Power, Performance and Reliability Optimisation of On-Chip Interconnect by Adroit Use of Dark Silicon

by

Haseeb Bokhari

A Thesis

Submitted in Accordance with the Requirements

for the Degree of

Doctor of Philosophy

School of Computer Science and Engineering

The University of New South Wales

September 2015

ORIGINALITY STATEMENT

'I hereby declare that this submission is my own work and to the best of my knowledge it contains no materials previously published or written by another person, or substantial proportions of material which have been accepted for the award of any other degree or diploma at UNSW or any other educational institution, except where due acknowledgement is made in the thesis. Any contribution made to the research by others, with whom I have worked at UNSW or elsewhere, is explicitly acknowledged in the thesis. I also declare that the intellectual content of this thesis is the product of my own work, except to the extent that assistance from others in the project's design and conception or in style, presentation and linguistic expression is acknowledged.'

Signed …… …….....

Date 9/06/2016 …….....

COPYRIGHT STATEMENT

Signed …… ……..

Date ……09/06/2016…….....

AUTHENTICITY STATEMENT

Signed …… .....

Date ……09/06/2016…….....

Thesis Publications

- **H. Bokhari**, H. Javaid, M. Shafique, J. Henkel, S. Parameswaran, SuperNet: Multimode Interconnect Architecture for Manycore Chips. In *Design Automation Conference*, DAC, 2015

- **H. Bokhari**, H. Javaid, M. Shafique, J. Henkel, S. Parameswaran, Malleable NoC: Dark Silicon Inspired Adaptable Network on Chip. In *Design, Automation and Test in Europe*, DATE, 2015

- **H. Bokhari**, H. Javaid, M. Shafique, J. Henkel, S. Parameswaran, darkNoC: Designing Energy Efficient Network-on-Chip with Multi-Vt Cells for Dark Silicon. In *Design Automation Conference*, DAC, 2014

- **H. Bokhari**, H. Javaid, S. Parameswaran, System-Level Optimization of On-chip Communication Using Express Links for Throughput Constrained MP-SoC. In *IEEE Symposium on Embedded System for Real Time Multimedia*, ESTIMedia, 2013

Other Publications

- H. Javaid, Y. Yachide, S. M. Min, **H. Bokhari**, S. Parameswaran, FALCON: A framework for HierarichAL Computation of Metrics for CompONent-Based Parameterized SoCs. In *Design Automation Conference , DAC*, 2014

Acknowledgements

My journey towards PhD has been full of excitement, frustration and gratification. I take this opportunity to acknowledge people who helped and encouraged me on every step of my journey.

First of all I thank Almighty Allah for His endless blessings. It was through His kindness that I was able to overcome all the challenges that I faced during my research.

I extend my deepest gratitude to my PhD supervisor, Prof. Sri Parameswaran. I had almost no research experience when I entered the PhD programme, but Prof. Parameswaran believed in me and taught me the basic skills of identifying and solving a research problem. He has always been there to listen to my problems, in either research or life. I will always be grateful to him for his supervision and support. I thank my friend and research colleague Dr. Haris Javaid for his valuable feedback on my research. I am also very grateful to Dr. Muhammad Shafique for collaborating with me on research ideas and providing useful feedback.

I acknowledge the valuable reviews of the research committee consisting of Prof. Gernot Heiser, Dr. Oliver Diesel and Dr. Annie Guo. Their timely feedback on my research was valuable for steering my research in the right direction. I take this opportunity to acknowledge the funding support I received from the School of Computer Science and Engineering, UNSW Faculty of Engineering and UNSW Graduate Research School.

I am very grateful to my colleagues at Embedded System Lab: Isuru, Dr. Tuo, Darshana, Pasindu, Lawrence, Dr. Liang, Dr. Su, Dr. Jorgen, Dr. Angelo, Dr. Joseph, Babak, Arash and Daniel. Thank you all for sitting through my research talks and giving me invaluable feedback.

I made some great friends when I came to Sydney who made my stay a wonderful experience. Thank you Murtaza and Zafar for hosting me for numerous dinners and lunches. Thank you Malik, Shoaib and Hamid for wonderful weekend meetups.

Last, I thank my family for their undaunted support throughout my life. Thank you mom and dad for encouraging me to pursue my passion and providing me all kinds of support I needed. Thank you my brothers Ammar Hassan Bokhari and Shoaib Bokhari for being there when I needed you the most.

Abstract

Continuous transistor scaling has enabled computer architecture to integrate increasing numbers of cores on a chip. As the number of cores on a chip and application complexity has increased, the on-chip communication bandwidth requirement increased as well. Packet switched *Network-on-Chip (NoC)* is envisioned as a scalable and cost effective communication fabric for multicore architectures with tens and hundreds of cores. Extreme transistor scaling (45nm and beyond) has its own share of technical challenges. Traditionally, semiconductor scaling resulted in reduction in both area and power of the transistor at the same time, a scaling law know as *Dennard's Scaling*. However, for recent technology nodes, the power per transistor is not reducing at the same rate as area. Failed Dennard's Scaling has resulted in a situation where we have abundant transistors, but not enough power to switch on these transistors at the same time, a phenomenon termed *Dark Silicon*. Previous research on dark silicon identified the energy inefficiency of general purpose computing cores as the cause of dark silicon and hence proposed integrating application specific accelerators or cores to improve energy efficiency and reliability, completely neglecting the interplay of dark silicon and NoC architecture.

