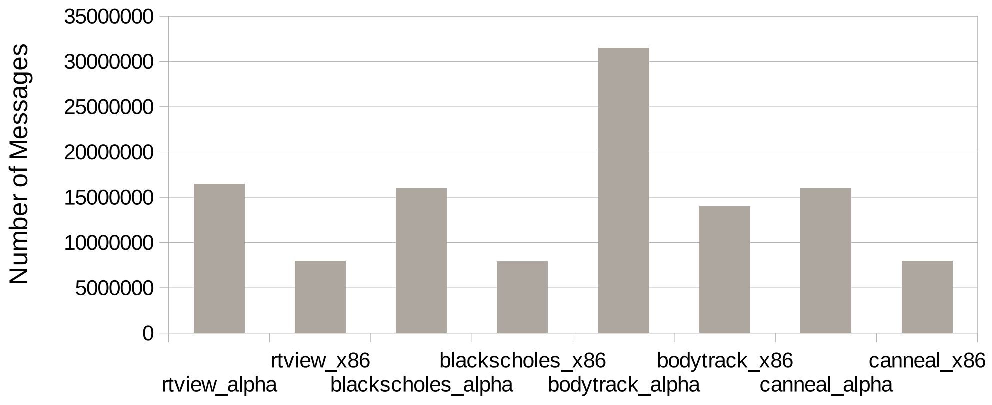


FIGURE 3.18: Comparison of ALPHA and x86 architecture's message volume in NoC traffic.

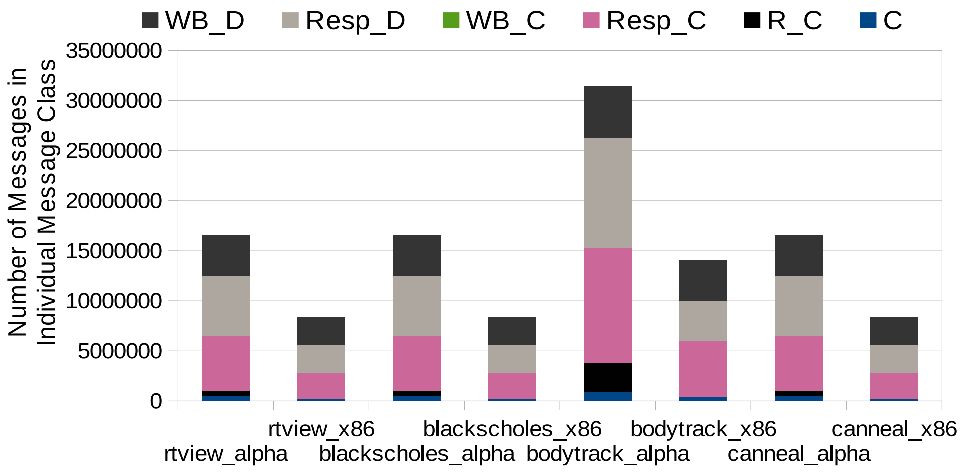


FIGURE 3.19: Middle-grain message classes distribution in the generated number of the messages for ALPHA and x86 processor.

3.7 Conclusions

In this chapter, we have introduced our experimental setup, evaluation methodology, and benchmarks. The necessary background about different characteristics of benchmark traffic is discussed to analyse NoC traffic characteristics for different applications of benchmarks. We figure out the proportion of NoC traffic over on-chip cache and off-chip memory traffic using PARSEC benchmark. The NoC traffic analysis helps to choose the experimental NoC parameters and benchmark parameters.

NoC traffic is analyzed with respect to single and dual NoC networks. We experiment both single NoC and dual NoC networks with benchmark parameters such as workload, number of threads, etc. We also vary NoC hardware configurations such as buffer size, the number of virtual channels to analyze the impact on NoC performance. Moreover, we examine message classes of PARSEC benchmark to understand middle-grain message distribution for different applications.

The discussed experimental methodology is used in all subsequent chapters to experiment with our research contributions. In the next chapter, we customise dual-network-NoC architecture to achieve static power efficiency. As we have surveyed different customisations of these architectures in Chapter 2 (Page 46), we shall explore customisation with dual-network-NoC architecture.

Chapter 4

Proposed 2-Network-NoC Architecture

In the previous chapter, we have observed that dual-network-NoC performs better than a single-NoC. Keeping networks to 2 in a multi-network NoCs is a cost-effective solution as area and power overheads remain comparable to single-NoC [43, 74]. In this chapter, we shall propose customisation in traditional dual-network-NoC to improve power efficiency.

In dual-network-NoC, the number of links from the core to router and router to router are doubled as compared to single-NoC, but each link retains half the bandwidth of its counterpart in single-NoC. For any router connected to source core, traffic injection is from the core which can generate traffic only at a certain rate. As more and more traffic is injected and buffers are filled, the backpressure mechanism shall send requests to source(s) to reduce the rate of data generation. So there is a limit on the maximum rate at which data can be injected into the network by a core.

In proposed customization to dual-network-NoC, only router-to-router links are doubled (similar to dual-network-NoC), but the number of core to/fro router links remain unchanged, i.e., same as that of single NoC and half of dual-network-NoC. The bandwidth, however, remains the same as that of a dual-network-NoC, i.e., half of that of a single NoC. In this chapter, we evaluate the performance of our proposal with the traditional single and dual-network-NoC. We shall refer to links between core and router as NI (Network Interface) links.

Reducing the number of NI links means half the bandwidth. If the injection rate is higher than the communication rate, buffers at NoC router shall begin to fill up, and the backpressure mechanism shall signal the core to reduce packet injection rate. The rate at which injected packets are routed shall ultimately limit the rate of packet injection into NoC. Thus, single NI links shall not affect traffic injection rate, and duplication of NoC links would speed up communications through the availability of two parallel links at each port. Less number of NI links shall mean less number of buffers and lower area overhead as well as power consumption. This is the motivation of our proposed customisation. We named the proposed customisation of dual-network-NoC as 2-network-NoC since only network-links (between router-to-router) of NoC participate in the formation of dual NoC networks. In the rest of the chapter, we evaluate the performance of our proposal.

4.1 Motivation and Contributions

The dual-network-NoC can be customized in different ways to achieve different performance metrics. As our objective is to address the modern processor's static power challenges, we have customised traditional dual-network-NoC architecture for this purpose. In this chapter, we explore the various aspects of dual-network-NoC and compare it with similar architectures. To understand the functionality of dual-network-NoC, how it is extended and customised from single-NoC, it is important to understand the hardware characteristics of single-NoC briefly. Using this, we propose a customized dual-network-NoC (i.e., 2-network-NoC) design that is a more cost-optimized version of dual-network-NoC networks.

Thus, our main contributions are:

- The dual-network-NoC is customised to proposed 2-network-NoC.
- We have placed a network selector hardware unit along with the routing unit to distribute message classes between 2-network-NoC.
- A detailed discussion to devise a suitable static traffic distribution for our proposed 2-network-NoC architecture.

4.2 Proposed Customisation in dual-network-NoC

In generic terms, an NoC be represented as $D \times P \times B$ wherein D is the number of planes, P is a number of NoC networks per plane, and $B = \frac{link-width}{P}$ is the bandwidth of a physical link. So in a single-NoC, $D = P = 1$, and a multiple-NoC[1] $D = 1, P > 1$ and a multi-plane NoC $D > 1$. For the dual-network-NoC and 2-network-NoC, the value of P is 2 as shown in Fig 4.1.

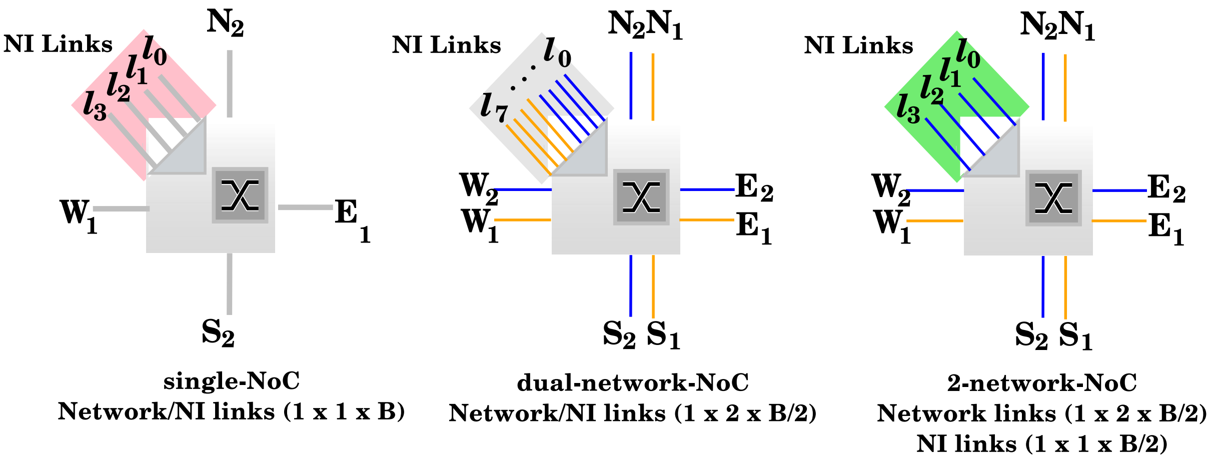


FIGURE 4.1: Proposed customisation in dual-network-NoC router to form 2-network-NoC.

Routers are connected with two types of links, namely i) network links which connect the router to another router and ii) NI links which connect router to a core. In single NoC (refer to Fig 4.1), four NI links connect the router to L_0 cache (l_0), L_1 cache (l_1), L_2 cache (l_2), and directory controller (l_3) through NI. These links are used for inter core's cache communication.

Dual-network-NoC is formed through slicing/partitioning of single-NoC which is sliced along NI links as well as networks links. As shown in Fig 4.1 (middle figure), NI links $\{l_0, l_1, l_2, l_3\}$ and network links $\{N_1, E_1, S_1, W_1\}$ interconnect the router with rest of the network and form NoC_1. Likewise, NI links $\{l_4, l_5, l_6, l_7\}$ and network links $\{N_2, E_2, S_2, W_2\}$ interconnect the router with rest of the network and form NoC_2. However, router and its primary components such as input unit, routing unit, switch allocator, crossbar and output unit are not replicated.

Fig 4.1 depicts how our proposed 2-network-NoC is different from traditional dual-network-NoC. Our proposal borrows number of NI links from single NoC and network links from dual-network-NoC with a difference that all links in our architecture have half the bandwidth of the counterpart in single NoC. The NI links are now single $\{l_0, l_1, l_2, l_3\}$ and have same link-width as in dual-network-NoC (half of the single-NoC). So this customisation improves the

[1]To understand the evolution of different multiple NoCs from single NoC, please refer to Fig 1.6 (12).

static power of 2-network-NoC as the number of NI links are half as compared to traditional dual-network-NoC.

Two different NoC networks originate from router. The network links $\{N_1, E_1, S_1, W_1\}$ interconnect the router with rest of the network and form NoC_1 and network links $\{N_2, E_2, S_2, W_2\}$ interconnect the router with rest of the network and form NoC_2. As network links are doubled unlike NI links, two NoC networks originate from the router. Therefore, network selector is placed along with the **R**outing **U**nit (RU) of the router instead of NI in 2-network-NoC.

4.3 Proposed 2-network-NoC vs. Existing Dual-network-NoC

The difference in hardware configuration of proposed 2-network-NoC vis-a-vis single NoC and dual-network-NoC are shown in Fig 4.2. The links between core to NI, NI to router, and router to router are respectively known as core links, NI links, and network links.

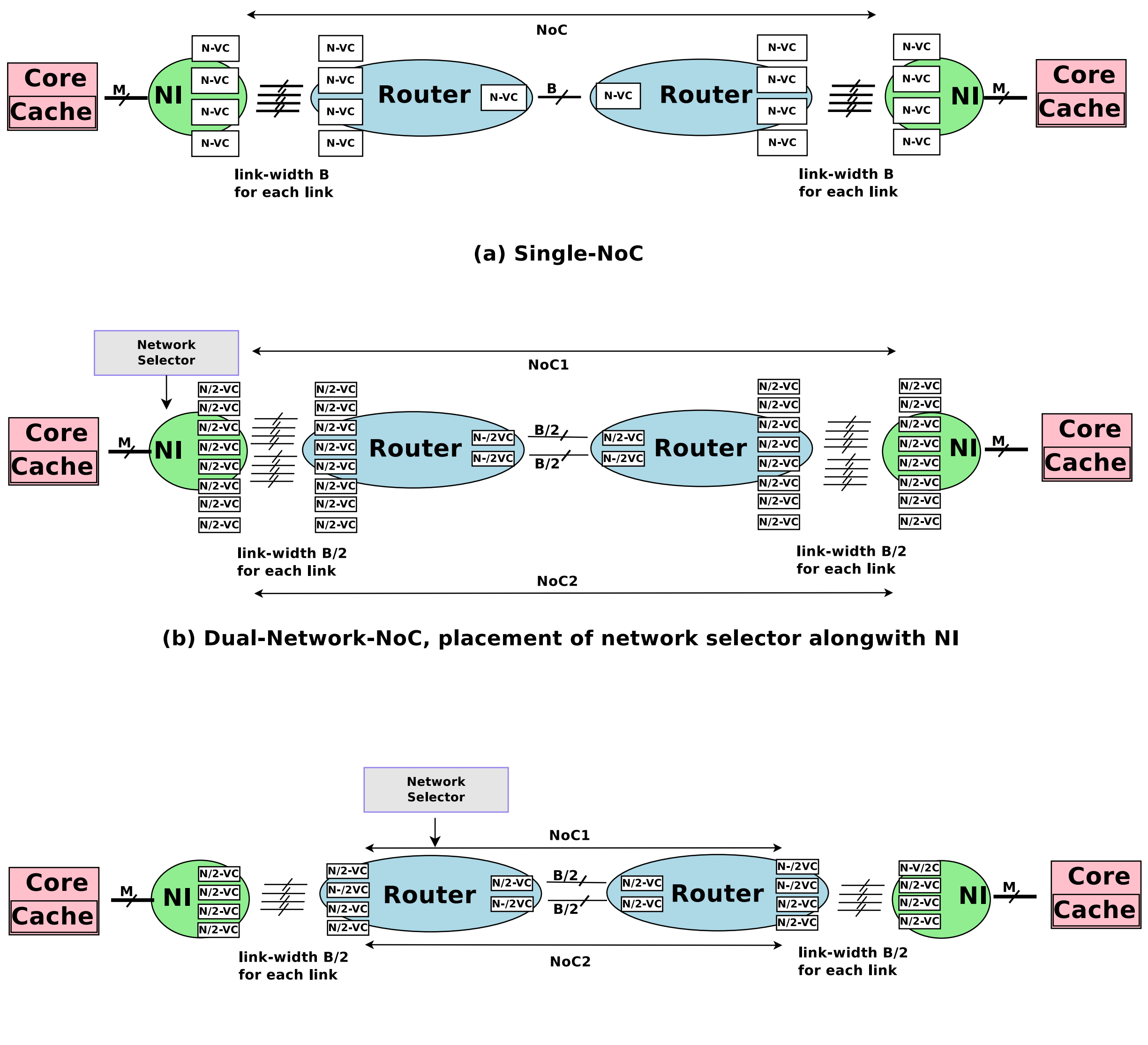
Single-NoC in Fig 4.2(a), consists of a single physical link between NI to router and router to router. Each link has B-width and transferred data can be demultiplexed to N number of VCs. The link-width of each link in dual-network-NoC is $\frac{N}{2}$ and the number of virtual channels is $\frac{V}{2}$ corresponding to each link as shown in Fig 4.2(b).

TABLE 4.1: Realization of 2-network-NoC from single-NoC through customisation in dual-network-NoC architecture.

Architecture	**`Net-Demux`**	**Link Type**	**Physical links (L)**	**Link-width [a,b]**	**VCs**
Single-NoC (Y.J. Yoon et al. [57])	Not needed	Core link	1	M	–
		NI link	4	B	N
		Network link	1		
Dual-Network-NoC (J. Balfour et al. [74])	Needed at NI	Core link	1	M	–
		NI link	8	$B/2$	$N/2$
		Network link	2		
Proposed 2-Network-NoC	Needed at Router	Core link	1	M	–
		NI link	4	$B/2$	$N/2$
		Network link	2		

[a] M≫ B, link-width unit is bits. Link-width is similar to bus-width.
[b] Link-width defines a group of links/wires that propagates signals, each link carry one-bit signal/information.

NI links and network links form two NoC networks, i.e., NoC$_1$ and NoC$_2$. A Network-Demultiplexer (`Net-Demux`), not needed in single-NoC, is generally placed at NI to distribute



(a) Single-NoC

(b) Dual-Network-NoC, placement of network selector alongwith NI

(c) 2-Network-NoC, placement of network selector alongwith router

FIGURE 4.2: Evolution of 2-Network-NoC from Dual-Network-NoC and formation of Dual-Network-NoC from Single-NoC.

the traffic to either NoC$_1$ or NoC$_2$. A `Net-Demux` is also known as network selector module/unit. As `Net-Demux` and 'network selector module' are synonyms, we shall be using these interchangeably.

The proposed 2-network-NoC as shown in Fig 4.2(c) has a similar configuration of link-width and virtual channels as in dual-network-NoC but the number of NI links are halved. Therefore two NoC are formed only with network links that connect the routers. Table 4.1 summarises the architectural differences between these NoC architectures.

In this chapter, we shall be evaluating proposed NoC architecture viz-a-viz dual-network-NoC [74]. On comparing, we find proposed 2-network-NoC is a subset of dual-network-NoC that necessitates placing `Net-Demux` at router due to a single NI links as shown in Fig 4.2(c). In essence, our proposal differs from existing NoC in two main respects

1) No duplication of links at NI though link-width is half

2) Requirements of a `Net-Demux` at router

4.4 Proposed 2-Network-NoC: Micro-architectural Details

Fig 4.3 demonstrates 2-network-NoC that is arranged in 4×4 mesh tiled[2] architecture of chip multiprocessor. The single physical link-width and respective resources like virtual channels of single-NoC are partitioned between the networks to form a 2-network-NoC. The number of router ports are doubled to connect routers through network links but port capacity is partitioned between the networks in proportion of link-width. These are the light ports as compared to single-NoC.

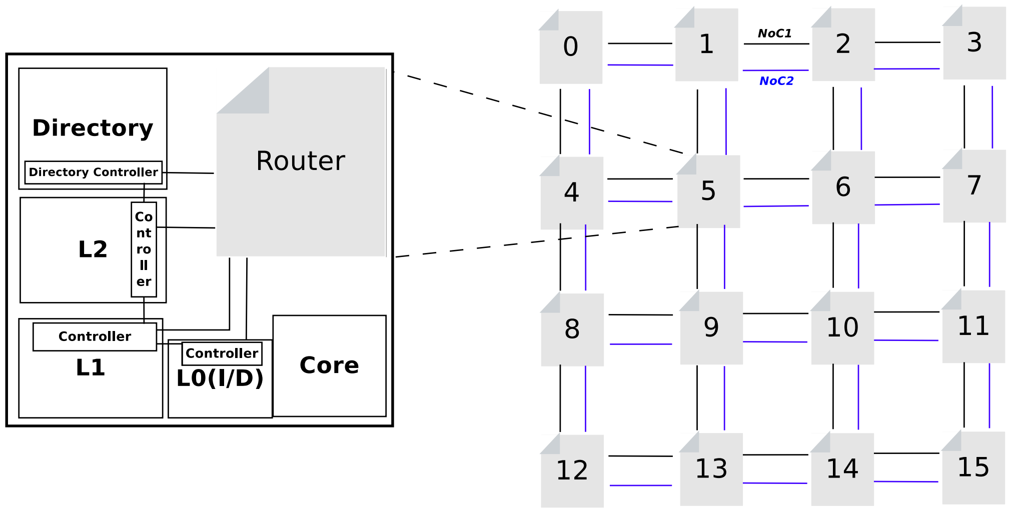


FIGURE 4.3: A typical 4×4 2-network-NoC mesh in tiled architecture of chip multiprocessor. Each router is connected with core through caches and directory controller. These components communicate with router through network interface whereas router-router communicates through network links that are arranged in a tiled architecture.

The router has four pipeline stages to route the flits [40]. The network selector hardware unit along with routing unit classify the traffic and distributes it between both mesh networks. The XY deterministic dimension order routing route the traffic between the source to destination cores. The virtual channel allocator and switch allocator select the winner flit for output channel

[2]More details of core, cache, and tiled chip multiprocessor architecture can be found in Appendix A.

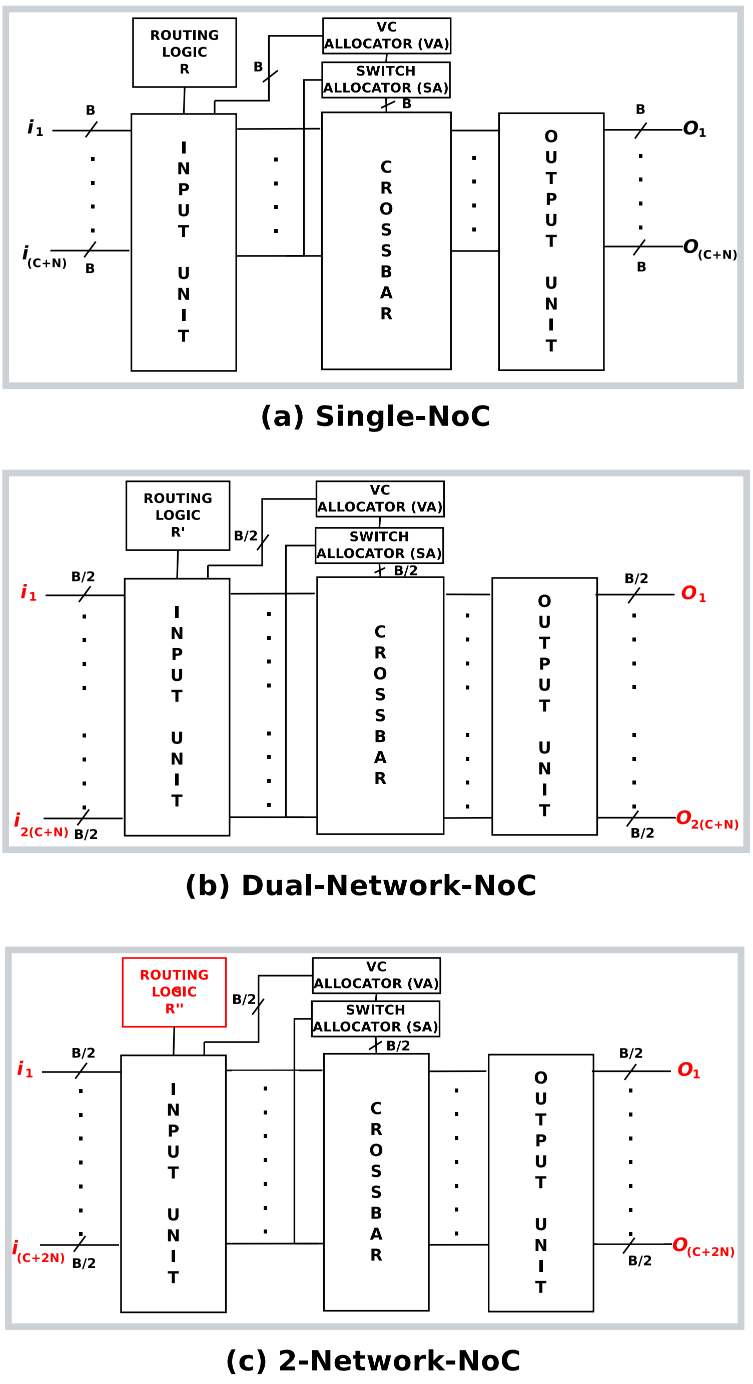


Figure 4.4: A comparison of different components of router micro-architecture for (a) Single-NoC, (b) Dual-Network-NoC and, (c) 2-Network-NoC.

among the contending flits by following round robin scheduling. We have considered virtual channel flow control to monitor the flit traversal through the network.

4.4.1 System based Comparison

The router microarchitecture of the single-NoC, contemporary dual-network-NoC, and proposed 2-network-NoC is shown in Fig. 4.4. The single-NoC router connects with NI through C number[3] of NI links, i.e., $\{i_0, \cdots i_{C-1}\}$. It connects with rest of the network via N number[4] of network links, i.e., $\{i_C \cdots i_{(C+N-1)}\}$. The existing dual-network-NoC replicates both NI links $(2 \times C = 2C)$, i.e., $\{i_0, \cdots i_{2C-1}\}$ and network links $(2 \times N = 2N)$, i.e., $\{i_C \cdots i_{2N-1}\}$, so total links are $\{i_0 \cdots i_{2(C+N)-1}\}$. Here, the router of single-NoC contains total $(C + N)$ links, and dual-network-NoC router contains total $2(C + N)$ links.

Table 4.2: A system based comparison of proposed 2-network-NoC with Dual-Network-NoC and Single-NoC. B_C and B_N are bandwidth (in bits) of NI and network link respectively.

NoC Parameters	Single-NoC	Dual-Network-NoC	2-Network-NoC
Total Bandwidth (BW) of all NI Links	CB_C	CB_C	$\frac{C\times B_C}{2}$
Total Bandwidth (BW) of all Network Links[a]	B_N	B_N	B_N
Bandwidth (BW) of a single NI Link	$\frac{C\times B_C}{C}=B_C$	$\frac{C\times B_C}{2C}=\frac{B_C}{2}$	$\frac{C\times B_C}{2C}=\frac{B_C}{2}$
Bandwidth (BW) of a single Network link[a]	B_N	$\frac{B_N}{2}$	$\frac{B_N}{2}$
Number of NI Links	C	$2C$	C
Number of Network Links	N	$2N$	$2N$
Number of Credit[b] Links	$C+N$	$2(C+N)$	$C+2N$
Number of Virtual Networks[c]	V_N	V_N	V_N
Number of Virtual Channels[c]	V_C	$\frac{V_C}{2}$	$\frac{V_C}{2}$
Buffer Size (Q) (in terms of number of flits)	Q	Q	Q
Flit Size (in bits)	F	$\frac{F}{2}$	$\frac{F}{2}$
Buffer Capacity	$Q\times F$	$Q\times\frac{F}{2}$	$Q\times\frac{F}{2}$
Number of Links to RU[d]/VA[e]/SA[f]	$C+N$	$2(C+N)$	$C+2N$
Number of Links to Crossbar	$C+N$	$2(C+N)$	$C+2N$

[a]In each direction

[b]Credit Links- These are the signal lines along with physical links that are placed between the routers to notify upstream router when a buffer is vacated. They help upstream router to keep track of the number of buffers available in the downstream router by sending a credit count (available buffer space).

[c]Corresponding to Each NI or network Link

[d]RU-Routing Unit

[e]VA-Virtual channel Allocator

[f]SA-Switch Allocator

In contrast, proposed 2-network-NoC replicates only network links $(2 \times N = 2N)$, i.e., $\{i_C \cdots i_{(C+2N-1)}\}$. The NI links are C $\{i_0, \cdots i_C\}$ that is similar to single-NoC. Thus, a typically 4×4 mesh has the number of links as follows:

Single-NoC $(C + N)$ < 2-Network-NoC $(C + 2N)$< Dual-Network-NoC $2(C + N)$.

[3]The value of C depends on cache hierarchy, in our case C=4.

[4]For mesh, the maximum number of network links is $N = 4$.

The proposed 2-network-NoC is a subset of dual-network-NoC. The bandwidth variations of the NI and network links are already discussed in Section 4.2 (Page 85) and Section 4.3 (Page 86). Other than NI and network links characteristics, router's micro-architectural components are also affected due to customisation in dual-network-NoC in our proposed architecture. The number of credit links changes in the same proportion as the total number of NI and network links changes. The number of virtual networks (V_N) remains same across each NoC architecture design as the minimum number of requisite virtual networks is fixed for each cache coherence protocol as discussed in Chapter 3 (Page 75).

The flits are stored in the buffer during the traversal of the network. If a buffer can accommodate N number of flits then total buffer capacity shall be number of flits (N) $\times$ buffer size (Q) $\times$ flit size (F). Since the buffers are associated with each input and output links, number of buffers needed in 2-network-NoC is less than dual-network-NoC and single NoC. Also, the number of virtual channels is half because of NI links being halved in 2-network-NoC. The total buffer capacity remains same for network links in each direction of router. Buffers are one of the major contributors to total router power consumption and router area.

Other control units that operate the flit traversal through router are routing unit (RU), virtual channel allocator (VA) and switch allocator (SA) as shown in Fig 4.4 (for more details refer Appendix C, Page 185). These units are also simplified in 2-network-NoC because of reduction in total number of input and output links. Likewise, crossbar size is also reduced. These micro-architectural variations of 2-network-NoC significantly contribute to total router area and power gains when compared with single and dual-network-NoC.

4.4.2 Port Number Allocation

Port numbering is used by the router to send the flits/packets to next router in the path[5] of destined router. Router ports bind with links to prepare communication channel corresponding to both NoC networks connected in mesh topology. Using port numbering, 2-network-NoC is connected with rest of the routers during formation of topology.

The port allocation of routers for proposed 2-network-NoC is devised from the single-NoC. The single-NoC router port numbers can be seen in Fig. 4.5 (i) and its directions of connectivity,

[5]Traversal direction is computed through routing unit (Section 4.5.1).

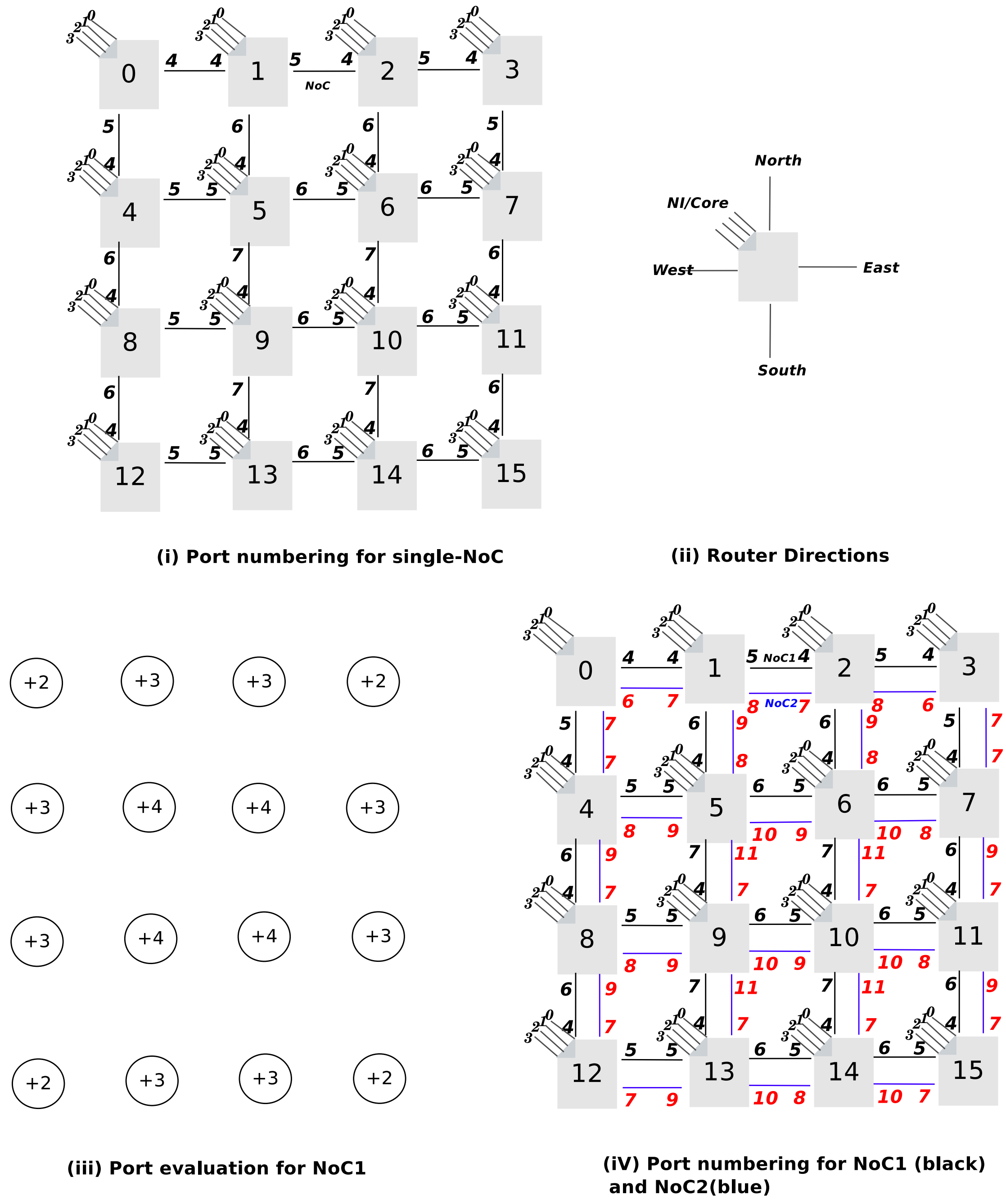

NoC
North
NI/Core
West
East
South
(i) Port numbering for single-NoC
(ii) Router Directions
+2 +3 +3 +2
+3 +4 +4 +3
+3 +4 +4 +3
+2 +3 +3 +2
(iii) Port evaluation for NoC1
NoC1
NoC2
(iV) Port numbering for NoC1 (black) and NoC2(blue)

FIGURE 4.5: Port allocation of (i) Single-NoC in Mesh, (ii) router directions (North, East, South, West, NI/Core) in mesh, (iii) values (store in array P) that will add in single-NoC port values to find 2-network-NoC port numbers (iv) 2-network-NoC.

i.e., North, East, South, West, and NI/Core in mesh is shown in Fig. 4.5 (ii). The router port numbers start with value 4 as port numbers 0, 1, 2, 3 are assigned for NI[6] links.

In terms of neighborhood connectivity, routers in a 2D mesh NoC can be classified as

1) corners: a router with 2 ports

[6]Port number 0 is for l_0 cache, 1 is for l_1 cache, 2 is for l_2 cache, and 3 is for directory controller.

2) border: a router with 3 ports

3) interior: a router with 4 ports

This classification is based on the placement positions of the routers in the mesh topology. After port numbering, the links connect the subsequent routers through binding of links with a given port number of neighboring router. At every router, first four ports 0–3 are reserved for NI links. Port 4–7 are assigned to NoC_1 and port 8–11 are assigned to NoC_2. For example, the port 4 (in East) of router 0 is binded with port 4 (in West) of router 1. Another port 5 (in South) of router 0 is connected with port 4 (in North) of router 4. Once the connection is done router 0 is connected with router 1 and router 4.

Similar to single NoC, port numbering is requisite for dual network mesh. We derive an algorithm for proposed 2-network-NoC to assign the port numbers of routers corresponding to the second link. The corner router ports of the second link will take the increment of +2, border router will add +3 and rest of the middle routers will take the increase of +4, that is shown in Fig. 4.5 (iii). After doing all such increments in port values of single-NoC, we get the port numbers for the proposed 2-network-NoC, as shown in Fig. 4.5 (iv). The network links connect the routers through these port numbers.

Now, for second NoC port binding, the links are connected with routers similar to single NoC but with different port numbers. For example, for NoC_2, the port 6 (in East) of router 0 is binded with port 7 (in West) of router 1. Another port 7 (in South) of router 0 is connected with port 7 (in North) of router 4. Now the router 0 is connected with router 1 and router 4 through NoC_1 and NoC_2 networks. The routers 0 is ready to communicate with router 4 and router 5 using two networks. Likewise, ports of the rest of the routers are binded with links and prepare a dual NoC network in mesh topology.

4.4.3 Network Selector Module

Routing unit does the route computation for the router. It comprises routing logic that computes the route for head flit. It connects with all channels of the input unit to route the traffic towards the destination. We have assumed XY routing in all NoC networks for their comparison. The following subsections explain the details of network selector logic:

4.4.3.1 Logic Design

The schematic diagram, in Fig. 4.7 shows all inputs (*North*, *East*, *South*, *West*, m_0, m_1, *b*, *S*) and output (*out′*) of the network selector logic. Lets assume this is the logic of router 5 (Fig. 4.5(i)) so port values of a 3-bit multiplexer (**MUX(4:1)**) initialize to (*North* = 4, *East* = 6, *South* = 7, *West* = 5). The port value 0, 1, 2 is for l_0, l_1, l_2 caches[7] and port value 3 is used for the directory controller.

The **OR** gate identifies the message class that we want to send on NoC$_2$ of 2-network-NoC. The variables m_0 and m_1 are used for selection of 2 message classes out of 6, i.e., `control` and `writeback control` as discussed in next Section 4.5 (Page 98). The flag value of m_0 and m_1 are set to true at the NI corresponding to *C* (m_0 = 1) and *WB_C* (m_1 = 1) messages. The flag *m* enables for specified class of messages. Further, an **DEMUX** makes the decision according to

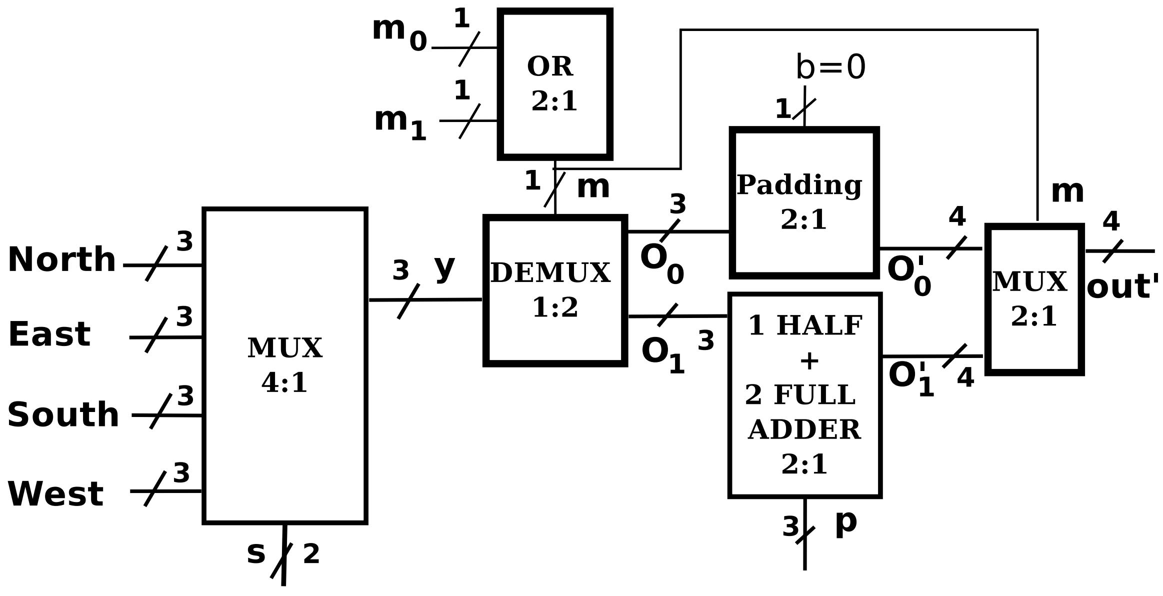


FIGURE 4.6: Block diagram of network selector logic comprising of 1 MUX (4:1) with four inputs (3-bits each) and one output (y) of 3-bits, 1 OR (2:1) with two inputs (m_0 and m_1) and one output (*m*) of one-bit each, 1 AND (2:1) with two inputs (one 3-bit and another is 1-bit) and one 3-bit output, and 1 ADDER (2:1) (one half and two full adders as they perform the addition of three bits) with two 3-bit inputs and two outputs of *out′* (*sum*) 3-bits and one *msb* (*carry*) of 1-bit. So final output is 4-bits to accommodate the port value of NoC$_2$ network.

select line *m* if the flag $m = 1$ is enabled then the message is redirected on O$_1$ to add the value of P (in Fig. 4.5(ii)) in existing port value of input. The output of adder is 4-bit O'$_1$ port value of NoC$_2$ and final output *out′* is assigned with port value of NoC$_2$. If flag $m = 0$ then O$_0$ line is selected by DEMUX. One MSB[8] as a fourth bit with value 0 is padded with 3-bit port value

[7]Three levels of the cache hierarchy.

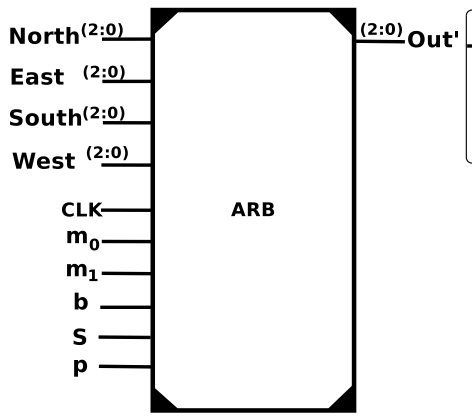
[8]**M**ost **S**ignificant **B**it

of NoC$_1$. Since $m = 1$ the O'$_0$ port value of NoC$_1$ is forwarded as a final output out' and hence NoC$_1$ is selected for traversal on input flit.

For example, let's say XY routing unit selects the output port for NoC$_1$ at router 5, i.e., the west port $West = 5$ is selected as the output of MUX, so $y = 5$. The message class belongs to C then flag $m_0 = 1$, so $m = 1$. Therefore, the 3-bit output of **DEMUX** will be $O = 5(101)$ on O$_1$ line. Now to calculate the port value of NoC$_2$ of 2-network-NoC the value of array $P = 4$ is added via **ADDER** so $out' = 9(1001)$ that is the port number of NoC$_2$. Finally, the message is redirected on NoC$_2$ of 2-network-NoC. If the message class does not belong to C and WB_C messages then flag $m_0 = 0$ or $m_1 = 0$, so $m = 0$. Therefore, the 3-bit output of **DEMUX** will be $O = 5(101)$ on O$_0$ line. Now one zero bit is padded as a fourth bit MSB in input port value of NoC$_1$. The 2:1 MUX selects O'$_0$ input line as a output because of select line $m = 0$. So the message is redirected on NoC$_1$ of 2-network-NoC.

4.4.3.2 RTL Synthesis

We have done **R**egister **T**ransfer **L**evel (RTL) synthesis in two steps. Initially, we use the Xilinx ISE framework to synthesize our logic on the Xilinx XC6VLX75T device of Virtex 6 family that is a **F**ield **P**rogrammable **G**ate **A**rray (FPGA) implementation. The area of the network selector logic in terms of the number of **L**ookUp **T**ables (LUTs) is 4. Further, we do **A**pplication **S**pecific **I**ntegrated **C**ircuits (ASIC) synthesis with Cadence RTL Encounter



Configuration Parameters

Parameter	Configuration
Synthesis Tool	Cadence RTL Compiler
Xilinx FPGA Implementation	Virtex 6 (XC6VLX75T)
Nanometer Technology	90 nm

Synthesis Results

Instance	Value
Number of Cells	5
Cell Area (um)^2	23
Dynamic Power (nW)	343.6
Leakage Power (nW)	55.06
Critical Path Delay (ns)	0.324

FIGURE 4.7: Schematic diagram of network selector logic with synthesis results at 90nm.

Design Compiler [42] with TSMC 90nm technology and target clock frequency set to $1GHz$. The number of cells is five that occupy $23um^2$ area. The leakage power of logic is $55nW$, as shown in Fig. 4.7. The dynamic power of logic can only be an estimate since the flow of the traffic also affects the switching activities of the network. So actual dynamic power can vary accordingly. The critical path delay with network selector logic is $0.324ns$.

Table 4.3: Hardware Synthesis Parameter Configuration

Parameters	**Configuration**
Synthesis Tool	Synopsis Design Compiler (version L-2016.03)
Hardware Design	**R**egister **T**ransistor **L**evel (RTL) synthesis using Verilog
Cell Technology	**L**ow **V**oltage **T**hreshold (LVT)
NoC Networks	Single-NoC, Dual- and 2-Network-NoC
Nanometer Technology	32 nm

4.4.4 Synthesis Results

To compare the proposed 2-network-NoC architecture with single-NoC, and dual-network-NoC, we have synthesized them on Synopsys Design Compiler (version L-2016.03). The synthesis results are evaluated on the 32nm library with **L**ow **V**oltage **T**hreshold (LVT) transistor cells.

Table 4.4: Synthesis results of different router architectures when path slack is zero.

NoC Architecture / **Slicing D.**[a] / **Parameters**	**Single-NoC**	**Dual-Network-NoC**	**2-Network-NoC**
	$1 \times 1 \times 256$	**$1 \times 2 \times 128$**	**$1 \times 2 \times 128$**
Area (μm^2)	3507	3766	2650
Leakage Power (μW)	74.47	77.66	54.73
Dynamic Power (μW)	11.57	12.22	24.91
Total Power (μW)	86.04	89.88	79.64
Clock Period[b]	0.2	0.24	0.12
Critical Path Delay[c] (ℂ)	0.18	0.23	0.10

[a]Slicing D-Slicing Dimensions
[b]The unit of clock period is nanoseconds.
[c]The unit of critical path delay is nanoseconds.

Table 4.4 shows the synthesis results on different clock periods as each design can run at different clock frequencies. The running frequency of a circuit is decided by its path slack that is the difference between the time data arrives, and the time the data is required. A clock frequency is known to be best if path slack is positive or ideal if slack is zero when the data arrival and data required by the circuit coincide.The lower value of a clock period shows a faster circuit as the clock period is inversely proportional to the clock frequency.

Table 4.5: Synthesis results for different router architectures at 2GHz clock frequency.

NoC Architecture / **Slicing D.**[a] / **Parameters**	**Single-NoC**	**Dual-Network-NoC**	**2-Network-NoC**
	$1 \times 1 \times 256$	**$1 \times 2 \times 128$**	**$1 \times 2 \times 128$**
Area (μm^2)	3529	3516	2643.8
Leakage Power (μW)	74.1	76.55	54.2
Dynamic Power (μW)	4.51	6.53	5.7
Total Power (μW)	78.61	83.08	59.9
Critical Path Delay[b] (ℂ)	0.21	0.28	0.12

[a]Slicing D-Slicing Dimensions
[b]The unit of critical path delay is nanoseconds.

As can be seen from Table 4.5, 2-network-NoC can run on a lower clock period as compared to single-NoC and dual-network-NoC. So it is fastest among these architectures on zero path slack. In addition to that, 2-network-NoC is more efficient in the area, total power, and critical path delay as compared to other NoC architectures. However, the dynamic power[9] consumption of 2-network-NoC is higher compared to other architectures. The higher clock frequency increases the switching activity of the circuit. Higher speed in our proposal comes at the cost of higher dynamic power. In next paragraph, we observe how our proposed architecture performs when clock frequency is kept the same for all architectures. When we fix the clock frequency (2 GHz) and synthesize these architectures, we get approximately same dynamic power for all these architectures as shown in Table 4.5. Our proposal 2-network-NoC exhibits more benefits in area, leakage and total power, and critical path delay. As can be seen here, 2-network-NoC area is reduced by 24.8%, leakage power is reduced by 29.2%, total power is reduced by 28% and critical path delay gets reduced by 57% as compared to dual-network-NoC. As compared to single-NoC, 2-network-NoC area is reduced by 25%, leakage power is reduced by 27%, total power is reduced by 24% and critical path delay gets reduced by 57%. There is an increase in

[9]As synopsis uses default wire load models for dynamic power evaluation, these dynamic power results are not reliable. We get the correct dynamic power during full system simulations with benchmarks when the traffic stresses hardware.

dynamic power as compared to single-NoC but that was to be expected owing to increase in number of network links. Though the dynamic power is less than that of dual-network NoC.

4.5 Message Classification, Distribution and Routing

Traffic can be classified on the basis of (i) volume and (ii) quality. The volume and quality of NoC traffic significantly affect the performance and dynamic power of the network. Earlier, the researchers have classified traffic on the volume basis because of limited support for cache coherence. The volume based distribution can be

1) **Uniform**[10]. The traffic is equally divided between both networks. The evenly distributed message classes are divided in the ratio of 3:3 combinations along both NoC networks of 2-network-NoC.

2) **Nonuniform**. The different combinations of messages can be distributed across either NoC networks. It can be further categorized into, (i) Control and Data (ii) Request and Response (iii) Read and Write (iv) Onchip and Offchip messages.

We have investigated the impact of message classification and distribution over our proposed 2-network-NoC using *canneal* benchmark which is a randomly selected benchmark. We want to find the best distribution of message for NoC$_2$ of 2-network-NoC. Cache traffic can be classified under six message classes as discussed in Chapter 3 (Page 77) for MESI multicast cache coherence protocol. Of these six message classes, NoC$_1$ can be reserved for K ($K < 6$) classes and NoC$_2$ for the rest. If we set $K = 4$, NoC$_2$ is reserved for 2 message classes. There can be $^6C_2 = 15$ possible combinations. Similarly, if NoC$_2$ is reserved for 3 classes, there are 20 possible combinations (for details refer Appendix D).

To identify a suitable combination, we have selected a random application from benchmark and run our simulation. Selected benchmark in this case was *canneal*. Fig 4.8 shows throughput for both single and 2-network NoCs. Fig 4.9 presents latency for both these NoC architectures. These results have been obtained for 4, 8, 16 and 32 core processors. In both figures, leftmost bar labelled 1-link shows results for single NoC. Subsequent bars represent performance for a given distribution of message classes across two networks.

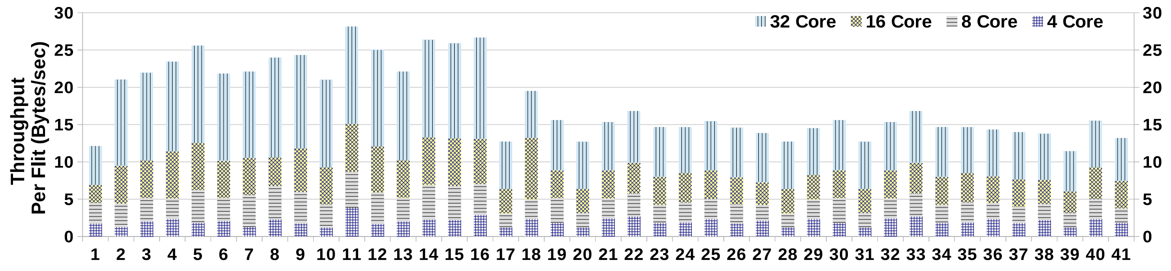


FIGURE 4.8: Throughput of *Canneal* benchmark with single-NoC and proposed 2-network-NoC (x-axis of graph show the possible static message distribution, refer Appendix D).

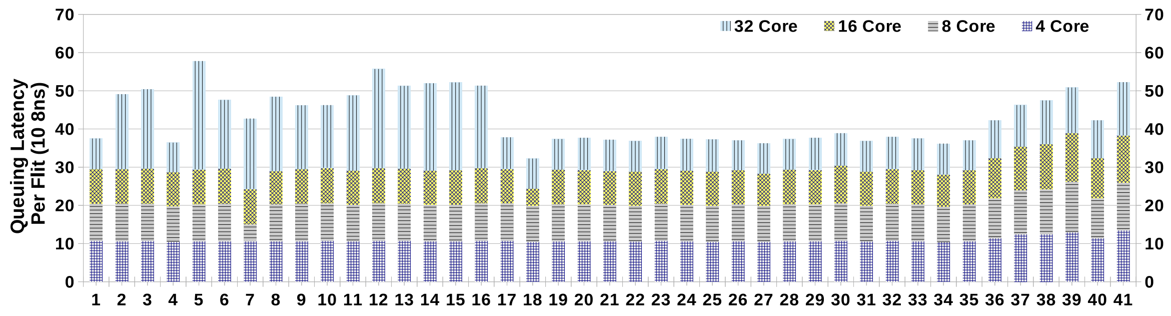


FIGURE 4.9: Queuing latency of *Canneal* benchmark with single-NoC and proposed 2-network-NoC (x-axis of graph show the possible static message distribution, refer Appendix D).

After experimentation with *canneal* benchmark, we have compared throughput and latency of 2-class as well as 3-class combinations of messages for 2-network-NoC with single-NoC. We observe that 2-class combination performs better than 3-class combination. However, 3-class combination is more suited from the perspective of queuing latency. We find that $C + WB\_C$ is the best message combination while considering both throughput and queuing latency for 2-network-NoC.

The message distribution between NoC_1 and NoC_2 are employed as `Control` and `Writeback Control` messages on NoC_1 and rest of the messages, i.e., `Request Control, Response Control, Writeback Data`, and `Response Data` on NoC_2.

4.5.1 Impact on Routing

The routing unit of a router gives the output direction for the input flit. Routing logic routes the flit through intermediate routers to the destination router which forwards the flits to its core. The route computation differs between dual-network-NoC and 2-network-NoC in terms of the number of total links participation. The number of NI links are half in 2-network-NoC

[10]Please note, here uniform message distribution means that half of the message classes are transmitted through NoC_1 and rest through NoC_2 irrespective of number of messages in that class.

as compared to dual-network-NoC (in Table 4.6[11]) whereas network links are same between existing and proposed NoC (in Table 4.7).

TABLE 4.6: Routing logic overhead in Dual-Network-NoC as compared to 2-Network-NoC because of the double NI links. The ✘indicates the incoming/outgoing traffic at NI. Total eight NI links participate in route computation in Dual-Network-NoC whereas only four NI links participate in route computation for 2-Network-NoC.

NoC→	**Dual-Network-NoC**								**2-Network-NoC**			
Link Types→	**NI Links**								**NI Links**			
Direction→	**Local to Core**								**Local to Core**			
Network→	**NoC$_1$**				**NoC$_2$**				**NoC**			
Link IDs→ / **Msg[a]↓**	l_0	l_1	l_2	l_3	l_4	l_5	l_6	l_7	l_0	l_1	l_2	l_3
Control	-	-	-	-	✘	✘	✘	✘	✘	✘	✘	✘
Request Control	✘	✘	✘	✘	-	-	-	-	✘	✘	✘	✘
Response Control	✘	✘	✘	✘	-	-	-	-	✘	✘	✘	✘
Writeback Control	-	-	-	-	✘	✘	✘	✘	✘	✘	✘	✘
Response Data	✘	✘	✘	✘	-	-	-	-	✘	✘	✘	✘
Writeback Data	✘	✘	✘	✘	-	-	-	-	✘	✘	✘	✘

[a]Msg=Message Classes

TABLE 4.7: The number of network links remain same in Dual-Network-NoC and 2-Network-NoC. So there is no difference in route computation because of these links. The ✓indicates the messages for network links on either NoC networks.

Link Types→	**Network Links**							
Direction→	**North**		**South**		**East**		West	
Network→	**NoC$_1$**	**NoC$_2$**	**NoC$_1$**	**NoC$_2$**	**NoC$_1$**	**NoC$_2$**	**NoC$_1$**	**NoC$_2$**
Link IDs→ / **Msg[a]↓**	$l_8(N_1)$	$l_9(N_2)$	$l_{10}(S_1)$	$l_{11}(S_2)$	$l_{12}(E_1)$	$l_{13}(E_2)$	$l_{14}(W_1)$	$l_{15}(W_2)$
Control	-	✓	-	✓	-	✓	-	✓
Request Control	✓	-	✓	-	✓	-	✓	-
Response Control	✓	-	✓	-	✓	-	✓	-
Writeback Control	-	✓	-	✓	-	✓	-	✓
Response Data	✓	-	✓	-	✓	-	✓	-
Writeback Data	✓	-	✓	-	✓	-	✓	-

[a]Msg=Message Classes

The 2-network-NoC routing logic shall be different as follows. In 2-Network-NoC, the number of NI links are four that is half of the dual-network-NoC. The NI links in 2-networks-NoC carry all types of messages whereas in dual-network-NoC these links carry separate traffic. The

[11]Table 4.6 and Table 4.7 list all NI links and network links. The NI links direction is *Local to Core* and network links connect the router with other routers in *North, South, East,* and *West* directions. The link IDs of both NI links and network links are associated with respective NoC networks. Here, we can see that separate networks links are used for NoC$_1$ and NoC$_2$ either in existing and our proposed NoC. Whereas, separate NI links are used for NoC$_1$ and NoC$_2$ in dual-network-NoC but same NI links of 2-network NoC are associated with router. Therefore, NI links of 2-network-NoC carry all type of messages and traffic is separated at the router. Whereas, NI links of dual-network-NoC are dedicated for specific message classes. Therefore, the traffic is separated from the the NI.

participation of the number of links is more in dual-network-NoC as compared to 2-network-NoC. Thus, architectural customisation of dual-network-NoC impacts a number of inputs and outputs of routing of 2-network-NoC.

4.6 Experimentation

In this section, we shall discuss the results of performance metrics (discussed below) evaluated with PARSEC benchmark.

4.6.1 Performance Metrics

We employ following performance metrics to evaluate proposed 2-network-NoC vis-a-vis single NoC.

1) Router Static Power: The power consumed by router component even without any circuit activity is static[12] power of the router. Static power is a big challenge for modern researchers with latest nanometer technology. It is drastically increasing as compared to dynamic power on shrinking of transistors. Therefore, saving of router static power is the primary objective of this chapter.

2) Router Dynamic Power: The internal[13] router components do switching activity to take decisions for flits traversal. During the switching activity, circuit changes the inputs and outputs from 0 to 1 and 1 to 0 with runtime variations of traffic. In higher traffic, the dynamic power is high otherwise it is low with lower router traffic.

3) Link Dynamic Power: Links that connect the NoC routers are also one of the power hungry component. The dynamic power dissipation in links is the major contributor to the NoC link power. Flit traversal affects switching and capacitance of the links. In the

[12]For more details on static power, please refer Appendix C (Page 187).

[13]The internal router component (as discussed in Appendix C) such as buffers perform read and write operations for the incoming flit. The routing logic performs the route computation for the head flit. The virtual channel allocator assigns the ID of the VC to the head flit during traversal to the output unit. Switch allocator selects the winner for crossbar traversal from the contending flits for the the crossbar input channel. So that flit reaches to the output unit through traversing crossbar of the router.

results (Subsection 4.6.2), we will compare the links dynamic power of the 2-network-NoC with single-NoC. As like router dynamic power, link dynamic power is dependent on the volume of traffic in the network.

4) Router Total Power: The total router power consumption includes static power consumption, dynamic power consumption, and clock power consumption of the router. With nanometer technology advancement, static power is more severe component in total power consumption. In our experiments with benchmarks, we have evaluated the impact through total router power. We evaluate the router power with 65nm technology and 1 GHz frequency.

5) Scalability: To understand the scalability of our proposed 2-network-NoC, we have compared the nanometer technology impact on total router power by evaluating it at 45nm and 32nm. The impact of frequency is also compared with 2GHz and 2.5GHz frequency.

6) Queuing Latency: The waiting time of a packet in the buffer of the router is queuing latency. Ideally, a packet expects zero load latency on the router before transmitting to the downstream router. Though this is possible only when buffers are not present on the router. We consider this metric as related works have employed it.

7) Throughput: It is defined as the volume of traffic (in bytes/flits) delivered to the destination of the network per unit[14] of time. If the network is below saturation, all the offered traffic is accepted by the network but the important factor is time required for routing it. In another words, throughput is the rate of offered traffic to reach its destination in the network. We have compared the throughput of 2-network-NoC with single NoC in the next subsection.

4.6.2 Results and Discussion

We have used the experimental methodology and tools as discussed in Table 3.1, Table 3.2 and Table 3.3. For the proposed architecture, the network router is configured as two VCs per port and 12 × 12 crossbar size. The physical link-width is 8-B and both control and data flits are 8-B.

[14]The unit of time is picoseconds in Gem5.

In Gem5, we have run PARSEC benchmark suite in full system simulation with 16 threads for a 8×8 mesh. We experiment with *simmedium* input data working set. Primarily, we are comparing the results of 2-network-NoC with single-NoC. We keep the total number of buffers constant in the 2-Network-NoC as compared to Single-NoC.

For example, if the Single-NoC consists of 6 virtual networks, each virtual network consist of four virtual channels (Single-NoC_6*vn*_4*vc*) then 2-Network-NoC consists of 6*vn* and 2*vc* corresponding to each link since the total number of virtual channels remain invariant across 2-Network-NoC and Single-NoC. Additionally, we vary the number of virtual channels (Single-NoC_6*vn*_3*vc*), and virtual networks (Single-NoC_3*vn*_4*vc*) of Single-NoC to see the impact of resource variation in comparison of 2-Network-NoC.

Normalised Results Representation. Let's say A is the power[15] consumption of single-NoC and B is the power consumption of proposed 2-Network-NoC. The power benefits for B is calculated as follows:

1) Normalize the power consumption of A to unity (1). The power consumption value (Q) of B re-scales over A's power consumption (P). B's proportion or normalized power consumption over A is $P' = Q/P$.
2) The power improvement of B with respect to A is $P'' = 1 - P'$.
3) The percentage (%) power improvement of B is P''x100%.

Each red bar in the figures, shows the individual normalized power consumption of B over A, for 13 different benchmarks. Let's denote them as b1 (Blackscholes), b2 (Swaptions), ..., b13 (Rtview) from left to right. The average of normalized power consumption for B is shown (Avg.(z) =((b1+b2+....+b13)/13) as the rightmost last red bar in each of the graphs.

For example, Fig 4.10 shows the router dynamic power consumption of benchmark 'Blackscholes' is b1 and A is normalized to 1. If A=1 and b1=0.82, the power improvement is ((1-0.82)/1)x100% = 18%.

Power Benefits. Fig 4.10 to Fig 4.12 show the power gain in 2-Network-NoC for all the applications of PARSEC benchmark suite. The results of different hardware resource combinations for Single-NoC and 2-Network-NoC are normalized with respect to Single-NoC_6*vn*_4*vc*.

[15]To understand the normalised representation of results, we take the example of power consumption calculations.

- Static Power: The static power and clock power of 2-Network-NoC are 58% and 29% efficient compared to Single-NoC_6*vn*_4*vc*, respectively. We have already estimated static power during system level analysis of design Subsection 4.4.1 and through hardware synthesis in Subsection 4.4.4. These are validated with full system simulation.

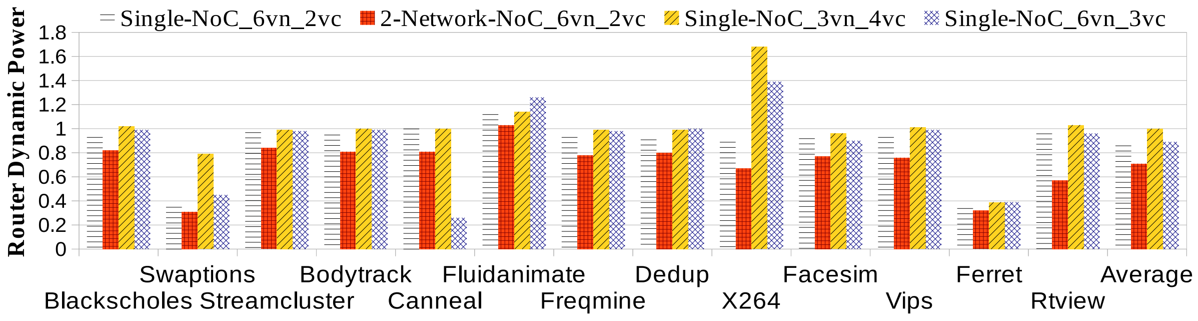


FIGURE 4.10: Dynamic router power comparison of proposed 2-Network-NoC with Single-NoC on PARSEC benchmarks.

- Router Dynamic Power: The improvement in router dynamic power of 2-Network-NoC is 29% as compared to Single-NoC_6*vn*_4*vc* as shown in Fig. 4.10. Other hardware configuration of Single-NoC perform similar to Single-NoC_6*vn*_4*vc* except *fluidanimate* and *x264*. So other opted hardware configurations are not suitable for these benchmarks.

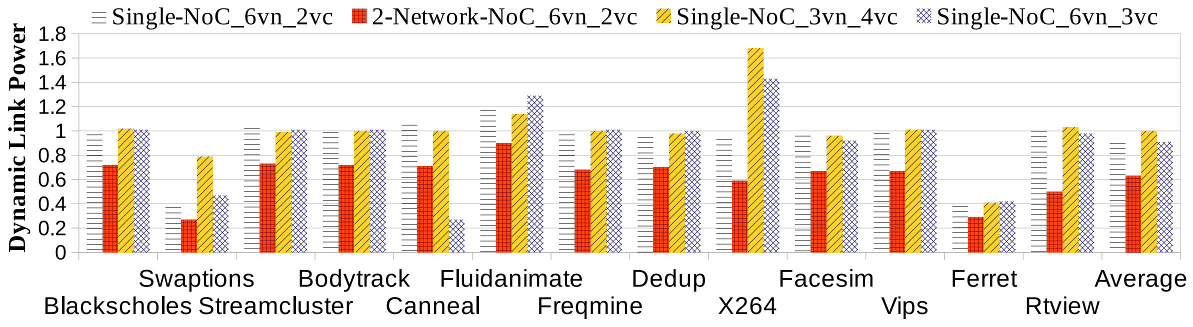


FIGURE 4.11: Link dynamic power comparison of proposed 2-Network-NoC with Single-NoC on PARSEC benchmarks.

- Dynamic Link Power: The improvement in dynamic link power of 2-Network-NoC is 37% as compared to Single-NoC_6*vn*_4*vc* as shown in Fig. 4.11. The trends in dynamic link power consumption for Single-NoC are similar to the trends observed in router dynamic power. Though, 2-Network-NoC shows more power benefits throughout all the applications.

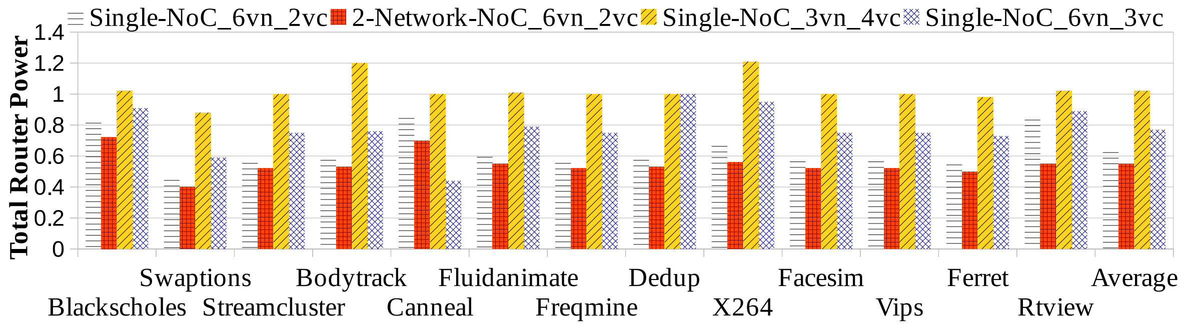


FIGURE 4.12: The total router power comparison of proposed 2-Network-NoC with Single-NoC on PARSEC benchmarks.

- Router Total Power: The total router power includes all components, i.e., static power, dynamic power, and clock power. Hence, total router power is 45% less compared to Single-NoC, as in Fig. 4.12. The 2-Network-NoC outperforms over Single-NoC_3*vn*_4*vc* and Single-NoC_6*vn*_3*vc* for all the applications of PARSEC.

The performance and other graphs are calculated in a similar way as we have computed power benefits.

Performance. The network performance is evaluated through measuring queuing latency and throughput of the network. Fig. 4.13 shows an average queuing latency faced by a single packet across the routers till its destination.

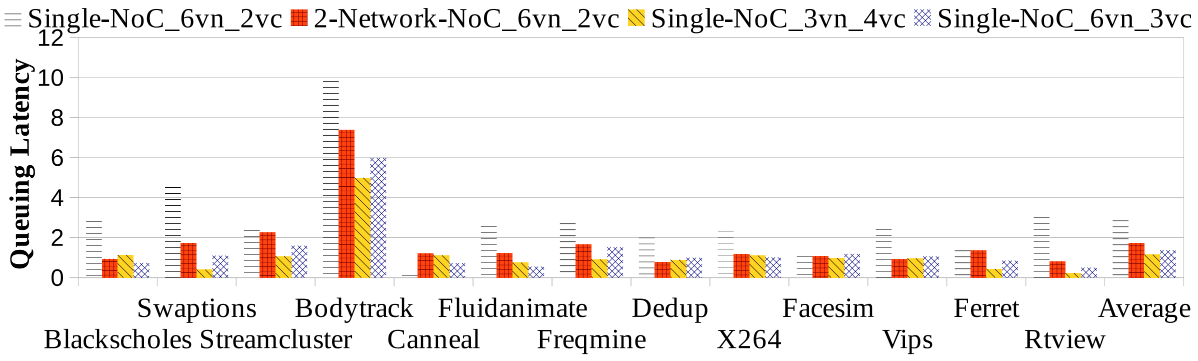


FIGURE 4.13: Latency comparison of proposed 2-Network-NoC with Single-NoC on PARSEC benchmarks.

In Fig. 4.13 and Fig. 4.14, there is a minor decrease in normalized latency (4%) and normalized throughput (5%) since the bandwidth of the core link in 2-Network-NoC is half of the Single-NoC and dual-network-NoC. While, the Single-NoC_6*vn*_3*vc* and Single-NoC_3*vn*_4*vc* have better queuing latency and throughput than 2-Network-NoC, because of, the better support provided to cache coherence classes by 3 virtual channels/networks.

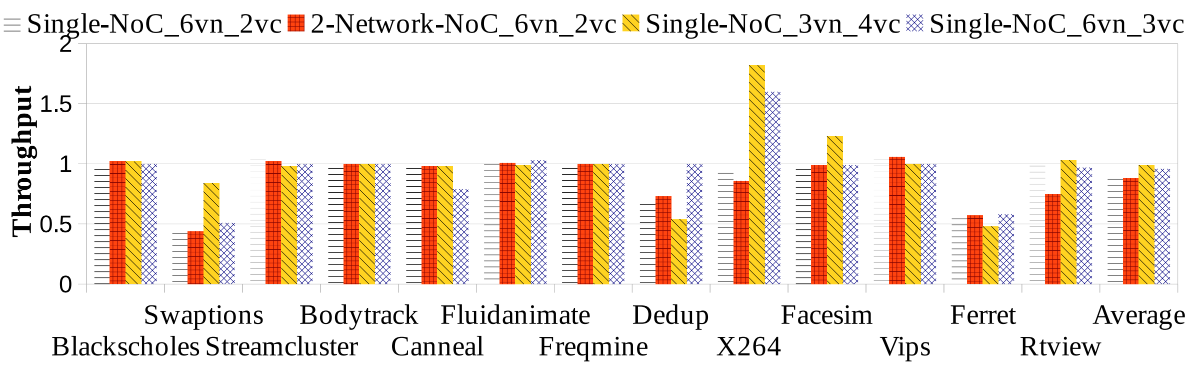


Figure 4.14: Normalized throughput comparison of proposed 2-Network-NoC with Single-NoC.

Although, queuing latency for 2-Network-NoC is 45% better than Single-NoC (Single-NoC_6*vn*_2*vc*) because 2-Network-NoC has the support of two NoC networks with 2-Network-NoC_6*vn*_2*vc*. Because of the queuing latency, the throughput of the network is affected. Thus power advantage of 2-network-NoC is achieved at a minor performance cost.

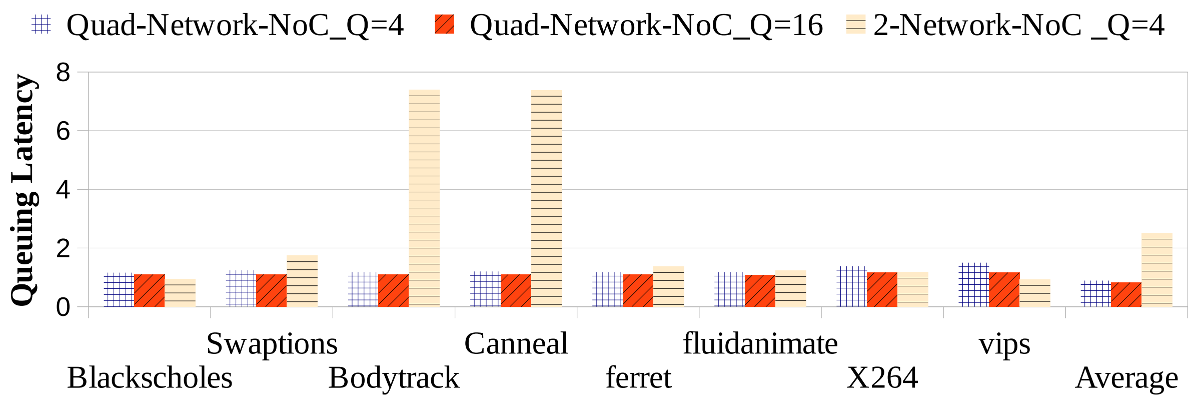


Figure 4.15: A normalized queuing latency of proposed 2-Network-NoC compared to Quad-Network-NoC with buffersize(Q) 4 and 16 [38, 57].

Comparison of queuing latency of proposed 2-Network-NoC with existing quad-network-NoC. Fig. 4.15 compares the queuing latency of 2-Network-NoC with quad-network-NoC[16] of buffer size $Q = 4$ and $Q = 16$ [38, 57]. While the buffer size of 2-Network-NoC is $Q = 4$ and number of NoC networks are dual. The buffer size Q=4 and Q=16 show that each queue buffer can store up to 4 and 16 flits, respectively.

[16]For more details on quad-network-NoC, please refer Chapter 1 (Page 9) and Chapter 2 (Page 50).

The benchmark `bodytrack` and `canneal` has high communication latency because they are more communication intensive. They are facing more congestion because of only two NoC networks in 2-Network-NoC.

Scalability of proposed 2-network-NoC with technology advancements. Technology scaling reduces lateral and vertical dimensions of the transistor. The supply voltage is scaled down to reduce power dissipation, proportionally threshold voltage is also scaled down to maintain the performance [140]. This significantly increases static power contribution in total chip power, contrary, modern multithreaded workload increases dynamic power of the chip. Therefore, consideration of both power factors are equally important in total chip power. Fig. 4.16 show that the total router power gain increases and approaches 58% as nanometer technology decreases.

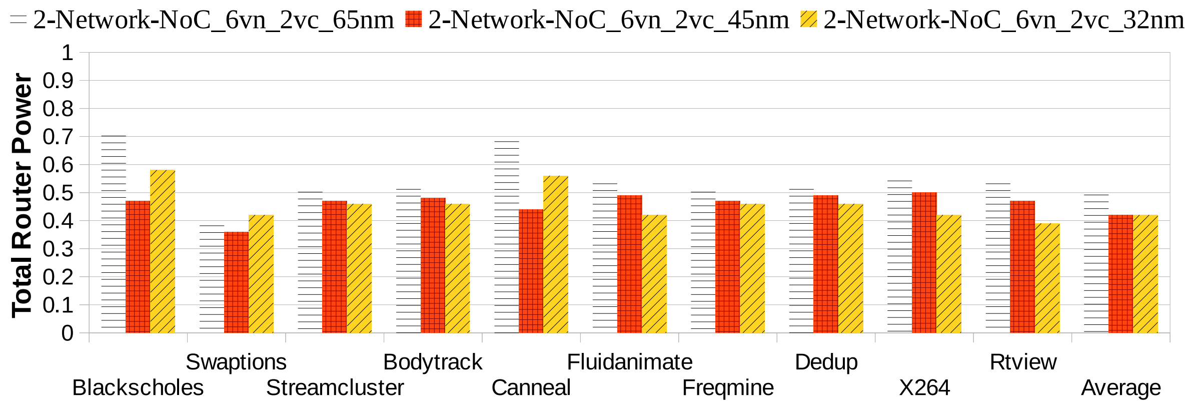


FIGURE 4.16: Normalized total router power for various nanometer (nm) technology.

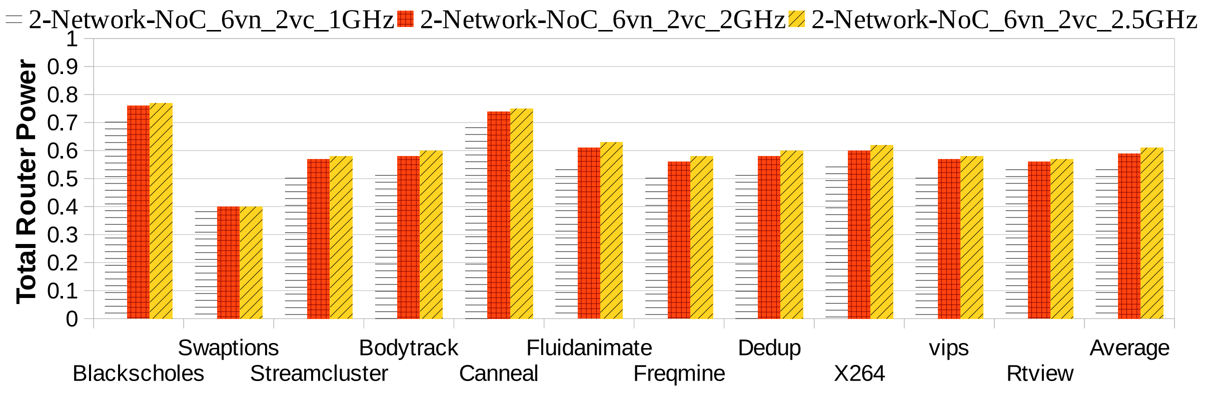


FIGURE 4.17: Normalized total router power for various frequencies.

Frequency of the circuit is another important factor that decides the performance and power of the circuit. The scaling/increasing of frequency is limited by the heat dissipation of processor.

One possible solution adopted in modern processors is scaling of the frequency as per the workload [150]. For example, Turbo Boost technology of Intel Core i7 processor with the workload variation of the processor.

For the proposed 2-Network-NoC, the power gains are limited to 40% as shown in Fig. 4.17. As switching activities of the router components increase with the frequency. Dynamic power gain of the router is reduced as operating frequency increases. Still, its power benefits are sufficient enough to make it suitable for modern processor technology.

4.7 Inferences

We have proposed 2-network-NoC architecture by customising dual-network-NoC. The number of NI links in 2-network-NoC is half of the dual-network-NoC. In single-NoC, the bandwidth is B-bit corresponding to each link while the bandwidth of dual-network-NoC and 2-network-NoC is $\frac{B}{2}$-bit. The total bandwidth is constant in each direction of network links across these architectures, though total bandwidth varies for NI links. We find that total bandwidth for NI links in 2-network-NoC < single-NoC $\simeq$ dual-network-NoC. So the bandwidth cost of NI links in our proposal lies between that of single-NoC and dual-network-NoC. We named the proposed architecture as 2-network-NoC since 1) we replicate **only network links** so separate networks exist only at the network link level, 2) we have sliced/partition NI links and removed half of the links, though the core is still **connected through two NoC** interconnects. The customisation seems simpler at an abstract level, but its consequences are significant.

1) The network selection hardware unit, placed in existing architecture at NI, needs to be placed at the router. We embed the network selection hardware unit at the routing unit of the router. Once the routing unit takes the decision of direction for the flit, one of the NoC networks is also selected for the traversal of the flit. In proposed 2-network-NoC, the routing unit hardware becomes simpler as customisation reduces the total number of input and output NI links that participate in the routing decisions.

2) These micro-architectural changes result in a significant reduction in static power consumption and surface area of the network. Since the hardware resources such as the bandwidth of NI links, number of credit links, number of virtual channels, buffer capacity, number of links for routing unit, VC allocator, switch allocator, and crossbar are

less for 2-network-NoC as compared to dual-network-NoC and single-NoC. The system based comparison and hardware synthesis show the efficiency of proposed architecture over existing ones.

3) The synthesis results are validated by running the PARSEC benchmark in full system simulation. The total router power benefits are 45% for 65*nm* that approach 58% for 32*nm* technology. Although the increase in frequency at 65*nm* limits the power gain to 40%. There is a minor performance overhead, throughput decreases (5%), and latency increases (4%) since the bandwidth of NI links are half of the single-NoC. The static power gains increase with each nanometer technology. We get a low power router based customised dual-network-NoC architecture, i.e., 2-network-NoC.

In the next chapter, we shall further explore alternate placement locations of network selection hardware unit (`Net-Demux` discussed in Section 4.3) other than routing unit on the router. As the placement impacts the digital characteristics of the circuits, it is interesting to explore such an impact in the next chapter.

Chapter 5

Placement of Network Selector

In the previous chapter, the proposed 2-network-NoC distributes traffic between two networks at the routing unit of the router. Architectural customization makes it mandatory to place a network selector somewhere on the router. One possible placement at the routing unit was presented in the previous chapter. As placement changes the characteristics of the digital circuit, it directly impacts the power-performance efficiency of the circuit. In this chapter, we explore alternate placement for the network selector.

5.1 Introduction

To the best of our knowledge, exploration for placement of network selector is done for the first time for multi-NoCs. Network selector behaves similarly to the digital demultiplexer. Demultiplexing is the process of taking the information from one input and transmitting the same over one of several outputs. Likewise, the network selector transmits different message classes[1] along physically different networks, in parallel wherever possible. In the following paragraphs, we shall be using the term Network Demultiplexer (`Net-Demux`) for this hardware unit. This is a more appropriate term as input message is demultiplexed into one of the available physical networks.

[1]`Control`, `Request Control`, `Response Control`, `Writeback Control`, `Response Data`, `Writeback Data` (MESI protocol).

Dual NoCs communicate messages of cache misses through the single physical channel to dual networks. Network selector requires `Net-Demux` to distribute the traffic between dual NoC networks.

In this chapter, we propose an improved placement of `Net-Demux` at switch allocator of the router as an alternative to the network interface and routing unit. Improvement is in terms of power efficiency. Placement changes the average number of signal transitions in a single cycle of the circuit, and hence it varies the switching activity of the circuit. Power dissipation at `Net-Demux` is related to the difference in input or output switching activity. The network selector in all proposed multiple NoC architectures is placed at the network interface. Exploring alternate placement of network selector at the router is a novel contribution. Following are the key contributions presented in this chapter:

1) Exploration of placement of network selector.
2) Analysis of network selector architecture with different placement.
3) Comparison of different placements with the traditional one.

5.2 Problem Formulation

Fig 5.1 shows the placement of `Net-Demux` in three different NoC architectures. Placement is necessary as each NoC architecture has to divide the traffic between dual NoC networks. The difference in placement of `Net-Demux` is driven through NoC architecture as discussed below:

1) **Dual-network-NoC.** The `Net-Demux` shall be placed at NI, as shown in Fig 5.1(a). The links of both NoC networks are connected with different NIs as follows- NI 0 (l_0, l_4), NI 1 (l_1, l_5), NI 2 (l_2, l_6), and NI 3 (l_3, l_7). Every NI has a choice of traffic distribution between any one of NoC networks, so four hardware units of `Net-Demux` shall be placed corresponding to each NI.
2) **2-network-NoC.** The `Net-Demux` shall be placed only at the router, as shown in Fig 5.1(b). The 2-networks are formed through routers. A single link connects different NIs with a single router as follows- NI 0 (l_0), NI 1 (l_1), NI 2 (l_2), and NI 3 (l_3). So `Net-Demux` shall be placed on the router.

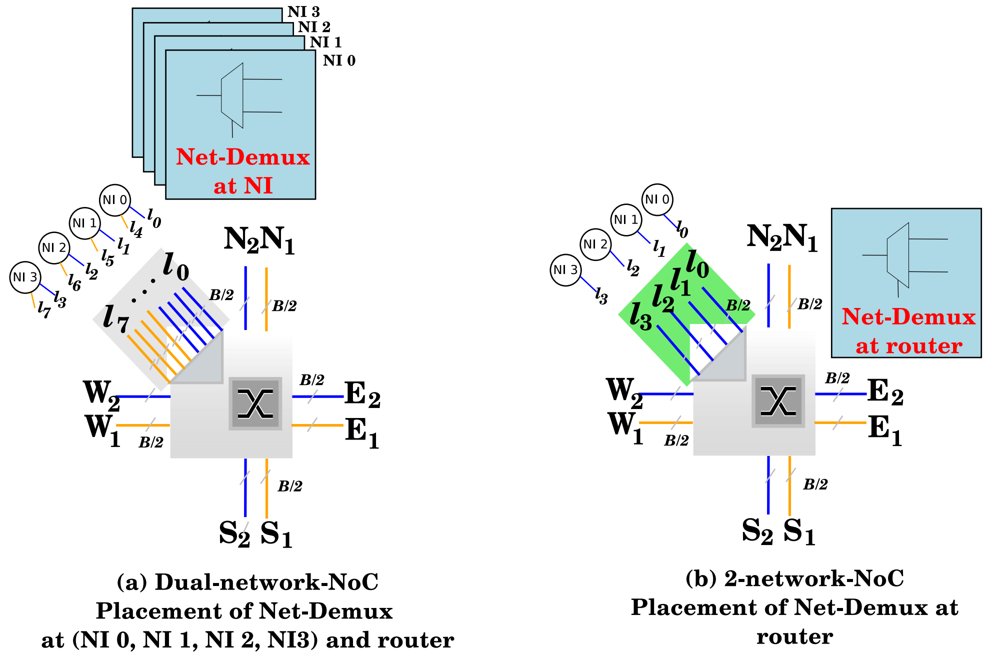


FIGURE 5.1: Placement of `Net-Demux` in (a) dual-network-NoC at NI 0, NI 1, NI 2, NI3 (b) 2-network-NoC at the router.