For the first time, this thesis proposes various NoC architectures that exploit dark silicon to improve the energy efficiency, performance and reliability of the on-chip interconnect. The first proposal is an on-chip interconnect, named *darkNoC*, that consists of multiple NoCs where each NoC is optimised at design time using multi-vt optimisation for different voltage-frequency (VF) levels. This architecture is based on the observation that instead of applying DVFS to a router that has been designed to operate at a higher VF level, using a router that has been designed specifically for a lower VF level is more energy efficient. The architecture operates autonomously for seamless switchover between different NoCs. This architecture can provide up to 52% saving in NoC energy delay product (EDP) for certain benchmarks, whereas

state-of-the-art DVFS scheme only saved 15% EDP. Then, the *Malleable NoC* architecture is proposed, which improves the energy efficiency of NoC by a combination of multiple VF optimised routers and per node VF selection. We exploit the heterogeneity of application workload and application-to-core mapping to compose a communication fabric at runtime by choosing a VF optimised router from each mesh node. In one of the benchmark applications, darkNoC saved 26.4% NoC EDP whereas Malleable NoC saved 46% EDP, showing the efficacy of Malleable NoC architecture in the presence of workloads with heterogeneous applications.

Next, this thesis proposes *SuperNet* NoC architecture, that exchanges dark silicon for optimising the energy, performance and reliability of on-chip interconnect. *SuperNet* consists of two parallel NoC planes that are optimised for different VF levels, and can be configured at runtime to operate in energy efficient mode, performance mode or reliability mode. Our evaluation with a diverse set of applications show that the energy efficient mode can save on average 40% NoC power, whereas the performance mode improves the average core IPC by up to 13% for high MPKI applications. The reliability mode provides safety against soft error in the data path through the use of strong byte oriented error correction codes, and in the control path through dual modular redundancy and lock step processing.

Finally, a design flow for designing custom on-chip communication for application specific MPSoCs targeting streaming applications is proposed. Streaming application, such as *H264 encoder*, are normally realised using a set of processors logically arranged in pipeline fashion, and data is passed to different pipeline processors using on-chip communication. The proposed framework considers both data volume and pipeline latency to insert an optimum number of express communication links between certain cores so that communication latency is reduced and the pipeline throughput meets a certain constraint. To reduce the runtime of the framework, a heuristic with linear time complexity is proposed for exploring exponential design space, reducing framework runtime by 27× compared to a state-of-the-art heuristic, yet producing optimal or near optimal solutions.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Multicore Architecture

The semiconductor industry has gone through radical reform in the last six decades. The first working transistor was demonstrated at Bell Labs in 1947. This was such an important invention that Shockley and his colleagues were awarded a Noble Prize for their contributions. From there on, there has been an uninterrupted effort to reduce the size of a transistor and make it cheaper in terms of both power and cost. In 1965, Moore [3] predicted that the number of transistors available on a chip will double every 18∼24 months, a statement commonly known as Moore's Law. Invariably, Moore's Law has been the main driving force behind dense transistor integration. The quest to harvest maximum performance from silicon and advancements in fabrication techniques has enabled us to produce chips such as the Intel Xeon Haswell-E5 that contains 5.5 billion transistors. This scientific development has revolutionised the whole computing paradigm.

Traditionally, single core processors have been used in most of the computing systems around us (Fig1.1). To fulfil the ever-increasing computation power requirements, computer architects were inclined towards increasing the operating frequency

of single core chips or introducing more complex single core designs. Reduced transistor delay and increasing transistor density enabled computer architectures to do so without any radical changes in core architecture. However about a decade ago, researchers observed two major issues with this trend. First, a large number of the applications do not exhibit any significant instruction level parallelism (ILP) and thread/task level parallelism is a more energy and silicon effective solution for improving the computing throughput. Second, as we approached 45nm and smaller node sizes, the CMOS transistor exhibited poor energy scaling and therefore the thermal density of the chips started increasing. With a limited thermal budget, operating the chip at higher frequency can induce permanent damage. It was not very hard to realise that we were approaching a practical limit on performance of a single core processor and the only way forward was to shift our attention from architectures containing complex high-frequency single core to architectures with multiple leaner cores running at lower frequency.

Multicore architectures are becoming a ubiquitous hardware architecture for almost every computing machine around us. From mobile phones to cloud computing

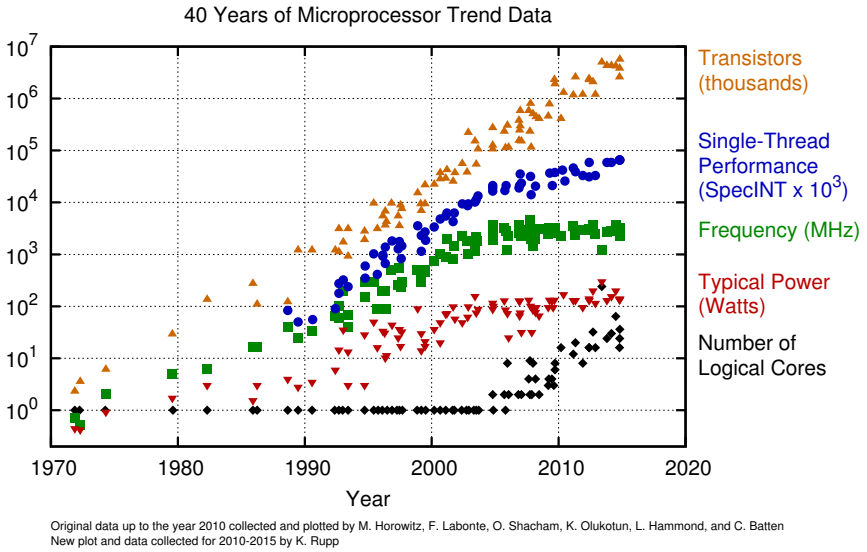


Figure 1.1: Effect of Semiconductor Scaling on Microprocessor Design

infrastructures, multicore architectures are playing a pivotal role in progress in technology. High performance multicore architectures have enabled humans to explore applications which were not realisable without the availability of such computing power. For example, gaming consoles nowadays can render life-like images at high FPS (frame per second) thanks to ever improving multicore GPU (Graphical Processing Unit) technologies. Similarly, we are getting closer to simulating atomic forces in proteins using application-specific multicores [4]. It is predicted that number of cores per chip will continue to climb, following Moore's Law. Researchers have predicted a 7.2× increase in computing power between now and 2024 [5]. This proves that multicore architectures will remain an area of interest for researchers in years to come.

1.2 On-Chip Interconnect Architecture

Every multicore chip has two major on-chip components; processing elements (*core*) and communication and memory architecture (*uncore*). Although high transistor density enables computer architects to integrate tens to hundreds of cores in a chip, the main challenge is to enable efficient communication between such a large number of on-chip components. The on-chip communication architecture is responsible for all memory transactions and I/O traffic, and provides a dependable medium for inter-processor data sharing. The performance of on-chip communication plays a pivotal role in the overall performance of the multicore architecture. The advantage of having multiple high performance on-chip processors can easily be overturned by an underperforming on-chip communication medium. Hence, providing a scalable and high-performance on-chip communication is a key research area for multicore architecture designers [6]. The main challenges faced by interconnect designers are:

- *Scalable communication for tens of cores:* It is fair to state that the performance of processing elements in multicore chips can be communication constrained [6]. Due to ever increasing improvement in processing capabilities, it

is quite possible to have a wide gap between the data communication rate and the data consumption rate. With tens of on-chip components, it is not possible to have single cycle communication latency between components placed at the far ends of a chip. Furthermore, with a large number of on-chip components, the on-chip interconnect is expected to support multiple parallel communication streams.

- *Limited power budget:* In 1974, Dennard predicted that the power density of transistor will remain constant as we move into lower node sizes. This is known as *Dennard's Scaling Law* [7]. However in the last decade or so, researcher have observed that the transistor's power cannot be reduced at the same rate as the area. Therefore, we are facing a situation where we have an abundance of on-chip transistors but do not have enough power to switch all these transistor at the same time, due to power and thermal constraints. Therefore, increasing the power efficiency of all on-chip components has become main the prerequisite to continue Moore's scaling. The on-chip communication architecture can consume roughly 19% of total chip power in a modern multicore chip [8]. Therefore, it is a challenging task to design a power-efficient on-chip interconnect that can still satisfy the latency and bandwidth requirements of current and future applications.

- *Heterogeneous applications:* A modern multicore chip is expected to execute a large set of diverse applications. Each application can interact with computing architecture in a unique way, hence the communication latency and bandwidth requirement can vary across different applications [9]. For example an application with a large memory footprint is expected to regularly generate cache misses and can hence be classified as a communication bound application. The performance of such applications is highly co-related with the efficiency of interconnect. On the other hand, an application with a smaller memory footprint is expected to be processor-bound and agnostic to on-chip

interconnect properties. Therefore, on-chip interconnects are often designed for worst-case scenarios (memory-bound applications in this case) and can therefore be inefficient for processor-bound applications. The situation is aggravated when both memory and processor bound applications are executed at the same time.

- *Selecting Interconnect performance metrics:* A major shortcoming in previous research is classifying the on-chip interconnect performance in terms of application-agnostic metrics such as transaction latency and memory bandwidth, instead of application-level performance metrics such as execution time and throughout [10] [11]. Therefore a major challenge is to extracting the correct metric to evaluate different possible interconnect architecture design points for a given set of applications.

- *Chip design cost:* The cost of designing a multicore chip has been increasing alarmingly due to high NRE cost[1] associated with small node sizes. A major portion of total chip cost is reserved for design verification and testing. Therefore, designers are expected to reuse previously designed and verified on-chip interconnects for new chip designs to reduce cost. With a multitude of interconnect architectures available, it is important to incorporate time-efficient design space exploration tools in research the phase to select the most suitable interconnect for a given set of target applications.

- *Interconnect Reliability:* With reducing node size, the concerns about the reliability of the digital circuits are on the rise. Any unexpected change in operating conditions such as supply voltage fluctuation, temperature spikes or a random alpha particle collision, can cause erratic behaviour in the output of a circuit. A soft error in on-chip interconnect can result in erroneous application output, or system deadlock if the control data is corrupted. Multicores

[1]NRE Cost: *Non Recurring Engineer Cost*. The term is used to classify the cost associated with researching, prototyping and testing a new product

are finding their ways into reliability-critical applications such as autonomous driving cars and medical equipment. Therefore designers are expected to integrate varying levels of reliability features in on-chip interconnect under given power and area constraints.

- *Co-Design of memory hierarchy and on-chip interconnect:* In modern multi-core architectures, on-chip memory hierarchy is closely coupled with on-chip interconnect architecture. In fact, for shared memory architectures, on-chip communication is the major factor in deciding the performance of memory hierarchy (cache, DRAM controllers, etc.). Therefore interconnect designers are often faced with the challenge of exploring the combined design space of memory hierarchy and on-chip interconnect.

1.3 Evolution of On-Chip Interconnect

1.3.1 Buses and Crossbars

Traditionally, System-on-Chip (SoC) designs used a very simple on-chip interconnect such as ad-hoc point-to-point connections or buses. The bus-based architecture is perhaps the oldest on-chip interconnect type in the computer industry and is still used in many SoC applications [12]. For a small number of on-chip components, bus interconnect is easier to integrate due to simple protocol design, and is efficient in terms of both power and silicon cost. However, poor global wire scaling prohibited building bus-based SoCs with large number of components due to increased delays. As the length of the wires in a bus interconnect increased the load capacitance also increased, and hence large wire buffers were required to drive the buses. This increase the on-chip interconnect's power budget. Furthermore, buses also showed performance scalability issues: As the number of on-chip components increased, the high traffic injection rates saturated the limited bus bandwidth, creating a performance bottleneck.