Therefore, placement not only changes the location of `Net-Demux` but also changes the architecture of `Net-Demux` hardware according to the functionality of the hardware unit where `Net-Demux` will be placed. In Chapter 4, we have proposed the placement at the routing unit of the router. In this chapter, we shall explore other placement possibilities within the router and compare them.

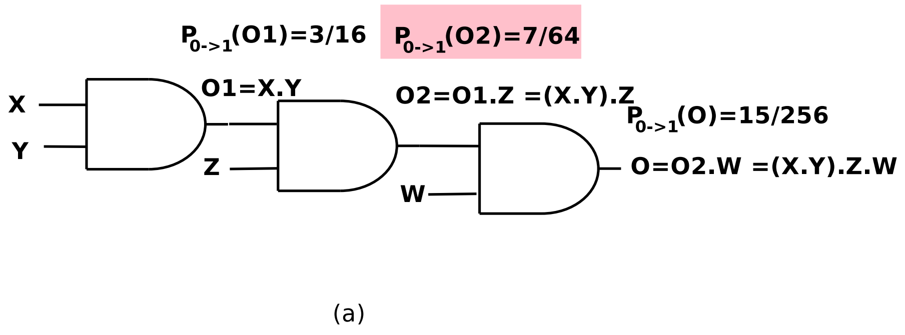
5.3 Placement Impact on Circuit Design Techniques

Messages are demultiplexed into different networks only after `Net-Demux`. Demultiplexing just at NI requires that parallel data streams be maintained at the router for sustaining performance gains. This shall require replication of all router components adding to cost. Another alternative is to demultiplex within the router, so that demultiplexing is part of router pipeline or switch allocation. Physical data flow is directly affected by the placement of `Net-Demux`. An inferior placement assignment might degrade the overall efficiency. Typical placement objectives include the improvement in timing delay, static and dynamic power for the overall gain in energy efficiency.

The dynamic power of a logic gate can be reduced by minimizing the physical capacitance and the switching activity. The physical capacitance can be minimized in a number of ways, including circuit style selection, transistor sizing, placement and routing, and architectural optimizations [151]. These are being part of the fabrication so beyond the scope of this thesis. The switching activity, on the other hand, can be minimized at all level of the design abstraction and is the focus of this chapter. In the following subsections, we discuss the parameters that affect the selection of placement location for `Net-Demux`.

5.3.1 Logic Restructuring

The topology of a circuit affects the overall power dissipation. The implementations of logic network O = X·Y·Z·W is possible in two alternate ways, as shown in Fig 5.2. Lets assume that



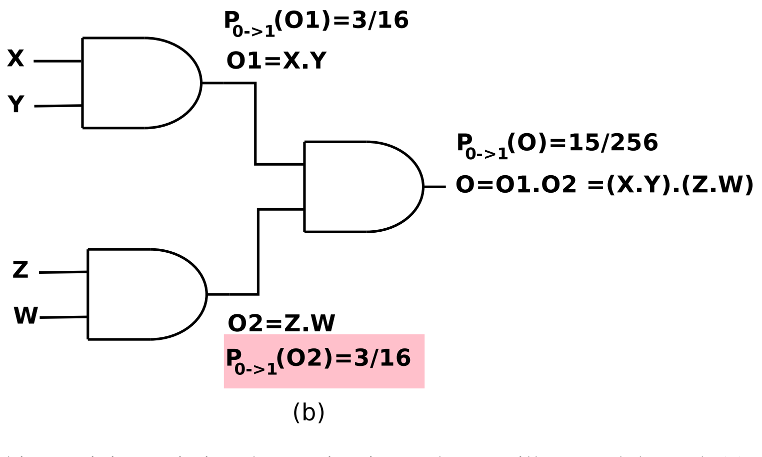


FIGURE 5.2: Switching activity variation due to circuit topology as illustrated through (a) chain structure, and (b) tree structure. The output transition probability is uniform, i.e., ($P_{1\ (X,\ Y,\ Z,\ W)}$ = 0.5) for all the inputs.

all primary inputs (X, Y, Z, W) are uniformly distributed (i.e., $P_{1\ (X,\ Y,\ Z,\ W)}$ = 0.5), i.e., P (X=1) = 0.5 and this holds good for all other inputs as well.

For an AND gate with input X and Y, the probability that the output is 1, i.e., $P_{1\ (X,\ Y)} = P_{1\ (X)} \times P_{1\ (Y)}$. The probability that output is 0, i.e., $P_{0\ (X,\ Y)} = 1 - P_{1\ (X,\ Y)}$. and the transition probability is

$$P_{(0\to1)\ (X,\ Y)} = P_{0\ (X,\ Y)} \times P_{1\ (X,\ Y)} \tag{5.1}$$

Likewise, the output signal transition probabilities $P_{0->1(O1)}$, $P_{0->1(O2)}$, $P_{0->1(O)}$ is calculated in Fig 5.2(a). The output signal transition probability $P_{0->1(O2)}$ for output O2 changes in a tree structure, as demonstrated in Fig 5.2(b). On comparing chain and tree topology, the results indicate that the chain implementation will have lower switching activity in intermediate outputs than the tree implementation as observed with random inputs. The lower switching activity during intermediate outputs is important as these intermediate signals may become the input of other circuits, therefore, they propagate the switching activity.

As `Net-Demux` integrates with different hardware modules of the NoC circuit, switching activity may significantly vary with placement of `Net-Demux`.

5.3.2 Input Signal Ordering

Another parameter affecting switching activity is the order of input signals. Fig 5.3 illustrates the impact of reordering of input signals on switching activity. Both the circuits in Fig 5.3 are identical in topology, but their output switching activity is different because of the input signal `X` is swapped with `Z`. The probabilities of input signal being 1 are $P_{(X=1)} = 0.8,\ P_{(Y=1)} = 0.2,\ P_{(Z=1)} = 0.1$.

1) **Circuit (a).** In the first circuit the output signal transition probability can be calculated as $(1 - 0.8 \times 0.2)\ (0.8 \times 0.2) = 0.1344$. The final output transition is $(1 - 0.1344 \times 0.1)\ (0.1344 \times 0.1) = 0.0132$.

2) **Circuit (b).** In the second case, the probability that a $0 \to 1$ transition takes place is $(1-0.2\times0.1)\ (0.2\times0.1) = 0.0196$. The final output transition is $(1-0.0196\times0.8)\ (0.0196\times 0.8) = 0.0154$.

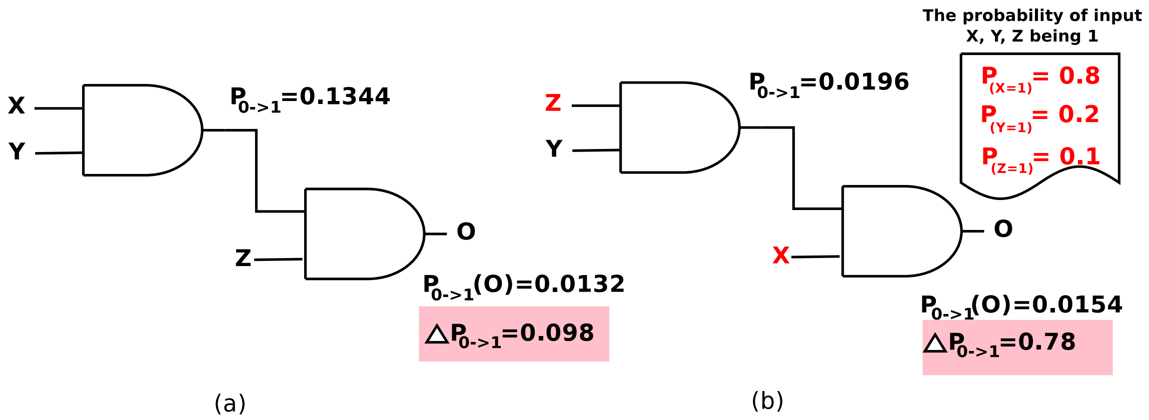


FIGURE 5.3: Reordering of input signals affects the output switching activity of the circuit (a) input are ordered as X, Y, Z (b) input order changes as Z, Y, X. The $\Delta P_{0->1}$ shows the reduction in output switching activity over intermediate output switching activity.

Reduction in signal transition rate at output vis-a-vis intermediate output is 0.098 for (a) and 0.78 for (b). We observe a substantial reduction in switching activity in the circuit (b) as compared to the circuit (a). So we conclude that it is beneficial to postpone the introduction of signals with a high transition rate [151]. A simple reordering of the input signals significantly reduces the signal transition rate (switching activity).

5.3.3 Timing Delay of Signal Path

A signal delivers from input pads to gate outputs and proceeds from the output pad to gate input [80]. So the delay of a path is the sum of interconnect and gate delays that make up the path. 'Timing delay' is referred to how much time a signal requires to reach from input to output pin of the circuit. The longest path that introduces maximum timing delay is known as 'critical path' of the circuit.

Delay along critical path is a significant parameter of circuit design as it decides the time period of the clock or, alternately, the maximum operating frequency of the circuit. For reliable operation of the circuit, difference between two clock arrivals should be more than the critical path delay. While integrating net-demux with the circuit, care should be taken that placement does not increase critical path delay. Placement of `Net-Demux` at various locations varies the following characteristics of signal paths

1) **Length of the signal path.** The integration of `Net-Demux` changes the length of the path itself.

2) **Total number of signal path.** If the total number of paths in the circuit is ρ_n and the number of paths in `Net-Demux` is q_n. The total number of paths of the hardware unit shall be changed due to addition of more q_n paths.

3) **Critical path delay may change.** The integration of `Net-Demux` may be along the critical path or the non-critical path. Due to addition of new paths of `Net-Demux`, the timing properties of the circuit may change.

Therefore, it is significantly important that

1) `Net-Demux` is integrated with a signal path that is not a critical path.

2) An optimal integration should avoid the critical path and also avoid any non-critical path that would become a new critical path. Violating longest path delay shall change circuit timing characteristics that may result in excess power dissipation.

Besides the path delay, the input signal ordering as per the switching activity needs to be considered during placement of `Net-Demux` on NoC. Though, we consider all these properties at the abstraction level of the micro-architecture. In the next section, we shall explore the impact on the architecture of `Net-Demux` with placement.

5.4 Architectural Implications of `Net-Demux` with Placement

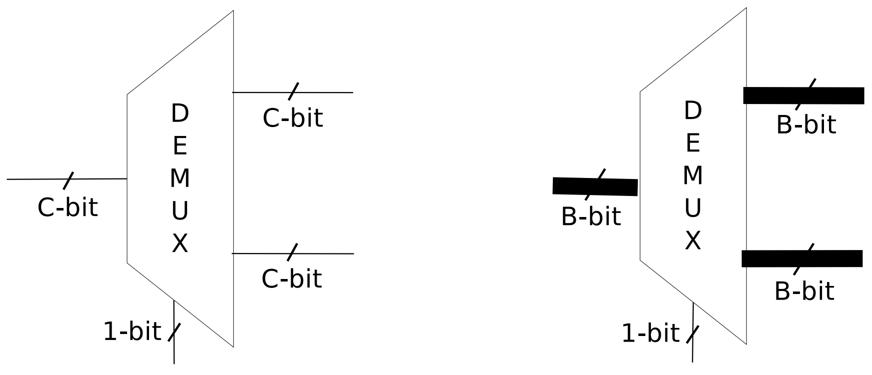
The `Net-Demux` can be placed in either control[2] plane or data[3] plane. We need to provide a provision in NoC hardware to separate the traffic between NoC networks through `Net-Demux`. Data plane has external data **I**nput/**O**utput (I/O), as well as a control I/O to/from the control planes. In Fig 5.4 (a) and (b), we have compared the variation in hardware implementation overhead[4] in Net-Demux with placement through demonstrating architectural variations in a single demultiplexer with control vs. data plane. A small[5] bus-width let's say C-bit input/output

[2]We shall synonymously use the term control plane or control path. Control planes refer to the part of the hardware that is responsible for generating control signals, i.e., coordinating the movement of packets through the data plane.

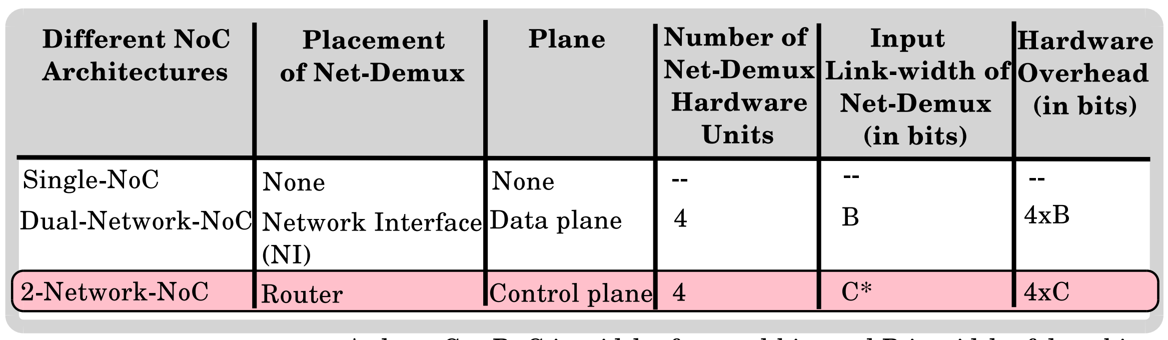
[3]Likewise, data plane or data path handles the storage and movement of packets. It comprises of a set of input and output buffers located in network interface as well as routers/switches.

[4]Placement also changes the architecture of the Net-Demux according to the functioning of the hardware unit where it is placed.

[5]3 and 4-bits for our proposed architecture.



(a) Control Plane (b) Data Plane



Different NoC Architectures	Placement of Net-Demux	Plane	Number of Net-Demux Hardware Units	Input Link-width of Net-Demux (in bits)	Hardware Overhead (in bits)
Single-NoC	None	None	--	--	--
Dual-Network-NoC	Network Interface (NI)	Data plane	4	B	4xB
2-Network-NoC	Router	Control plane	4	C*	4xC

*where C<<B, C is width of control-bits and B is width of data-bits

(c) Frontend Comparison of Control Plane vs Data Plane Placement

FIGURE 5.4: Demultiplexer in (a) control plane (b) data plane (c) control v/s data plane placement in single NoC, dual- and 2-network-NoC.

is sufficient to find the NoC network as compared to a large[6] bus-width input/output, let's say $B-$bit input/output, on placing demultiplexer in data plane. In Fig 5.4 (c), we have compared the hardware implementation overhead of data plane (for dual-network-NoC) and our proposed control plane (for 2-network-NoC) placement and observe a significant overhead[7] with data plane placement.

The static power and area are affected by placement as follows:

1) Placement in the control plane is independent of the network link-width.[8]

2) The benefit with control plane is $(I_{NI} \times B)/(I_R \times C)$ times more over data plane placement hardware overhead. Where $I_{NI} = 4$ is the number of NI links input to the router, and $I_R = 4$ is the number of inputs from other routers (for one NoC as another NoC has

[6]128/256/512-bits (128-bits for our proposed architecture).

[7]Large area and static power as compared to control plane because of a wide bus-width is used for input/output of Net-Demux hardware.

[8]Recent NoC architectures are using higher bandwidth networks (for example, a link width of 512 bits is required to sustain today's per-core bandwidth [46]). So the placement in data path significantly increases hardware overheads.

already separated traffic). The placement in control plane shall be $(4 \times B)/(4 \times C) = B/C$ times beneficiary as compared to data plane placement since link-width B>>C.

3) As the number of NoC networks increases, the width of select line as well as the number of output lines of `Net-Demux` increase. This shall be a huge overhead if the placement is in a data path. Contrary, the control path has a minor impact due to a lower input link-width C, which is sufficient to implement the functioning of `Net-Demux` with negligible overheads.

5.5 Hardware Implementations of `Net-Demux` with Placements

Placement causes the variations in circuit paths. The physical design of circuit significantly impacts power dissipation, area, and timing delays of a circuit. Incompatible placement degrades the overall efficiency, which cannot be offset even with intelligent routings. Placement of `Net-Demux` impacts the control and data path differently. As discussed in the previous section, placement in data plane significantly affects the area and static power, whereas placement in control path affects the switching power of the circuit. As static power saving is more significant for modern processors, we have restricted `Net-Demux` placement exploration to control path only.

5.5.1 Network Interface (NI)

Till date, all commercial [68–71] and academic [57] architectures consist of parallel NI and network links, therefore `Net-Demux` is placed at NI to distribute the traffic between multiple NoCs. **N**etwork **I**nterface (NI) is located between the core and router. Core messages enter into NI, and then into the router through NI links as shown in Fig 5.5. A router consists of eight core links and eight networks links for dual-network-NoC. The odd-numbered links constitute NoC_1 network whereas the even-numbered links represent NoC_2 network.

Messages are the logical unit of communication and may be arbitrarily long. They are divided into packets that are further segmented into fixed length FLITs (FLow control unITs). If the number of generated packets is k, then single packet size is $S = \frac{M-2}{k}$ and flit size is $N = \frac{M-2}{k \times f}$ for the f number of flits per packet. On receiving a message at NI, the flits are checked for the

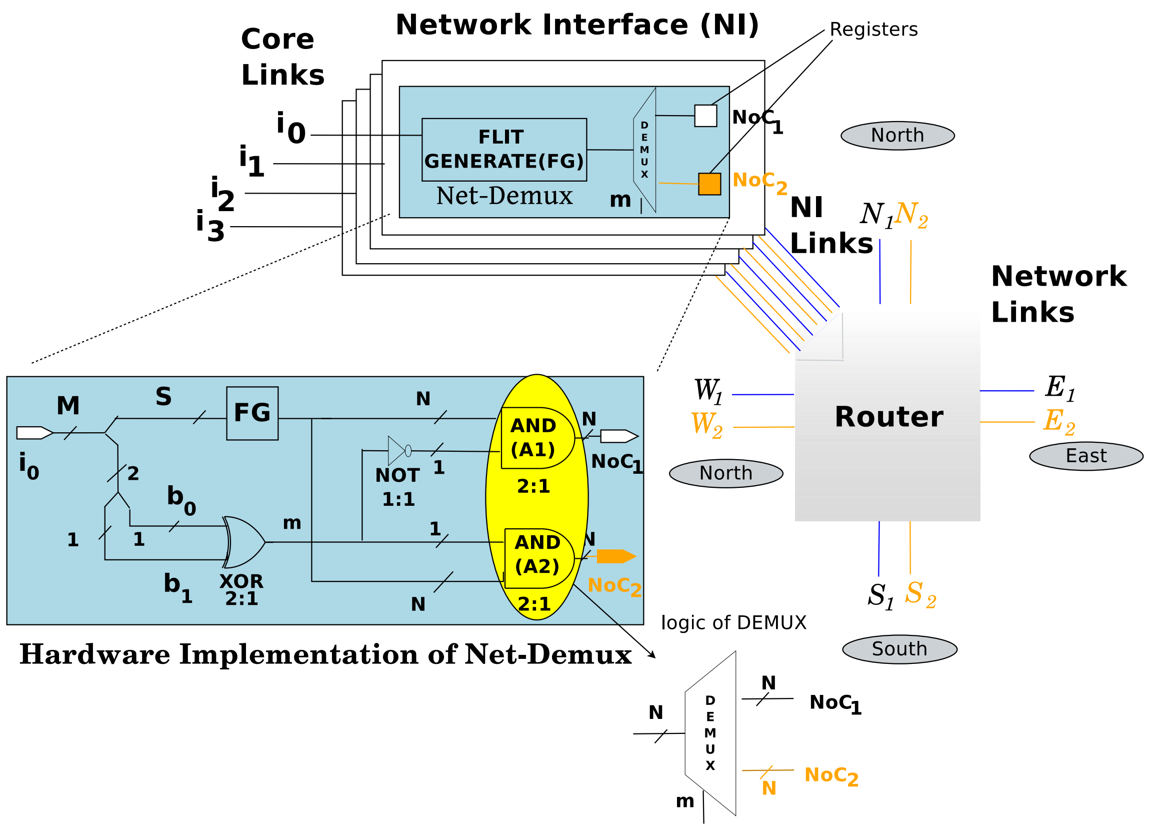


FIGURE 5.5: Network interface with `Net-Demux` hardware implementation details and interface with router for dual-network-NoC. FG refers to flit generator.

Control or Data packets since the Control and Data flits are generated separately. A Control packet is composed of one control flit. It consists of a coherence command and the memory address. On the contrary, data packets are made up of five[9] flits. It consists of a head flit that contains the destination address, three body flits and a tail flit that indicates the end of a packet and triggers a signal for generating next flit. In the case of data packet, the first flit is Head flit. The control packet is a single flit though it keeps all the required information of destination similar to head flit. So only control flit, and head flit keep the control and routing information. The Data and Tail flits follow the Head flits status to reach their respective NoC networks. These flits consist of payload, however, the last data flit is padded with zeros (if required).

Every message is prepended with two bits identifying the message class as per encoding specified in column 3 of Table 5.1. These bits are passed to inputs $i_1[0]$ and $i_2[0]$ of the `Net-Demux` circuit shown in Fig 5.5. These two bits are XORed. Output m of XOR gate is connected to two AND gates $A1$ and $A2$. The other input to these gates is a message. So whenever message belongs to `Control` or `Writeback Control`, $m = 1$ AND gate A2 passes message bits. This

[9]Five flits for our NoC architecture

TABLE 5.1: Flag bits setting for different message classes. The fourth column depicts *m* bit computed from these flag bits. Network selection is decided by the value of *m* bit.

Type	Message Classes	2-bit Encoding[a]	m[b]	NoC Network
1	Control	$b_0 = 0,\ b_1 = 1$	1	NoC$_2$
2	Writeback Control	$b_0 = 1,\ b_1 = 0$	1	
3	Request Control	$b_0 = 0,\ b_1 = 0$	0	NoC$_1$
4	Response Control			
5	Writeback Data			
6	Response Data			

[a]2-bit codes assigned
[b]m-flag that indicates message type

ensures the selection of NoC$_2$ for these messages. For other message classes, NoC$_1$ is selected.

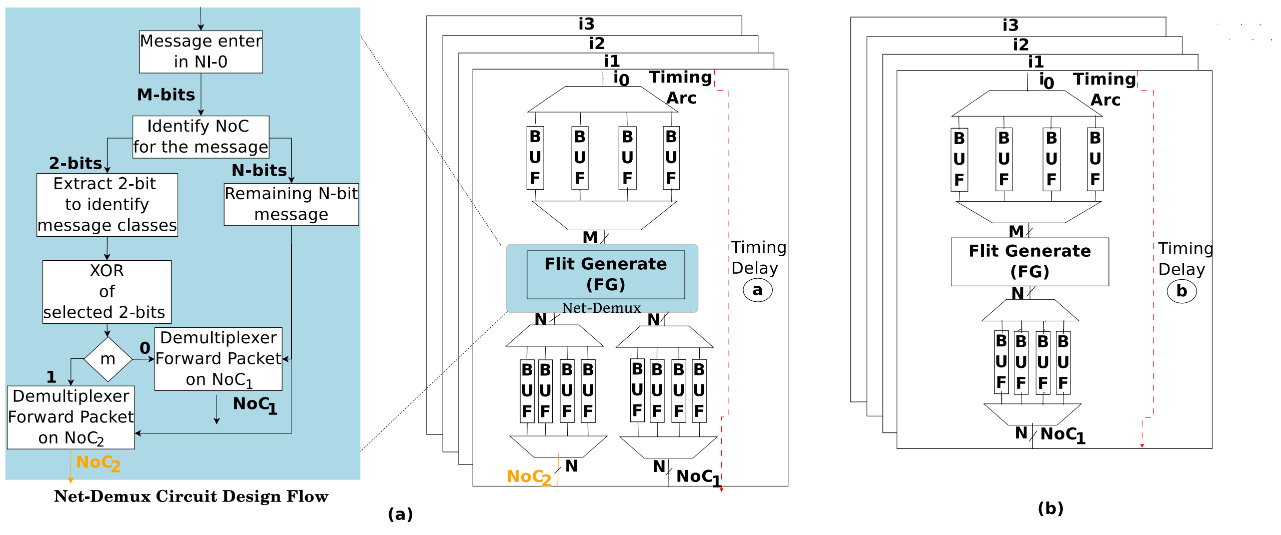


FIGURE 5.6: Comparison of Network Interface (a) with placement of `Net-Demux` in dual-network-NoC (b) without placement in single and 2-network-NoC. This comparison demonstrates the overhead in power, area and timing delay at NI with placement of `Net-Demux` as compared to without any `Net-Demux`. The circuit design flow of `Net-Demux` also demonstrates the complexity of placement over NI on comparing timing delay of both the circuits $a > b$.

The `Net-Demux` is integrated with **F**lit **G**eneration (FG) as shown in Fig 5.6 (a) as compared to single-NoC Fig 5.6 (b). Once the credit signal indicates the availability of empty VC, the FG check the counter(c), if it is zero, the first flit is checked for signal *m*. If *m* is true, the outport address of NoC$_2$ is assigned to flit, else NoC$_1$ address is allocated. The complete `Net-Demux` circuit design flow from the incoming of a message in NI to a selection of NoC network for the packet is shown in Fig 5.6 (a). Placement of `Net-Demux` at NI increases the critical path delay as NI is on the critical path itself. As a result, the timing delay with placement *a* is larger than the timing delay b without placement of `Net-Demux` at NI. Since the $a > b$, the placement of `Net-Demux` increases the critical path of the circuit.

5.5.2 Router

Conventional placement of `Net-Demux` is at NI that is part of the data plane. On the router, placement is possible at both control as well as data plane. In the datapath, the placement is possible at the input/output of the crossbar, but it is costly in area and power consumption as compared to control plane. The required number of `Net-Demux` hardware units shall be as many as the number of inputs to NoC_1 from the router, and area and static power shall be proportional to the data path width. Due to high hardware cost, we have not explored the placement of `Net-Demux` at crossbar. In the control plane, we propose placement at switch allocator of the router because it selects the outport on the crossbar for flit traversal.

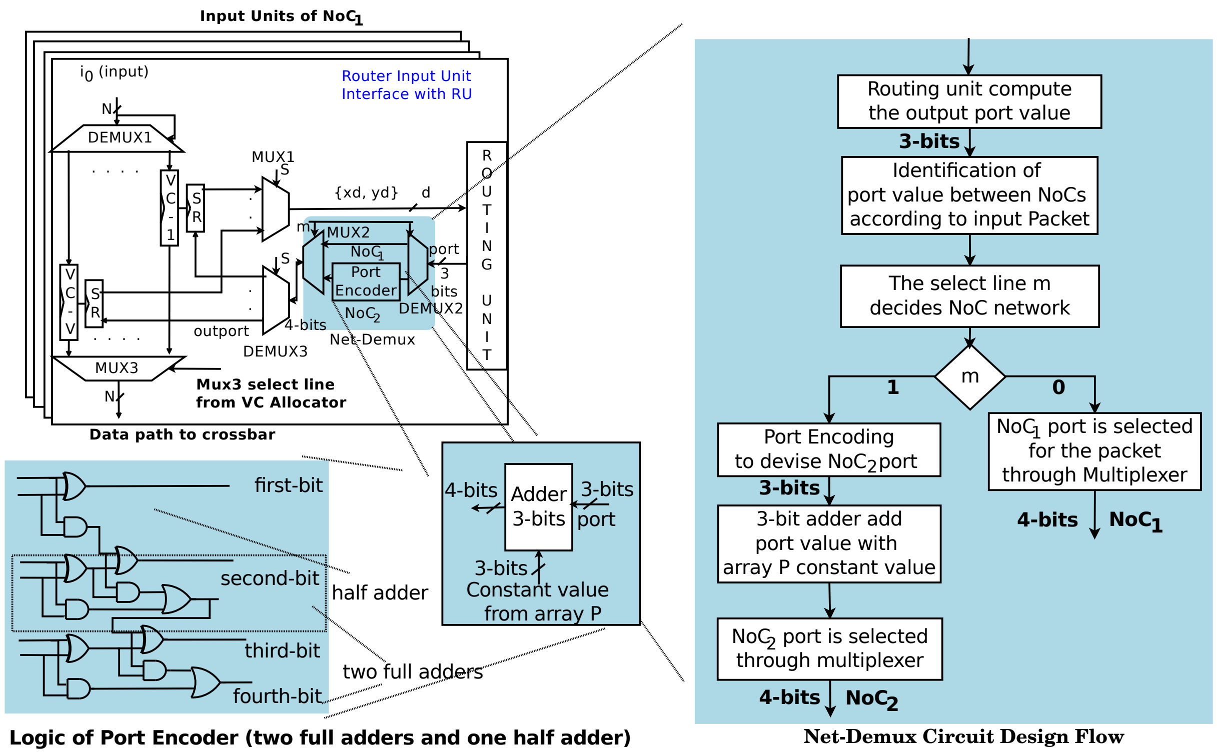

Input Units of NoC1
i0 (input)
Router Input Unit Interface with RU
N
DEMUX1
MUX1
S
VC-1
SR
{xd, yd}
d
m
MUX2
NoC1
Port Encoder
NoC2
port
3 bits
DEMUX2
ROUTING UNIT
VC-V
SR
outport
4-bits
DEMUX3
Net-Demux
MUX3
N
Mux3 select line from VC Allocator
Data path to crossbar
first-bit
second-bit
third-bit
fourth-bit
half adder
two full adders
4-bits
Adder
3-bits
3-bits
port
3-bits
Constant value from array P
Logic of Port Encoder (two full adders and one half adder)
Routing unit compute the output port value
3-bits
Identification of port value between NoCs according to input Packet
The select line m decides NoC network
m
1
0
Port Encoding to devise NoC2 port
3-bits
3-bit adder add port value with array P constant value
NoC2 port is selected through multiplexer
4-bits
NoC2
NoC1 port is selected for the packet through Multiplexer
4-bits
NoC1
Net-Demux Circuit Design Flow

FIGURE 5.7: Placement of `Net-Demux` between the input unit and the routing unit of the router. One half adder and two full adders shall be required to implement the logic of port encoder. Flowchart demonstrates the hardware implementation of `Net-Demux` placement after route computation of RU for the selection of either of NoC networks.

5.5.2.1 Routing Unit (RU)

Flits are divided into head, body, and tail flits. The least significant three bits of head flit store the information of the destination. This information is stored in status registers and is utilised during route computation. After computation of routing **S**tatus **R**egisters (SR) are updated with output port information. The port encoder of `Net-Demux` hardware unit converts this into four-bit outport. The port encoder is made of one half adder and two full adders. As in single-NoC, the possible directions for the router is eight (four network links and four NI) for 2-network-NoC. In the case of dual NoC, twelve directions are possible (eight network directions and four NI).

In 2-network-NoC, `Net-Demux` is placed at the routing unit of the router. Routing unit reads the status register to get the routing information for Head flit. As illustrated in Fig5.7 (a), two control lines read the destination address x_d and y_d from the status register and feed this information to the input of `Net-Demux` that checks the message class of current Head flit and assign its NoC network. The selection of NoC network is dependent on signal m_1. If signal m_1 holds the value zero, the outport of NoC_1 is selected by `Net-Demux`, else the outport of NoC_2 network.

The flowchart in Fig5.7(b) illustrates the control path of the routing unit. The dashed line shows the critical path delay of the control path of the circuit. Integration of `Net-Demux` increases the length of each path for routing unit.

5.5.2.2 Switch Allocator (SA)

Switch **A**llocator (SA) schedules the crossbar connections which are established between input and output ports for flit traversal in each cycle. Timing arc (red dash-line) exhibits a critical path which lies between three submodules: local arbiter, global arbiter, and masking logic as shown in Fig 5.8. The local arbiter selects a winner among the competing **V**irtual **C**hannels (VCs) at each input port. The output of local arbiter becomes the select line of MUX1 that selects only one wire consisting of outport information with respect to flit of winner VC.

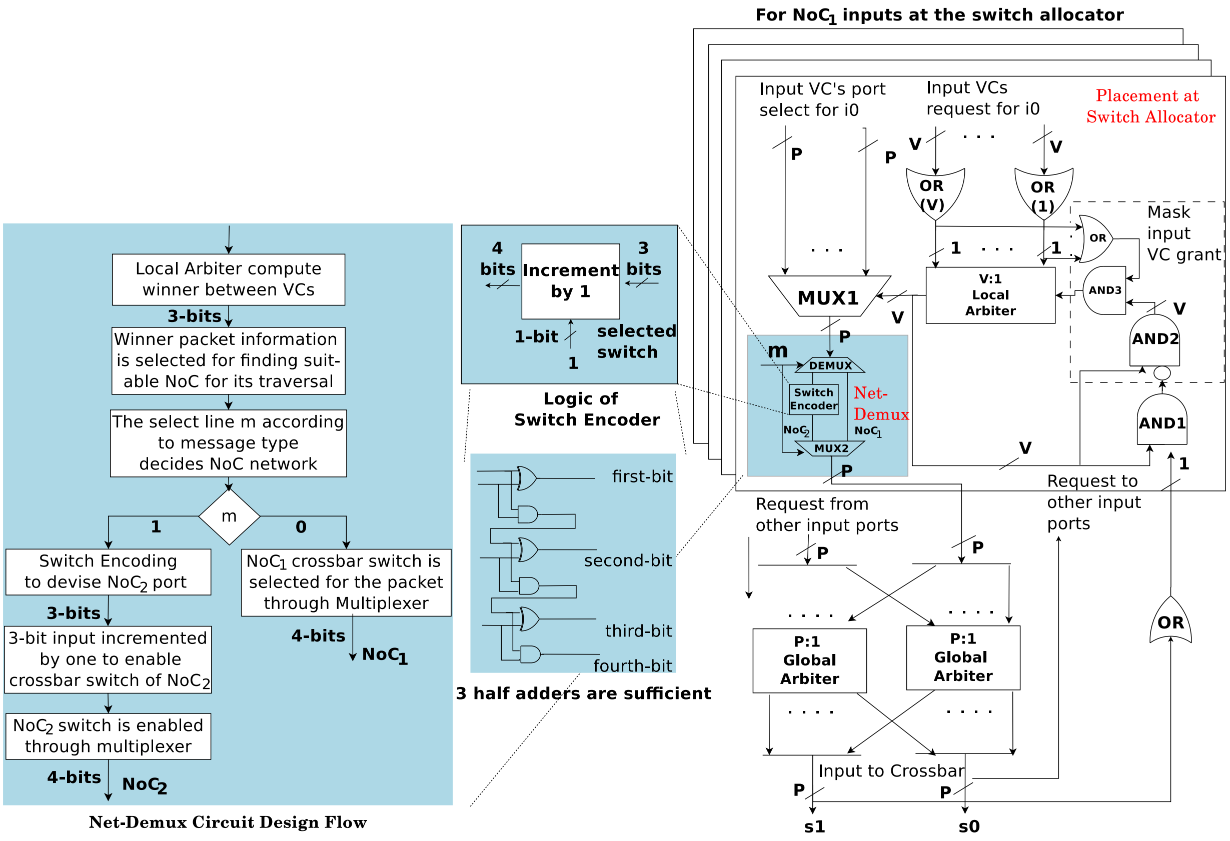

For NoC1 inputs at the switch allocator
Input VC's port select for i0
Input VCs request for i0
Placement at Switch Allocator
OR (V)
OR (1)
Mask input VC grant
MUX1
V:1 Local Arbiter
AND3
AND2
AND1
DEMUX
Switch Encoder
Net-Demux
MUX2
Request from other input ports
Request to other input ports
P:1 Global Arbiter
Input to Crossbar
s1
s0
OR
4 bits Increment by 1 3 bits
1-bit selected switch
Logic of Switch Encoder
first-bit
second-bit
third-bit
fourth-bit
3 half adders are sufficient
Local Arbiter compute winner between VCs
3-bits
Winner packet information is selected for finding suitable NoC for its traversal
The select line m according to message type decides NoC network
m
Switch Encoding to devise NoC2 port
NoC1 crossbar switch is selected for the packet through Multiplexer
3-bit input incremented by one to enable crossbar switch of NoC2
4-bits
NoC1
NoC2 switch is enabled through multiplexer
NoC2
Net-Demux Circuit Design Flow

FIGURE 5.8: `Net-Demux` placement at switch allocator of the router. Three half adders are sufficient to implement switch encoder logic. The flowchart demonstrates the selection of either of NoC networks with the placement of `Net-Demux`.

The `Net-Demux` is placed in between local and global arbiters. A 1-bit select line *m* does the selection of NoC network according to a type of message class as listed in Table-5.1. If *m* is '0', flit proceeds to NoC$_1$, else to NoC$_2$ network.

The competition between different inputs requesting for the same output physical channel is resolved by **G**lobal **R**ound-**R**obin (G-RR) arbiter. Each input bit processed through G-RR arbiters corresponding to each output channel. The grants generated by global arbiter are used to set up the crossbar control registers.

The output of global arbiter is also fed into the input of mask logic. Another input of mask logic is V-bit input line from the local arbiter. The masking logic activates the next VC request in lieu of recent VC.

5.6 Comparative Analysis of Placement

In this section, we shall compare the variation in the architecture of `Net-Demux` with placement. The first comparison is made in respect of hardware component and is shown in Table 5.2. Impact of placement on hardware metrics is compared in Table 5.3. In the first part, we discuss the placement at network interface with respect to placement at the router. Then, in the second part, we compare placement within the router, i.e., routing unit vs. switch allocator.

TABLE 5.2: Comparative changes in hardware components architectures with different placements of network selector.

Hardware Components	Network Interface	Routing Unit	Switch Allocator	Comments
Input Signals	N-bits input	K-bits input	K-bits input	$N >> K$
Demultiplexer	One with N-bits	One with K-bits	One with K-bits	Total required four
Encoder	None	Port Encoder	Crossbar Switch Encoder	Negligible overhead f
Adder	None	5 half adder (2 full and 1 half adder)	3 half adder	Switch allocator adder is more efficient
Multiplexer	None	One with K-bits	One with K-bits	Negligible overhead

TABLE 5.3: Gain in hardware metrics with different placement of network selector.

Hardware Metric	Network Interface	Routing Unit	Switch Allocator	Comments
Critical Path Delay	Increase path length	No impact	No impact	Router critical path decided by crossbar so no impact of other components
Minimum required units	4	4	4	NI input overheads are four times more
Power Overheads	$4 \times N$	$4 \times K$	$4 \times K$	NI overhead is much higher
Area Overheads	$4 \times N$	$4 \times K$	$4 \times K$	NI overhead is much higher

5.6.1 Network Interface vs. Router Placement

At the network interface, `Net-Demux` is placed in the data plane whereas at the router the placement is possible at the control plane. The placement at the control plane of the router is beneficiary over network interface placement as the data plane placement of network interface requires more area and power overhead for the implementation of `Net-Demux` as we can see in Table 5.2. This is reflected as overhead in area and power hardware metrics in Table 5.3.

5.6.2 Routing Unit vs. Switch Allocator Placement

The `Net-Demux` can be placed on the router either at the routing unit or at switch allocator. The architecture of `Net-Demux` varies with placement. Each functional unit of the router performs a specific task accordingly, `Net-Demux` inputs and outputs vary. For example, the routing unit computes the outport for the incoming header flit. On the other side, switch allocator resolves the contention between channels and enables the switch of the crossbar that forwards the flit across the crossbar. In Table 5.2, we can see the encoder for both the placement shall be different, and accordingly, the minimum number of required half adders is different. The architecture of `Net-Demux` is less complicated for switch allocator placement as compared to routing unit placement. Though, the inputs signals are of the same width for these placements.

5.6.3 Impact on Digital Circuit Characteristics

Though, we have not explored backend impact of placement in detail. We can presume impacts with following brief discussion.

1) Input Signal Ordering. The placement of `Net-Demux` at the switch allocator is more efficient compared to routing unit placement. This is concluded from the digital circuit characteristics of **input signal ordering**. It is beneficial to postpone the introduction of input signals with a high transition rate, as discussed in Subsection 5.3.2. Thus, the output of the routing logic becomes the input of `Net-Demux` because signal passes through routing algorithm then becomes the input of `Net-Demux` the input signal transition rate is higher as compared to the signal for switch allocator. At the switch allocator signal just passes through one multiplexer and become the input for `Net-Demux`. Such input signal ordering becomes the deciding factor, and it is beneficiary to introduce a low transition rate signal early for the `Net-Demux`.

2) Timing Delay. Critical path delay is affected by network interface placement, as discussed in Table 5.3. The router microarchitecture is the four-stage pipelined architecture. The critical path delay is dominated by the crossbar of the router. The placement of the `Net-Demux` does not add to the critical path delay of the crossbar of the router. Since the frequency does not need to change on placement, so there shall not have any impact on data arrival in each cycle of the router.

3) Topology of Circuit. The topology of the circuit is decided by the optimisation algorithms used by the backend tools. Therefore, with placement, the topology of the circuit changes as discussed in Subsection 5.3.1 and hence switching activity[10] that is the primary factor affecting the dynamic power.

5.6.4 Impact on Scalability

Latest technology breakthrough makes NoC more fault-prone, more severe congestion, traffic hotspot, and thermal issues. So routings of these processors are more complex. The complex routing circuits have high switching activity because of the more number of digital units in every path. It is better to place `Net-Demux` at switch allocator of the router.

5.7 Experimental Results

In this section, we discuss our experiment for different placements. The power and changes in the signal paths are evaluated through synthesis of the circuit. For the dynamic power comparison, we have extended Gem5 and Garnet simulator for comparing the placement of `Net-Demux` with PARSEC benchmarks.