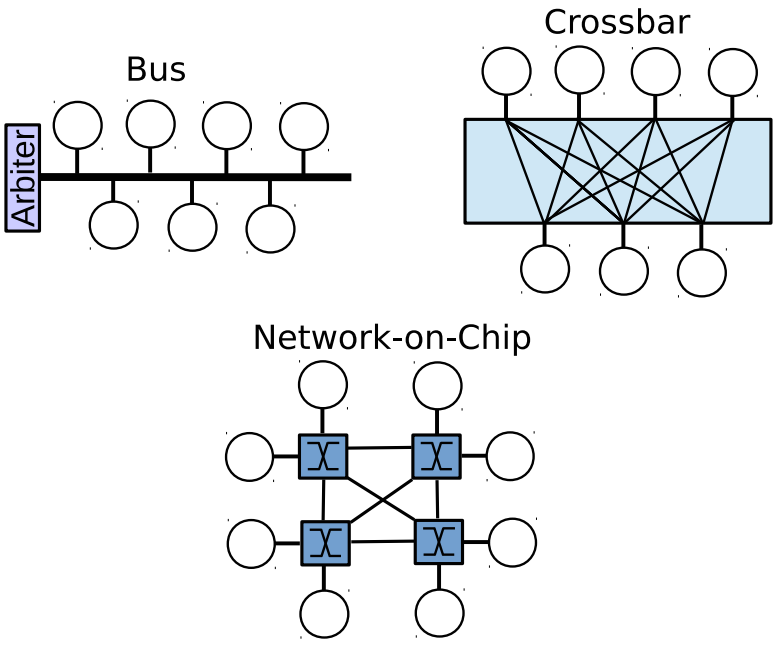


Figure 1.2: Evolution of On-Chip Interconnect

Crossbars were proposed as a solution to the limited bandwidth problem of the buses as it provides multiple communication paths. However, as the number of on-chip components increased, crossbars also showed poor scalability in terms of area and power. Furthermore, designing high radix crossbars increased the wire layout complexity and hence the overall chip design complexity [13].

1.3.2 Network-on-Chip

Packet switched *Network-on-Chip (NoC)* [14] interconnect was proposed as a solution for scalability issues of bus and crossbar-based on-chip interconnect. Essentially, a NoC consists of routers spread across the chip (Figure 1.2). These routers are connected with point-to-point links. Data is exchanged between different SoC components by generating data packets according to a predefined format. The data packets travel across multiple routers depending on physical location of source-destination pairs and routing algorithm.

The key advantages of using NoC as the on-chip interconnect are:

- NoCs inherently support multiple communication paths through a combination of physically distributed routers and links, which greatly increases the available on-chip data bandwidth. This enables different cores to exchange data in parallel without any central arbitration. This makes NoC an ideal candidate interconnect for supporting increasing communication needs of multicores chips with tens and hundreds of cores. Multiple communication paths between given source and destination cores give NoC an inherent fault tolerance. In case of permanent error in the router or link on a given path, data can be rerouted through an alternate path between source and destination cores.

- NoC architecture use short electric wires that have highly predictive electric properties. Compared to bus interconnect, smaller drive transistors are required to switch short wires between routers. This helps to improve the *energy/bit* metric of interconnects. Moreover, due to shorter wire delays, NoCs can be switched at higher frequencies than buses and crossbars without any significant increase in power. Deep sub-micron semiconductor manufacturing introduces integrity issues in wires. Having shorter wires reduces the probability of manufacturing faults and hence improve the production yield. The predictive electric properties of short wires also help in reducing design verification cost.

- NoCs follow a modular design paradigm by allowing reuse of existing hardware intellectual property blocks (IPs). For most designs, NoCs can be easily scaled for different number of cores and applications by simply instantiating multiple copies of existing designed and verified router IPs. This reduces the overall complexity of the chip design process.

- NoCs provide a clear boundary between computation and communications Therefore all on-chip components, irrespective of their types (memory controllers, processing cores, hardware IPs, etc.), can communication with each other using a standard packet format without caring about the underlying

implementation of on-chip components. This is a useful feature for designing heterogeneous SoCs with hardware components sourced from different internal or external sources.

1.4 Thesis Overview

1.4.1 Semiconductor Scaling and Dark Silicon

With transistor scaling, the supply voltage is reduced to proportionally reduce the power, and the transistor threshold voltage is also reduced to maintain the transistor switching frequency [6]. However, reducing the threshold voltage increases the subthreshold current ($I_{subthreshold}$) which has a large share in transistor leakage current. Furthermore, with reduction in transistor size, the thickness of the gate oxide layers is also reduced. With a thinner oxide layer, the number of electrons crossing from gate to substrate increases and hence the I_{gate} is increased. Overall, the effects of transistor scaling can be summarised as:

- The contribution of leakage power towards total chip power increases with transistor scaling [15]. To hold leakage power in check, the threshold voltage must be reduced, which in turn can reduce the switching frequency of the transistor.

- The effect of process variation has increased with transistors scaling [16, 17]. This limits the reduction of supply voltage beyond a certain limit without affecting the reliability of the transistors. This is a major cause of failing Dennard's Scaling.