TABLE 5.4: Hardware Synthesis Parameter Configuration

Parameters	**Configuration**
Synthesis Tool	Synopsis Design Compiler
Hardware Language	System Verilog
Hardware Design	**R**egister **T**ransistor **L**evel (RTL) synthesis
Cell Technology	**L**ow **V**oltage **T**hreshold (LVT)
NoC Networks	Single-NoC, dual- and 2-Network-NoC
Nanometer Technology	32 nm
Clock Frequency	2 GHz

[10]Switching activity has two components 1) a static component – the function of the logic topology, and 2) a dynamic component – the function of the timing behavior (glitching). We are not detailing dynamic component as it is out of the scope of the thesis.

5.7.1 Synthesis Result Analysis

The configuration of hardware synthesis parameters is listed in Table 5.4. We have done Register Transistor Level (RTL) synthesis to evaluate the hardware area, power and critical path delay of the RU (Routing Unit) and SA (Switch Allocator) placement of `Net-Demux` in 2-network-NoC and this is compared with dual-network-NoC. We have written these hardware RTL designs using system verilog. We have used 32nm standard-cell libraries for synthesis using LVT[11] cell technology with Synopsys Design Compiler. Target clock frequency was set to 2Ghz.

A number of signal paths (execution paths) are compared in respect of slack values. Table 5.5 shows a different number of execution paths and respective variations in slack values. When we compare slack variations and number of paths between NI, RU, and SA, we find a significant difference between the number of paths and slack values for different placements of NI. These slack values are measured in nanoseconds (*ns*).

TABLE 5.5: Path Slack variations in different execution paths due to `Net-Demux` placements.

Placement of `Net-Demux`					
at NI[a]		at RU[b]		at SA[c]	
# Paths	ΔSlack[d] (10^{-3})	# Paths	ΔSlack (10^{-2})	# Paths	ΔSlack (10^{-2})
32	0-0.1	8	0.36-0.4	32	0.5-0.6
4	0.32-0.35	8	0.45-0.56	8	0.6-0.7
14	0.78-0.88	20	0.62-0.70	10	0.11-0.14
		14	0.72-0.78		

[a]Network Interface
[b]Routing Unit
[c]Switch Allocator
[d]The unit of slack measurement is picoseconds.

A slack shows the permissible delay of a cell activity without delaying overall circuit output. So path slack is the difference of time when data arrives, and data is required. When the data arrival coincides with the time this data is required, slack is zero (ideal). However, the paths that are close to zero slack values are critical. If a higher number of paths in a circuit are close to zero, a slight variation in the slack may render the circuit unreliable at times. The time when a signal arrives can vary due to variation in input data, variation in temperature and voltage, manufacturing differences in the exact construction of each part. Hence the design

[11]Low Voltage Threshold [117]

should ensure that despite these variations, all signal will arrive neither too early nor too late. The positive slack implies the arrival time of a cell may be postponed by slack value without affecting the overall delay of the circuit. It shows that data arrives before being required by the circuit. So the researchers emphasize on the structuring of a schematic for fewer equally critical paths. Instead of looking at the slack of just the most critical path, the design should consider the entire distribution of slacks [81]. The path slack value for NI is of the order of 10^{-3} *ns* which is very close to zero slack as compared to RU and SA. Rest of the positive slack values reduces the risk of violating timing delay of a circuit [154].

Likewise, the endpoint slack is an observation point at the end of a path. The primary output or scan flipflop referred to as an endpoint [82]. Each endpoint is associated with a path delay distribution. Due to the complexity of finding all paths to an endpoint, only least slack path to each endpoint is considered.

TABLE 5.6: Endpoint Slack variations for different `Net-Demux` placements.

Placement of `Net-Demux`					
at NI[a]		at RU[b]		at SA[c]	
# Paths	ΔEP-Slack[d]	# Paths	ΔEP-Slack	# Paths	ΔEP-Slack
256	0-0.042	144	0-0.05	32	0-0.5
8	0.24-0.31	948	0.37-0.42	8	0.32-0.37
4	0.32-0.36			1004	0.38-0.42
1260	0.37-0.42				

[a]Network Interface
[b]Routing Unit
[c]Switch Allocator
[d]The unit of endpoint slack measurement is picoseconds.

Table 5.6 shows the number of endpoints with the least slack divided across the entire cycle period for the clock. Execution paths that have positive sufficient slack values are safe paths as the probability of violating the delay is lower. The NI has 256 paths, RU has 144 paths, and SA has 32 paths that are close to zero endpoint slack value. It shows that SA is less sensitive for slack variation and uncertainty. These variations in slack values also cause variation in critical path delay, which is important because the clock frequency of a circuit is decided by its critical path.

5.7.1.1 Area

The area of different components of a circuit for various `Net-Demux` placements is compared in Fig 5.9. The total area is the sum of four factors: Combinational, Noncombinational, Net

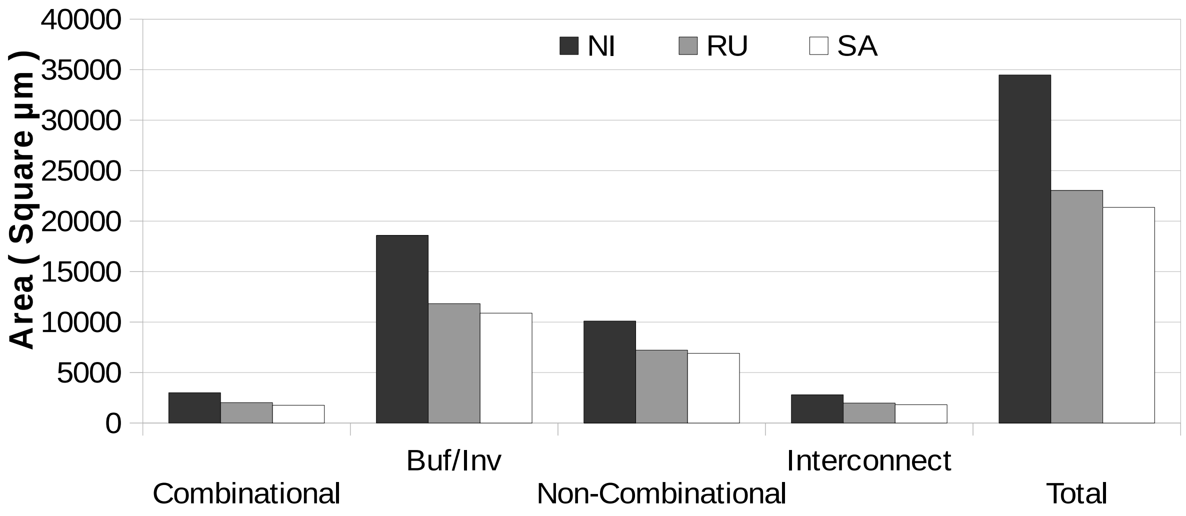


Figure 5.9: Area comparison

Interconnect area, and Buffers (Buf). The area due to logic cells in design is shown by the combinational logic gates like ANDs, ORs, etc., whereas the noncombinational factors are registers. The third factor affecting the area is the wires connecting these cells that is net interconnect area defined/computed by the library of wire load models. The buffer is the primary contributor to area as compared to other NoC components. The total gain [12] in the area is 39% in SA over NI and 36% in RU over NI placement.

5.7.1.2 Energy Analysis

The energy consumption is proportional to the network capacitance ($E \propto C_{Net}$). Higher values of network capacitance indicate higher energy consumptions. Placement of `Net-Demux` significantly affects the energy consumption of dual NoCs, and our proposed optimally sliced NoC. For example, 4050 networks in NI and 2795 networks in RL have 0-140 picofarad capacitance, which is significantly higher than SA, as shown in Table 5.7. On comparing the network capacitance, we can infer that `Net-Demux` placement at switch allocator is more energy efficient.

[12]The actual area advantage would be more for SA over NI. We get these area results when we have skipped the virtual channels and virtual networks in the circuit.

TABLE 5.7: Variation in net capacitance depict the energy consumption for various placements of `Net-Demux`.

Placement of `Net-Demux`					
at NI[a]		at RU[b]		at SA[c]	
# Nets	Capacitance[d]	# Nets	Capacitance	# Nets	Capacitance
4050	0-140	2795	0-140	2658	0-120
1	950-1050	1	950-1055	1	950-1055

[a]Network Interface
[b]Routing Unit
[c]Switch Allocator
[d]The unit of capacitance is picofarad.

Thus, with the help of Table 5.7, we conclude that `Net-Demux` placement significantly affects the energy efficiency[13] of a circuit.

5.7.1.3 Power and Critical Path Delay

We compare `Net-Demux` placements at NI in dual-network-NoC with RU and SA placement in 2-network-NoC. These placements are compared in respect of the area, power, and critical path delay. The area is the total of combinational, non-combinational, net interconnect and buffers area.

The total power dissipation is the static/leakage power and switching/dynamic power. The leakage power is dependent on the static current flows from voltage supply to ground in the absence of switching activity. It consists of (dis)charging of the capacitor and short circuit power. It varies according to the switching activities of the circuit. Critical path delay is also an important design metric. It affects the arrival of data to the input of the circuit. It determines the NoC frequency by estimating the worst path taken by the data.

We normalized the results of placements with respect to single-NoC by considering all values of design metrics are considered as ‘1’ for single-NoC to estimate the gain with dual-network-NoC and 2-network-NoC.

Table 5.9 is drawn from Table 5.8 to show the percentage improvement in the area, power, and delay for different placements (in row2) whereas row3 of the table depicts the architectures without `Net-Demux` placement. The ‘+’ sign indicates the advantage, ‘-’ shows the drawback.

[13]These results are validated when we evaluate energy gains in full system simulation with PARSEC benchmark for SA with respect to NI and RU.

TABLE 5.8: Area-power and delay comparison relative to single-NoC for different placements (area-power and delay of single-NoC is normalized to one unit time).

Hardware Metric		Placement of `Net-Demux`					
		NI[a]		RU[b]		SA[c]	
		NoC					
	single	dual-net[d]	2-net[e]	dual-net	2-net	dual-net	2-net
Area	1	0.52	0.47	0.68	0.55	0.80	0.64
Total Power	1	0.51	0.48	0.66	0.56	0.89	0.68
Static Power	1	0.50	0.47	0.61	0.52	0.75	0.59
Dynamic Power	1	0.51	0.48	1.13	0.96	1.46	1.04
CP[f] Delay	1	1.08	1	1.13	0.96	1	0.67

[a]Network Interface
[b]Routing Unit
[c]Switch Allocator
[d]dual-network-NoC
[e]2-network-NoC
[f]Critical Path Delay

TABLE 5.9: Improvement in area-power and delay for different placements.

Hardware Metric	% Improvement in					
	dual-network-NoC		2-network-NoC		2-network-NoC	
	`Net-Demux` placement at					
	NI[a]		RU[b]		SA[c]	
	gain over		gain over		gain over	
	single	2-net[d]	single	dual-net[e]	single	dual-net
Area	+48	-10	+45	+19	+36	+20
Total Power	+49	-6	+44	+15	+32	+24
Static Power	+50	-6	+48	+15	+41	+21
Switching Power	+49	-6	+4	+15	-4	+29
Critical Path Delay	-8	-8	+4	+15	+33	+33

[a]Network Interface
[b]Routing Unit
[c]Switch Allocator
[d]2-network-NoC
[e]dual-network-NoC

The advantage in the area and static power with all three placement at NI, RU, and SA are quite high over respective single-NoC. Because the flit-width in dual-network-NoC and 2-network-NoC are half of the single-NoC and the buffer size is determined in the proportion of flit-width. A significant gain is observed as buffers are the primary component of area occupation and static power consumption [73].

However, the downside is perceived in the design metrics of NI placement in dual-network-NoC over 2-network-NoC. The number of core links is eight, whereas 2-network-NoC has only four core links. Less number of links imply less number of buffers. Therefore, NI placement in dual-network-NoC has more area and static power as compared to 2-network-NoC.

The area and power overhead of `Net-Demux` are the same for either of RU and SA placements of 2-network-NoC, though they are different than NI placement of dual-network-NoC. Former places `Net-Demux` in the data plane whereas later places it in the control planes of the router. So overall gain in the area and static power is observed with 2-network-NoC placement over dual-network-NoC.

Adding `Net-Demux` at Network Interface (NI) shall increase critical path delay as shown in Table 5.9. The data arrival due to NI placement in dual-network-NoC is delayed by 8% over single-NoC and 2-network-NoC. However, the increase in critical path delay at RU and SA does not impact the overall router frequency. As the router has a pipelined[14] architecture, its clock frequency shall not be affected.

Different placements vary switching power because of the variation in input and output switching activity. Except for the SA placement in 2-network-NoC over single-NoC, switching power gain is observed with rest of the placements. As these results are calculated with the default wire load[15] model of low traffic, we evaluate the results with a larger workload of PARSEC benchmark as well.

TABLE 5.10: Configuration of Simulation Parameters

Parameters	**Configuration**
Simulator	Gem5 (Full System Simulation)
NoC simulator	GARNET
Cores	64 cores
ISA[a]	ALPHA
Topology	8×8 Mesh
NoC Networks	Single-NoC, dual- and 2-Network-NoC
Routing	Dimension Order XY
Caches	Three levels of cache hierarchy
Cache Coherence[b]	Directory-based MESI protocol
Power	ORION 2.0
Benchmark	PARSEC (medium workload, 16 threads)

[a]ISA-Instruction Set Architecture
[b]For more details, refer to Appendix B.

[14]The crossbar of the router dominates critical path of the router.
[15]Sometimes the default wire load model shows incorrect dynamic power.

5.7.2 Result Analysis with Benchmarks in Full System Simulation

To validate our proposal under larger workload, we compare the energy efficiency of router placement with NI placement using PARSEC benchmark. For full system simulation, we have used Gem5 simulator integrated with GARNET NoC simulator. A Linux 2.6.27 kernel image is booted with ALPHA instruction set architecture. For power results, GARNET is integrated with ORION.

The simulation configuration[16] parameter details are listed in Table 5.10. The volume of messages varies across different message classes as well as different applications of PARSEC benchmark. A specific NoC is assigned to each message class during inter-core communications as we get the static[17] message distribution with our initial profiling with *canneal* benchmark.

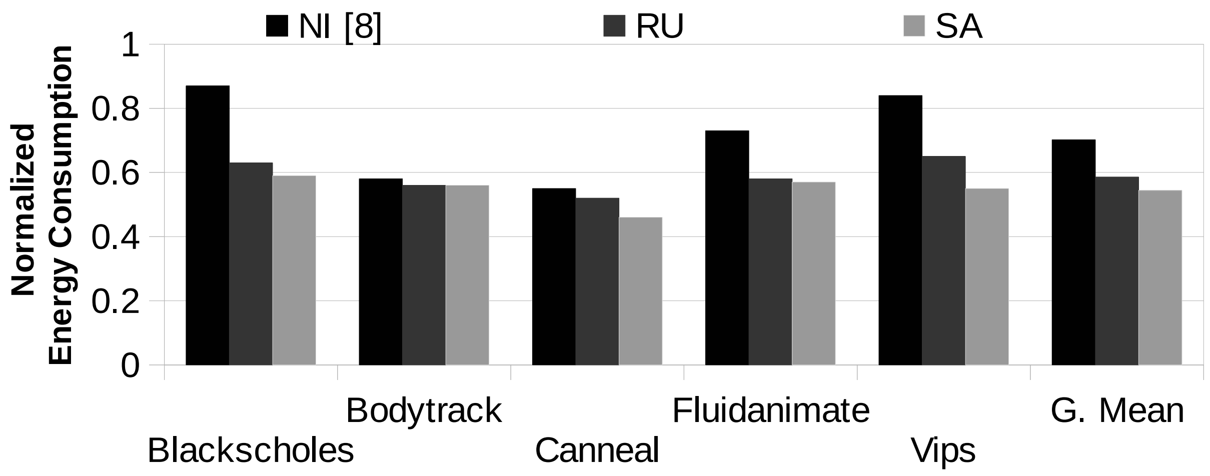


FIGURE 5.10: Energy consumption with different placements for communication-intensive PARSEC benchmarks. Here energy consumption of S-NoC is normalized to 1 unit.

The energy efficiency of distinct placements with communication intensive PARSEC benchmarks is shown in Fig 5.10. The rightmost bar shows the geometric(G) mean[18] of different placements. It shows that under PARSEC workload conditions SA, RU, and NI [8] placements are 46%, 40% and 30% efficient over single-NoC. Also, we observe SA and RU are approximately 33% and 26% more energy efficient over NI.

[16]For more details of Gem5 simulator, memory, and NoC parameters refer to Table 3.1, 3.2 and Table 3.3 of Chapter 3 (Page 58).

[17]Refer to Chapter 4 (Page 98).

[18]A geometric mean is a suitable measurement to summarize the normalized values of benchmarks [153]. The geometric mean of N values is the $(1/N)th$ power of the product of all values.

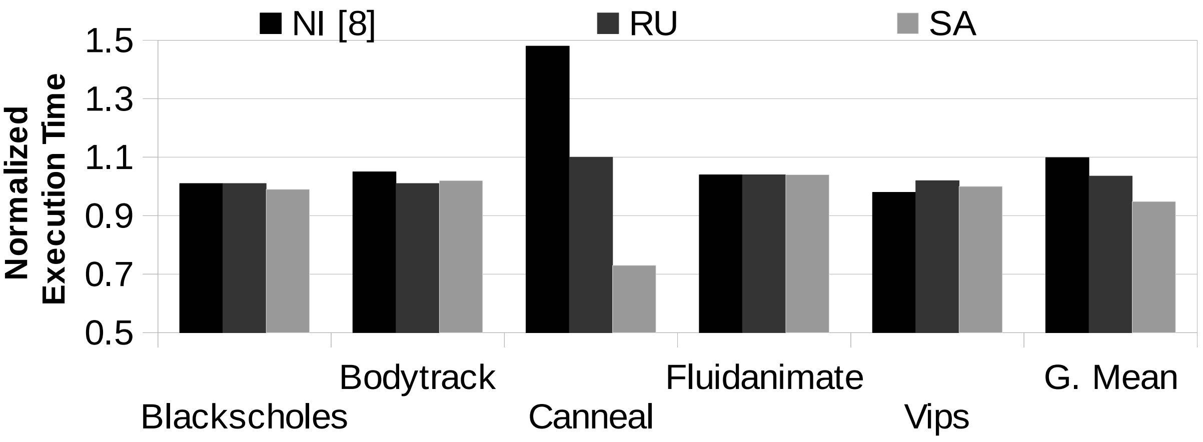


FIGURE 5.11: Execution time for different placements (the execution time of Single NoC is normalized to one unit time).

We also compare the execution time for the placement at NI [8], RU and SA (please refer to Fig 5.11). Minor degradation of 10% and 4% is observed in execution time for NI and RU. However, 6% improvement is achieved with SA over single-NoC, and also RU and SA placements are 6% and 14% efficient over NI.

5.8 Inferences

Different multiple dual- and 2-network-NoC (our proposed architecture) require to place a network selector to distribute traffic between NoC networks. We named network selector as network demultiplexer (`Net-Demux`) since it demultiplexes input messages into one of the NoC networks. In this chapter, we propose `Net-Demux` placement at switch allocator of the router for 2-network-NoC and compare it with the routing unit placement on router and conventional **N**etwork **I**nterface (NI) placement of dual-network-NoC. We infer that

1) placement of `Net-Demux` at NI comes into the data plane. On router, placement is possible at both control plane and data plane. As we have a choice at the router, we have selected the hardware units of control planes that are routing unit and switch allocator. Since placement modifies the architecture of `Net-Demux` according to the functionality of hardware units, we observe switch allocator placement has the lowest hardware implementation overhead in `Net-Demux` architecture as compared to routing unit and NI placement.

2) placement also impacts digital circuit characteristics such as input/output wire-width and critical path delay of the circuit. The placement in control plane requires a small bus-width of wires as well as small size of respective registers as compared to data plane `Net-Demux` hardware unit. Similarly, placement on the control plane of router does not impact critical path delay as the router is a pipelined architecture. Contrary, `Net-Demux` at NI increase critical path delay up to the length of the `Net-Demux` hardware critical path.

3) on comparing synthesis results, we observe significant benefits with switch allocator placement in a number of execution paths, path delay, area, and energy as compared to conventional placement. Switch allocator placement improves 21% static power, 29% dynamic power, and 33% critical path delay of the circuit over conventional placement.

4) with PARSEC benchmark, the `Net-Demux` placement at switch allocator, routing unit, and network interface [8] is 46%, 40% and 30% efficient over single-NoC. The placement at switch allocator and routing unit are approximately 33% and 26% more energy efficient over NI. The execution time with switch allocator placement improves 6% over single-NoC. Subsequently, the routing unit and switch allocator placement improves execution time 6% and 14% over NI. Thus, hardware synthesis and PARSEC benchmark results favor to place Net-Demux at switch allocator of the router for 2-network-NoC.

In the next chapter, we shall explore software level issues for 2-network-NoC. Traffic distribution is the next essential part for dual networks NoC architectures. We have briefly explored static traffic distribution for 2-network-NoC in Chapter 4 (Page 98) with a single benchmark of PARSEC. We shall analyze static traffic distribution impact on power-performance efficiency through a case study with general purpose PARSEC applications. Such case study with general purpose applications of PARSEC benchmark shows a significant impact of traffic distribution on power-performance and energy efficiency of the network. To mitigate the limitations of static traffic distribution, we shall propose a runtime adaptive traffic distribution in the next chapter.

Chapter 6

Runtime Adaptive Traffic Distribution

In Chapter 4 and Chapter 5, we have explored hardware-based approaches to improve power or energy efficiency of NoC. In this chapter, we shall explore traffic distribution at the software level of the NoC. In Chapter 4 (Page 98), we have defined traffic distribution policy on the basis of the experiment on *Canneal* benchmark. Static approach of Chapter 4 transmits `control` and `writeback control` messages through one NoC network and the rest through another NoC network. In Chapter 5, we have explored suitable placement of NoC (NoC_1 or NoC_2) selector hardware unit on router for 2-network-NoC. It follows the same traffic distribution policy as devised in Chapter 4.

This chapter highlights the limitations of static traffic distribution and proposes solutions in terms of adaptive-traffic distribution, which can adapt itself to runtime dynamics of messages contrary to the static approach. Modern chip multiprocessor applications are divergent in message volume[1] and traffic patterns, for example, web traffic is more bursty in nature (asymmetric request-response) than video traffic (symmetric). Different traffic patterns vary the volume of control and data messages. Static traffic distribution cannot yield good results for different traffic patterns as are encountered in general purpose multiprocessor chips.

Static traffic distribution rules are based on initial profiling of network traffic at design time according to overall traffic statistic. Static traffic distribution is popular for multiple NoCs because it is simple and easier to implement. It is very efficient for application specific

[1]The number of messages

processors. But for general purpose processors, it may show poor power-performance tradeoff for various applications.

In this chapter, we propose an adaptive on-the-fly traffic distribution that automatically intercepts NoC traffic communication and distributes traffic accordingly to improve network performance. These techniques consider underlying NoC architecture, available resources and their utilization, data characteristics, especially in terms of criticality to estimate the underutilization (overutilization) of one NoC (another NoC) network.

Our key contributions are:

1) We identify the limitations of conventional static traffic distribution for multiple NoCs through a case study on the impact of static traffic distribution on power-performance and energy efficiency with PARSEC benchmark.
2) We propose a novel adaptive on-the-fly traffic distribution to overcome the limitations of static message distributions.
3) To automate the adaptation, we also propose a policy for traffic distribution through analysis of fine-grained cache state messages.
4) The proposed technique is evaluated against two existing techniques: static traffic distribution (Chapter 4) and single-NoC [73]. Our experimental results show the improvement in energy efficiency and link utilisation of multiple NoCs with proposed adaptive approach.

6.1 Static Traffic Distribution

The static policy of traffic distribution is based on a priori knowledge of NoC architecture and traffic analysis. Static traffic distribution is consequential only when the best probabilistic distribution of traffic is devised resulting in efficient resource allocation. The static message distribution distributes the following message classes[2] on dual NoC networks[3].

1) `Control` (C_C)
2) `Request Control` (R_{Q_C})
3) `Response Control` (R_{P_C})
4) `Writeback Control` (W_C)
5) `Response Data` (R_{P_D})
6) `Writeback Data` (W_D)

The static message distribution for dual NoC network classifies the message classes in two sets. One that is sent on NoC_1 and another is forwarded on NoC_2. This non-overlapping distribution of message classes on both NoC networks does not change with time.

The possible number of ways[4] to distribute six different subclasses of messages on 2-network-NoC is as computed in the following expressions (1) - (3).

$$(NoC_1 : NoC_2) = \text{if} \begin{cases} (1:5) \rightarrow {}^6C_1 \rightarrow 6 & \cdots(1) \text{ /*Of six classes of messages, one is} \\ (2:4) \rightarrow {}^6C_2 \rightarrow 15 & \cdots(2) \text{ transmitted via NoC}_1 \text{ and rest via} \\ (3:3) \rightarrow {}^6C_3 \rightarrow 20 & \cdots(3) \text{ NoC}_2\text{*/} \end{cases}$$

As $C_r^n = C_{n-r}^n$, and there are only two physical channels in dual-network, C_4^6 shall generate the same combinations as C_2^6 albeit role of NoC_1 and NoC_2 shall be reversed. As a result, the total number of distinct combinations[5] are 41 only. For complete PARSEC benchmark suite consisting of 13 applications, a total number of $13 \times 41 = 533$ full system simulations are needed for analyzing the impact of static message distribution on 2-network-NoC.

[2]`Control` (C_C) messages are associated with invalidation and upgrade events. `Request Control` (R_{Q_C}) messages are generated at shared caches for data block replacement. `Response Control` (R_{P_C}) are acknowledgements which are initiated on receiving of response data. `Writeback Control` (W_C) are acknowledgements that originate from memory and inform private caches about the modified copy of data. `Response Data` (R_{P_D}) carry data messages between on-chip caches and off-chip memory to cache. `Writeback Data` (W_D) traverse from cache to off-chip memory.

[3]We have used MESI cache coherence protocol that classifies messages in six message classes (for more details refer to Appendix B).

[4]The possibilities may vary with cache coherence protocol and a number of networks in multiple NoC.

[5]The details of distinct distribution are listed in Table D.1 of Appendix D.

TABLE 6.1: Percentage improvement in the distribution of control messages with respective variance in result metric ($\Delta\mathfrak{R}$) for PARSEC benchmark.

Benchmark	**Proportion of Message Classes (P) (in %)**								$\Delta\widehat{\mathfrak{R}}$ *(in %)*[†]		
	C_C	R_{Q_C}	R_{P_C}	W_C	Σ_C	R_{P_D}	W_D	Σ_D	$\Delta\xi$	$\Delta\tau$	$\Delta\rho$
Blackscholes	2.8	7.64	38.32	0.05	**48.81**	41.29	9.9	51.19	**28**	**36**	**55**
Swaptions	2.39	6.28	37.84	0.03	**46.54**	41.45	12.01	53.46	**40**	**27**	**500**
Canneal	2.38	6.97	37.96	0.08	**47.39**	41.44	11.17	52.61	**42**	**58**	**66**
Fluidanimate	0.78	6.16	37.89	0.33	**45.16**	42.51	12.33	54.84	**59**	**55**	**18**
X264	2.07	4.81	38.41	0.1	**45.39**	39.1	15.51	54.61	**511**	**69**	**96**
Freqmine	1.73	4.6	39.97	0.11	**46.41**	34.18	19.41	53.59	**9**	**3**	**2**
Streamcluster	0.01	8.87	33.14	0.02	**42.04**	52.81	5.15	57.96	**60**	**34**	**1**
Bodytrack	0.61	3.11	35.12	0.09	**38.93**	46.86	14.21	61.07	**9**	**26**	**1**
Dedup	0.17	3.58	38.32	0.03	**42.1**	40.02	17.88	57.9	**25**	**21**	**2**
Facesim	0.87	0.33	30.86	0.09	**32.15**	42.33	25.52	67.85	**12**	**78**	**27**
Ferret	0.65	1.31	42.31	0.12	**44.39**	34.23	21.38	55.61	**233**	**18**	**5**
Vips	0.54	1.9	38.93	0.35	**41.72**	40.06	18.22	58.28	**50**	**40**	**1**
Rtview	4.94	7.88	43.58	0.13	**56.53**	34.69	8.78	43.47	**560**	**30**	**76**
Variation (Δv)	4.93	8.54	13.7	0.33	**24.4**	18.6	20.4	24.4	**551**	**75**	**499**

[†] η = Cluster no., $\Delta\widehat{\mathfrak{R}}$ = Variance in result metric, ξ = Energy, τ = Throughput, ρ = Power

6.1.1 Case Study to Unveil the Problems of Static Traffic Distribution with CMPs

We have done a case study with all 13 applications of PARSEC benchmark to identify how static traffic distribution impacts power, performance, and energy efficiency. The different message classes viz. C_C, R_{Q_C}, R_{P_C}, W_C, R_{P_D}, and W_D for all the applications of PARSEC suite have different percentage[6] of messages as listed in Table 6.1[7]. The variation can also be seen in the benchmark graph, Fig 6.1.

The benchmarks are kept into three clusters according to the proportion of control messages $45\% \leq \sum_C \leq 55\%$ (top), $\sum_C < 45\%$ (middle), and $\sum_C > 55\%$ (bottom). To observe the impact on result metric with variation in the proportion of messages. We can see the minimum to a maximum proportion of messages classes[8] in the column of proportion of message classes in Table 6.1 across the PARSEC benchmark. On comparing the proportion of total control ($\sum_C$) and Data ($\sum_D$) messages columns, a significant difference is observed in *Bodytrack*, *Facesim*,

[6]We shall interchangeably use the term percentage/fraction/ratio/proportion.

[7]We have used different types of lines for boxes wherein dotted box exhibits volume of control messages. Whereas the percentage improvement/variance of result metric is highlighted with blue lined boxes as shown towards the rightmost column of the table.

[8]In column C_C, R_{Q_C}, R_{P_C}, W_C, R_{P_D}, and W_D of Table 6.1.

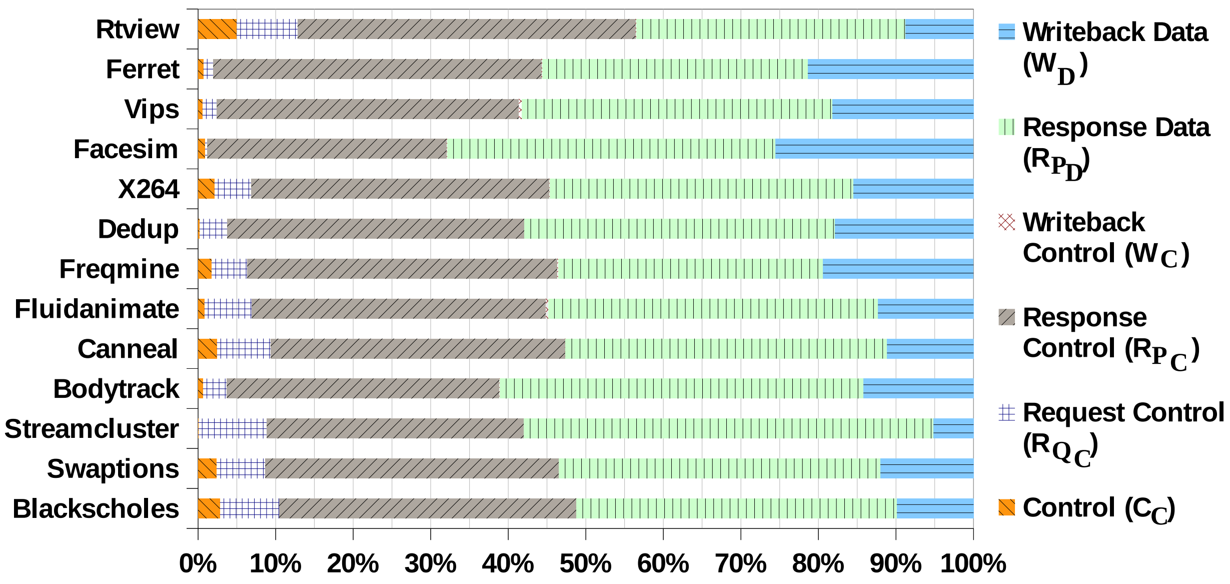


FIGURE 6.1: Variation in the quantity of Control ($\sum_C$) messages in PARSEC benchmark for a MESI protocol of cache coherence. The message classes are explained in Section 6.2 (Results are computed on 8×8 mesh in full system simulation).

and *Rtview* benchmarks as compared to other benchmarks. For specified proportional distribution of messages, the static message distribution changes according to expressions (1) - (3). The variance in result metric across 41 static message distribution is listed in the rightmost column of the table corresponding to each benchmark.

Thus, variance in result metric[9,10] ($\Delta\widehat{\mathfrak{R}}$) measures the variation in percentage improvement of energy, throughput, and power for 2-network-NoC over single-NoC when 41 combinations of static message distribution run for each application.

6.1.2 Experiment Result Discussion on Impact of Power-Performance Efficiency

Table 6.1 as discussed in the previous section shows the upper and lower bounds for Σ_C and Σ_D message proportion as well as the variance for $\Delta\widehat{\mathfrak{R}}$ with 41 different combinations of static message distribution for PARSEC benchmark. The maximum variation ($\Delta\nu$) within $\sum_C$ is 13.7% (for R_{P_C}), $\sum_D$ is 20.4% (for W_D), and $\Delta\widehat{\mathfrak{R}}$ is observed 551% (for $\Delta\xi$).

[9]The benchmarks highlighted with red fonts show maximum variance for at least one out of three metrics in $\Delta\widehat{\mathfrak{R}}$ from each cluster.

[10]Overall variation ($\Delta\nu = V_{Max} - V_{Min}$) for each message subclass is computed in the last row. The three different circled value from left to right shows the highest variance among control messages, data messages, and in result metric.

We have drawn following inferences with the proposed case study on general purpose applications of PARSEC benchmark:

1) Communication intensive benchmarks such as *swaptions* (power), *x264* (energy), and *rtview* (throughput) show a significant approximately 5× variations in result metric.

2) Power varies between 3% to 78%, throughput varies 1% to 500%, and energy varies 9% to 560%. This indicates that a single static distribution strategy may not benefit general purpose applications alike.

3) The benchmark *freqmine* has a negligible variance for all result metrics as it is a computation intensive application. Therefore, it exhibits negligible impact of traffic distribution variations. Similarly, relatively computation intensive. Benchmarks such as *bodytrack* and *dedup* display minor variations in result metrics.

Table 6.1 results show that the static message distribution does not provide universally optimum message distribution for general purpose processors. This case study illustrates how unfair static traffic distribution can be while considering the execution of different types of application on the general purpose processor. It shows that offline implementation with the assumption of hardware agnostic static traffic distribution may not be the right way to decide traffic distribution for multiple NoC networks.

Network power can be as high as 150W for a manycore die [53]. So a high proportion of chip power is consumed by NoC. It continues rising with the increasing number of cores and communication traffic. Due to the impact of static message distribution on power-performance trade-off and thus energy efficiency, a significant investment [50, 51] is being made to improve the energy efficiency of modern manycore NoC designs. The static message distribution is implicitly limited as

1) it does not provide a universally optimum message distribution for all applications.

2) it does not remain an ideal traffic distribution even for the entire[11] execution time of a given application as the volume of different messages classes often varies during the execution of the application.

[11]This is discussed in detail in the following section.

Then, what could be the right way to segregate different classes? How to assign message classes to dual NoC networks for resolving unfair message distribution. To find the solution, it is important to understand traffic characteristics in more details as we do in the next section.

6.2 Criticality Analysis of Fine-Grain Messages of Caches

In this section, we demonstrate runtime variations in the volume of message classes that further motivates us to study messages at their finer granularity. We demonstrate the inter-dependency in the sequence of fine-grain messages during serving of messages in response to a cache miss. Ordering, as well as inter-dependency of these messages, is result of cache state transitions dictated by coherence protocol. Cache states trigger a variety of fine-grain cache message[12] to maintain the coherence between caches and memory. We identify criticality priority of these fine-grain messages.

6.2.1 Runtime Dynamics of NoC Traffic

We observe the runtime variation in the volume of different message classes across different applications. We randomly selected traces of '*Swaption* benchmark' to demonstrate runtime variations in message volume. In Fig 6.2, the y-axis shows the cumulative number of injected flits and the x-axis shows the simulated number of **I**nstructions **P**er **S**econds (IPS) that are ranging from 1×10^6 to 9×10^6.

As shown in Fig 6.2, the volume of transmitted `Response Control` (R_{P_C}) and `Response Data` (R_{P_D}) flits are less during 5×10^6 – 6×10^6 IPS as compared to 6×10^6 – 7×10^6 IPS. If static message distribution is employed, performance degradation may occur.

Therefore, non-flexibility of static traffic distribution techniques are inadequate to handle the runtime changes in the volume of individual message class. This motivates to consider the runtime variability and unpredictability of traffic volume for designing adaptive message distribution.

[12]These messages belong to six message classes for MESI.

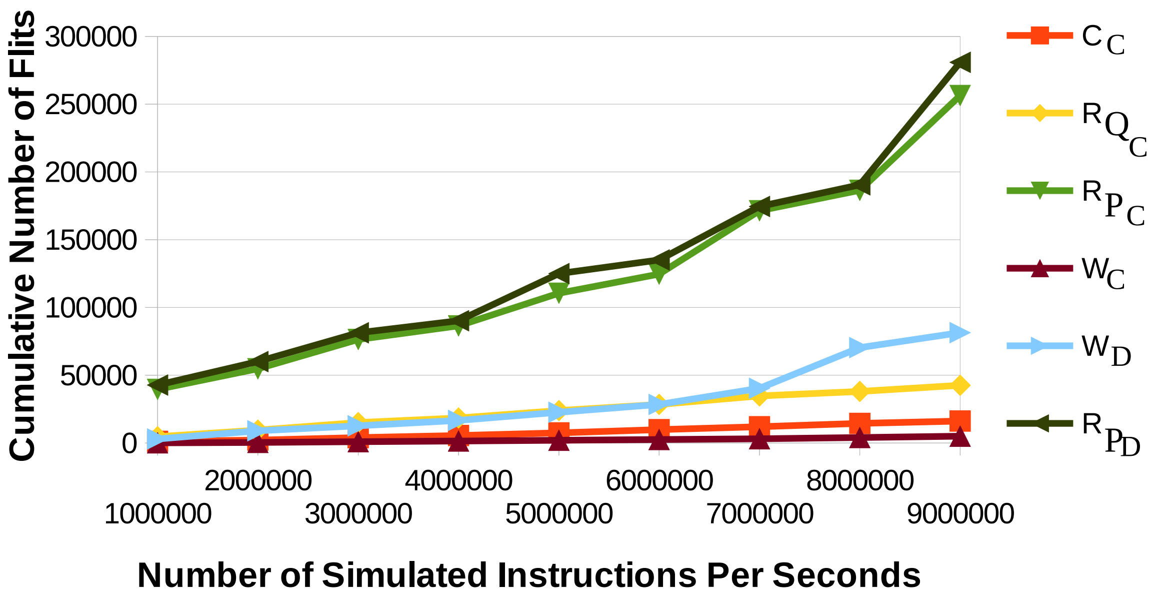


FIGURE 6.2: Runtime traffic traces of *Swaption* benchmark on 8×8 mesh with Gem5 simulator.

6.2.2 In-Transit Cache State Fine-Grain Messages

When a processor core does not find requested data in cache, a cache miss occurs and a request for data message is transmitted through Networks-on-Chip. Additional coherence messages also traverse through NoC to maintain the consistency between caches and memory data. Every message is associated with a cache state transition that occurs on a cache miss. Since the transition between different cache states results in different types of messages, an exploration of fine-grain messages during cache state transition is really important.

6.2.2.1 Message Flow

The number of messages is dependent on the cache coherence protocol, the property of cache hierarchy, and method of sharing among caches.

- **Cache coherence protocol.** We consider strictly inclusive MESI cache coherence protocol [135] (Appendix B). This is a multicast cache coherence protocol used with three levels of the cache hierarchy, i.e., L_0 private, L_1 private and L_2 shared among the cores.
- **Inclusive property of cache hierarchy.** We have used inclusive property for the hierarchy of caches of the same core. The message block existing in a lower level cache needs

to be forwarded to the higher level cache. Similarly, if a block is evicted from a higher level cache, it has to be evicted from lower level cache too.

- **Sharing status.** We have used multicast protocol wherein core valid bits are used by shared L_2 for each block to maintain the sharing status of the cache block.

In following paragraphs we briefly summarize events in case of a cache miss.

1) L_0 **is empty**. Initially, when core fetches an instruction and data cache L_0 is empty, it will check the data in L_1 and load the data from shared L_2 or memory. The cache block would be in **E**xclusive (E) state (since this is the only copy of data in cache identical to memory copy). When this block is modified with a write operation, it is converted to **M**odified (M) state. When a cache block is replaced, a $PUTX$ event is generated to write the block into the memory to maintain consistency only if it is in a modified state.

2) L_0 **is a Hit**. Data is available in L_0 cache, so it is serviced to the core.

3) L_0 **is a Miss,** L_1 **is a Hit**. When a miss occurs at L_0, the data request is forwarded to L_1 cache. Either exclusive or shared copy is sent back to L_0 cache.

4) L_0 **is a Miss,** L_1 **is a Miss,** L_2 **is a Hit**. L_1 Miss is forwarded to L_2 shared cache. Shared data or exclusive data is sent back to the L_1 cache of requested core and, then, it is serviced to L_0 cache by L_1 cache.

5) L_0 **is a Miss,** L_1 **is a Miss,** L_2 **is a Miss**. The L_1 miss is forwarded to L_2 and data is fetched from memory and serviced back to L_0 cache of the requested core. The stable states, i.e., **I**nvalidation (I) and M of MESI three level coherence protocol are used by every individual cache. Transient states, i.e., IM, SM, MT, MT_MB are used to recycle and enqueue–dequeue cache request queue during transition among stable states.

   Fig 6.3 illustrates cache state transitions on L_1 miss. In this simulation, miss happens at core 0 and requisite data is in L_3 cache of core 3. A request for data is initiated by core 0 on L_1 miss, the request is forwarded to L_2 cache by using control messages. Once again miss at shared L_2 cache of core 0 denotes the nonavailability of data in L_2 cache. In this simulation scenario, the requested copy is available with L_2 cache of core 3. Core 0 sends the request to L_2 cache of core 3, but before receiving the request, the copy is replaced at

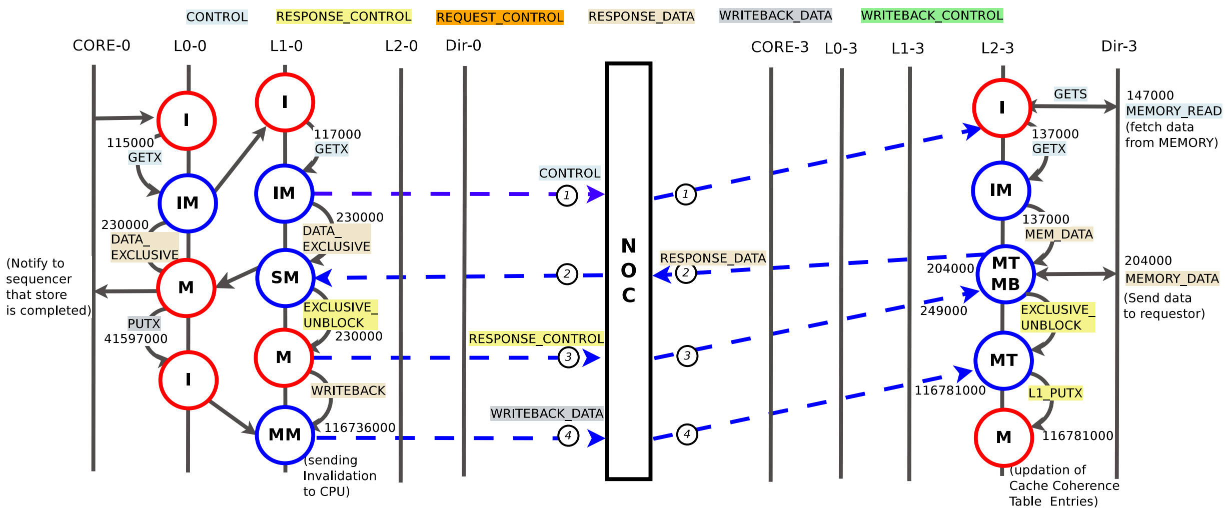


FIGURE 6.3: A test case scenario for NoC traffic communication with cache state transitions when L_0 is a miss, L_1 is a miss, and L_2 is a miss (NoC–**N**etworks-**on**-**C**hip).

L_2 cache of core 3. Then, L_2 cache miss of core 0 is handled in the following steps while maintaining cache coherence.

- When L_2 cache of core 3 is replaced, the request of core 0 is forwarded to memory through $GETS$ control message.
- The L_2 cache of core 3 fetches the data from memory and as a response gets MEM_DATA from memory.
- Now MEM_DATA message is forwarded to requested L_2 cache of core 0. The inclusive property triggers the cache event $DATA\_EXCLUSIVE$ to L_1 and L_0 caches of core 0 The state of both these caches are changed to M on triggering of the cache event $DATA\_EXCLUSIVE$ message because this block is modified with a memory write operation.
- A response control, i.e., $EXCLUSIVE\_UNBLOCK$ is sent by L_1 of core 0 to L_2 cache of core 0 as an acknowledgement.
- The $L1\_PUTX$ is initiated by L_2 cache of core 3 to replace the block that is modified by L_1 cache of local core (if any).
- Whereas the $L1\_WRITEBACK$ is initiated by core 0 before replacing the only copy of the block that is modified by L_1 cache of core 0.
- Repeat steps subsequently on L_2 miss, again.

Message flow in the above example shows how NoC traffic vis-a-vis messages classes vary and is dependent on cache state transitions. In the next section, we prioritise messages according to their criticality[13] that is devised through the proportion of generated messages and their inter-dependency.

6.2.3 Criticality-Aware Prioritisation of Fine-Grain Messages

The fine-grain cache messages include load, store, acknowledgements etc. The switching between two stable cache states is followed by multiple transient states. Every state transition stable-to-transient (transient-to-stable) generates such messages.

These messages related to

1) cache itself and local cache hierarchy[14] updation: `Control` and `Writeback Control` belong to local cache updation. `Writeback Control` communicates with L_0 and L_1 local caches for cache block replacement on the eviction of the block from L_2 cache. Whereas `Control` messages are communicated by all local L_0, L_1, and L_2 caches to communicate with each other to update local caches.

2) updation of other on-chip shared caches and offchip memory: `Request Control` messages communicate between the caches having shared copies of the cache block. The `Response Control` messages originate at both local as well as other shared caches before replacement of original copy to invalidate stale data.

Figures 6.4 and 6.5 show cache state transition diagrams for L_2 and L_3 caches respectively. As different messages are generated for different transitions, we shall call these as fine grain messages. These messages can be mapped to six message classes (`Control, Request Control, Response Control, Writeback Control, Response Data, Writeback Data`), which shall be referred to as mid-level messages. Mapping of fine grain message to middle-grain or mid-level messages is shown in Figures 6.4 and 6.5, a different color is used for mapping to a given mid-level message. From the cache state transition diagram, we can see that

1) each cache can switch between `Stable` and `Transient` states. Two additional states are used by L_2 cache, i.e., `Blocking` and L_2 `Replacement` because of its shared nature.

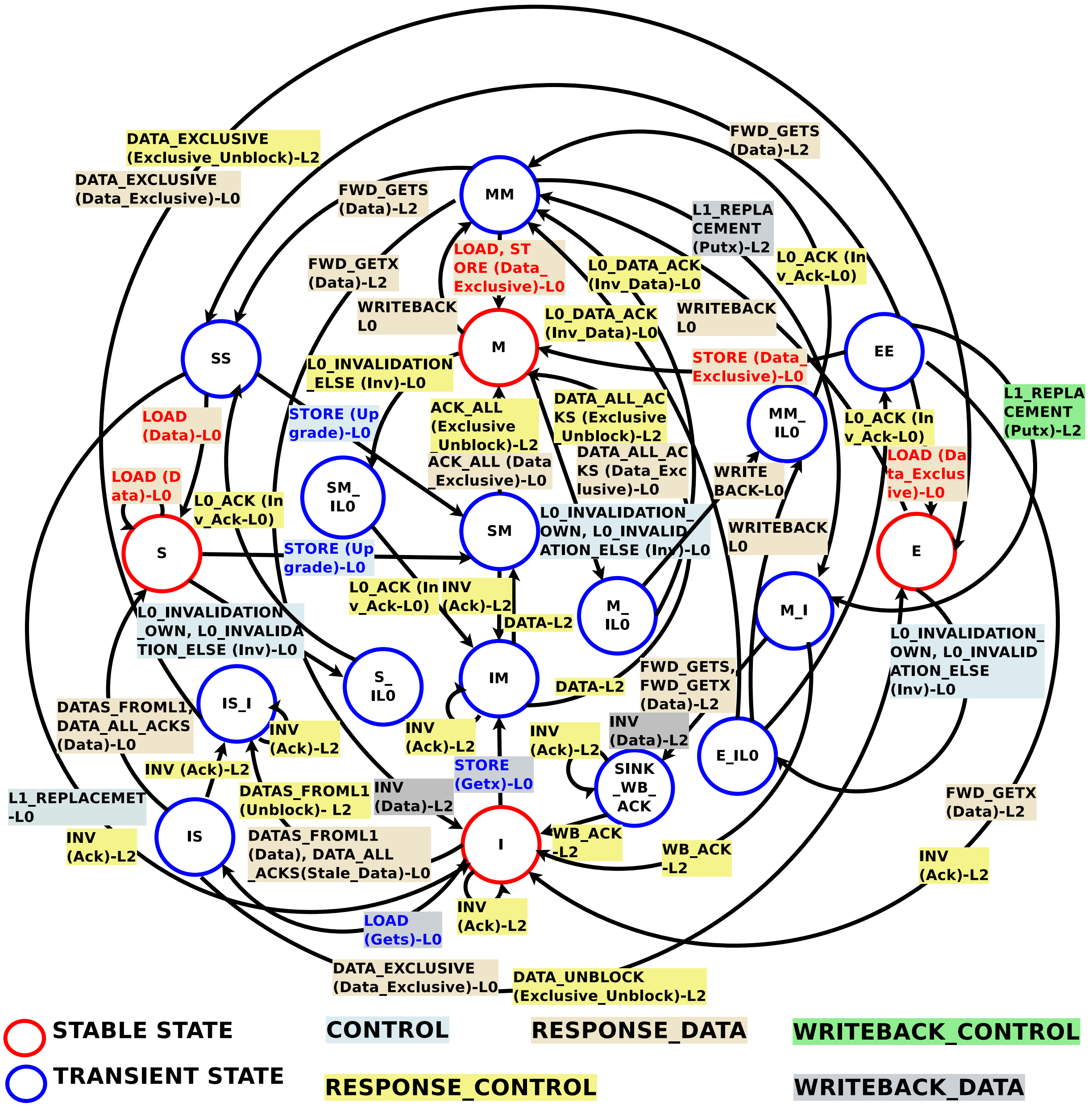
[13]How promptly an application uses data after its required message is fetched from memory.

[14]Forwarded to/fro lower level caches.

2) fine-grain messages are triggered in-between cache state transitions by the cache controller.

3) these messages can be aggregated as mid-level class/category of messages, i.e., `Control`, `Request Control`, `Response Control`, `Writeback Control`, `Response Data`, and `Writeback Data`. Every class of fine-grain messages is highlighted with background color (blue, orange, yellow, green, peach, gray) as per the category.

4) the red text indicates the messages generated on a cache hit and blue text indicates the messages on a cache miss.



DATA_EXCLUSIVE (Exclusive_Unblock)-L2
DATA_EXCLUSIVE (Data_Exclusive)-L0
FWD_GETS (Data)-L2
MM
FWD_GETS (Data)-L2
L1_REPLACEMENT (Putx)-L2
FWD_GETX (Data)-L2
LOAD, STORE (Data_Exclusive)-L0
L0_DATA_ACK (Inv_Data)-L0
L0_ACK (Inv_Ack-L0)
WRITEBACK L0
M
L0_DATA_ACK (Inv_Data)-L0
WRITEBACK L0
SS
L0_INVALIDATION_ELSE (Inv)-L0
STORE (Data_Exclusive)-L0
EE
LOAD (Data)-L0
STORE (Upgrade)-L0
ACK_ALL (Exclusive_Unblock)-L2
ACK_ALL (Data_Exclusive)-L0
DATA_ALL_ACKS (Exclusive_Unblock)-L2
DATA_ALL_ACKS (Data_Exclusive)-L0
MM_IL0
L0_ACK (Inv_Ack-L0)
L1_REPLACEMENT (Putx)-L2
LOAD (Data)-L0
SM_IL0
L0_ACK (Inv_Ack-L0)
LOAD (Data_Exclusive)-L0
WRITE BACK-L0
S
SM
L0_INVALIDATION_OWN, L0_INVALIDATION_ELSE (Inv)-L0
WRITEBACK L0
E
STORE (Upgrade)-L0
L0_ACK (Inv_Ack-L0)
INV (Ack)-L2
M_IL0
M_I
L0_INVALIDATION_OWN, L0_INVALIDATION_ELSE (Inv)-L0
DATA-L2
L0_INVALIDATION_OWN, L0_INVALIDATION_ELSE (Inv)-L0
S_IL0
IM
DATA-L2
FWD_GETS, FWD_GETX (Data)-L2
IS_I
DATAS_FROML1, DATA_ALL_ACKS (Data)-L0
INV (Ack)-L2
INV (Ack)-L2
INV (Ack)-L2
INV (Data)-L2
E_IL0
INV (Ack)-L2
STORE (Getx)-L0
SINK_WB_ACK
L1_REPLACEMET-L0
DATAS_FROML1 (Unblock)- L2
INV (Data)-L2
FWD_GETX (Data)-L2
IS
INV (Ack)-L2
DATAS_FROML1 (Data), DATA_ALL_ACKS(Stale_Data)-L0
I
WB_ACK-L2
WB_ACK-L2
INV (Ack)-L2
LOAD (Gets)-L0
INV (Ack)-L2
DATA_EXCLUSIVE (Data_Exclusive)-L0
DATA_UNBLOCK (Exclusive_Unblock)-L2
STABLE STATE
CONTROL
RESPONSE_DATA
WRITEBACK_CONTROL
TRANSIENT STATE
RESPONSE_CONTROL
WRITEBACK_DATA

FIGURE 6.4: State space diagram of L_1 cache for MESI 3–level cache coherence protocol.

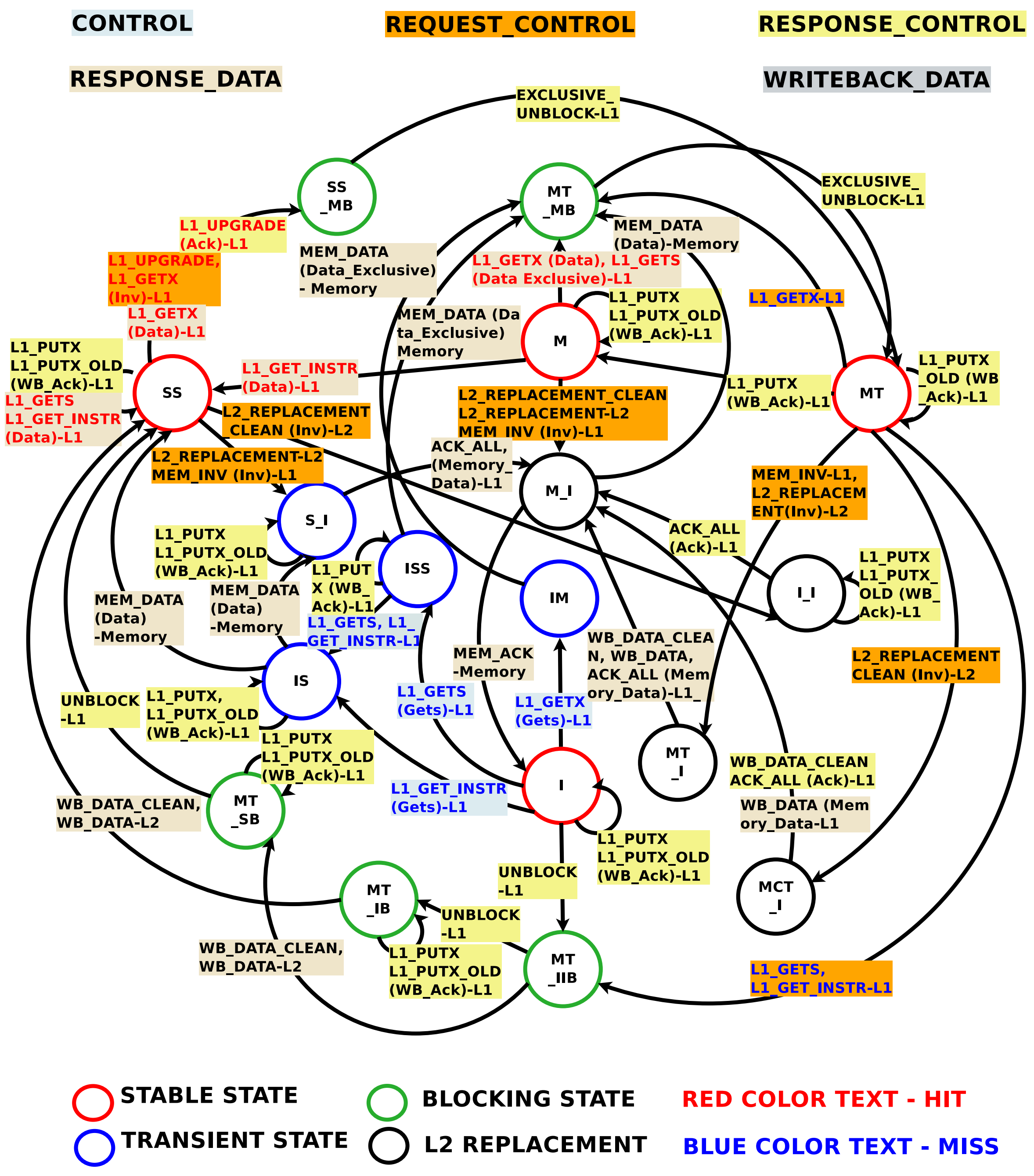

CONTROL
REQUEST_CONTROL
RESPONSE_CONTROL
RESPONSE_DATA
WRITEBACK_DATA
EXCLUSIVE_ UNBLOCK-L1
SS _MB
MT _MB
EXCLUSIVE_ UNBLOCK-L1
L1_UPGRADE (Ack)-L1
L1_UPGRADE, L1_GETX (Inv)-L1
MEM_DATA (Data_Exclusive) - Memory
MEM_DATA (Data)-Memory
L1_GETX (Data), L1_GETS (Data Exclusive)-L1
L1_GETX (Data)-L1
MEM_DATA (Data_Exclusive) Memory
M
L1_PUTX L1_PUTX_OLD (WB_Ack)-L1
L1_GETX-L1
L1_PUTX L1_PUTX_OLD (WB_Ack)-L1
L1_GETS L1_GET_INSTR (Data)-L1
SS
L1_GET_INSTR (Data)-L1
L2_REPLACEMENT CLEAN (Inv)-L2
L2_REPLACEMENT_CLEAN L2_REPLACEMENT-L2 MEM_INV (Inv)-L1
L1_PUTX (WB_Ack)-L1
MT
L1_PUTX _OLD (WB _Ack)-L1
L2_REPLACEMENT-L2 MEM_INV (Inv)-L1
ACK_ALL, (Memory_ Data)-L1
M_I
MEM_INV-L1, L2_REPLACEMENT(Inv)-L2
S_I
ACK_ALL (Ack)-L1
L1_PUTX L1_PUTX_OLD (WB_Ack)-L1
ISS
L1_PUTX (WB_ Ack)-L1
IM
I_I
L1_PUTX L1_PUTX_ OLD (WB_ Ack)-L1
MEM_DATA (Data) -Memory
MEM_DATA (Data) -Memory
L1_GETS, L1_ GET_INSTR-L1
WB_DATA_CLEAN, WB_DATA, ACK_ALL (Memory_Data)-L1
MEM_ACK -Memory
L2_REPLACEMENT CLEAN (Inv)-L2
IS
UNBLOCK -L1
L1_PUTX, L1_PUTX_OLD (WB_Ack)-L1
L1_GETS (Gets)-L1
L1_GETX (Gets)-L1
MT _I
L1_PUTX L1_PUTX_OLD (WB_Ack)-L1
WB_DATA_CLEAN ACK_ALL (Ack)-L1
WB_DATA_CLEAN, WB_DATA-L2
MT _SB
L1_GET_INSTR (Gets)-L1
I
WB_DATA (Memory_Data-L1
L1_PUTX L1_PUTX_OLD (WB_Ack)-L1
UNBLOCK -L1
MCT _I
MT _IB
UNBLOCK -L1
WB_DATA_CLEAN, WB_DATA-L2
L1_PUTX L1_PUTX_OLD (WB_Ack)-L1
MT _IIB
L1_GETS, L1_GET_INSTR-L1
STABLE STATE
BLOCKING STATE
RED COLOR TEXT - HIT
TRANSIENT STATE
L2 REPLACEMENT
BLUE COLOR TEXT - MISS

FIGURE 6.5: State space diagram of L_2 for MESI 3–level cache coherence protocol.

Additional states, i.e., blocking and replacement states in L_2 cache-state space block the transitions while waiting for data and replace a cache block with **L**east **R**ecently **U**sed (LRU) replacement policy.

The proportion of these messages vary as per the memory access pattern of the application [15]. To identify the criticality of messages, we

1) compare the proportion of fine-grain message mapping to each class (mid-level granularity) of messages.
2) understand the inter-dependency of messages to discern the order, i.e., which messages need to be sent first so that data transfer is not delayed.

6.2.3.1 Analysis using Proportion of Fine-Grain Messages in-between Cache States Transitions

On comparing the proportion of the messages in Fig 6.4 and Fig 6.5, our observations with fine-grain messages in-between cache state transitions are as follows:

1) We find data messages[16] are more than control messages[17]. Because of the bulk of proportion of these messages, we have considered them as noncritical messages as these are preceded by control messages.
2) The proportion of `Writeback Control` messages is less[18] as compared to `Control` messages, so we prioritize them over `Control` messages. As these messages belong to local caches, the quick arrival of these messages speeds up the communication. We consider these messages urgent, so we keep them as high priority messages.
3) The proportion of `Request Control` is less, i.e., $R_{Q_C} < R_{P_C}$ as compared to `Response Control` messages. As the proportion of `Response Control` is the highest among the control messages, we put them at low priority among the control messages.

We also observe that the proportion of fine grain messages independent of benchmark parameter variations. Our discussed comparative proportion of messages is also scalable[19] for the higher volume[20] of traffic as demonstrated in Fig 6.6[21].

[15] In the gem5, there is a cache coherence protocol rules, messages, and their transitions are defined under protocol directory. We have used these rules to draw cache state space graphs.

[16] Split into multiple flits (five flits for our NoC architecture).

[17] Single flit

[18] Messages are generated less frequently.

[19] Criticality prioritisation is independent of the volume of traffic.

[20] The number of messages

[21] In Fig 6.6, the legends represent the fine-grain messages of Fig 6.4 and Fig 6.5. The name of the legend exhibits message class, message type and belongingness with the cache. For example, in legend of fine-grain message WB_DATA_INV, the WB_DATA (first word of the name) is `Writeback Data` message class, and message type is INV (invalidation), and the message belongs to the L_2 cache. In case of presence of L_1 label in legend name, the message belongs to L_1 cache.

1) On varying workload from low to high in Fig 6.6[22] (a), the volume of messages significantly increases though their prioritisation order still remains the same. For example - in Fig 6.6 (a) with *canneal* benchmark, we get huge volume increase for $R\_C\_L1\_GETX$. These messages are classified under `Response Control` (low priority among the control messages) and belong to L_1 cache.

2) Similar trends are observed when the traffic increases on a varying number of threads from 8 to 64, as shown in Fig 6.6[23] (b).

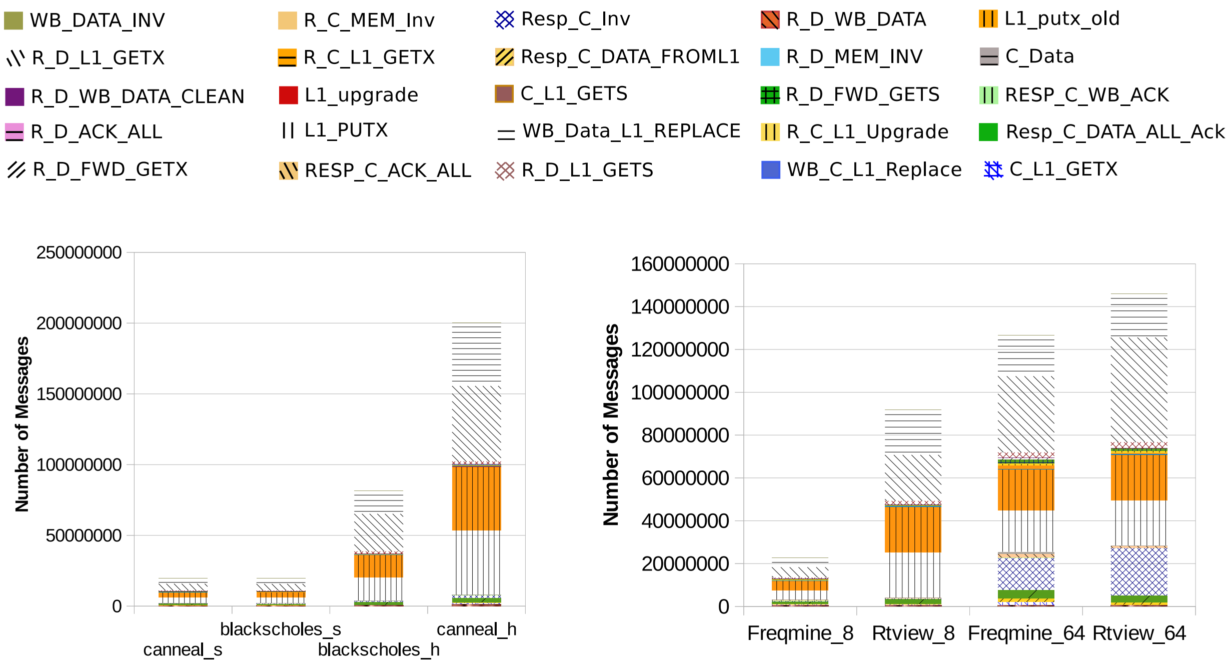


FIGURE 6.6: Message variations (a) with Low (L)(*canneal_s, blackscholes_s*) and High (H) (*canneal_h, blackscholes_h*) workloads of the benchmarks (b) for *freqmine* and *rtview* benchmarks when the number of threads vary from 8 to 64.

Other factors such as the number of cores, **I**nstruction **S**et **A**rchitecture (ISA) of a processor, cache hierarchy levels, cache type and size, type of cache coherence protocol, etc., also influence the variation in proportion of various message classes in the traffic.

6.2.3.2 Analysis-Based on Quality of Messages

In our another perspective of analysis, our consideration is the quality of message instead of volume/proportion of messages. We devise the criticality through an understanding of the

[22]refer Appendix E for the tabular representation of data
[23]refer Appendix F for tabular representation of data

inter-dependency of messages. The control messages are more critical over data messages ($\text{Criticality}_{\sum_C} > \text{Criticality}_{\sum_D}$) as the state-of-the-art literature reported [58–60]. The request for data is initiated with control messages and is followed by data responses. Once the data messages are received, the ACK[24] is sent back to the sender.

For all control messages, the criticality priority (P) within subclasses of control messages ($\sum_C$) is considered as $P(\sum_C) \Rightarrow W_C > C_C > R_{Q_C} > R_{P_C}$ by analyzing the necessity of messages for caches/cores. The most important undertaking by cache coherence is cache block replacement and eviction of the block from shared cache through `writeback control` (W_C) messages. Once these messages are received, caches proceed to write the latest modified data to the rest of the caches. Next, `control messages` (C_C) are considered as top priority messages. These messages invalidate the older copies of the data and update the caches. Otherwise, it may lead to inconsistency between caches. Receiving of these messages are significantly important for initiating another coherence event. The dependence of other messages on these message classes (W_C and C_C) combinedly make them the most critical (S_0) messages.

The remaining control messages are considered as middle-level critical (S'). Among these the `request control` messages (R_{Q_C}) are prioritized, these messages unblock or allow the caches for further read/write operations. Then `response control` messages (R_{P_C}) are considered that are the acknowledgments of the cache writes whereas the data messages are considered as non-critical (S_1).

6.3 Strategy for Runtime Adaptive Message Distribution

Prior analysis of criticality shall be utilised to devise traffic distribution policy for hardware implementation of runtime adaptive scheme. Out of six message classes of MESI protocol, we define three levels of criticality priority (P) as follows

$$\text{P} = \text{if}\begin{cases} \text{Most } (\text{S}_0) & : \begin{cases} W_C \\ C_C \end{cases} ; \text{ Flags}\begin{cases} m_0{=}1 \\ m_1{=}1 \end{cases} & \cdots(4) \\ \text{Middle } (\text{S}') & : \begin{cases} R_{Q_C} \\ R_{P_C} \end{cases} ; \text{ Flags}\begin{cases} m_2{=}1 \\ m_3{=}1 \end{cases} & \cdots(5) \\ \text{None } (\text{S}_1) & : \begin{cases} W_D \\ R_{P_D} \end{cases} & \cdots(6) \end{cases}$$

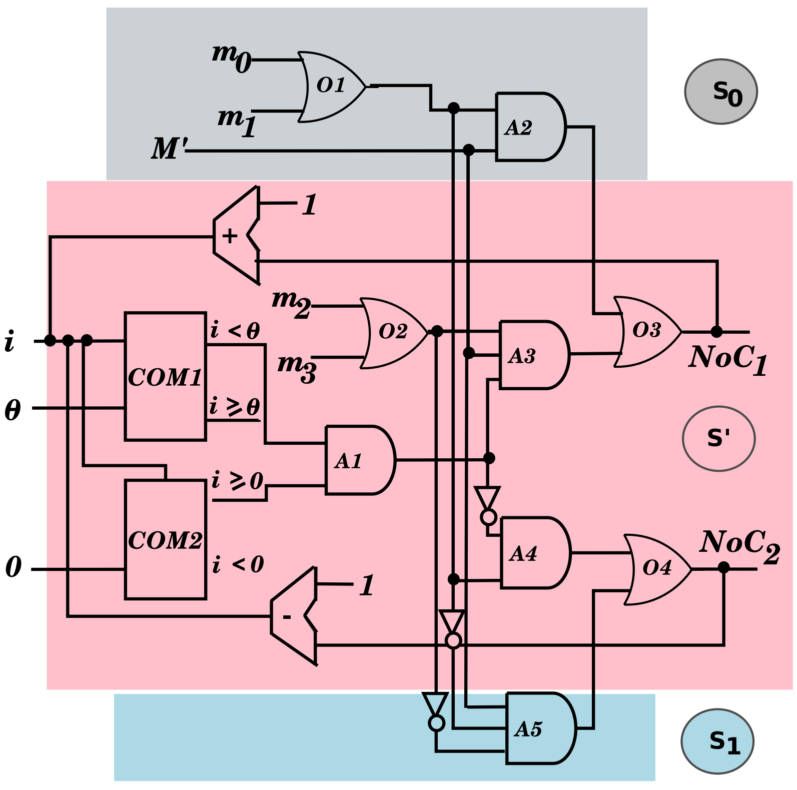
[24]Acknowledgement

Each level comprises of two messages classes. Their priorities are defined under the set S_0 *(Most)*, S'*(Middle)*, and S_1 *(non-critical/none)* as discussed in the previous section[25] . The set S_0 and S_1 are dedicated to NoC_1 and NoC_2 networks. The set S' is adaptive, it can be assigned to either NoC_1 or NoC_2. The selection of NoC network according to message classes are demonstrated through Algorithm 1.

Algorithm 1 Runtime Adaptive Message Distribution

```
Input: $m_0, m_1, m_2, m_3, M', i, \theta$
Output: Selection of NoC (either $NoC_1$ or $NoC_2$) for input Message M $'$
Initialize: $i \leftarrow 0$   //Global initialization
 1: while network active do
 2:   $m_0 \leftarrow 0, m_1 \leftarrow 0, m_2 \leftarrow 0, m_3 \leftarrow 0$   //Local initialization
 3:   if $M' = W_C$ then
 4:     $m_0 \leftarrow 1$   //$W_C$ message sends through $NoC_1$
 5:     $i \leftarrow i + 1$
 6:   else if $M' = C_C$ then
 7:     $m_1 \leftarrow 1$   //$C_C$ message sends through $NoC_1$
 8:     $i \leftarrow i + 1$
 9:   else if $i < \theta$ && $i \geq 0$ then
10:     if $M' = R_{Q_C}$ then
11:       $m_2 \leftarrow 1$   //$R_{Q_C}$ transfers through $NoC_1$ when $i < \theta$
12:       $i \leftarrow i + 1$
13:     else if $M' = R_{P_C}$ then
14:       $m_3 \leftarrow 1$   //$R_{P_C}$ transfers through $NoC_1$ when $i < \theta$
15:       $i \leftarrow i + 1$
16:     else
17:       $i \leftarrow i - 1$
18:     end if
19:   else
20:     $i \leftarrow i - 1$   /*This condition indicates arrival of either a data message $R_D/W_D$ or control
        message $R_{Q_C}/R_{P_C}$ while $i > \theta$ that transfers through $NoC_2$*/
21:   end if
22: end while
```

[25]As the message criticality is defined through consideration of message volume/quality.



(a)

```
if (M'=W_C ) then
|   m_0 ← 1 ;
|   i ← i + 1 ;
else if (M'=C_C ) then
|   m_1 ← 1;
|   i ← i + 1 ;
else if (i < θ  &&  i ⩾ 0) then
|   if (M'=R_QC)  then
|   |   m_2 ← 1;
|   |   i ← i + 1;
|   else if  (M'=R_PC) then
|   |   m_3 ← 1;
|   |   i ← i + 1;
|   else
|   |   i ← i - 1 ;
|   end if
else
|   i ← i - 1  ;
end if
```

(b)

Figure 6.7: Schematic of adaptive message distribution logic (a) hardware implementation of NoC selection logic for each message M' (b) pseudocode demonstrates the functioning of NoC selection for the messages belong to different priority levels along with hardware implementation. (Where COM: Comparator, +: Adder, –: Subtractor, A: AND Gate, and O: OR Gate). The highlighted red color circuit is for S' set of messages while S_0 and S_1 set of messages logic are on top (in black) and bottom (in blue) of S' circuit/logic.

If the message type is either `Writeback Control (WB`$_C$`)` (step 4) or `Control (C)` (step 7), the messages always traverse through NoC$_1$ network.

Data messages (`Response Data (Resp`$_D$`)` and `Writeback Data (WB`$_D$`)`) always dedicated to NoC$_2$ network. Whereas remaining control messages (`Request Control (R`$_{QC}$`)` and `Response Control (R`$_{PC}$`)`) adaptively changes NoC network according to the number of messages of NoC$_1$ network. Initially, these messages traverse through NoC$_2$ network. The decision to change the NoC network for these messages is taken through a threshold value θ that declare the underutilisation of NoC$_1$ network. Once the condition i< θ && i≥ 0 is true (where i is a counter), it indicates the underutilisation of NoC$_1$ network. The messages of middle level critical switch to traverse through NoC_1. Meanwhile, counter i is incremented on traversing messages through NoC_1. But if messages traverse through NoC_2, the counter i is decremented. Thus, counter i is continuously updated and results in a right prediction of under-utilisation of NoC_1 network. The hardware implementation of the algorithm is shown in Fig 6.7.

We design a hardware logic for adaptive traffic distribution using the defined priority/criticality of messages. The hardware implementation for selecting three different sets S_0, S', and S_1 of message classes are shown in Fig 6.7 (highlighted with three different colors), and its working is already explained in Algorithm 1. The flag of corresponding message class M' is enabled to identify respective network. Initially, all flags $\{m_0, m_1, m_2, m_3\}$ are initialized with zero. The most critical messages are assigned to NoC_1, and non-critical messages always traverse through NoC_2 whereas middle-level messages are initially assigned on NoC_2. But network selection changes with runtime dynamics of traffic and utilisation of NoC networks.

In the next section, we monitor the runtime dynamics of traffic for the selection of NoC networks of middle level critical messages.

6.3.1 Monitoring of Runtime Traffic Dynamics for Implementation of Adaptivity

Ideally, at every execution instance, both NoC networks should be minimally underutilised or overutilised. The adaptive scheme is expected to balance the traffic between multiple NoC networks. Though, this is entirely dependent on the prediction accuracy of

1) message criticality: we have defined criticality through analysis of the proportion of fine-grain messages of cache state transitions, and it is also validated through the analysis of inter-dependency of messages.

2) over/under utilisation measurement threshold value θ: when NoC_1 channel is underutilized (overutilized), the middle-level messages start to traverse through NoC_1 (NoC_2). So adaptive method balances the utilization of both networks. As middle level messages traverse on the underutilized NoC network, they can switch between NoC_1 and NoC_2 according to network traffic conditions. We have done the offline analysis of threshold value θ with multiple enough number of samples.

3) offline sampling for threshold θ: we have done an offline sampling for selecting threshold θ by taking $\theta = \{2, 4, 8, 16\}$. This sampling helps to find an exact value of θ that can monitor the most recent traffic trends. If we take it too small, it will not exploit the utility of adaptivity, or if we take it very large, it will increase the hardware cost while benefits are the same as with the lower value of θ. With experiments, we observe that threshold value $\theta = 8$ is sufficient to give the latest or the most recent observations about the traffic behavior of the network. For example, we can observe in Fig 6.10 – Fig 6.12, the energy saving and improvement in throughput (in Fig 6.13 – Fig 6.14) is the same as for $\theta = \{8, 16\}$.

Network assigned to S' depends on the runtime utilization of NoC_1 evaluated through a threshold θ. For the implementation of adaptivity, a certain threshold value needs to be check when specific NoC network can be declared as underutilised/overutilised. If the condition θ is *True* (*False*), S' is forwarded through NoC_1 else NoC_2. The utilization (U) of NoC_1 is measured via a counter i locally evaluated against the threshold θ at a router. If $(i < \theta \,\&\&\, i \geq 0) \implies U_{\text{NoC}_1}, \forall \quad i \in \{0, 1, \cdots, \theta - 1\}, \theta \in N$. if the condition $0 \leq i < \theta$ is true, it shows the underutilization of NoC_1.

Initially counter $i = 0$, if the condition $0 \leq i < \theta$ is true and flag m_2 or m_3 is enabled, then the messages are sent on NoC_1. The counter i is incremented by one if the message is forwarded through NoC_1 else it is decremented by one if any message is sent through NoC_2 as shown in Fig 6.7 and explained in steps 10-17 of Algorithm 1. It shows adaptive message distribution at runtime for dual NoC networks, and it employs feedback correction mechanism by incrementing/decrementing the counter as per channel utilization of NoC_1.

The adaptive mechanism continuously monitors NoC networks for improving underutilisation of the networks. This runtime adaptation of traffic makes network better with the more balanced workload on each channel.

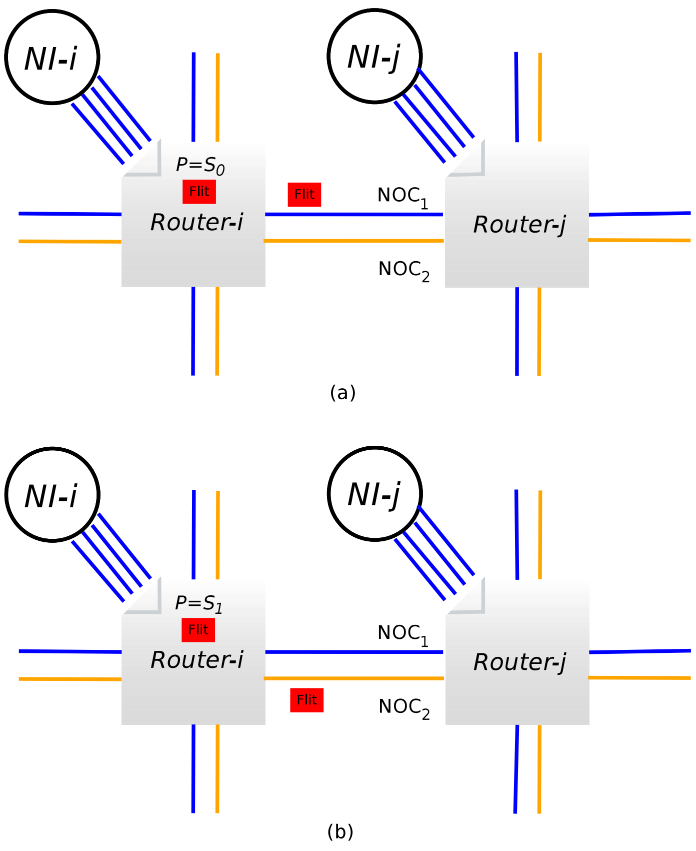


FIGURE 6.8: Runtime adaptive message distribution mechanism, the flit (red box) at the router selects the NoC network (a) when flag m_0 or m_1 is enabled, the NoC_1 is selected (b) when all flags are disabled, the NoC_2 is selected.

6.3.2 A Walkthrough Example

We have demonstrated working of adaptive message distribution in Fig 6.8 and Fig 6.9 with a walkthrough example using different test cases of message classes. The red box at the router shows the flit that selects the NoC networks as per the decision of adaptive message distribution hardware unit. This hardware unit is embedded with the routing unit, and a decision of NoC selection is taken as per message class of the flit just after the route computation. Once the network is selected, the flit traverses in the destined direction as computed through routing logic. The routing can handle only the volume of data traffic, but the proposed adaptive method can handle both quality, volume, and utilization of NoC networks with runtime traffic.

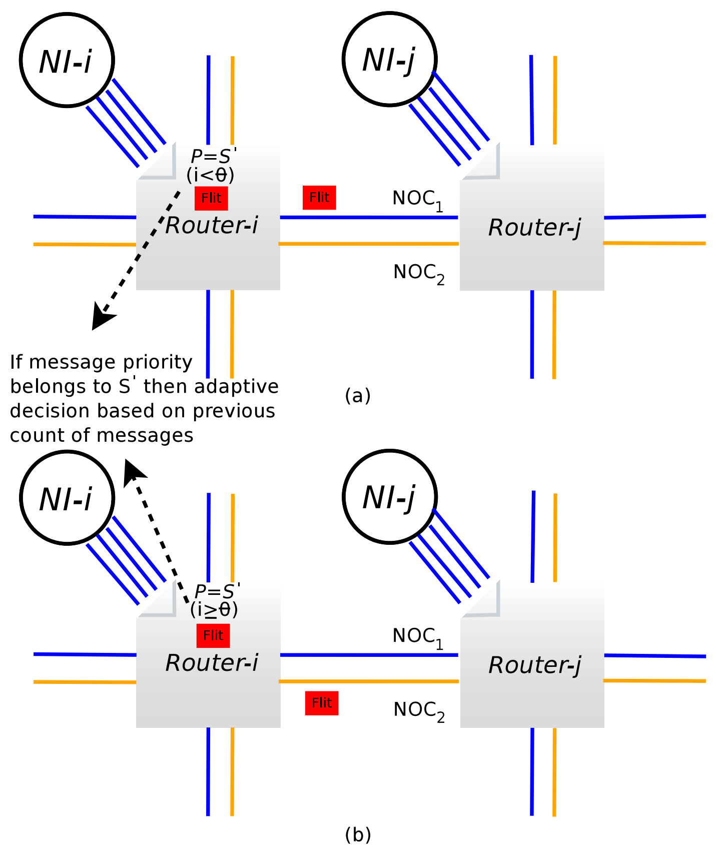


FIGURE 6.9: Runtime adaptive message distribution mechanism, the flit (red box) at the router selects the NoC network (a) when flag m_2 or m_3 is enabled while $i < \theta$, the NoC$_1$ is selected because of its underutilisation and (b) when flag m_2 or m_3 is enabled while $i \geq \theta$, the NoC$_2$ is selected as message counter exceeds θ.

The signal lines (each 1-bit) along with links from NI to the router carry information (flag bits) to identify the message classes (S_0 and S'). Signal lines are enabled on the arrival of the messages as defined in expressions (4)-(6). If the message type belongs to the class S_0, the message will redirect on NoC$_1$ as shown in Fig 6.8(a) which is decided by the adaptive message distribution hardware unit; if all flags/signals are disabled, it belongs to S_1 then redirect it on NoC$_2$ as shown in Fig 6.8(b). If the class is S' and $i < \theta$, then the message is forwarded on NoC$_1$ else on NoC$_2$ as demonstrated in respective Fig 6.9(a) and Fig 6.9(b).