The net effect of the combination of continuous transistor scaling and poor power per transistor scaling is that, for lower node size, we have more than enough transistors available on a chip but do not have enough power to switch on these transistors

at the same time. Esmaeilzadeh [18] names this phenomenon *Dark Silicon* and defines it as:

"the fraction of chip that needs to be powered off at all times due to power constraints."

Henkel et al. [19] further expanded this definition in terms of power and thermal budget. They argue that dark silicon result from a limited power budget (e.g., mobile phones) or due to a limited thermal budget (e.g., cloud computing infrastructure).

Irrespective of the criteria used to define *dark silicon*, it is currently a hot topic for research in the architecture and EDA community [5, 19, 20]. It is anticipated that innovations at circuit, architecture and software levels are required to mitigate the effect of dark silicon and rejuvenate Dennard's Scaling [21].

1.4.2 Thesis Motivation

Researchers believe that a key cause of dark silicon is the energy inefficiency of general purpose processing cores [18]. Most prior research in dark silicon has been dedicated to improve the energy efficiency of processing elements through device and architecture level heterogeneity [19]. Design flows introduced recently exploit the available dark silicon available to integrate high performance, low energy, application-specific accelerators and cores to general purpose multicores [22–30].

NoCs can potentially consume a large share of total chip power [6]. For example, NoC consume 19% power in experimental 36 core SCORPIO chip [8] and 28% of total power in Intel's TeraFlop chip [31]. The share of NoC power will remain high for dark silicon chips as a number of processing cores remain switched off yet the whole NoC needs to be functional to enable communication between working cores. In fact, as shown in Figure 1.3, Bejestan et al. [1] estimate that NoC routers will consume 18∼24% power in different configurations of dark silicon multicore chips fabricated in 7nm technology.

Following our literature review, we conclude that prior research on dark silicon

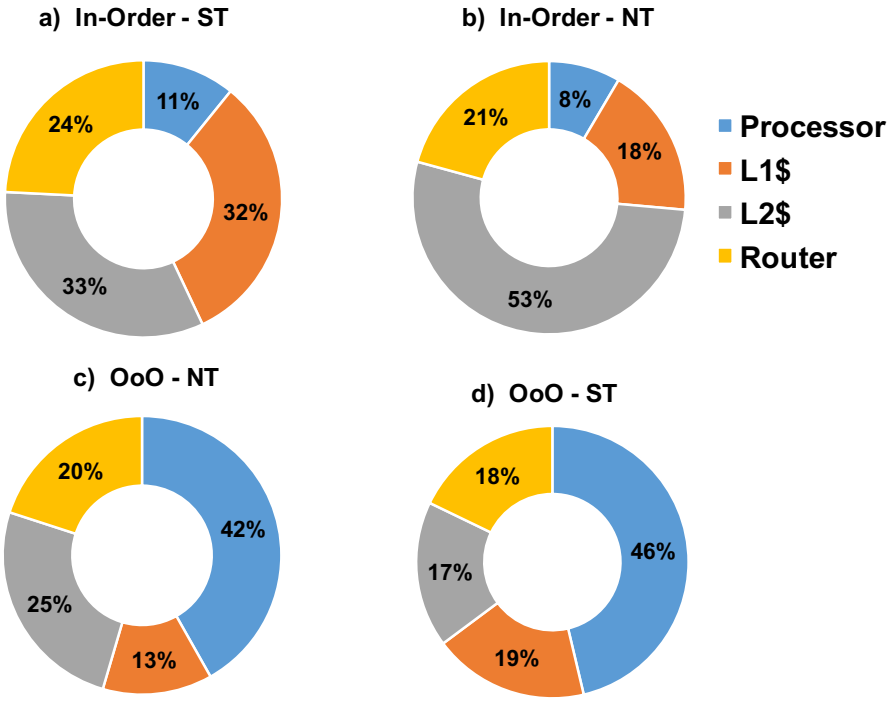


Figure 1.3: Power breakdowns of the In-order multicore processor in (a) super-threshold (ST) and (b) near threshold (NT) regimes, and the OoO multicore processor in (c) ST and (d) NT regimes under 7nm FinFET technology. [1]

does not investigate the interplay of dark silicon and NoC. Therefore, the main motivation for this thesis is to study how dark silicon can be used to improve the energy efficiency and other NoC design metrics, such as reliability and performance. Furthermore, in the competitive financial market, the product design cost and time-to-market are the key factors that decide the success or failure of a product. Therefore, NoC design flow should be time efficient and the resultant architectures should be robust to match different application requirements.

1.4.3 Research Problems

Failing Dennard's Scaling and continuous transistor scaling have changed the designer's perspective on the priorities of chip design. Maximising chip performance under given power and area constraint has always been the primary objective. For

45nm and higher node sizes, after performance optimisation, effective use of area was the primary design objective, whereas reducing power was a secondary design objective. However, smaller nodes are more adversely effected by the failure of Dennard's Scaling. Therefore power has become the primary optimisation goal, whereas area is considered a cheap commodity(dark silicon). Furthermore, with increasing chip design costs, designers are interested in consolidating as many features as possible in a single chip to cut down on NRE costs.

In the perspective of varying optimisation goals and commonplace use of NoCs in modern multicores, the following questions are investigated in this thesis:

1. Is it possible to use available dark silicon to improve the energy efficiency of exiting NoC architectures?

2. Can we design a NoC that can be configured at runtime to target different optimisation goals such as power, performance and reliability?

3. What kind of cost-effective design flow is required to design a new NoC for a given set of applications by reusing the existing NoC IP?

1.4.4 Thesis Contributions:

This thesis makes the following contributions:

- **Energy Efficient NoC Architectures:**

  – *Dynamic Voltage and Frequency Scaling* (DVFS) is a well known technique for reducing the power consumption of NoCs in modern multicore architectures. During our experiments with logic synthesis of NoC router architecture, we observed that the efficiency of DVFS technique can be improved by instantiating at a given node multiple routers that are designed specifically for a particular voltage-frequency (VF) level using multi-vt optimisation circuit design technique. Based on this observation, we introduce a novel NoC

architecture which consists of multiple architecturally homogenous yet VF optimised NoC layers. We present the architectural features that are necessary for fast and seamless inter layer switching. Furthermore, we present a scalability study of our proposed architecture. We present procedures that must be followed for seamless handover of NoC traffic from one NoC layer to another while ensuring lossless property of on-chip networks.

– During our experiments, we observed that NoC-based multicores are expected to execute workloads with heterogeneous applications, and every application's execution time may or may not be dependent on NoC latency. We also observed that most NoC multicore are built on the principle of non-uniform memory access (NUMA) which essentially means that application-to-core mapping is vital in studying the effect of NoC latency on application performance. Based on these observations we present a runtime configurable NoC architecture that consists of routers designed for different VF levels through multi-vt optimisation. Depending on runtime application cache profiling and application-to-core mapping, a controller selectively connects different VF optimized NoC routers. Based on the observation that NoC has to compete with other computing components such as accelerators for dark silicon, we propose an architecture where only a subset of nodes have replicated routers for lower VF. We present an analysis that shows how the amount of dark silicon affects the energy efficiency of our proposed architecture.

- **Multimode NoC Architecture:** To tackle growing chip design costs and to cater the need of SoCs that can be used for multiple-use scenarios, we propose a multimode NoC architecture. Our proposed NoC architecture consists of multiple VF optimised NoC layers that can be configured at runtime to run in one of the three modes: energy efficient mode, reliability mode and performance mode. In energy efficient mode, NoC switches between NoC layers designed for low and high VF, depending on the application's communication

needs. For performance mode, multiple NoC layers are powered on at the same time and data is injected in one of the NoCs depending on how critical is the data latency for the application's performance. For reliability mode, multiple NoCs operate in lock step to ensuring control path protection whereas strong error detection and correction codes are transmitted on secondary NoC for data path protection. We discuss power and performance tradeoffs of different modes, and based on experimental observation, we provide general recommendations for which mode is suited to certain applications.

- **Application-Specific NoC Design:** We propose a design flow for designing application-specific on-chip interconnect for *streaming applications*. It is becoming common to design specific multicores for performance-constrained applications such as H264 encoder, etc. The proposed scheme modifies a given baseline on-chip interconnect such that a certain application throughput constraint can be met. Low latency express links are added between communicating cores so that a certain fraction of data that is communicated through these links rather than the baseline interconnect. A key idea of this scheme is choosing application-level metrics (the streaming application's throughput) rather than network level metrics for evaluating different design space points.

1.4.5 Thesis Organization

The Chapter 2 of this thesis introduce the preliminaries of multicore design in general, and on-chip interconnect design in particular. This chapter reviews the previous research on NoC power, performance and reliability optimisation. Some of the circuit design techniques used in this thesis are also discussed briefly.

Chapter 3 describes in detail the *darkNoC* NoC architecture that uses abundantly available transistors to optimize the energy efficiency of the DVFS scheme for NoCs.

Chapter 4 describes the *MalleableNoC* architecture that exploits application heterogeneity for connecting heterogeneous VF optimised routers at runtime.

Chapter 5 describes the multimode NoC architecture called *SuperNet* that can be configured at runtime for three possible goals: energy efficiency, performance or reliability.

Chapter 6 covers the framework for design of on-chip interconnect for application specific MPSoCs.

Chapter 7 concludes the thesis and discuss future research directions.

Chapter 2

Preliminaries and Literature Review

There is a plethora of prior research on optimising Network-on-Chip design metrics such as power, performance and reliability. The purpose of this chapter is to introduce the reader to concepts that are necessary to understand various design techniques used in this thesis and provide a philosophical overview on how techniques proposed in this thesis are different from prior research.

The chapter starts with an introduction on power consumption in CMOS digital circuits. We then discuss circuit and architecture level techniques used in digital system to cut down dynamic and leakage power. This is followed by a study on how transistor scaling is changing different aspects of power aware design.

We then shift our focus on on-chip communication architecture design and introduce the reader to some essential concepts of Network-on-Chip design. This is followed by a discussion on the commonly used power saving techniques used for NoCs and the drawbacks and limitations of these techniques.

We then concentrate on performance optimisation through intelligent mapping of applications on multicores. We conclude the chapter with a discussion of various application specific on-chip interconnect design methods.

2.1 Reducing Power in Digital System

Interest in reducing the power of digital circuit is motivated by several factors. First, a large portion of digital chips manufactured today are used in battery powered devices such as mobile phone and tablets. These devices can store only a very limited amount of energy in batteries.Therefore one of the basic selling point of these devices is the expected battery life. As digital chips used in these devices claim most of the energy [32], power optimisation at various stages of chip design, is common practice. Second, even for devices such as personal computers and server farms that are connected to power grids, the thermal management of computing infrastructure is important due to the rising cost of ownership. The heat dissipation of a digital system is directly related to its electric power consumption. Therefore the cost of cooling infrastructure can be reduced considerably by efficiently managing the chip power. Moreover, a strong motivating reason to rethink the power consumption of digital system is failing Dennard's Scaling [5, 7]. For 45nm and smaller nodes, transistor power is not reducing in proportion to the reducing area, so the overall power density of digital chips is increasing. It is therefore anticipated that failing Dennard's Scaling will confound Moore's Law. [18].