6.4 Experimental Evaluation

The full system simulation configuration parameters[26] with Gem5 are listed in Table 6.2.

[26]For detailed parameter configuration, refer Table 3.1 to Table 3.3 of Chapter 3 (Page 58). Table 3.1 contains the details of the processor and benchmark parameters. Table 3.2 lists cache and memory details such as cache hierarchy levels, cache

TABLE 6.2: Configuration of Simulation Parameters

Parameters	**Configuration**[a]
Simulator	Gem5 integrated with GARNET for NoC interconnect
Cores	64 cores
Topology	8×8 Mesh
NoC Networks	Single-NoC/2-Network-NoC
Routing	Dimension Order XY
Traffic Distribution	Runtime Adaptive[b]
Caches	Three levels of cache hierarchy (two are private and last level is shared)
Cache Coherence	Directory-based MESI protocol
Main Memory	ddr3_1600_x64
Benchmark	PARSEC (medium workload, 16 threads)

[a]The hardware implementation cost is measured through synthesis using Synopsis Design Compiler.

[b]Adaptive distribution is compared with Static Traffic Distribution of Chapter 4.

The network traffic follows MESI cache coherence protocol which multicasts the coherence messages across the on-chip cores. We run multithreaded PARSEC benchmark in full system simulation of Gem5. The benchmarks are classified into three clusters as per the characteristics of the traffic as follows:

1) Communication-intensive workloads of PARSEC are *blackscholes, swaptions, canneal, fluidanimate, x264,* and *rtview*. These benchmarks generate a comparatively high volume of shared data. So communication traffic is high in NoC.

2) Offchip memory-intensive workloads are observed in *vips, facesim,* and *ferret*. These benchmarks generate a comparatively high volume of `Writeback Data`.

3) Computation-intensive workloads find in *bodytrack, dedup,* and *freqmine*. Such remaining benchmarks of PARSEC suite generate less number of on-chip traffic. So they are considered as computation-intensive applications.

Thus, we have compared proposed adaptive approach using different types of workloads.

size, associativity, and cache coherence protocol. Table 3.3 contains single- and dual-NoC parameters such as topology, its layout/size, number of routers, router pipeline, flow control, etc. The total hardware resources are kept the same between single- and dual-NoC for a fair comparison of both networks.

The PARSEC benchmarks with 16 threads and medium workloads require simulation time varying from 2-5 days with 64-bit Ubuntu 14.10, on HP ProLiant BL460c G7–Xeon E5649@2.53–2.93GHz blade server, having 6 Cores, 1TB Disk, and 16 GB memory for a single run of PARSEC application(s) with medium workloads[27]. Machine with 16 GB RAM is, sometimes, insufficient for large dataset, and time taken is more than a week for a single run of any benchmark. For the sake of experimental feasibility[28], we have selected medium workloads for all benchmarks.

6.4.1 Experiment Results and Discussions

In this section, we shall compare energy, throughput, and link utilisation of proposed adaptive approach with static traffic distribution and single-NoC. Fig 6.10 – Fig 6.12 demonstrate the relative energy consumption viz-a-viz single NoC. In these graphs, the static message distribution on NoC_1 and NoC_2 is compared with the runtime adaptive message distribution with different samples of threshold value $\theta = \{2, 4, 8, 16\}$ (Case I – IV). The additional energy consumption overhead of the adaptive distribution circuit is considered in results for the case I, case II, case III and case IV, however, the overall gains in energy with proposed approach make the overhead negligible.

The rightmost histogram of graphs indicates geometric mean[29] for all cases (static distribution, runtime adaptive distribution with different θ values) computed over all applications of the benchmark suite. The different graphs of static, dynamic, and total energy include the energy of the following NoC components.

- Static energy graph shows the total static energy consumption for the link, router, and clock energy of the network.
- Dynamic energy graph exhibits the links and router's dynamic energy.
- Total energy manifests static and dynamic energy for all the links and routers of 8×8 mesh network.

[27]Input dataset as in terminology of PARSEC.

[28]We were able to run all 41 combinations, as explained in Section 6.1.

[29]The geometric mean is only correct mean when averaging normalized results. The ranking/efficiency remains to preserve in comparison by the geometric mean, stays the same as the one obtained with unnormalized values [153].

The architecture of 2-network-NoC is more static energy efficient as compared to single NoC due to half of the core link-width. Since every router is connected to a core, this energy saving is appreciable as supported by experimental results. Likewise, crossbar energy of each router is proportional to (flit-width)^3 [61]. The flit-width is used in the proportion of link-width in 2-network-NoC that improves the saving in crossbar energy consumptions. Hence, the savings in router energy consumption contributed to an overall gain in energy consumption.

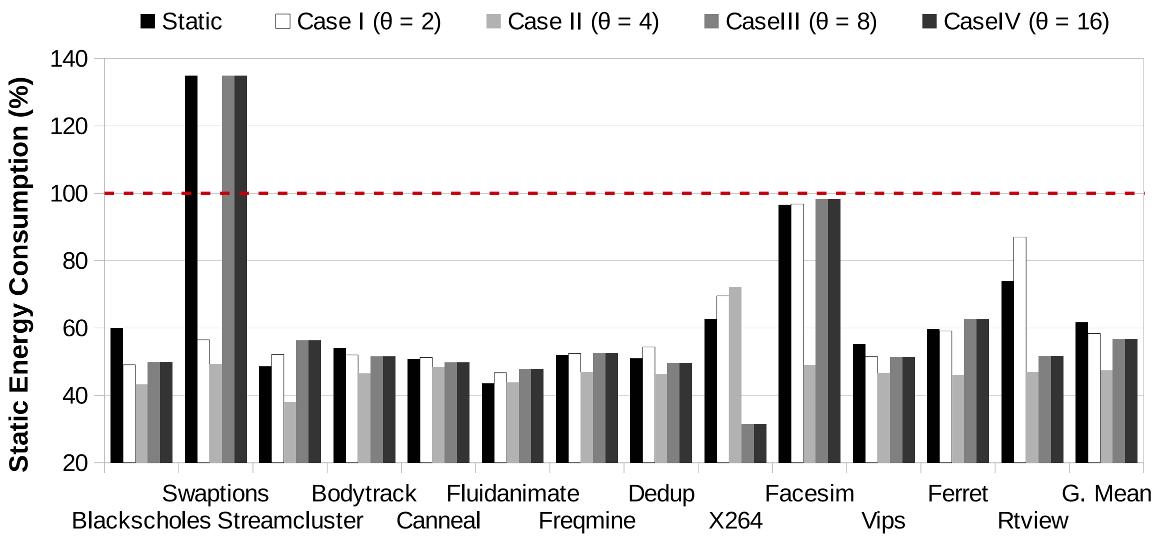


FIGURE 6.10: Static energy consumption for PARSEC benchmark suite. Each application compares runtime adaptive message distribution for the case I, case II, case III, and case IV on $\theta = 2, 4, 8, 16$ respectively with static message distribution (Chapter 4) and single-NoC [46] (The energy consumption of single-NoC scales to 100%, it depicts in graph with red dashed line).

The static energy consumption is saved up to 43%, as shown in Fig 6.10, and the dynamic energy consumption is saved up to 23%, as shown in Fig 6.11 as compared to single-NoC. The benefits in static energy come through the crossbar as data transmission bit rate (R) and the maximum number of data flow (N) across the switching fabric is different for 2-network-NoC as compared to single-NoC. The total power consumption of $N \times N$ crossbar is proportional to $R^3 \times N$ [66]. The value of R and N for single-NoC[30] (2-network-NoC) is 16 (8) and 8 (12). Thus, power consumption for single NoC router crossbar is 32768 nW[31], and 2-network-NoC crossbar is 5144 nW. This is a significant gain in 2-network-NoC because of reducing the value of R and N as compared to single-NoC.

[30]Refer to Table 3.3
[31]nW-nanowatt

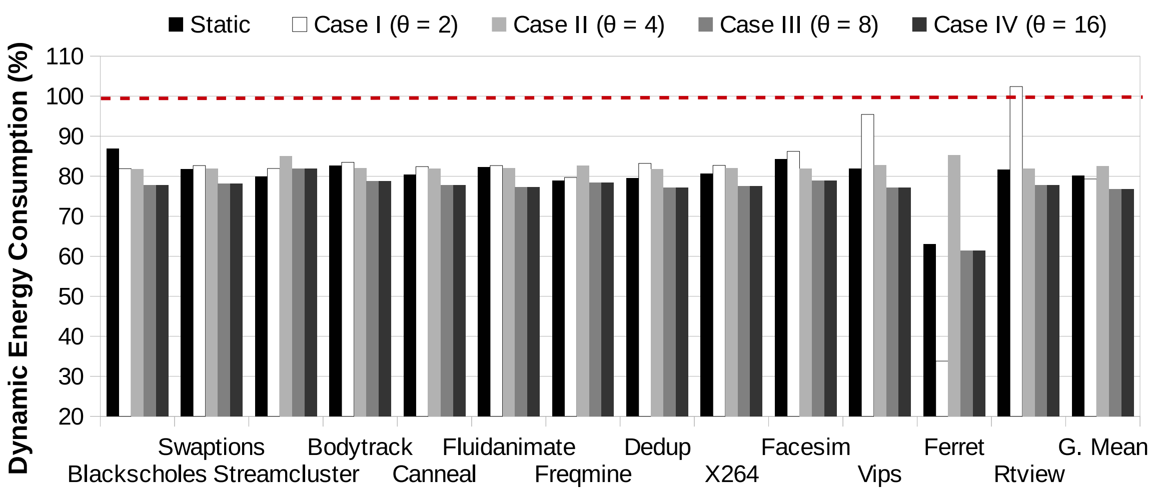


FIGURE 6.11: Dynamic energy consumption for PARSEC benchmark suite. The rightmost histogram compares the geometric mean for all the cases.

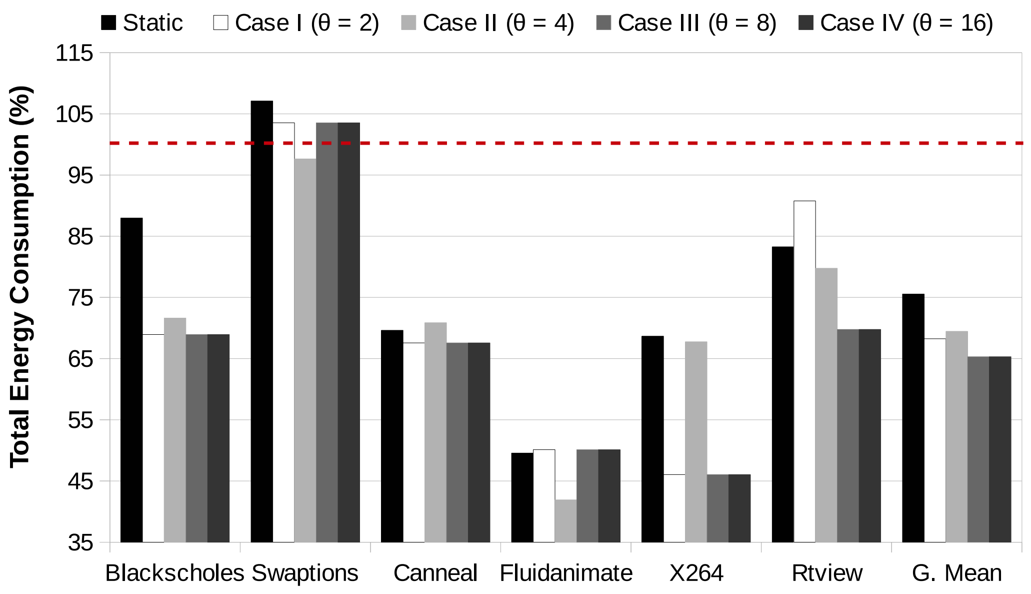


FIGURE 6.12: Saving in total energy (ξ) consumption for communication-intensive benchmarks.

On average, best results are obtained for adaptive traffic distribution with threshold $\theta = 8$. For this value of θ, we observe a maximum reduction in static energy up to 8% as shown in Fig 6.10 and dynamic energy consumption up to 5% as shown in Fig 6.11 as compared to static distribution.

The dynamic energy reduces because of the adaptivity in the selection of a network $\{NoC_1, NoC_2\}$ that reduces waiting time of control messages in buffers. The total energy, as shown in Fig 6.12, comprises the total of static and dynamic energy of all the links and router components of the network. For accurate analysis, we are more focused on communication-intensive applications of PARSEC as it comprises communication-intensive, memory-intensive, and computation-intensive workloads.

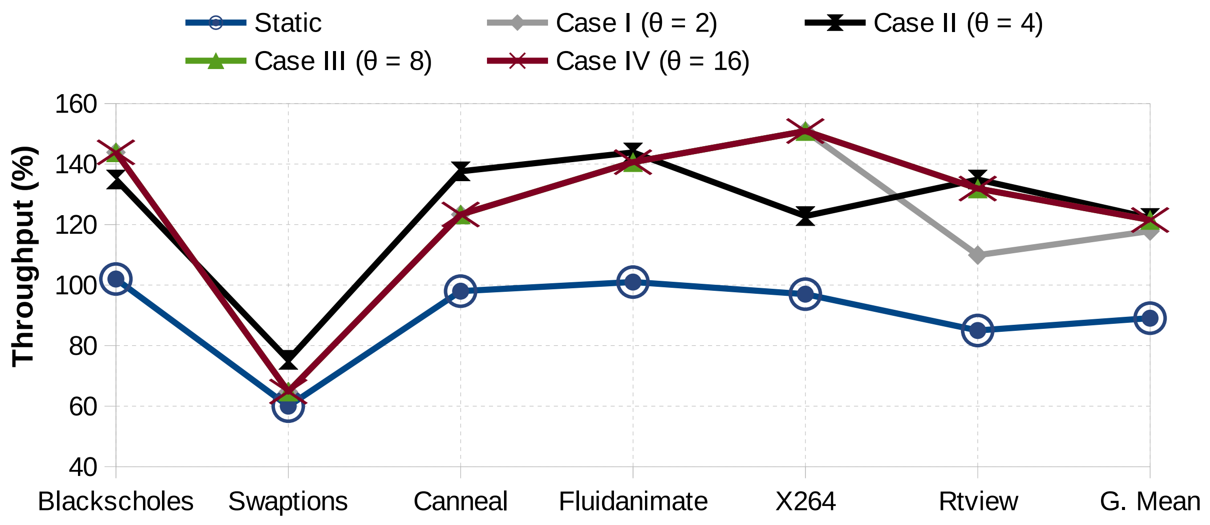


FIGURE 6.13: Percentage-wise improvement in throughput (τ) for communication-intensive benchmarks across static (Chapter 4) and runtime adaptive cases.

Communication-intensive applications show 14% improvement in mean total energy that is more than the memory-intensive (10%) and the computation-intensive (4%) application's benefits. The proposed adaptive approach reduces the communication latency of the channel by supporting the communication of coherence traffic. Therefore, we get more benefits in communication-intensive applications as compared to other PARSEC applications.

With the proposed adaptive scheme, throughput (τ) improves 36% as shown in Fig 6.13 for the communication-intensive and improves 23% (in Fig 6.14) for the memory-intensive applications of PARSEC as opposed to static approach wherein 10% decrease is observed[32].

Except for the *swaption*, communication intensive benchmarks have no delay overhead[33]. The maximum variance of 5× is observed for *swaption* benchmark (refer Fig 6.15). This shows that the proposed traffic distribution or threshold θ[34] value is not suitable for the *swaption*

[32]We are not considering the throughput improvement for computation-intensive workloads as they have a minor gain for the same.

[33]The flit size of single-NoC is 16 B whereas the flit size for 2-network-NoC is 8 B.

[34]case-III (θ = 8)

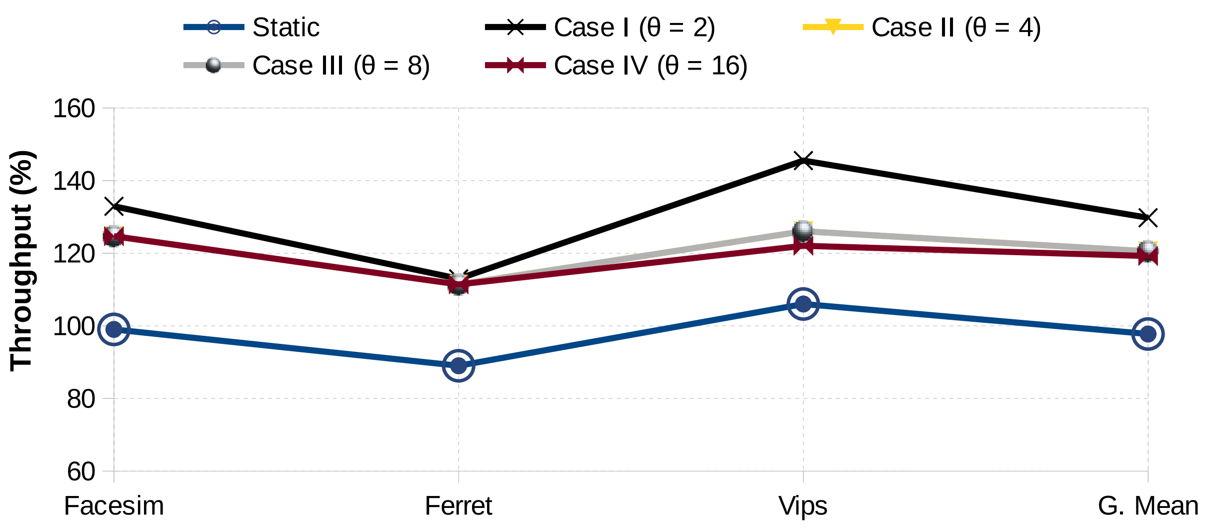


FIGURE 6.14: Percentage-wise improvement in throughput (τ) for memory-intensive benchmarks across static (Chapter 4) and runtime adaptive cases.

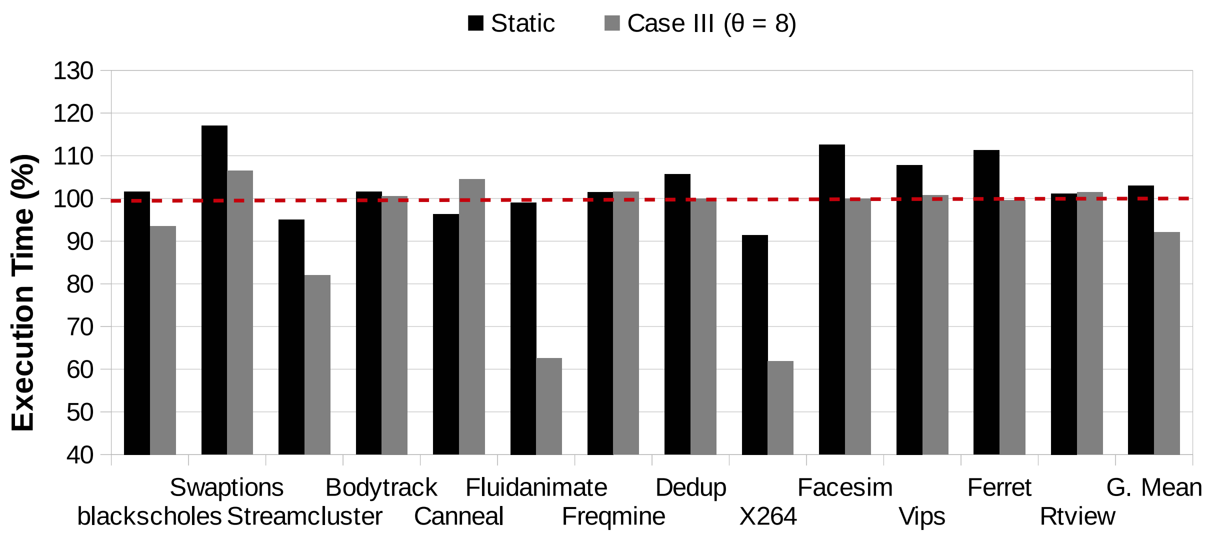


FIGURE 6.15: Percentage improvement in the execution time of 2-network-NoC, which is normalized with respect to single-NoC.

benchmark. While for the computation-intensive benchmarks, the execution time of 2-network-NoC does not exceed the time of single-NoC.

In memory-intensive benchmarks, the execution time for *facesim* and *ferret* have some overhead with static distribution over single-NoC. The proposed adaptive approach is effective for communication intensive benchmarks. We get 8% average gain in execution time over single-NoC. While static distribution has the 3% delay over single-NoC.

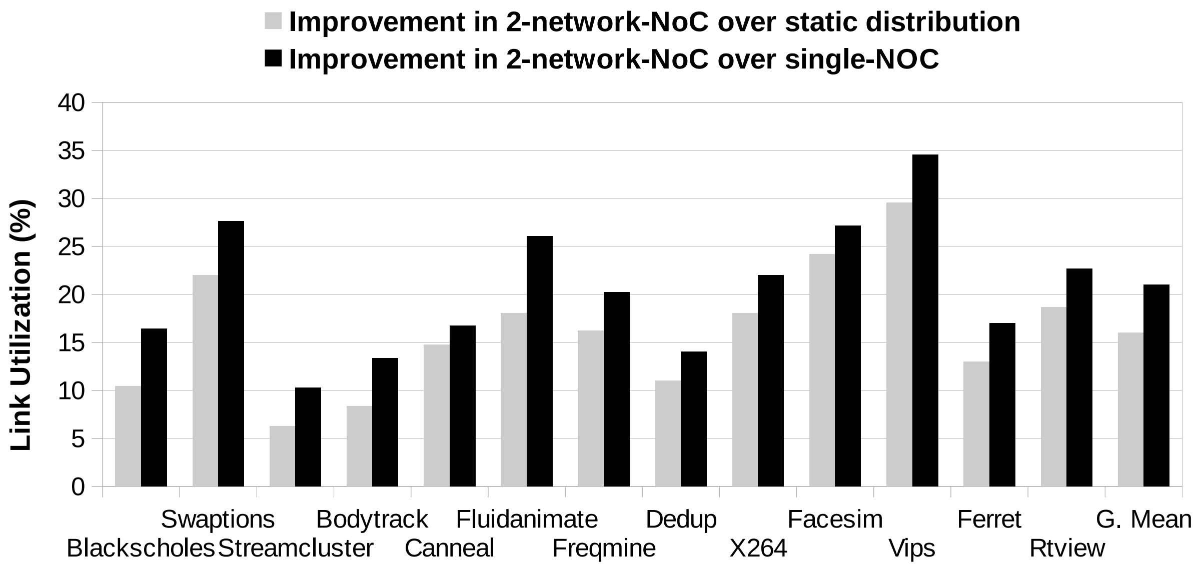


Figure 6.16: Percentage improvement in the link utilization of 2-network-NoC with respect to single-NoC and static traffic distribution.

The link utilization also increases by 16% and 21% over static message distribution and single-NoC, respectively, with proposed adaptive approach, as shown in Fig 6.16. It shows the percentage improvement on the average of all the link utilization values. A link is known as utilized if it sends a flit during a particular clock cycle. The average link utilization value indicates the efficiency of NoC to utilize hardware resources and thus links of NoC networks. Hence, the improvement in link utilization shows more utilization of physical channels and remains less idle as compared to single-NoC architectures. Although the benefits are different for individual PARSEC benchmarks, as each application has different runtime traffic patterns, so it causes the variation in results.

We have also done synthesis of adaptive message distribution hardware unit and static traffic distribution in Verilog using Synopsys Design Compiler[35] to compare their hardware overheads. The area and power overhead for adaptive distribution are $81.1 \times 10^{-12}\ m^2$ and $78.89 \times 10^{-6}\ w$ respectively. The power consumption of our proposed approach is 13% less as compared to

[35]At 90nm technology library

static distribution due to gain in dynamic power though area overhead is 4% more as compared to static traffic distribution. But, it can be considered almost negligible.

Thus, the offered flexibility of 2-network-NoC can exploit traffic characteristics to efficiently distribute workload between the networks. The proposed adaptive scheme is flexible, scalable, efficiently handle real-time variations in traffic dynamics.

6.5 Inferences

Static traffic distribution for multiple NoC network is suitable for only application-specific processors. For general purpose processors, static traffic distribution is unsuitable. This limitation is substantiated by the case study of static traffic distribution on the PARSEC benchmark. Since static traffic is configured once at design time without taking into account the traffic dynamics. One NoC network is underutilised while another network is suffering from over-utilisation by traffic.

We propose the integration of an adaptive message distribution hardware unit to the router to alleviate the deficiencies of static traffic distribution. The router can adapt itself for traffic distribution between NoC networks with workload variations at different execution instances of the underlying application(s). Adaptive hardware unit offers the ability to create customised workload flows at runtime between multiple NoC networks. The workload variations are dynamic, and there is a continuous variation while the application is running on the processor because of the fine-grained traffic variations. These messages are generated during the transition between cache states.

We monitor the runtime dynamics of the traffic through adaptive hardware unit to adapt fine-grain message distribution between multiple networks of multi-NoCs. The adaptivity policy is devised through analysis of the volume and quality of fine-grain control messages using cache states transitions. The adaptive message distribution hardware unit enables efficient utilization of multiple NoC networks for high throughput execution of network resources. It quickly detects underutilised NoC at run-time, and it switches the mid-level critical messages to underutilised NoC. The most critical and noncritical are dedicated to individual NoC networks. The three levels of criticality are devised through analysis of fine-grain messages. The allowed window size of checking for underutilised/overutilised is done through offline sampling. Thus, adaptive router quickly responds to variations under dynamic workload and resource variations.

Our proposed adaptive traffic distribution attains energy saving up to 36% and 14% as compared to the single-NoC and static traffic distribution respectively for communication-intensive PARSEC benchmarks suite, and the link utilization improves 16% and 21% over static message distribution and single NoC. The proposed adaptive technique can further optimize for different power-performance or energy tradeoff as it considers unknown dynamic workload scenarios, diverse application requirements, and characteristics.

Thus, software level solutions proposed in this chapter complements the hardware level custom implementations proposed in Chapter 4 and 5. The next chapter summarizes our contributions proposed throughout the thesis.

Chapter 7

Conclusions

In this last chapter of the thesis, we review the primary findings. We present a consistent overall picture of our novel contributions. We also detail here a few remaining open problems and some interesting future research ideas for continuation of this thesis work.

7.1 Thesis Contributions and Outcomes

The objectives of modern general purpose processors are

1) achieving good power-performance tradeoff and energy efficiency
2) efficient distribution of traffic through NoC while considering the variety of workloads.

Our novel thesis contributions include the following.

1) Proposed custom-made NoC architectures to improve static power efficiency. Increase in static power with increased miniaturization of IC components is more as compared to dynamic power.
2) Efficient traffic distribution even when workload is a mix of computation, communication, and memory-bound applications. Another problem of CMPs as the workload variations are not taken into consideration by conventional static traffic distribution methods. The older workloads, based on multiprogramming, used for performance evaluation were

largely computation intensive. Recent multithreaded workloads on shared cache architectures may be communication intensive owing to memory-bound traffic that needs to be catered by NoC.

Therefore, recent research is more focused on improvement of power and energy metrics by custom hardware implementations and efficient traffic distribution through NoCs. These objectives are achieved using multiple NoC architectures. Various research prototypes and commercial implementations motivate us to work on multiple-NoC and customise these architectures at hardware and software levels in different ways to achieve aforementioned design objectives.

The most of the multi-NoC work has targeted application specific processor, we, however, address power-performance efficient multiple NoC for the general purpose processors. In brief, the outcomes of this thesis work are

1) a novel custom-made 2-network-NoC architecture for improved static power efficiency (Chapter 4).
2) placement of network demultiplexer on non-critical path along router for improving static power efficiency (Chapter 5).
3) an adaptive message distribution technique for fair distribution of traffic (Chapter 6).

In Chapter 4, we customize dual-network-NoC for static power efficiency without affecting the flexibility of dual networks. These NoC architectures have parallel dual NoCs which originate from NI. In proposed customisation, NI links remain single and now parallel NoC networks start from the router. We named our proposed architecture as **custom-made 2-network-NoC** architecture.

The single/common NI links carry the traffic of both NoC networks. Backpressure mechanism ensures that nearly-full buffers trigger a signal to reduce injection rate at the source core. Thus, single NI links shall not affect traffic injection rate, and duplication of NoC links speed up communications through the availability of two parallel NoC networks.

This significantly reduces hardware cost (area) and improves static power and energy because of half number of the core links that reduces the number of I/O ports of the router, the number of VNs and VCs, routing logic and control logic overhead, size of the crossbar, etc.

In addition to these changes, we need to place a network selector hardware unit somewhere on router. We propose its placement at the routing unit within the router. For synthesized proposed NoC

- The results show 62% and 58% improvement in static power, and 30% and 25% area efficiency of proposed 2-network-NoC over conventional dual-network-NoC and single-NoC respectively at 32*nm* technology.
- Experiment on PARSEC benchmarks finds 45% efficiency in total router power at $(65nm, 1GHz)$. The power efficiency approaches 58% as technology shrink to 32*nm* at 1*GHz* frequency. In contrast, the power efficiency limits to 40% as frequency increases to 2.5*GHz* at 65*nm* technology.
- A minor performance overhead incurred as throughput decreases (5%) and latency increases (4%) since the bandwidth of core links is half of the single-NoC.

Our objective of low power router design with dual NoC networks is successfully achieved.

In Chapter 5, placement of network selection hardware unit is explored for static power and energy efficiency with proposed 2-network-NoC router design. We have proposed placement of `Net-Demux` at the switch allocator of the router for 2-network-NoC rather than routing unit. As the input and output of network demultiplexer are driven by different modules of the circuit, placement changes the switching of the circuit through variation in the average number of signal transitions per cycle. We have compared switch allocator placement with routing unit and traditional network interface placements and have obtained the following results.

- Through synthesis, we determine that the switch allocator placement is 21% and 41% static power efficient over network interface placement of dual-Network-NoC and single-NoC respectively. This is 29% improvement over routing unit placement.
- Experiments show that switch allocator, routing unit, and network interface placements are 46%, 40% and 30% energy efficient over single-NoC on PARSEC benchmarks.
- We also observe switch allocator and routing unit are approximately 33% and 26% more energy efficient over the network interface.

- We achieve 6% improvement in execution time with switch allocator over single-NoC whereas routing unit and switch allocator placements are 6% and 14% faster over the network interface.

Thus, our proposed switch allocator and routing unit placement significantly improve the energy efficiency of proposed NoC.

In Chapter 6, we propose adaptive traffic distribution to consider runtime dynamics of messages. For general purpose processor, static message distribution offers limited benefits to accommodate the worst-case traffic requirements, or inevitably lead to a degradation in final power-performance and energy efficiency. Static message distribution may cause underutilisation of one NoC network during different runtime instances while another NoC network is overutilised due to an imbalance in traffic loads assigned to two networks. This makes traffic traversal slower and degrades the performance while underutilised network resources are not in use. Our proposed case study on static traffic distribution using PARSEC benchmark shows up to $\approx 5\times$ variation in power and energy.

To alleviate these deficiencies, we have proposed runtime adaptive traffic distribution. Adaptivity allows messages to change the NoC networks as per their utilization status. The hardware area and power overhead of proposed microarchitecture are measured by implementing in Verilog and synthesized at 90 nm technology.

- Compared to the static method, adaptive hardware has a minor area overhead of 4% whereas it is 13% power efficient.
- Adaptive method offers energy saving up to 36% and 14% compared to the single-NoC and static distribution of traffic on multiple NoCs, respectively.
- The link utilization improves 16% and 21% over static message distribution and single-NoC.

Thus, our proposed adaptive traffic distribution overcomes the limitations of static traffic distribution. It reshuffles the traffic distribution on both the networks to improve the runtime underutilisation/overutilisation of the network.

Our thesis contributions are scalable with large network size, static power, energy, and area efficient hardware and software level implementation of 2-network-NoC. Our contribution at software level traffic customisations exploits criticality of fine-grain messages of cache memory hierarchy for distributing NoC traffic communication.

7.2 Future Directions

Some interesting open problems that are related to the work in this thesis are as follows.

7.2.1 Routing Customisations

The 2-network-NoC architectures can be explored for **different routing** algorithms. Different routings on separate NoC networks distribute the workload across the network. These may reduce traffic hotspots and improve network performance. The proposed architecture can also be extended for increasing the number of NoC networks and compare its power performance trade-offs with other multi-NoC networks.

7.2.2 Crossbar and 2-/dual-network-NoC Customisations

As the crossbar of the router primarily consumes power and covers a large area, a variety of crossbar integration such as **thin versus wide links of multiple crossbars** for energy efficiency can be explored for different multi-NoC architectures. Different placements of `Net-Demux` with a different number of the crossbar and multiple NoC networks may be explored. These changes in the router architecture affect the area, power and path delay of NoC. Therefore, these customised architectures have different power-performance trade-offs.

It will be interesting to explore the impact of placement on critical path delay of the router designed with different pipelined architectures. Significant research has been done on reducing router pipeline stages through bypassing and speculation techniques for improving network throughput. As bypassing or speculation failure results in a longer critical path delay, exploration of placement with reduced pipelined router architecture is suitable only for low traffic workloads.

7.2.3 Traffic distribution with approximate load balancing

Other than architectural improvements, analysis of traffic distribution methods plays a vital role in improving multi-NoC efficiency. The proposed case study on static traffic distribution, in Chapter 6, can also be extended for broadcast cache coherence protocols.

Approximate load balancing algorithms can be used for further improvement in runtime under-utilization of network links as in Chapter 6. These algorithms can be used to further explore the runtime variations in mid-level critical messages by allowing runtime changes in message priorities and the threshold value. Though the careful design is required for **approximate algorithms** to maintain power-performance trade-off.

7.2.4 3D planar 2-Network-NoC

The advantage of using 3D is the decreased hop count due to smaller network size in each layer. Though, these architectures scale inefficiently because of the area and power overhead with a higher number of ports of the router. This increases the complexity of microarchitecture components, i.e., crossbar, buffer space, routing logic and arbitration logics (in both virtual channel allocation (VA) and switch allocation (SA)) [20]. Though, our proposed 2-network-NoC can be extended in 3D planar to combine the benefits of plane extension with multiple NoC to achieve area, power, and energy efficiency over conventional 3D planar single-NoC.

7.2.5 Asynchronous 2-Network-NoC

The proposed 2-network-NoC can be extended for asynchronous multiple NoCs architectures that partitioned networks into multiple independent frequency domains. Each domain is clocked synchronously, while inter-domain communication is achieved through specific circuit design techniques.

Appendix A

Chip Multi-Processor Architecture

Chip Multi-Processor (CMP) are designed as arrays of identical building blocks known as tiles. Each tile comprises a processing core, caches, network interface, and router. The routers of all tiles are connected through a mesh NoC interconnect. The tiled architecture of CMPs supports families of products with varying number of tiles by including the option of connecting multiple separately tested dies within a single package. Therefore, they become the choice for modern many-core CMPs. In our thesis, we have considered a tiled CMP with three levels of cache

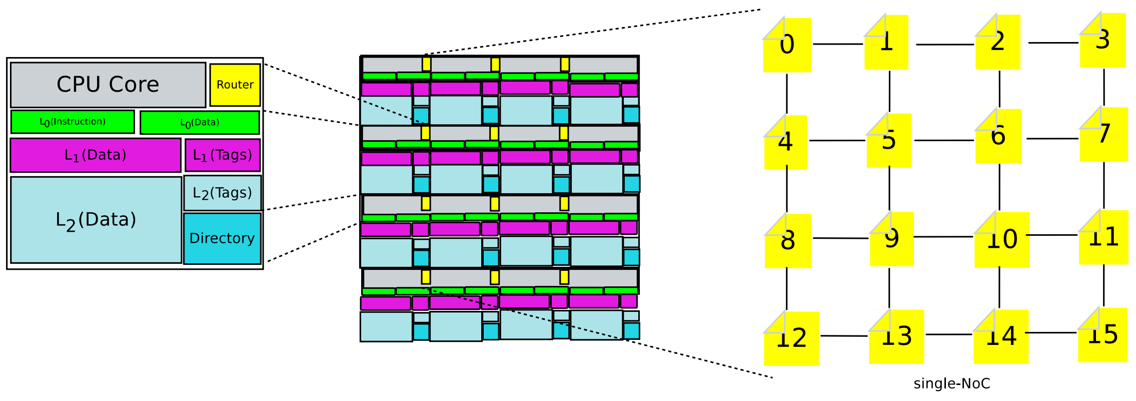


FIGURE A.1: Tiled architecture of chip multi-processor with three level of cache hierarchy and one directory along with core and router. The presented tiled architecture of CMP is extended from two level cache hierarchy architecture proposed in [143].

hierarchy as shown in Fig A.1. The L_0 and L_1 cache is private to its local processing core. In contrast, the third one (L_2 cache) is logically shared but physically distributed among the processing cores. Therefore, each cache block maps to a particular L_2 cache bank, which is called the home tile for that block. The home bank of each block is obtained from its address bits. The bits chosen for the mapping to a particular bank are the less significant ones without considering the block offset [144–146].

Appendix B

Directory-Based MESI Protocol

Each tile includes an on-chip directory cache that stores the sharing and owner information for the blocks that it manages. This cache is used for the blocks that do not hold a copy in the shared cache. Besides the tags' part of the shared cache also includes a field for storing the sharing information of those blocks that have a valid entry in that cache. The directory-based protocol[1] keeps track of the sharer through a full-map (or bit-vector) that allows the protocol to send invalidation messages just to the caches currently sharing the block. On every cache miss, the core that causes the miss sends the request only to the local home tile, which is the serialization point for all requests issued for the same block.

Once the home tile decides to process the request, it accesses the directory, and it performs the appropriate coherence actions. These coherence actions include forwarding the request to the owner tile and invalidating all copies of the block in case of write misses. When a tile receives a forwarding request, it provides the data to the requester if it is already available or, in another case, the request must wait until the data is available. Since the home tile sends this information that knows the number of invalidation messages issued to the requester along with the forwarding and data messages. When the requester receives all acknowledgments and the data block, data can be accessed.

Fig B.1 shows an example of how Directory-CMP solves a cache-to-cache transfer miss. The request (`1.GetX`) is sent to the home tile, where the directory information is stored. Then, the home tile forwards (`2.Fwd`) the request to the provider of the block, which is obtained from the directory information. The provider sends the unblock message (`3.Unbl`) to the home

[1]Example of directory protocol implementation is Piranha. This is a research prototype developed by Compaq [152].

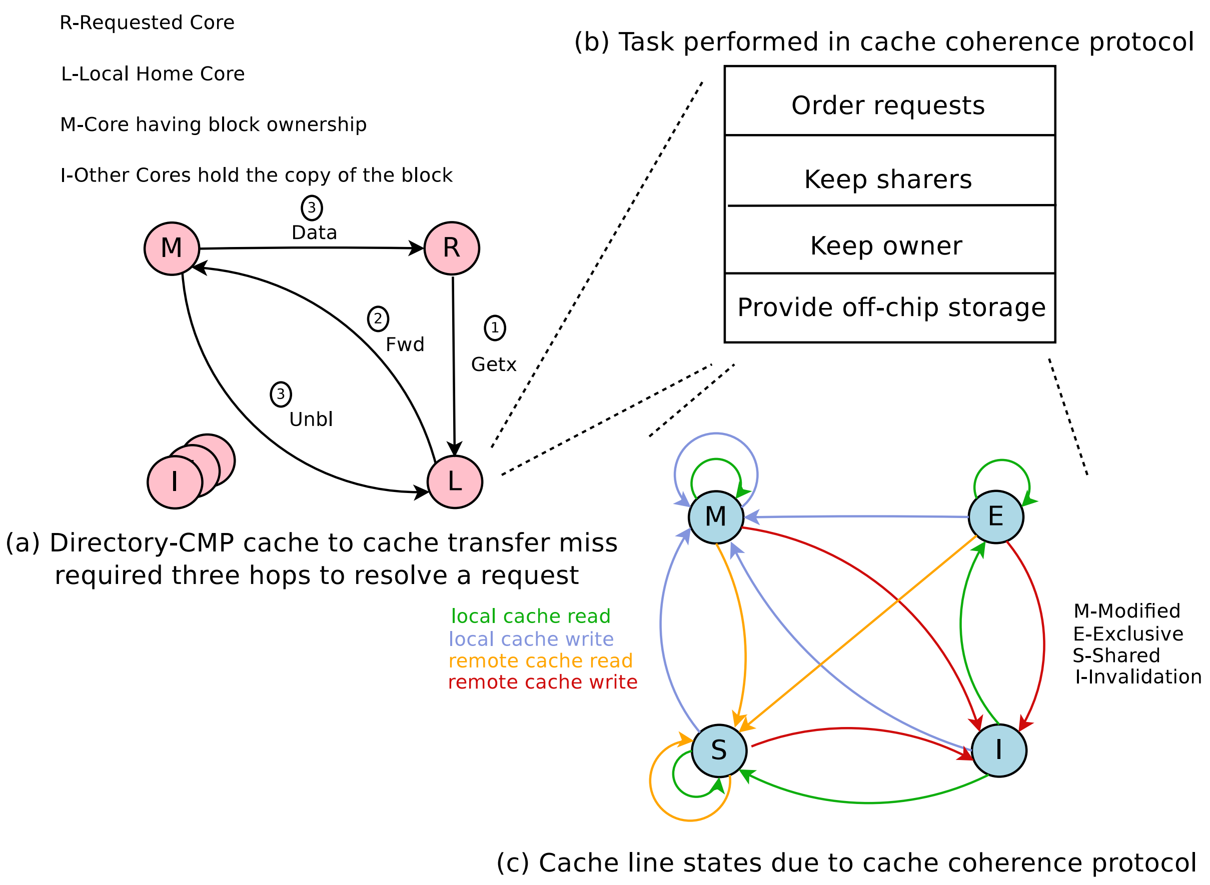

R-Requested Core
L-Local Home Core
M-Core having block ownership
I-Other Cores hold the copy of the block
3 Data
2 Fwd
1 Getx
3 Unbl
M
R
L
I
(a) Directory-CMP cache to cache transfer miss required three hops to resolve a request
(b) Task performed in cache coherence protocol
Order requests
Keep sharers
Keep owner
Provide off-chip storage
local cache read
local cache write
remote cache read
remote cache write
M
E
S
I
M-Modified
E-Exclusive
S-Shared
I-Invalidation
(c) Cache line states due to cache coherence protocol

FIGURE B.1: Directory Protocol (a) cache to cache transfer miss required minimum three hops to resolve a request (b) tasks performed in cache coherence protocols by local home tile (c) the cache line state of local home cache varies the states between `M, E, S,` and `I`.

tile to allow subsequent requests to be processed, and it also sends the data (`3.Data`) to the requester. When the data block arrives at the requester, the miss is considered solved. As we can see, although this protocol introduces three hops in the critical path of the miss to solve cache misses, few coherence messages are required to solve them which eventually translates into savings in network traffic and less power consumption[2].