2.1.1 Energy Dissipation in CMOS Circuit

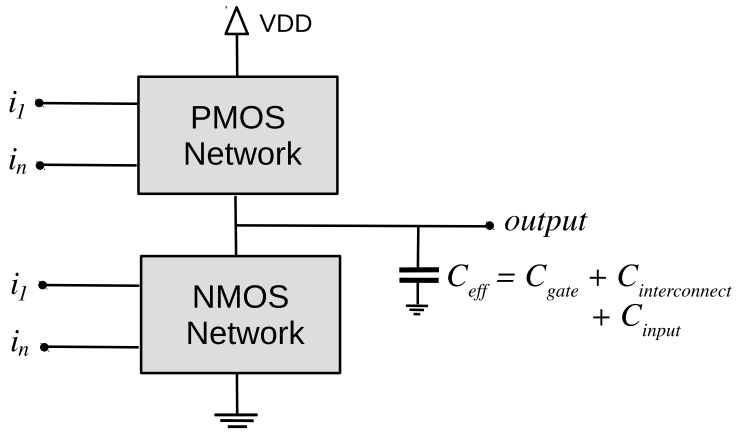


Figure 2.1: CMOS Gate

Figure 2.1 shows the structure of a generic CMOS logic gate. Each CMOS gate contains a network of PMOS transistors (pull-up network) and a network of NMOS transistors (pull-down network). Depending on the boolean function implemented by the gate and the inputs, either the PMOS or NMOS network conducts current. Similarly, the output of the logic gate is either *'0'* or *'1'*. The output is normally connected as an input to another set of CMOS gates, depending on the architecture of the digital circuit.

The power consumption of a CMOS circuit is defines as:

$$P_{CMOS} = P_{dyn} + P_{leakage}$$

where P_{dyn} is the power consumed whenever the output of the logic gate changes and $P_{leakage}$ is the power consumed irrespective of output switching. The dynamic power can be further defined as:

$$P_{dyn} = C_{eff}.V_{DD}^2.f.\alpha$$

In the equation above V_{DD} is the supply voltages, f is the operating frequency and α is the average number of output transitions. The term C_{eff} is the total effective capacitance of the CMOS gate and it calculated by:

$$C_{eff} = C_{gate} + C_{interconnect} + C_{input}$$

In the equation above, C_{gate} is the sum of all internal gate capacitances, $C_{interconnect}$ is the capacitance of interconnect wires between outputs of the transistors and inputs of the next stage, and C_{input} is the input capacitance of all the gates connected to the output of this gate. Essentially dynamic power is consumed whenever the capacitance C_{eff} is charged. The leakage power of the CMOS gate is defined as:

$$P_{leakage} = V_{DD}.I_{leakage}$$
$$I_{leakage} = I_{reverse} + I_{subthreshold} + I_{gate}$$

In the equations above, $I_{reverse}$ is the reverse current induced due to the PN junction characteristic of transistors, $I_{subthreshold}$ is the current due to carrier diffusion between

the source and drain regions of a transistor, and I_{gate} represents the current flowing from gate oxide to substrate and vice versa [33].

2.1.2 Power Saving Techniques

Having discussed the sources of power consumption in CMOS circuits, we will now provide an overview of classical and emerging power saving techniques that are relevant to the research problems explored in this thesis.

2.1.2.1 Gate Sizing

A source of dynamic power consumption is the charging and discharging of internal gate capacitance. This capacitance can be reduced by decreasing the gate size. However, reducing the gate size negatively affects its ability to quickly charge or discharge the load capacitances, and hence the maximum operating frequency. However, modern circuit synthesis tools exploit the positive slack of different circuit paths to size the gates in ways that reduced the overall dynamic power of the circuit while abiding by the latency requirement [34–36].

2.1.2.2 Clock Gating

A basic building block of synchronous digital circuit is *clock*. A set of flip-flops are inserted in the digital circuit to keep it synchronous. These flip-flops are connected to a high fanout clock network. Both flip-flops and clock network can consume 30-40% of the total dynamic power budget [37]. This is because a) the clock network and the flip-flops have a very high switching activity factor which can be close to 1 for a standard circuit, b) a high fanout clock network normally has a very high interconnect capacitance. Clock gating [37, 38] reduces the dynamic power consumption by reducing the activity factor. Essentially, in clock gating, the clock for a given set of flip-flops and clock network is disconnected depending on the usage

scenario.

Figure 2.2 shows an example circuit which uses the clock gating technique. The *Gating Logic* unit decides if this cirucit is active or inactive. The output of the *Gating Logic* is *ANDed* with the original clock signal and the output of this *AND* operation is fed in as the clock signal for sequential elements (flip-flops in this case). Disabling the clock reduces the switching activity of the two flip flops shown in the Figure 2.2 and reduces the load on the global clock network. Furthermore, as the output of the flip-flop is constant, there is no switching activity in the combinational logic l, and therefore the dynamic power of the combinational logic gates is also reduced.

2.1.2.3 Power Gating

Power Gating is the most commonly used technique to reduce the leakage power of a CMOS digital circuit. With increasing leakage power in modern multicore chips, aggressive Power Gating technique are being employed [6]. In this technique, the power supply to circuit blocks that are idle or not required is removed. This essentially cuts down both dynamic and leakage power as almost zero current flows through the circuit block that is power gated [39].