The following tasks are performed in directory coherence protocols by local home tile, as illustrated in Fig B.1 (b).

1) Order requests: Cache coherence maintenance requires to serialize the requests issued by different cores to the same block. Directory protocols assign this task to the home tile of each memory block.

[2]This characteristic allows the directory protocol to scale up to a higher number of cores.

2) Keep coherence information: Coherence information is used to track blocks stored in private caches. Directory protocols store coherence information at the home tile, where cache coherence is maintained.

3) Provide the data block: If the valid copy of the block resides on chip, data is always provided by the owner tile, since it always holds a valid copy. The owner of a block is either a tile holding the block in the exclusive or the modified state, the last core that wrote the block when there are multiple sharers or the shared cache bank within the home tile in case of an eviction of the owner block from private cache.

4) Provide off-chip storage: When the valid copy of a requested block is not stored on a chip, off-chip access is required to obtain the block. The local home tile is responsible for detecting that the owner copy of the block is not stored on chip. It is also responsible for sending the off-chip request and receiving the data block.

We have used the MESI protocol to maintain cache coherence [84]. The selected cores share the same copy of data in multicast communication, and these cores communicate with each other through NoC. Each home tile cache states vary between **M** (Modified), **E** (Exclusive), **S** (Shared), and **I** (Invalidation). The protocol is named after the four states a cache line can be in when using the MESI protocol.

1) Modified (M): The local core has modified the cache line. This also implies it is the only copy in any cache.

2) Exclusive (E): The cache line is not modified but known not to be loaded into any other cache.

3) Shared (S): The cache line is not modified and might exist in another core's cache.

4) Invalid (I): The cache line is invalid, i.e., stale copy of data.

The cache line does transition between these four stable states by triggering fine-grained messages to communicate with local/other caches and memory. These messages maintain the data consistency as well as cache coherence among all the shared caches and off-chip memory. The fine-grained[3] messages perform the specific job of respective mid-level message class as

[3]Here, we have not discussed fine-grained messages because of nonrelevance with the scope of this section.

highlighted with their matching color in Fig B.2. This figure is the L_0 cache state graph that demonstrates all cache state transitions and respective messages (control and data types).

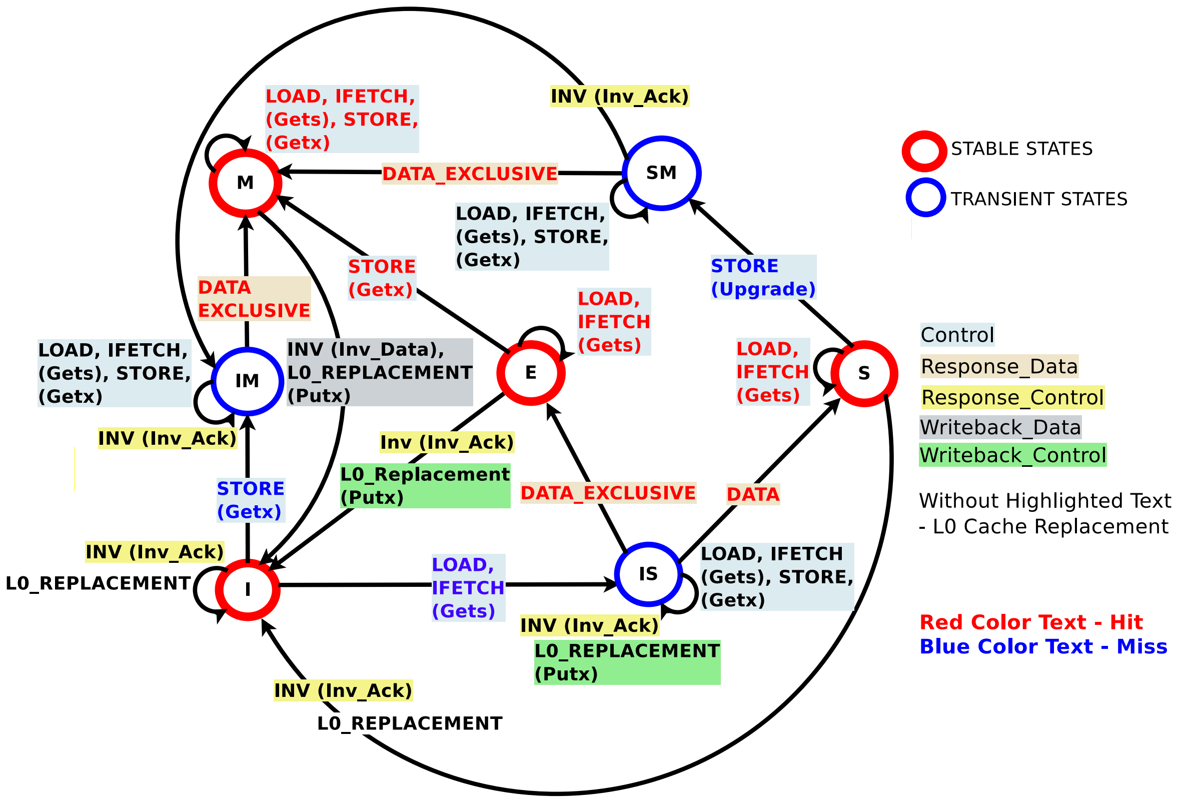


FIGURE B.2: State space diagram of L_0 cache for MESI 3–level cache coherence protocol.

Each mid-level message class of MESI is associated with specific functionality as follows.

1) Control (C_C) messages are associated with invalidation and upgrade events.

2) Request Control (R_{Q_C}) messages are generated at shared caches for data block replacement.

3) Response Control (R_{P_C}) are acknowledgements initiated on receiving response data.

4) Writeback Control (W_C) informs private caches about the modified copy of data.

5) Response Data (R_{P_D}) carry data messages between on-chip caches and off-chip memory to cache.

6) Writeback Data (W_D) traverse from cache to off-chip memory.

Initially, all cache lines are empty and hence also Invalid. If data is loaded for writing, the cache state changes to Modified. If the data is loaded for reading from another core, the new cache state is Shared, otherwise Exclusive. If a Modified cache line is read from or written to on the local core, the instruction can use the current cache content, and the state does not change. If another core wants to read from the cache line, the first core has to send the content of its cache to another core and then it can change the state to Shared. The data sent to another core is also received and processed by the memory controller, which stores the content in memory.

If another core wants to write to the cache line of the first core. It sends the cache line content and marks the cache line locally as Invalid. Performing this operation in the last level cache, just like the I→M transition is comparatively expensive. Since this transition happens between multiple shared caches located on a different core; basically, this is one-to-many (many-to-one) transition occurs for multicast MESI protocol.

If a cache line is in the Shared state and the local core reads from it, no state change is necessary, and the read request can be fulfilled from the cache. If the cache line is locally written to the cache line can be used as well but the state changes to Modified. It also requires that all other possible copies of the cache line in other cores are marked as Invalid. If the cache line is requested for reading by another core, nothing has to happen. The main memory contains the current data, and the local state is already Shared. In case, another core wants to write to the cache line that will be marked as Invalid.

The Exclusive state is mostly identical to the Shared state except a local write operation does not have to be announced on the NoC. The local cache copy is known to be the only one. This can be a huge advantage so the core will try to keep as many cache lines as possible in the Exclusive state instead of the Shared state. The latter is the fallback in case the information is not available at that moment. The Exclusive state can also be left out completely without causing functional problems. It is only the performance that will suffer since the E→M transition is much faster than the S→M transition.

From this description of the state transitions, it should be clear where the costs specific to manycore operations are. Yes, filling caches is still expensive but now we also have to look out for M→I transition. Whenever such a message has to be sent, things are going to be slow.

Appendix C

Network-on-Chip: Preliminaries

This section introduces Network-on-Chip, and discusses its basics in sufficient detail for understanding the thesis. Necessary background on metrics of NoC design validation is also presented that will be used by thesis chapters to assess our contributions.

C.1 Evolution of Networks-on-Chip

Traditional buses interconnect cores through a single communication channel. Few examples of the bus interconnect architectures are ARM, AMBA [9], IBM CoreConnect, and Tensilica PIF Interface [76]. The simplicity of buses makes them resource efficient, but scalability is limited to a modest number of cores. Per core capacity of bus interconnect is defined as $C = \frac{2B_B}{N} = \frac{2bB_C}{N}$, where B_B is bisection bandwidth, N is the number of cores, b is channel bandwidth, and B_C is bisection channel bandwidth. An AMD Athlon[1] processor is exemplified in Fig C.1 wherein two cores are connected through the bus interconnect having bandwidth $\frac{8GB}{s}$ as illustrated in Fig C.2. Further increase in the number of cores reduces per core capacity of delivered bandwidth. The bus reaches its practical limit after a certain extent. Bandwidth is saturated and finally network traffic jams on exceeding delivered bandwidth (B_D) over the offered bandwidth (B_O), $B_D \geq B_O$, as shown in Fig C.3.

Crossbar implementation of IBM Cyclops64 emerges as an alternative interconnect architecture. It connects 80 custom processors and about 160 memory banks. A 4×4 crossbar is exemplified in Fig C.4, and its floorplan in Fig C.5 wherein each core is connected with rest of

[1] http://www.overclock.net/products/athlon-64x2-6400-black-edition

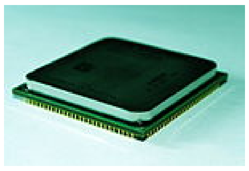


FIGURE C.1: AMD Athlon[1] X2 6400+ dual–core processor(90nm technology, 8GB/s on chip bandwidth, 20.8 GB/s offchip bandwidth, 3.2GHz clock frequency)

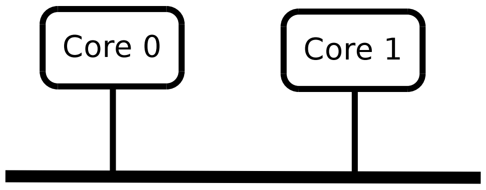


FIGURE C.2: 1 Dimensional (D) bus connects two cores[1]

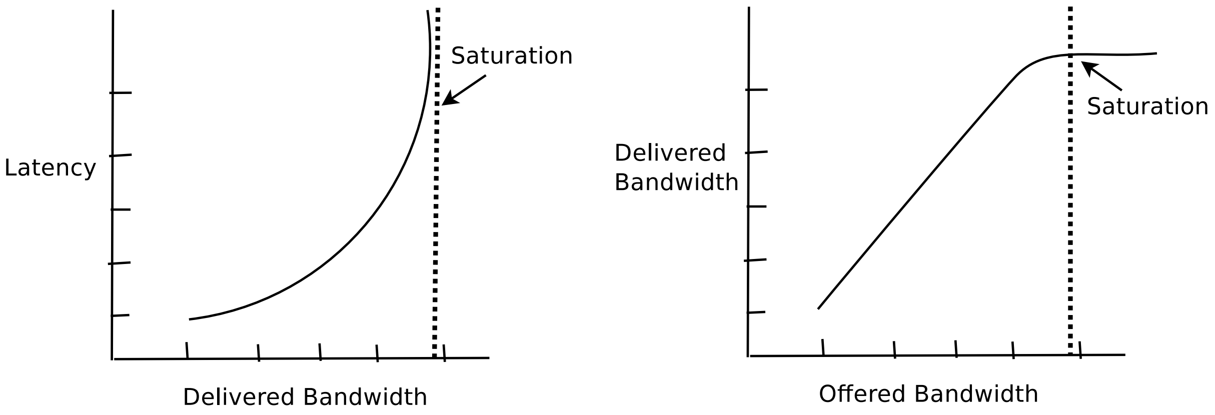


FIGURE C.3: Relation between Network Bandwidth vs Latency[3]

the cores. Floorplan of 128 × 128 crossbar at 90nm technology is proposed by Passas[2] et al., it provides 24/s bandwidth at the expense of 150mm^2 area. The complexity of area = O(N^2W), delay = O(N $\sqrt{W}$), and power = O(N^2) make it unsuitable due to the non-linear relation between the number of ports, latency and wire costs [3] for the increasing number of cores. Where N is the radix of a node, and W is the width.

Network **o**n **C**hip (NoC) emerges as a promising interconnect solution for manycore processors over the last decade. NoC is a subset of a broader class of interconnection networks that facilitate the transporting of data between cores. It consists of routers, links and network interfaces that are arranged in some specific pattern. Scalability, modularity and structured nature of wires are the main advantages of NoC interconnect in comparison to traditional means of communication. MIT's RAW was the first chip with four on-chip mesh networks [68]. STNoC is the product of ST Microelectronics [14]. It is targeted to replace the widely-used STBus in **M**ulti**P**rocessor **S**ystems-on-**C**hip (MPSoCs) using ring topology for interconnections. Subsequently, Tilera's TILE64 and TILE64Pro architectures [12, 13] are optimized for intelligent networking, multimedia and cloud applications, and delivers remarkable computing

[2]G. Passas, M. Katevenis, D. Pnevmatikatos, "Crossbar NoCs Are Scalable Beyond 100 Nodes," Computer–Aided Design of Integrated Circuits and Systems, IEEE Transactions on , vol.31, no.4, pp.573-585, April 2012.

[3]www.ece.eng.wayne.edu/ czxu/ece7660_f05/network.pdf

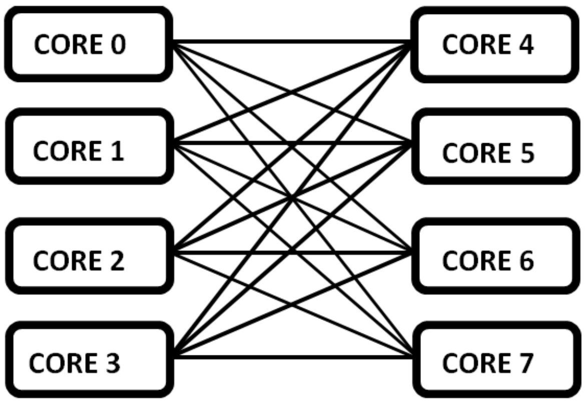


FIGURE C.4: A 4×4 crossbar interconnect

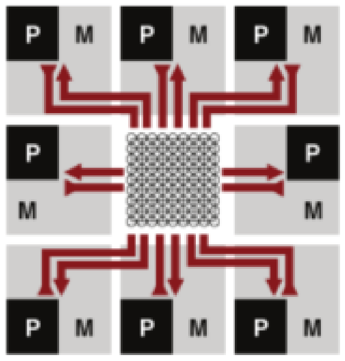


FIGURE C.5: Floorplan of 4×4 crossbar[2]. Centralized crossbar interconnect tiles (Ps denote processors and Ms memories.)

and **I**nput/**O**utput (I/O) with complete **S**ystem-**o**n-a-**C**hip (SoC) features. For better scalability, Tilera GX series SoCs have adopted tiled architecture. Each tile comprises a processor engine, cache engine, and switch engine that is capable of running an operating system. These components are connected in parallel and conform the four mesh networks.

Intel TeraFLOPS [15, 16] is targeted for exploring future processor designs with high core counts. It is a 65nm, 275 mm^2 chip with 80 tiles running at a targeted frequency of 5GHz connected through 10×8 mesh. Each tile has a **P**rocessing **E**ngine (PE) connected to an on-chip network router. Intel's **S**ingle **C**hip **C**loud (SCC) computer are connected through a 4×6 mesh NoC that features 48 cores of the x86 architecture.

C.2 Network-on-Chip (NoC) Basics

The design of NoC encompasses various basic building blocks, i.e., topology, routing, flow control, and router architecture. In the next few subsections, we briefly explain these basic building blocks.

C.2.1 Topology

Topology specifies the placement and connections between routers and wires. It affects the routing, latency, throughput, and hence execution time of the network. A 2-D mesh is a single core non-uniform edge symmetric direct network. It is non-uniform since every router does not have the same number of ports. Corner routers have two ports, peripheral routers have three ports, and remaining routers have four ports to connect with rest of the network. Mesh is a direct network since each tile behaves as a router as well as **I**ntellectual **P**roperty (IP) core. It is edge symmetric because there exists an auto-morphism that maps any channel *a* (let's say) into another channel *b* (let's say). A typically 4×4 mesh, as shown in Fig 2.6 is arranged in a two-dimensional grid layout. It becomes popular because of its simplicity and regularity. IP core is a hardware block (processor, memory, controller, etc.) that is instantiated in a design by purchasing from third parties. The router routes the packet in a network that is connected through channels. These channels carry parallel signals between routers and cores. A number of traversed routers between the source-destination pair are considered as the hop count of a network whereas the maximum distance between any two routers is considered as diameter. A set of channels that divides the network into two halves are known as bisection bandwidth. And, the maximum number of links removal which disconnects the router/core is known as the connectivity of the network. These different topology terminologies are discussed in Table C.1 using an example of 4×4 mesh, as shown in Fig 2.6.

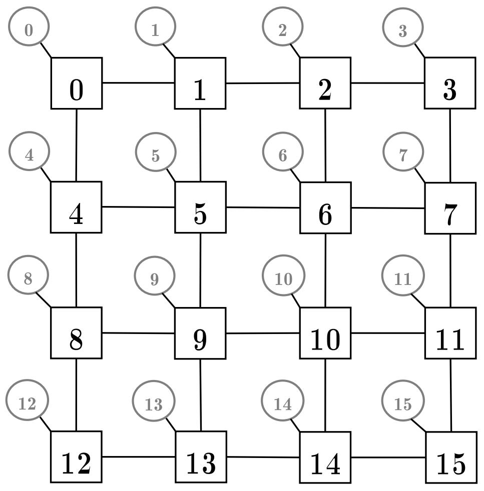


Figure 2.6: 2-D Mesh topology

N/W Parameters	$n \times n$	4×4
# IP cores (m)*	m^2	16
# of NI	m^2	16
# of routers (n)	n^2	16
IP core–router	m^2	16
Router–2 Links	$2n(n-1)$	24
Total Links	$2n(n-1)+m^2$	40
Switch ports	2, 3, 4	2, 3, 4
Path Diversity†	$\frac{D!}{H_x! \times H_y!}$	$20^{\ddagger}$
Diameter (D)	$2(n-1)$	6
Bi. Bandwidth	n	4
Connectivity	2, 3, 4	2, 3, 4

Table C.1: Parameters of Mesh[4]

In the early 1960s, machines like Solomon, Illiac, and MPP, were based on simple 2–D mesh or torus networks because of their physical regularity [73]. In the late 1970s, hypercube [24] networks became popular because of their low diameter. It is adopted by Ametek, Cosmic Cube, the nCUBE computers [25], and Intel iPSC series machines. In the mid-1980s, due to realistic packaging constraints, low-dimensional networks outperformed compared to hypercubes. So most of the machines returned to 2-D or 3-D mesh or torus networks. The examples of these

[4]$\star$ Here $n = m$, $\dagger$ H_x– # hops in X dimension, H_y– # hops in Y dimension, $\ddagger$ Path diversity is 20 when Source(0)–Destination(15)

machines are J-machine, Cray T3D and T3E, Intel DELTA, and Alpha 21364. Mesh is still a popular topology due to its simplicity.

C.2.2 Routing

The routing unit of the router computes the route between a source and the destination node using a routing algorithm. A route is a path that a message will take through the network to reach its destination. These routes are decided by the routing algorithm which computes the available feasible paths through the network topology. Routing algorithm finds the sequence of routers traversed by a flit between source and destination. **F**low contro**L** un**IT** (FLIT) is the smallest unit of a packet. The selection of a routing algorithm is based on the objective of power-performance metric that wishes to achieve. A good routing algorithm evenly distributes the traffic across all the routers to maximize the saturation throughput of the network. Power can be optimized by keeping the routing circuit simple and work in less hop count. Routing algorithms can broadly be classified as deterministic, oblivious, and adaptive methods. Dimension order XY routing is an example of deterministic routing. Its simplicity makes it popular wherein all flits follow the same path from a particular source to a destination. In our proposed

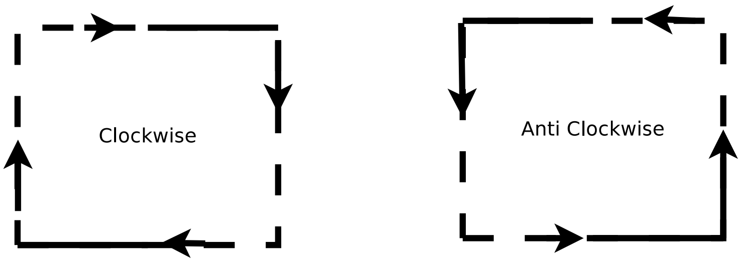


FIGURE C.6: XY routing turn restrictions

work, we have used XY dimension-ordered routing for the mesh topology. Flit always traverses the X direction either in the West (left) or East (right) first and then turns towards Y either in the North (top) or South (bottom). Other oblivious and adaptive routings allow flit traversal for different routes between source and destination. These routing methods have different priority metrics like adaptive schemes monitor network congestion while oblivious schemes prioritize link utilization or randomness. All routing schemes conform to turn model [17], which disallows certain turns to avoid cyclic resource dependency to avoid deadlocks. Fig C.6 shows the turn models for three deadlock-free routing algorithms viz. XY dimension-ordered, West-First and South-Last with the disallowed highlighted turns.

C.2.3 Flow Control

Flow control determines the allocation of network resources to messages during traversal of flits through the network. The buffers and channel bandwidth allocation is the responsibility of a flow control mechanism, which regulates resource utilization through ensuring efficient sharing of resources during flit traversal. The buffer space requirements of the flow control scheme directly affect performance and hardware implementation cost in terms of area and power consumption of each router [18]. We have used **V**irtual **C**hannel (VC) flow control which allocates flits to channel and buffers rather than packets. The primary advantage of VC flow control is its degree of freedom in buffer allocation from one router to another router as illustrated in Fig C.7. At a given time, the different flits of the same packet may be queued on different routers. When a flit of packet *B* is blocked at router-3, the VC1 of router-3 is allocated to flit of packet *A* if it gets the chance to proceed first. It moves from VC0 of router-2 whereas flit of packet *B* wait for either unblocking of VC0 of router-3 or availability of VC1 at router-3 with its turn to transfer flit. So VC flow control primarily relies on VCs for high-

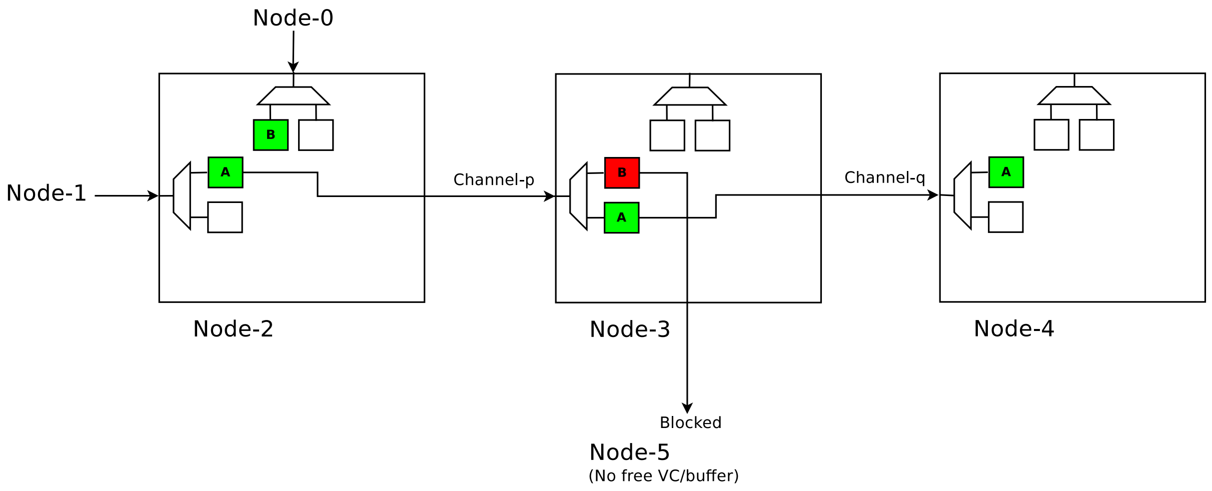


FIGURE C.7: Virtual channel flow control

performance and deadlock-free communications [57]. Each VC is composed of multiple flit buffers. It efficiently manages the flit buffers by allowing packets to pass blocked packets by using idle virtual (logical) channel bandwidth.

VC based routing prevents the head-of-line blocking and enhances throughput using multiple VCs for different flows at the router. A head flit allocates a VC and arbitrates for the output physical channel bandwidth before it proceeds to the next router. The body and tail flits use the same VC, but still, need to compete for the channel bandwidth with flits in other VCs. A VC is freed once the tail flit leaves.

For buffer management, an upstream router sends flits to its downstream (neighboring) router on receiving a credit signal that indicates the number of free buffers at its adjacent downstream router. When a flit leaves the downstream router, it sends a credit bit back to the upstream router which increments its credit count. For VC flow control, the credit count is maintained on each downstream router for individual VCs, and each credit signal carries credit bit, VC id, and an additional bit to indicate the availability of VC.

C.2.4 Router Architecture

A router has datapath and different control units which implement routing and flow control functions. These control units operate datapath operations of flit that are buffered and forwarded to their destinations. Whereas datapath form collectively with registers, switches, and functional units. As earlier networks were not demanding high throughput, a very simple unpipelined router model was used. They had a limited buffering without virtual channels to lower the area and power overhead. Modern routers expect low latency and high throughput networks that cause complex router design. Modern router architecture consists of input buffers, virtual channels, routing logic, allocators, and a crossbar, as shown in Fig C.8.

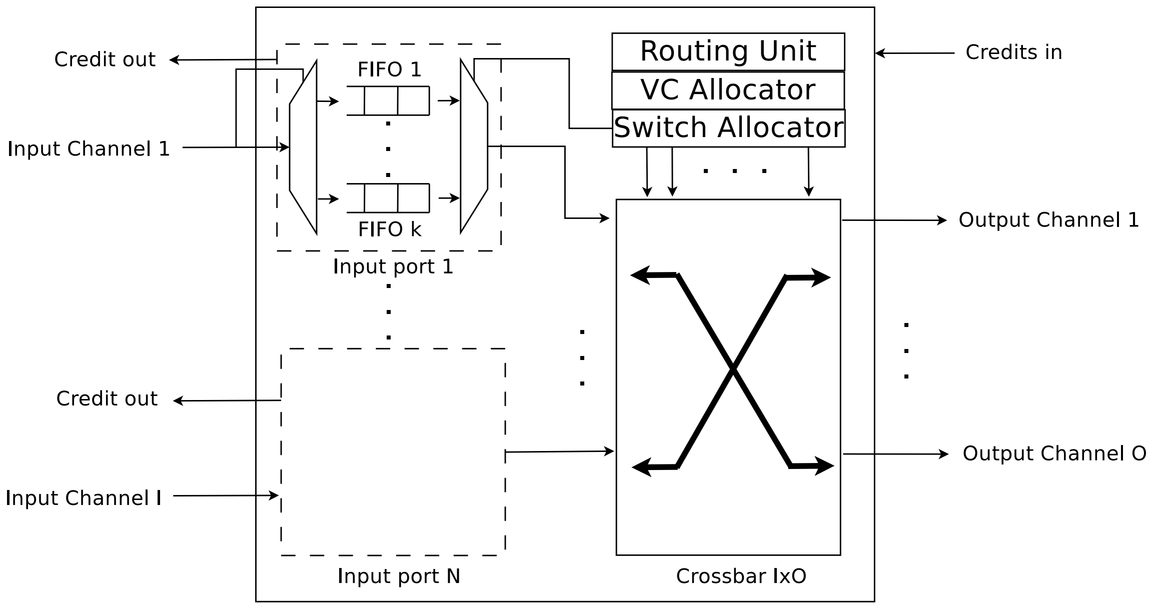


FIGURE C.8: Router architecture

As the routers are connected in a mesh topology, each single-NoC baseline router has four input and output ports that connect the router with rest of the routers through network links. The router connects to the core using other additional ports wherein core inject/eject traffic to the router through core links. The router pipeline consists of the **B**uffer **W**rite (*BW*), the **R**oute **C**omputation (*RC*), the **V**irtual channel **A**llocation (*VA*), **S**witch **A**llocation (*SA*), **S**witch **T**raversal (*ST*) and **L**ink **T**raversal (*LT*) stage.

Flit arrival at the router is decoded and buffered to its specified input virtual channel in *BW* stage. Route computation is performed in the second stage *RC* to determine suitable output port through routing unit. Next *VA* stage arbitrates flits for a virtual channel as the previous stage already determined the output port. Once a virtual channel has been successfully allocated, the flit proceeds to the fourth *SA* stage. Here, the flit arbitrates for access to switch based on its input-output port pair. Once the switch has been allocated to the flit, fifth stage *ST* proceed the flit from the crossbar towards the router output. Finally, link traversal stage *LT* carries the flit to the next router in its path on receiving credit signal from the downstream router.

As each packet splits into several flits, the head flit is responsible for route computation and virtual channel allocation whereas body and tail flits reuse such computation and allocation.

C.3 Performance Assessment Metrics for NoC Design

General purpose desktop computing still covers the largest market. Desktop computing spans from low-end systems to high-end, heavily-configured workstations. Fig C.3.1 shows the power contribution of NoC in processor power that is rapidly increasing as compared to other on chip components with each generation of processor.

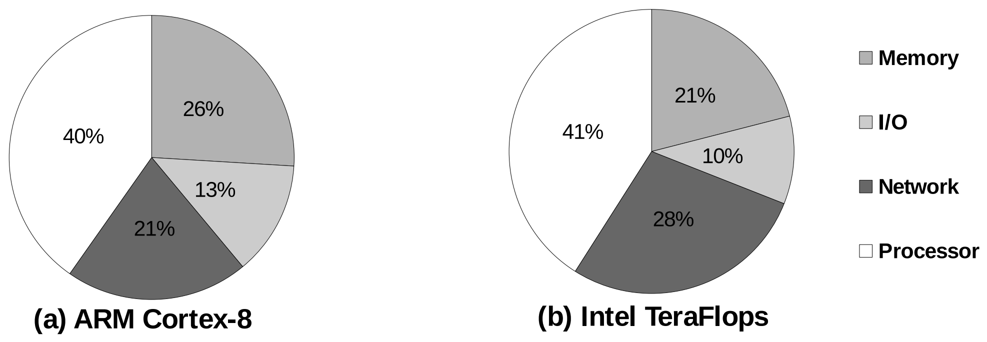


FIGURE C.9: Comparison of NoC power between two different general purpose processors

C.3.1 Power

The scaling down of silicon technology facilitates a phenomenal increase in the number of processing cores (Moore's Law). Scaling improves processor functionality, and performance, but it results in high power dissipation. Each generation of processor significantly advances power consumption. The first microprocessors consumed tenths of watts. A Pentium-4 operating on 2 GHz frequency is close to 100 watts. The fastest workstation in the year 2001 consumes power between 100 and 150 watts. In the near future, it is expected that power consumption rather than raw transistor count will become the major limitation since distributing power, removing heat, and preventing hot spots are increasingly difficult challenges for designing energy-efficient processors.

Reduction in inter-transistors physical distance and consequently, a thinner insulating layer contributes to an increase in static power dissipation. Fig C.10 shows that the gap between static and dynamic power continue increasing with shrinking nanometer technology. NoC contributes significantly to the total chip power, in fact up to 40% in RAW [36, 37] architecture.

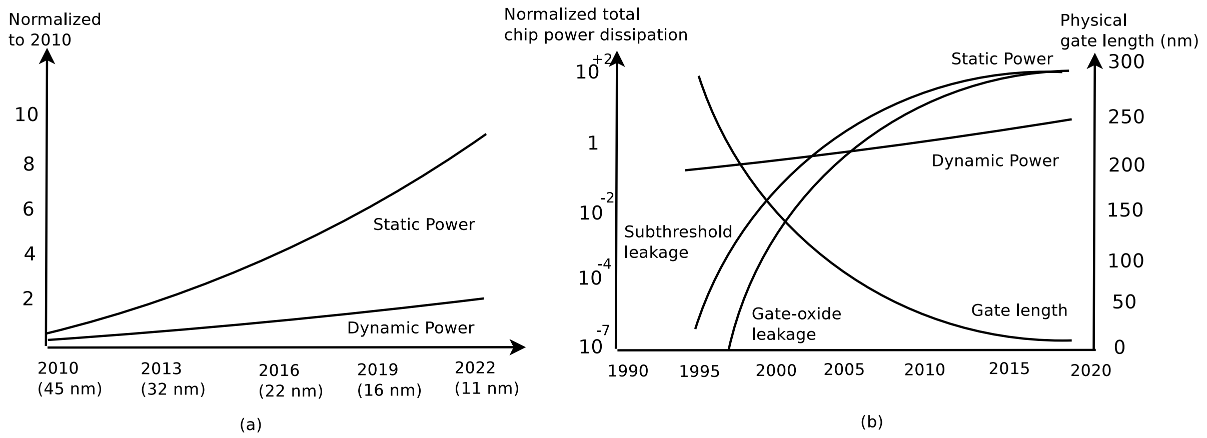


FIGURE C.10: Static and dynamic power (a) yearwise analysis and (b) variations with shrinking gate length[5]

Excessive power dissipation increases packaging and cooling costs, and adversely affects hardware reliability by elevating temperature [23] that contradict one of the objectives of NoC to work under tight power budget. The power consumption of a **C**omplementary **M**etal-**O**xide **S**emiconductor (CMOS) circuit is defined as

$$P_{CMOS} = P_d + P_s$$

[5]http://www.mdpi.com/2079-9268/1/1/131/htm

The dynamic power is $P_d = C_e \times_{DD}^2 \times \times \delta$. This is power consumed whenever the output of the logic gate changes. V_{DD} is the supply voltage, f is the operating frequency and δ is the average number of output transitions. The term C_e is the total effective capacitance of the CMOS gate. It is calculated as $C_e = C_g + C_n + C_i$ Where C_g is the sum of all internal gate capacitances, C_n is the capacitance of interconnecting wires between outputs of the transistors and input of the next stage, and C_i is the input capacitance of all the gates connected to the output of this gate. The dynamic power is consumed whenever capacitance C_e is charged.

Technology advancements have resulted in a large share of static power contributions in total processor power. It is essential to reduce static power for sustaining continuous scaling of the CMOS process. As gate length reduces with nanometer technology advancements, it results in lower threshold voltage. This leads to a significant increase in sub-threshold leakage current, and thinner gate oxides increase the gate tunneling leakage current. Static power is $P_s = V_{DD} \times_l$ wherein the power consumed irrespective of output switching. Where $I_l = I_r + I_s + I_g$, I_r is the reverse current induced due to the PN junction characteristic of transistors, I_s is the current due to carrier diffusion between the source and drain regions of a transistor, and I_g represents the current flowing from gate oxide to substrate and vice versa [11].

C.3.2 Execution Time

The duration between the start time and the completion of an event is referred to as execution time. One realistic measure of performance is the execution time of applications. As performance and execution time are reciprocals, increasing performance decreases execution time. We usually say ‘improve performance’ or ‘improve execution time’ when we mean increase performance and decrease execution time. Whether we are interested in throughput or response time, the key measurement is time. The computer that performs the same amount of work in the least time is the fastest.

C.3.3 Energy

Energy efficiency is one of the primary design challenges for processors. It is significantly dependent on the power and execution time of the network

$$Energy = Power \times\ Time$$

A better match between the workload and the execution hardware can improve overall energy efficiency. The required energy for a transistor is proportional to the product of the transistor load capacitance, switching frequency, and the square of the voltage. Variation in the number of transistors changes switching frequency from one process to the next that varies load capacitance and voltage.

C.3.4 Area

Computation of the router area includes the input unit, output unit, buffers, and crossbar of the router. We use SRAM-based **F**irst **In** **F**irst **O**ut (FIFO) buffer whose area is the product of word line and bit line lengths of the buffer.

$$A_{fifo} = L_{word-line} \times_{bit-line}$$

The word line is $L_{word-line} = F \times (w_{cell} + 2 \times (P_r + P_w) \times_w)$ where F, w_{cell}, d_w, P_r, and P_w are flit size in bits, memory cell width, wire spacing, number of read ports and number of write ports, respectively. Whereas bit line is $L_{bit-line} = B \times (h_{cell} + (P_r + P_w) \times_w)$, where B is buffer size, and h_{cell} is memory cell height. To calculate the total area for a B entry buffer and flit size F, the gate area model [72] is used to calculate h_{cell} and w_{cell}. Other router components, namely, crossbar and arbiter firstly decompose into gate-level netlist to estimate the area of individual circuit components, and finally, the area of the entire block is computed.

The area occupied by links is due to wires and repeaters. We use the gate area model to estimate the area of repeaters. The area of global wiring is calculated as

$$A_{link} = F \times (w_w + s_w) + s_w$$

where A_{link} denotes the wire area, F is the flit size in bits, and w_w and s_s are the wire width and spacing that are computed from the layer width and spacing of the global or intermediate wires which are routed as per the design style.

C.4 Conclusions

This section introduces the evolution of networks-on-chip with a brief discussion on commercial NoC architectures. The basic building blocks of network-on-chip such as topology, routing, flow control, and router architecture are discussed on which we have implemented our proposed work in thesis chapters. The metrics for NoC design validation is discussed.

Appendix D

Static Message Distribution

Table D.1: Each row except the first row indicates static message distribution out of 41 combinations. The static message distribution is **N**ot **A**pplicable (NA) for the single-NoC (first row).

	Control Messages				**Data Messages**	
	Control (***C***)	Request (***R_C***)	Response (***Resp_C***)	Writeback (***WB_C***)	Response (***Resp_D***)	Writeback (***WB_D***)
1	NA	NA	NA	NA	NA	NA
2	✓	✓	-	-	-	-
3	✓	-	✓	-	-	-
4	✓	-	-	✓	-	-
5	✓	-	-	-	✓	-
6	✓	-	-	-	-	✓
7	-	✓	✓	-	-	-
8	-	✓	-	✓	-	-
9	-	✓	-	-	✓	-
10	-	✓	-	-	-	✓
11	-	-	✓	✓	-	-
12	-	-	✓	-	✓	-
13	-	-	✓	-	-	✓
14	-	-	-	✓	✓	-
15	-	-	-	✓	-	✓
16	-	-	-	-	✓	✓
17	✓	✓	✓	-	-	-
18	✓	✓	-	✓	-	-
19	✓	✓	-	-	✓	-
20	✓	✓	-	-	-	✓
21	✓	-	✓	✓	-	-
22	✓	-	-	✓	✓	-
23	✓	-	-	-	✓	✓
24	-	✓	✓	✓	-	-
25	-	✓	-	✓	✓	-
26	-	✓	-	-	✓	✓
27	✓	-	✓	-	✓	-
28	✓	-	✓	-	-	✓
29	✓	-	-	✓	-	✓
30	-	✓	✓	-	✓	-
31	-	✓	✓	-	-	✓
32	-	✓	-	✓	-	✓
33	-	-	✓	✓	✓	-
34	-	-	✓	✓	-	✓
35	-	-	✓	-	✓	✓
36	-	-	-	✓	✓	✓
37	✓	-	-	-	-	-
38	-	✓	-	-	-	-
39	-	-	✓	-	-	-
40	-	-	-	✓	-	-
41	-	-	-	-	✓	-
42	-	-	-	-	-	✓

Appendix E

Message Variation with Workloads

Table E.1: Message variations with Low (L)(canneal_s, blackscholes_s) and High (H) (canneal_h, blackscholes_h) workloads of the benchmarks.

Messages	***Canneal_s***	***Blackscholes_s***	***Blackscholes_h***	***Canneal_h***
C_L1_GETS	536918	536152	601004	1444118
C_L1_GETX	107984	103271	189577	882106
Resp_C_DATA_FROML1	162879	161770	197290	355408
Resp_C_DATA_ALL_Ack	725183	716826	1838556	3066199
Resp_C_Inv	473280	471737	857885	1991832
RESP_C_WB_ACK	46	45	58	97
RESP_C_ACK_ALL	157340	156224	191081	302102
C_Data	23187	22815	34707	54842
L1_PUTX	3857879	3837198	16223126	45244117
L1_putx_old	3857879	3837198	16223126	45244117
L1_upgrade	14	14	16	23
WB_C_L1_Replace	40724	40505	108464	639728
R_C_L1_GETX	99253	98483	122733	152930
R_C_L1_Upgrade	134134	133386	156343	226973
R_C_MEM_Inv	432	423	369	482
R_D_FWD_GETS	162879	161770	197290	355408
R_D_FWD_GETX	76026	75625	87978	98024
R_D_MEM_INV	34255	33877	34405	34180
R_D_ACK_ALL	325	357	366	29904
R_D_WB_DATA	122	118	167	6868
R_D_WB_DATA_CLEAN	76	76	49	905
R_D_L1_GETS	564360	560745	1595708	2140911
R_D_L1_GETX	4803771	4778689	26549376	53334665
WB_Data_L1_REPLACE	3817155	3796693	16114662	44604389
WB_DATA_INV	198	194	216	8989

Appendix F

Message Variation with Threads

TABLE F.1: Message variations for freqmine and rtview benchmarks when the number of threads vary from 8 to 64.

Messages	*Freqmine_8*	*Rtview_8*	*Freqmine_64*	*Rtview_64*
C_L1_GETS	554056	594833	540571	576411
C_L1_GETX	379508	332473	1489189	312866
Resp_C_DATA_FROML1	359587	177881	1840240	938632
Resp_C_DATA_ALL_Ack	944880	2296450	3891718	3251208
Resp_C_Inv	494486	434899	15014795	22223803
RESP_C_WB_ACK	34	40	531	109
RESP_C_ACK_ALL	352161	172316	1823720	930912
C_Data	17979	22828	911697	197910
L1_PUTX	4476803	21280149	19261894	21140479
L1_putx_old	4476803	21280149	19261894	21140479
L1_upgrade	10	11	56	37
WB_C_L1_Replace	45037	285594	134901	659793
R_C_L1_GETX	132328	95181	1648757	480453
R_C_L1_Upgrade	333551	148933	912013	733002
R_C_MEM_Inv	596	501	267	234
R_D_FWD_GETS	359587	177881	1840240	938632
R_D_FWD_GETX	114334	72337	731345	282057
R_D_MEM_INV	4341	9019	23301	26780
R_D_ACK_ALL	1525	1200	427	162
R_D_WB_DATA	174	292	152	15
R_D_WB_DATA_CLEAN	24	55	4	57
R_D_L1_GETS	469017	1914468	2582881	2854195
R_D_L1_GETX	4819685	21443288	35615929	48885380
WB_Data_L1_REPLACE	4431766	20994555	19126993	20480686
WB_DATA_INV	209	359	156	72

Appendix G

Publications

1) Sonal et al., "A Survey of Architectural and Evaluation Approaches for Multiple On-Chip Interconnects," **Journal of Parallel & Distributed Computing (JPDC)**, Elsevier, 2019. (ready to communicate)

2) Sonal et al., "An Energy Efficient Customise 2-Network Router Architecture for Multiple On-Chip Interconnect," **Microprocessors and Microsystems (MICPRO), Embedded Hardware Design**, Elsevier, 2019. (ready to communicate)

3) Sonal et al., "Energy Efficient Optimal Placement of Network Demultiplexer for Multiple Networks-on-Chip," **Integration, The VLSI Journal**, Elsevier, 2019. (ready to communicate)

4) Sonal et al., "Runtime Adaptive Traffic Distribution to Maximize Energy Efficiency of Multiple NoC by Message Redistribution over Underutilised NoC," **Journal of Supercomputing**, Springer, 2019. (ready to communicate)

5) S. Yadav, V. Laxmi, M. S. Gaur, and H. K. Kapoor, "Late Breaking Results: Improving Static Power Efficiency via Placement of Network Demultiplexer over Control Plane of Router in Multi-NoCs," in Proc. of 56th Int. Conf. Design Automation Conference (DAC), **ACM/EDAC/IEEE**, Las Vegas, NV, US, June 2-6 2019.

6) S. Yadav, V. Laxmi, H. K. Kapoor, M. S. Gaur, and M. Zwolinski, "A Power Efficient Crossbar Arbitration in Multi-NoC for Multicast and Broadcast Traffic," in Proceedings of the 4th IEEE International Symposium on Smart Electronic Systems (iSES'18), **IEEE**, Hyderabad, India, December 17-19, 2018. (Best Paper Award)

7) S. Yadav, V. Laxmi, M. S. Gaur, and H. K. Kapoor, "Comprehensive State Space Model of Caches with respect to On-Chip Interconnect Traffic," in Proceedings of the International Conference on Recent Innovations in Electrical, Electronics & Communication Engineering - (RIEECE), **IEEE**, Bhubaneswar, India, July 27-28, 2018.

8) A. Sharma, Y. Gupta, S. Yadav, L. Bhargava, M. S. Gaur, and V. Laxmi, "A Power, Thermal and Reliability-Aware Network-on-Chip," in Proceedings of **IEEE** International Symposium on Nanoelectronic and Information Systems (IEEE-iNIS), Bhopal, December 2017.

9) S. Yadav, V. Laxmi, and M. S. Gaur, "A Power Efficient Dual Link Mesh NOC Architecture to Support Nonuniform Traffic Arbitration at Routing Logic," in Proceedings of the 29th International Conference on VLSI Design (VLSID), **IEEE**, Kolkata, pp. 69-74, Jan. 4-8, 2016.

10) S. Yadav, V. Laxmi, M. S. Gaur, and M. Bhargava, "C^2-DLM: Cache Coherence aware Dual Link Mesh for On-chip Interconnect," in Proceedings of 19th **IEEE** International Symposium on VLSI Design and Test, Ahmedabad, 2015.

11) S. Yadav, M. S. Gaur, V. Laxmi, and S. Sharma, "Fault tolerant adaptive XY routing for HPC mesh," in Proceedings of 9th **IEEE** International Conference on Industrial and Information Systems (ICIIS), Gwalior, pp. 1-6, 2014.

12) S. Yadav, M. S. Gaur, V. Laxmi, and M. Bhargava, "Dynamic fault injection model for on-chip 2D mesh network," in Proceedings of the 7th **ACM** Computing Conference (COMPUTE '14). ACM, New York, NY, USA, Article 18, 6 pages.

13) S. Yadav et al. "Tool Chain for Performance Analysis of Chip-Multiprocessor (CMP)," in 29th National Convention of Computer Engineers and National Seminar on ETICE-2015, Organized by IET and SKIT Jaipur, India. February 7-8, 2015.

Posters

1) Poster presented on "Traffic Distribution in Multi-NoCs," in International Workshop on Network on Chip supported by INDO-UK project HiPER NIRGAM, December 10-12, 2015, MNIT Jaipur, India.

2) Participating Creating Effective Poster from Project/Paper/Research, Sponsored by TEQIP-II, SVNIT Surat, Gujrat, April 16-18, 2015.

3) One Day Workshop on Network on Chip, March 29, 2015, IIIT Delhi, India.

Achievements

1) DAC 2019 (Secured 3rd position out of 18 selected papers)

2) iSES 2018 (Best Paper Award)

3) DATE 2019 PhD-Forum (Travel Grant)

Research Outcome

We develop a multiple-NoC based full system simulation tool using conventional Gem5 and GARNET[1] simulator for industry and academic research on multiple NoC. It supports up to four multiple mesh NoC (for more networks it can be extended). It is integrated with PARSEC benchmark to run in full system simulation.

[1]A detailed cycle-accurate network-on-chip model inside Gem5 full-system simulator

Glossary

Network Interface It is located between core and router. Core messages enter into NI, and then into the router through NI links. Messages are the logical unit of communication that may be arbitrarily long. NI divides these messages into packets which are further segmented into FLITs (FLow control unITs). 9

Virtual Networks These networks logically separate message classes to avoid deadlock in coherence protocols. 68

Buffers A buffer holds the flit and also used to save its state needed to coordinate the handling of the flits of a packet over a channel. 68

Multi-Network-NoC All links including NoC as well as NI links are replicated. 11

Multi-Planar-NoC Complete NoC including routers and links are replicated and can be viewed as a stack of NoCs, each NoC in a different plane of the stack. 11

Multi-Router-NoC Only routers are replicated, they are connected through single links in the network. 12

Multi-Switch-NoC Only switching hardware within router (for example, crossbar switch) is replicated. 12

Virtual Channels Virtual channels allow the upstream router to use a second free lane (a VC with buffer space available) when a first packet is blocked in downstream router [20]. 74

References

[1] G. E. Moore, “Cramming more components onto integrated circuits, Reprinted from Electronics, volume 38, number 8, April 19, 1965, pp.114 ff.,” IEEE Solid-State Circuits Society Newsletter, vol. 11, no. 3, pp. 33-35, Sept. 2006., vol. 86, no. 1, pp. 82-85, 1998.

[2] X. Huang et al., “Sub-50 nm P-channel FinFET,” IEEE Transactions on Electron Devices, vol. 48, no. 5, pp. 880-886, May 2001.

[3] Y. P. Zhang, T. Jeong, F. Chen, H. Wu, R. Nitzsche, and G. R. Gao, “A study of the on-chip interconnection network for the IBM Cyclops64 multi-core architecture,” in Proceedings of the 20th International Conference on Parallel and Distributed Processing (IPDPS), IEEE Computer Society, Washington, DC, USA, 64-64. 2006.

[4] S. Borkar and A. A. Chien, “The future of Microprocessors,” Communications of the ACM, Magazine, vol. 54, pp. 67-77, May 2011.

[5] C. Hamacher, Z. Vranesic, and S. Zaky, Computer Organization, McGraw-Hill Education, Fifth Edition, 2017.

[6] Computer System Architecture Research Tool, “Gem5 Full System Simulator,” January 28, 2018. http://www.gem5.org/Main_Page.

[7] RTL Design & Verification Synthesis Tool, “Synopsys Design Compiler,” January 28, 2018. https://www.synopsys.com/implementation-and-signoff/rtl-synthesis-test/dc-ultra.html

[8] J. S. Miguel and N. E. Jerger, “Data Criticality in Network on Chip Design,” in Proceedings of the 9th IEEE/ACM Internation Symposium on Network on Chip (NOCS), Vancouver, BC, Canada, Sept. 28-30, 2015.

[9] ARM Limited, “AMBA bus interconnect specification,” Oct. 14, 2017. https://www.arm.com/products/system-ip/amba-specifications.

[10] H. P. Hofstee, “Power efficient processor architecture and the cell processor,” in Proceedings of the International Symposium on High Performance Computer Architecture, pages 258-262, February 2005.

[11] D. A. Neamen, Semiconductor physics and devices, McGraw-Hill Education, 2003.

[12] A. Agarwal, L. Bao, J. Brown, B. Edwards, M. Mattina, C. C. Miao, C. Ramey, and D. Wentzlaff, “Tile processor: Embedded multicore for networking and multimedia,” in Proceedings of Hot Chips: Symposium on High Performance Chips, 2007.

[13] S. Bell et al., “TILE64 processor: A 64-core SoC with mesh interconnect,” International IEEE Solid-State Circuits Conference, pages 88–89, Feb 2008.

[14] M. Coppola, R. Locatelli, G. Maruccio, L. Pieralisi, and A. Scandurra, “Spidergon: a novel on chip communication network,” in International Symposium on System on Chip, page 15, November 2004.

[15] Y. Hoskote, S. Vangal, A. Singh, N. Borkar, and S. Borkar, “A 5-GHz mesh interconnect for a Teraflops processor,” IEEE MICRO, 27(5):51–61, 2007.

[16] S. R. Vangal et al., “An 80-tile sub-100-w TeraFLOPS processor in 65-nm CMOS,” IEEE Journal of Solid State Circuits, 43(1):29-41, January 2008.

[17] C. J. Glass and L. M. Ni, “The turn model for adaptive routing,” Proceedings of the International Symposium on Computer Architecture (ISCA), pages 278-287, May 1992.

[18] R. Hesse, Fine Grained Adaptivity for Dynamic On-Chip Networks, PhD Thesis, University of Toronto, 2016.

[19] A. B. Kahng, B. Li, L.-S. Peh, and K. Samadi, “ORION 2.0: a fast and accurate NoC power and area model for early-stage design space exploration,” in Proc. of the Conference on Design, Automation and Test in Europe (DATE), Belgium, pp. 423-428, 2009.

[20] J. Flich, D. Bertozzi. Designing Network-on-Chip Architectures in the Nanoscale Era, Taylor & Francis, Chapman and Hall/CRC, 1st Edition, 2010.

[21] J. G. Anjana and M. Prasanth, “A three level cache structure,” IEEE International Conference on Advanced Communications, Control and Computing Technologies, Ramanathapuram, pp. 426-430, 2014.

[22] J. Hruska. L2 vs. L3 cache: What’s the Difference?. 2017. https://www.extremetech.com/computing/55662-top-tip-difference-between-l2-and-l3-cache.

[23] I. Seitanidis, C. Nicopoulos, and G. Dimitrakopoulos, “PowerMax: An Automated Methodology for Generating Peak-Power Traffic in Networks-on-Chip,” in Proceedings of Tenth IEEE/ACM International Symposium on Networks-on-Chip (NOCS), Japan, 2016.

[24] H. P. Katseff, “Incomplete hypercubes,” IEEE Transactions on Computers, vol. 37, no. 5, pp. 604-608, May 1988.

[25] J. F. Palmer, “The NCUBE family of high-performance parallel computer systems,” in Proceedings of the third conference on Hypercube concurrent computers and applications: Architecture, software, computer systems, and general issues - Volume 1 (C3P), Geoffrey Fox (Ed.), Vol. 1. ACM, New York, NY, USA, 847-851, 1988.

[26] W. Wolf, A. A. Jerraya, and G. Martin, “Multiprocessor System-on-Chip (MPSoC) Technology,” in IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 27, no. 10, pp. 1701-1713, Oct. 2008.

[27] M. Duric et al., “Dynamic-vector execution on a general purpose EDGE chip multiprocessor,” International Conference on Embedded Computer Systems: Architectures, Modeling, and Simulation (SAMOS XIV), Agios Konstantinos, pp. 18-25, 2014.

[28] M. Grammatikakis, M. Coppola, and F. Sensini, “Software for multiprocessor networks on chip,” In Networks on chip, Axel Jantsch and Hannu Tenhunen (Eds.). Kluwer Academic Publishers, Norwell, MA, USA 281-303, 2003.

[29] S. Saponara, L. Fanucci and M. Coppola, “Many-core platform with NoC interconnect for low cost and energy sustainable cloud server-on-chip,” 2012 Sustainable Internet and ICT for Sustainability (SustainIT), Pisa, pp. 1-5, 2012.

[30] A. Daglis, S. Novaković, E. Bugnion, B. Falsafi, and B. Grot, “Manycore Network Interfaces for in-memory rack-scale computing,” 2015 ACM/IEEE 42nd Annual International Symposium on Computer Architecture (ISCA), Portland, OR, pp. 567-579, 2015.

[31] Ultimate Cloud Computing Processor, GIGENET Cloud, “https://www.gigenetcloud.com/archives/118-ultimate-cloud-computing-processor/”, 2018.

[32] L. D. Paulson, “Squeezing supercomputers onto a chip,” in Computer, vol. 38, no. 1, pp. 21-23, Jan. 2005.

[33] J. Zhan et al., “NoC-sprinting: Interconnect for fine-grained sprinting in the dark silicon era,” in 51st ACM/EDAC/IEEE DAC, San Francisco, CA, pp. 1-6, 2014.

[34] R. Parikh et al., “Power-aware NoCs through routing and topology reconfiguration,” 51st ACM/EDAC/IEEE DAC, San Francisco, CA, pp. 1-6, 2014.

[35] T. P. Morgan, “Tilera etches ‘*ss-kicking’ 72-core system-on-chip for network gear,” Feb 19, 2013. https://www.theregister.co.uk/2013/02/19/tilera_tile_gx72_processor/.

[36] S. Volos, C. Seiculescu, B. Grot, N. K. Pour, B. Falsafi, and G. D. Micheli, “CCNoC: Specializing On-Chip Interconnects for Energy Efficiency in Cache-Coherent Servers,” in Proceedings of Sixth IEEE/ACM International Symposium on Networks on Chip (NoCS), pp.67,74, 9-11 May 2012.

[37] J. Zhan, J. Ouyang, F. Ge, J. Zhao, and Y. Xie, “DimNoC: a dim silicon approach towards power-efficient on-chip network,” in Proceedings of the 52nd Annual Design Automation Conference (DAC ’15), ACM/EDAC/IEEE, New York, USA, Article 10, 6 pages, 2015.

[38] Y. J. Yoon, N. Concer, M. Petracca, and L. P. Carloni, “Virtual channels vs. multiple physical networks: A comparative analysis,” in Proceedings of 47th ACM/IEEE Conference on Design Automation Conference (DAC), pp.162, 165, 13-18 June 2010.

[39] F. Gilabert, M. E. Gómez, S. Medardoni, and D. Bertozzi, "Improved Utilization of NoC Channel Bandwidth by Switch Replication for Cost-Effective Multi-processor Systems-on-Chip," in Proceedings of Fourth ACM/IEEE International Symposium on Networks-on-Chip (NOCS), pp.165, 172, 3-6 May 2010.

[40] N. Agarwal, T. Krishna, L.-S. Peh, and N. K. Jha, "GARNET: A detailed on–chip network model inside a full–system simulator," in International Symposium on Performance Analysis of Systems and Software (ISPASS), IEEE, Boston, MA, pp. 33-42. 2009.

[41] N. Binkert et al., "The gem5 simulator," in proceedings of ACM, SIGARCH Comput. Archit. News 39, 2. August 2011.

[42] Cadence Design Systems Inc., "Encounter RTL Compiler," July 13, 2018. https://www.cadence.com/content/cadence-www/global/en_US/home/training/all-courses/84441.html.

[43] A. Ejaz and A. Jantsch, "Costs and benefits of flexibility in spatial division circuit switched networks-on-chip," in Proceedings of the Sixth International Workshop on Network on Chip Architectures (NoCArc). ACM, New York, NY, USA, 41-46. 2013.

[44] N. Barrow et al., "A communication characterization of Splash-2 and Parsec," in Proceedings of the IEEE International Symposium on Workload Characterization (IISWC), IEEE Computer Society, Washington, DC, USA, 86-97. 2009.

[45] C. Ciordas, K. Goossens, and T. Basten, "NoC Monitoring: Impact on the Design Flow," in International Symposium on Computer Architecture (ISCAS), pp. 1981-1984, 2006.

[46] R. Das, S. Narayanasamy, S. K. Satpathy, and R. G. Dreslinski, "Catnap: energy proportional multiple network-on-chip," in Proceedings of the 40th Annual International Symposium on Computer Architecture (ISCA), ACM, New York, USA, 320-331. 2013.

[47] J. Sepúlveda, D. Flórez, and G. Gogniat, "Reconfigurable security architecture for disrupted protection zones in NoC-based MPSoCs," in Proceedings of 10th International Symposium on Reconfigurable Communication-centric Systems-on-Chip (ReCoSoC), Bremen, pp. 1-8, 2015.

[48] B. Grot, J. Hestness, S. W. Keckler, and O. Mutlu, "Express Cube Topologies for on-Chip Interconnects," in Proceedings of 15th IEEE International Symposium on High Performance Computer Architecture, Raleigh, NC, pp. 163-174, 2009.

[49] C. Bienia, Benchmarking Modern Multiprocessors, Ph.D. dissertation, Princeton University, January 2011.

[50] A. Olofsson, "Adapteva announces 1,000 Teraflops accelerator chip for deep learning," April 1, 2016. http://www.adapteva.com/andreas-blog/adapteva-announces-1000-teraflops-accelerator-chip-for-deep-learning/.

[51] C. Mills, "A 1,000-core processor is such wonderful overkill," November 3, 2016. http://bgr.com/2016/06/21/most-powerful-processor-uc-davis-kilo-core/

[52] A. Flores, J. L. Aragon, and M. E. Acacio, “Heterogeneous Interconnects for Energy–Efficient Message Management in CMPs,” IEEE Transactions on Computers (TC), vol. 59, no. 1, pp. 16-28, Jan. 2010.

[53] S. Borkar, “Thousand core chips: a technology perspective,” in Proceedings of the 44th annual Design Automation Conference (DAC ’07), ACM, New York, NY, USA, 746-749, 2007.

[54] M. R. Marty, “Cache Coherence Techniques for Multicore Processors,” PhD dissertation, Dept. of Computer Sciences, University of Wisconsin-Madison, USA, 2008.

[55] T. Chen, Q. Guo, O. Temam, Y. Wu, Y. Bao, Z. Xu, and Y. Chen, “Statistical Performance Comparisons of Computers,” IEEE Transactions on Computers (TC), vol. 64, no. 5, pp. 1442-1455, May 1, 2015.

[56] T. C. Huang, U. Y. Ogras, and R. Marculescu, “Virtual Channels Planning for Networks-on-Chip,” in Proceedings of the 8th International Symposium on Quality Electronic Design (ISQED’07), San Jose, CA, pp. 879-884, 2007.

[57] Y. J. Yoon, N. Concer, M. Petracca, and L. P. Carloni, “Virtual Channels and Multiple Physical Networks: Two Alternatives to Improve NOC Performance,” IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCADICS), vol. 32, no. 12, pp. 1906-1919, Dec. 2013.

[58] Z. Fang, L. Cheng, and S. R. Vangal, “Using Criticality Information to Route Cache coherency Communications,” U.S. Patent US20090300292 A1, December 03, 2009.

[59] Z. Li, J. Wu, L. Shang, R. P. Dick, and Y. Sun, “Latency criticality aware on-chip communication,” in Proceedings of the Twelfth Design, Automation & Test in Europe Conference & Exhibition (DATE), IEEE/ACM, Nice, pp.1052-1057, 20-24 April 2009.

[60] A. Boroumand, S. Ghose, M. Patel, H. Hassan, B. Lucia, K. Hsieh, K. T. Malladi, H. Zheng, and O. Mutlu, “LazyPIM: An Efficient Cache Coherence Mechanism for Processing-in-Memory,” IEEE Computer Architecture Letters (CAL), June 2016.

[61] A. B. Kahng, B. Li, L.-S. Peh, and K. Samadi, “ORION 2.0: A Power-Area Simulator for Interconnection Networks,” IEEE Transactions on Very Large Scale Integration (TVLSI) Systems, vol.20, no.1, pp.191,196, Jan. 2012.

[62] D. J. Sorin, M. D. Hill, and D. A. Wood, A Primer on Memory Consistency and Cache Coherence, Synthesis Lectures on Computer Architecture, Morgan & Claypool Publishers, 2011.

[63] N. D. E. Jerger, “Chip MultiProcessor Coherence and Interconnect System Design,” Ph.D. dissertation, Dept. of Electrical Engineering, University of Wisconsin-Madison, USA, 2008.

[64] C. A. Nicopoulos, D. Park, J. Kim, N. Vijaykrishnan, M. S. Yousif, and C. R. Das, “ViChaR: A Dynamic Virtual Channel Regulator for Network-on-Chip Routers,” in Proceedings of the Thirty-ninth IEEE/ACM International Symposium on Microarchitecture (MICRO’06), Orlando, FL, pp. 333-346, 2006.

[65] M. Lai, Z. Wang, L. Gao, H. Lu, and K. Dai, “A dynamically-allocated virtual channel architecture with congestion awareness for on-chip routers,” in Proceedings of the Forty-fifth ACM/IEEE Design Automation Conference, Anaheim, CA, pp. 630-633, 2008.

[66] A. Bianco, P. Giaccone, G. Masera, and M. Ricca, “Power Control for Crossbar-Based Input-Queued Switches,” IEEE Transactions on Computers, vol. 62, no. 1, pp. 74-82, Jan. 2013.

[67] J. Zhan, J. Ouyang, F. Ge, J. Zhao, and Y. Xie, “Hybrid Drowsy SRAM and STT-RAM Buffer Designs for Dark-Silicon-Aware NoC,” IEEE Trans. on Very Large Scale Integr. Syst., vol. 24, no. 10, pp. 3041-3054, Oct. 2016.

[68] M. Taylor et al., “The Raw Microprocessor: A Computational Fabric for Software Circuits and General-Purpose Programs,” IEEE Micro, vol. 22, pp. 25-35, 2002.

[69] P. Gratz, C. Kim, K. Sankaralingam, H. Hanson, P. Shivakumar, S. Keckler, and D. Burger, “On-Chip Interconnection Networks of the TRIPS Chip,” IEEE Micro, vol. 27, pp 41-50, 2007.

[70] D. Wentzlaff et al., “On-Chip Interconnection Architecture of the Tile Processor,” IEEE Micro, vol. 27, pp. 15-31, 2007.

[71] A. Varghese, B. Edwards, G. Mitra, and A. P. Rendell, “Programming the Adapteva Epiphany 64-Core Network-on-Chip Coprocessor,” in Proc. of the 29th IEEE Int. Parallel & Distribut. Processing Symp. Workshops, Phoenix, AZ, pp. 984-992, 2014.

[72] D. D. Gajski, F. Vahid, S. Narayan, and J. Gong, Specification and Design of Embedded Systems, Prentice Hall, NJ, 1994.

[73] W. J. Dally and B. Towles, Principles and Practices of Interconnection Networks, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2003.

[74] J. Balfour and W. J. Dally, “Design tradeoffs for tiled CMP on-chip networks,” in Proceedings of the 20th International Conference on Supercomputing (ICS), ACM, NY, USA, pp. 187-198, 2006.

[75] G. M. Tang, K. Takata, M. Tanaka, A. Fujimaki, K. Takagi, and N. Takagi, “4-bit Bit-Slice Arithmetic Logic Unit for 32-bit RSFQ Microprocessors,” IEEE Transactions on Applied Superconductivity, vol. 26, no. 1, pp. 1-6, Jan. 2016.

[76] H. Bokhari, “Power, Performance and Reliability Optimisation of On-Chip Interconnect By Adroit Use of Dark Silicon,” The University of New South Wales, Australia, 2015.

[77] H. Esmaeilzadeh, E. Blem, R. St. Amant, K. Sankaralingam, and D. Burger, “Dark Silicon and the End of Multicore Scaling,” IEEE Micro, vol. 32, no. 3, pp. 122-134, May-June 2012.

[78] R. Das, “Application-aware on-chip networks,” PhD thesis, The Pennsylvania State University, 2010.

[79] A. K. Mishra, O. Mutlu, and C. R. Das, “A heterogeneous multiple network-on-chip design: An application-aware approach,” in Proceedings of 50th ACM/EDAC/IEEE Design Automation Conference (DAC), Austin, TX, pp. 1-10, 2013.

[80] K. W. Kang and G. Do, “Layout Structures of Data Input/Output Pads and Peripheral Circuits of Integration Circuit Memory Devices,” US Patent 2004/0004897 A1, Jan 2004.

[81] X. Bai, C. Visweswariah, P. N. Strenski, and D. J. Hathaway, “Uncertainty-aware circuit optimization,” in Proceedings of Design Automation Conference (DAC), IEEE/ACM , pp. 58-63, 2002.

[82] M. H. Tehranipoor and N. Ahmed. Nanometer Technology Designs: High-Quality Delay Tests. Springer, 2010.

[83] K. Duraisamy, Y. Xue, P. Bogdan, and P. P. Pande, “Multicast-Aware High-Performance Wireless Network-on-Chip Architectures,” IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 25, no. 3, pp. 1126-1139, March 2017.

[84] L. Ivanov and R. Nunna, “Modeling and verification of cache coherence protocols,” in Proceedings of IEEE International Symposium on Circuits and Systems (ISCAS), Sydney, NSW, pp. 129-132, vol. 5, 2001.

[85] T. Cross, “After moores law,” July 13, 2018. https://www.economist.com/technology-quarterly/2016-03-12/after-moores-law.

[86] D. Abts, N. D. E. Jerger, J. Kim, D. Gibson, and M. H. Lipasti, “Achieving predictable performance through better memory controller placement in many-core CMPs,” in Proceedings of the 36th annual International Symposium on Computer Architecture (ISCA ’09), ACM, New York, NY, USA, 451-461, 2009.

[87] W. Hung, C. Addo-Quaye, T. Theocharides, Y. Xie, N. Vijaykrishnan, and M. J. Irwin, “Thermal-aware IP virtualization and placement for networks-on-chip architecture,” in Proceedings of the International Conference on Computer Design, IEEE, 2004.

[88] J. Hu and R. Marculescu, “Engergy-aware mapping for tile- based NoC architecture under performance constraints,” in Proceedings of the Asia and South Pacific Design Automation Conference (ASP-DAC), IEEE/ACM, 2003.

[89] K. Srinivasan and K. Chatha, “A technique for low energy mapping and routing in network-on-chip architectures,” in Proceedings of the International Symposium on Low Power Electronics and Design, IEEE/ACM, 2005.

[90] H. Lu, G. Yan, Y. Han, Y. Wang, and X. Li, “ShuttleNoC: Boosting on-chip communication efficiency by enabling localized power adaptation,” The 20th Asia and South Pacific Design Automation Conference, Chiba, pp. 142-147, 2015.

[91] H. Lu, Y. Chang, G. Yan, N. Lin, X. Wei, and X. Li, "ShuttleNoC: Power-adaptable Communication Infrastructure for Many-core Processors," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, July 2018.

[92] J. Wu, D. Dong, X. Liao, and L. Wang, "Energy-efficient NoC with multi-granularity power optimization," Journal of Supercomputing, vol. 73, no. 4, pp. 1654-1671, April 2017.

[93] J. Wu, D. Dong, and L. Wang, "NoC Power Optimzation using Combined Routing Algorithms," in Proc of ICIC, Wuhan, China, 2017.

[94] S. Hesham, D. Goehringer, and M. A. A. E. Ghany, "A call-up for circuit-switched NoCs in the Dark-Silicon Era," IEEE Nordic Circuits and Systems Conference (NORCAS): NORCHIP and International Symposium of System-on-Chip (SoC), Linkoping, pp. 1-6, 2017.

[95] J. Wu, D. Dong, X. Liao, and L. Wang, "Chameleon: Adaptive energy-efficient heterogeneous network-on-chip," 33rd IEEE International Conference on Computer Design (ICCD), New York, NY, pp. 419-422, 2015.

[96] S. H. Gade and S. Deb, "HyWin: Hybrid Wireless NoC with Sandboxed Sub-Networks for CPU/GPU Architectures," in IEEE Transactions on Computers, vol. 66, no. 7, pp. 1145-1158, July 1, 2017.

[97] J. Lee, C. Nicopoulos, H. G. Lee, and J. Kim, "TornadoNoC: A lightweight and scalable on-chip network architecture for the many-core era," ACM Trans. Architec. Code Optim. 10, 4, Article 56, 30 pages, Dec. 2013.

[98] J. Fang, S. Liu, S. Liu, Y. Cheng, and L. Yu, "Hybrid Network-on-Chip: An Application-Aware Framework for Big Data," Complexity, Volume 2018, Article ID 1040869, 11 pages, 2018.

[99] M. Junginger and Y. Lee, "The Multi-Ring Topology: High-Performance Group Communication in Peer-to-Peer Networks," In Proceedings of the Second International Conference on Peer-to-Peer Computing (P2P '02). IEEE Computer Society, Washington, DC, USA, 2002.

[100] D. Yoo, I. Jung, S. R. Maeng, and H. Roh, "Multistage ring network: a new multiple ring network for large scale multiprocessors," Parallel Processing, Proceedings. International Workshops on, Aizu-Wakamatsu, pp. 290-294, 1999.

[101] M. A. Wani and H. R. Arabnia, "Parallel Edge-Region-Based Segmentation Algorithm Targeted at Reconfigurable MultiRing Network," The Journal of Supercomputing, vol. 25, issue 1, pp 43–62, May 2003.

[102] M. A. Wani and H. R. Arabnia, "Parallel stereocorrelation on a reconfigurable multi-ring network," The Journal of Supercomputing, vol. 10, issue 3, pp 243–269, Sept. 1996.

[103] H. R. Arabnia, "The stereo correspondence problem on a ring-based network," Proceedings of IEEE International Symposium on Parallel Algorithms Architecture Synthesis, Aizu-Wakamatsu, pp. 265-275, 1997.

[104] A. S. T. Lee, D. K. Hunter, D. G. Smith, and D. Marcenac, "ROUTING IN WDM RINGS AND MULTI-RING NETWORKS," published by The Institution of Electrical Engineers (IEE), Savoy Place, London, UK, 1998.

[105] M. A. Wani and H. R. Arabnia, "Parallel Polygon Approximation Algorithm Targeted at Reconfigurable Multi-Ring Hardware," Proceedings of the 2006 International Conference on Computer Graphics Virtual Reality (CGVR'06), Las Vegas, Nevada, USA, June 26-29, 2006

[106] A. K. Abousamra, R. G. Melhem, and A. K. Jones, "Déjà Vu Switching for Multiplane NoCs," 2012 IEEE/ACM Sixth International Symposium on Networks-on-Chip, Copenhagen, pp. 11-18, 2012.

[107] A. Abousamra, A. K. Jones, and R. Melhem, "Proactive circuit allocation in multiplane NoCs," In Proceedings of the 50th Annual Design Automation Conference (DAC '13). ACM, New York, NY, USA, Article 35, 10 pages, 2013.

[108] T. C. Xu, V. Leppänen, P. Liljeberg, J. Plosila, and H. Tenhunen, "Trio: A Triple Class On-chip Network Design for Efficient Multicore Processors," IEEE 17th International Conference on High Performance Computing and Communications, IEEE 7th International Symposium on Cyberspace Safety and Security, and IEEE 12th International Conference on Embedded Software and Systems, New York, pp. 951-956, 2015.

[109] Z. Li, J. S. Miguel, and N. E. Jerger, "The runahead network-on-chip," 2016 IEEE International Symposium on High Performance Computer Architecture (HPCA), Barcelona, pp. 333-344, 2016.

[110] V. Akhlaghi, M. Kamal, A. A. Kusha, and M. Pedram, "An efficient network on-chip architecture based on isolating local and non-local communications," Computers Electrical Engineering, Volume 45, Pages 430-444, 2015.

[111] J. Lee, C. Nicopoulos, H. G. Lee, and J. Kim, "Sharded Router: A novel on-chip router architecture employing bandwidth sharding and stealing," Parallel Computing, Volume 39, Issue 9, Pages 372-388, 2013.

[112] P. Kumar, Y. Pan, J. Kim, G. Memik, and A. Choudhary, "Exploring concentration and channel slicing in on-chip network router," 2009 3rd ACM/IEEE International Symposium on Networks-on-Chip, San Diego, CA, pp. 276-285, 2009.

[113] Cray X1. 2019. http://www.cray.com/products/x1/.

[114] S. Scott, D. Abts, J. Kim, and W. J. Dally, "The blackwidow high-radix clos network," In Proc. of the International Symposium on Computer Architecture (ISCA), pages 16–28, Boston, MA, June 2006.

[115] B. Grot, J. Hestness, S. W. Keckler, and O. Mutlu, "Express cube topologies for on-chip interconnects," In International Symposium on High-Performance Computer Architecture (HPCA), pages 163–174, Raleigh, NC, 2009.

[116] M. Buckler, W. Burleson, and G. Sadowski, "Low-power Networks-on-Chip: Progress and remaining challenges," International Symposium on Low Power Electronics and Design (ISLPED), Beijing, pp. 132-134, 2013.

[117] H. Bokhari, H. Javaid, M. Shafique, J. Henkel, and S. Parameswaran, "SuperNet: Multimode interconnect architecture for manycore chips," 52nd ACM/EDAC/IEEE Design Automation Conference (DAC), San Francisco, CA, pp. 1-6, 2015.

[118] E. Carara, F. Moraes, and N. Calazans, "Router architecture for high-performance NoCs," In Proceedings of the 20th annual conference on Integrated circuits and systems design (SBCCI '07). ACM, New York, NY, USA, pp. 111-116, 2007

[119] S. Noh, V. Ngo, H. Jao, and H. Choi, "Multiplane Virtual Channel Router for Network-on-Chip Design," First International Conference on Communications and Electronics, Hanoi, pp. 348-351, 2006.

[120] C. R. Jesshope and C. Izu, "The MP1 Network Chip and its Application to Parallel Computers," The Computer Journal, Volume 36, Issue 8, Pages 763–777, 1 January 1993.

[121] P. Ezhumalai, C. Arun, S. Manojkumar, P. Sakthivel, and D. Sridharan, "High Performance Hybrid Two Layer Router Architecture for FPGAs Using Network-On-Chip," (IJCSIS) International Journal of Computer Science and Information Security, Vol. 7, No. 1, 2010.

[122] M. Shafique and S. Garg, "Computing in the Dark Silicon Era: Current Trends and Research Challenges," in IEEE Design Test, vol. 34, no. 2, pp. 8-23, April 2017.

[123] H. Bokhari, H. Javaid, M. Shafique, J. Henkel, and S. Parameswaran, "darkNoC: Designing energy-efficient network-on-chip with multi-Vt cells for dark silicon," 51st ACM/EDAC/IEEE Design Automation Conference (DAC), San Francisco, CA, pp. 1-6, 2014.

[124] J. Wu, D. Dong, and L. Wang, "HM-Mesh: Energy Efficient Hybrid Multiple Network-on-Chip," 2016 International Symposium on Computer, Consumer and Control (IS3C), Xi'an, pp. 404-407, 2016.

[125] F. Wang, X. Tang, and Z. Xing, "Applying Partial Power-Gating to Direction-Sliced Network-on-Chip," Journal of Electrical and Computer Engineering, Article ID 862387, 16 pages, Volume 2015.

[126] N. Teimouri, M. Modarressi, A. Tavakkol, and H. Sarbazi-Azad, "Energy-optimized on-chip networks using reconfigurable shortcut paths," In Conf. on Architecture of Computing Systems, pages 231–242, Feb. 2011.

[127] C. Gomez, M. Gomez, P. Lopez, and J. Duato, "Exploiting wiring resources on interconnection network: Increasing path diversity," In Work. on Parallel, Distributed and Network-Based Processing, pages 20–29, Feb. 2008.

[128] B. Sethuraman, P. Bhattacharya, P. Bhattacharya, J. Khan, and R. Vemuri, "LiPaR: A Light-Weight Parallel Router for FPGA-based Networks-on-Chip," Proc. of the 15th ACM Great Lakes Symposium on VLSI, Chicago, Illinois, USA, April 17-19, 2005.

[129] H. Bokhari, H. Javaid, M. Shafique, J. Henkel, and S. Parameswaran, “Malleable NoC: Dark silicon inspired adaptable Network-on-Chip,” 2015 Design, Automation Test in Europe Conference Exhibition (DATE), Grenoble, pp. 1245-1248, 2015.

[130] M. Lodde, J. Flich, and M. E. Acacio, “Heterogeneous NoC Design for Efficient Broadcast-based Coherence Protocol Support,” IEEE/ACM Sixth International Symposium on Networks-on-Chip, Copenhagen, pp. 59-66, 2012.

[131] Y. J. Yoon, P. Mantovani, and L. P. Carloni, “System-level design of networks-on-chip for heterogeneous systems-on-chip,” Eleventh IEEE/ACM International Symposium on Networks-on-Chip (NOCS), Seoul, pp. 1-6, 2017.

[132] M. Evripidou, C. Nicopoulos, V. Soteriou, and J. Kim, “Virtualizing virtual channels for increased network-on-chip robustness and upgrade-ability,” in Proc. ISVLSI, Aug. 2012, pp. 21–26.

[133] J. S. Kim, M. B. Taylor, J. Miller, and D. Wentzlaff, “Energy characterization of a tiled architecture processor with on-chip networks,” Proceedings of the 2003 International Symposium on Low Power Electronics and Design, ISLPED ’03., Seoul, South Korea, pp. 424-427, 2003.

[134] M. B. Taylor et al., “Evaluation of the Raw microprocessor: an exposed-wire-delay architecture for ILP and streams,” Proceedings. 31st Annual International Symposium on Computer Architecture, Munchen, Germany, pp. 2-13, 2004.

[135] A. GÉGO. Study and performance analysis of cache-coherence protocols in shared-memory multiprocessors. Master Thesis. École polytechnique de Louvain. 2015-2016.

[136] C. Bienia, S. Kumar, J. P. Singh, and K. Li, “The PARSEC benchmark suite: Characterization and architectural implications,” 2008 International Conference on Parallel Architectures and Compilation Techniques (PACT), Toronto, ON, Canada, pp. 72-81, 2008.

[137] C. Bienia, S. Kumar, J. P. Singh, and K. Li, “The PARSEC Benchmark Suite: Characterization and Architectural Implications,” Princeton University Technical Report TR-811-08, January 2008.

[138] F. Zeng, L. Qiao, M. Liu, and Z. Tang, “Memory performance characterization of spec cpu2006 benchmarks using tsim,” Physics Procedia, vol.33, no. 0, pp. 1029-1035, 2012.

[139] S. Lian, Y. Wang, Y. Han, and X. Li, “BoDNoC: Providing bandwidth-on-demand interconnection for multi-granularity memory systems,” In Proc. the 22nd Asia and South Pacific Design Automation Conf., pp.738-743, Jan. 2017.

[140] A. Islam, “Technology scaling and its side effects,” 19th International Symposium on VLSI Design and Test, Ahmedabad, pp. 1-1, 2015.

[141] Dynamic branch prediction. http://ece-research.unm.edu/jimp/611/slides/chap4_5.html

[142] A. Flores, J. L. Aragón, and M. E. Acacio, "Sim-PowerCMP: A Detailed Simulator for Energy Consumption Analysis in Future Embedded CMP Architectures," Proc. Fourth Int. Symp. Embedded Computing (SEC-4), pp. 752-757, May 2007.

[143] A. Ros, M. E. Acacio, and J. M. García, "Cache Coherence Protocol for CMPs," Universidad de Murcia, Spain, 2019. http://ditec.um.es/ aros/papers/pdfs/aros-pdc10.pdf

[144] J. Huh, C. Kim, H. Shafi, L. Zhang, D. Burger, and S. W. Keckler, "A NUCA substrate for flexible CMP cache sharing," 19th Int'l Conference on Supercomputing (ICS), pp. 31-40. 2005.

[145] M. Shah et al., "UltraSPARC T2: A highly-threaded, power-efficient," SPARC SoC, IEEE Asian Solid-State Circuits Conference, pp. 22–25, 2007.

[146] M. Zhang and K. Asanovic, "Victim replication: Maximizing capacity while hiding wire delay in tiled chip multiprocessors," 32nd Int'l Symp. on Computer Architecture (ISCA), pp. 336–345, 2005.

[147] M. Palesi and M. Daneshtalab. Routing Algorithms in Networks-on-Chip. Springer New York Heidelberg Dordrecht London. 2014

[148] C. Izu, "On the Use of Multiplanes on a 2D Mesh Network-on-Chip," International Conference on Algorithms and Architectures for Parallel Processing (ICA3PP) Lecture Notes in Computer Science, vol 7017. Springer Lecture Notes in Computer Science (LNCS), Berlin, Heidelberg, 2011.

[149] M. Daneshtalab. Exploring Adaptive Implementation of On-Chip Networks, University of Turku, 2011.

[150] K. Choi, R. Soma, and M. Pedram, "Dynamic voltage and frequency scaling based on workload decomposition," in Proc. of Int'l Symp. on Low Power Electronics and Design, CA, USA, pp. 174-179, 2004.

[151] J. M. Rabaey. Digital Integrated Circuits: A Design Perspective. Prentice-Hall Inc., NJ, USA, 1996.

[152] L. A. Barroso et al., "Piranha: A scalable architecture based on single-chip multiprocessing," 27th Int'l Symp. on Computer Architecture (ISCA), pp. 12-14, 2000.

[153] P. J. Fleming and J. J. Wallace, "How Not to Lie With Statistics: The Correct Way to Summarize Benchmark Results," Communications of the ACM (Computing Practices), Vol. 29, No. 3, March 1986.

[154] R. Fung, V. Betz, and W. Chow, "Slack Allocation and Routing to Improve FPGA Timing While Repairing Short-Path Violations," in IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 27, no. 4, pp. 686-697, April 2008.

[155] A. Akram. A Study on the Impact of Instruction Set Architectures on Processor's Performance. Master Thesis. Western Michigan University. 2017.