The circuits in Figure 2.3 demonstrate two possible methods of applying Power Gating techniques. In the circuit on the left, a NMOS *sleep transistor* is added between the CMOS logic block and the ground (V_{SS}). When the *control signal* is at logic *one*, the sleep transistor is conducting and the logic block can perform normal operation. However, when the *control signal* is *zero*, the logic block is disconnected

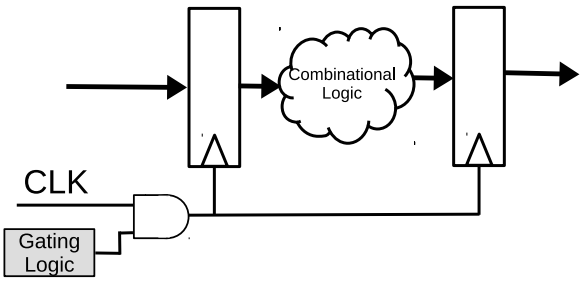


Figure 2.2: Clock Gating

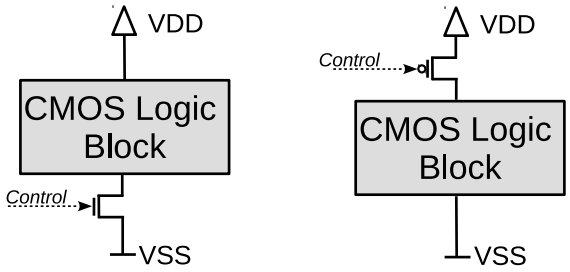


Figure 2.3: Power Gating

from ground. Similarly, for the circuit on the right in Figure 2.3, a PMOS *sleep transistor* is added between CMOS power supply (V_{DD}) and the logic block. However, a logic *one* on the input of this PMOS transistor power gates the circuit whereas a logic *zero* enables normal operation. The challenges in applying Power Gating technique are:

- It takes some time to turn on the sleep transistor back to the normal operation. Therefore, Power Gating can impose performance penalty if not done intelligently [40].
- Turning on and off certain parts of a chip can induce noise in the global voltage supply [41]. Mitigating supply noise requires cosierable effort during chip design phase.
- Some energy is required to switch on and off the sleep transistors. If power gating is performed frequently, the overhead cn be more than the actual power saving. Therefore, it is important, when in implementing a power gating technique, to determine the *break even cycles*: the minimum number of cycles for which the circuit should remain switched off to cover the overhead of power gating.

2.1.2.4 Multiple Voltage Domains

The relation between the switching delay of a transistor and the supply voltage can be defined by:

$$t \propto V_{DD} / (V_{DD} - V_t)^2$$

where V_{DD} is the supply voltage and V_t is the transistor threshold voltage. According to this equation, reducing the supply voltage increases the transistor delay [42] . It has been observed that SoC designs often contain elements that operate at different frequencies [43] depending on application and microarchitecture. One such heterogeneous SoC architecture is shown in Figure 2.4 which shows a CPU, an SRAM memory and an I/O device. The CPU in this example has the most stringent operating frequency requirement and therefore must be powered by higher supply voltage. However, using a high V_{DD} for other SoC components which are required to be operated at a lower frequency can result in power wastage. Therefore, the SRAM and I/O in this example can be powered by a separate lower V_{DD} power supply. The efficacy of such an architecture has been shown in previous studies [44] [45] [46] [47].

The only concern about multiple voltage supply technique is the design and verification cost of the extra voltage rails required for this scheme. Furthermore, special *voltage level shift transistors* are required for logic signal crossing from one voltage domain to another.

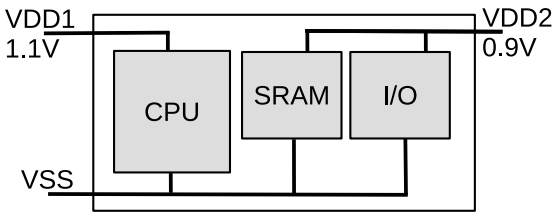


Figure 2.4: SoC with Multiple Voltage Supply Rails

2.1.2.5 Dynamic Voltage and Frequency Scaling (DVFS)

The dynamic power of a transistor can be reduced quadratically by reducing the supply voltage. However reducing the voltage can also increase the transistor delay. This energy-delay characteristic of CMOS transistor can be used to strike a balance between power and performance. However, selecting a certain static voltage-frequency

(VF) operating point at design time can be non-optimal for microprocessors that are expected to execute heterogeneous application workloads [48]. Therefore, a more robust technique is to monitor the workload characteristics and dynamically select the operating frequency depending on available execution slack. Once a certain frequency is selected, an appropriate voltage can be selected, one that is required to safely operate a digital circuit at that particular frequency [49].

It is difficult to predict the minimum voltage supply required to support a certain frequency [50]. Therefore microprocessor manufacturers, such as Intel, characterise the voltage and frequency response of the chips after fabrication and store this information as a look-up tables [51]. This information can be used at runtime by the operating system or firmware to implement DVFS schemes.

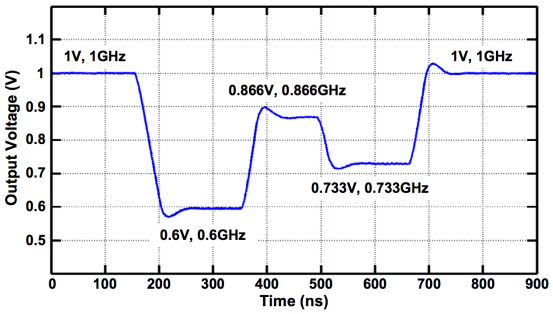


Figure 2.5: Fast DVFS using On-Chip Voltage Regulators [2]

Changing the VF level at runtime can incur performance loss depending on the switching speed of the voltage regulator controller (VRC). VRCs normally use either off-chip or on-chip Switching Voltage Regulators (SVR) to dynamically change the supply voltage [52]. Off-chip SVRs can perform voltage scaling with high efficiency. However, they suffer from high latency, which can often be in the range of hundreds of microseconds to a few milliseconds [51]. On-chip SVRs, on the other hand, can switch between voltage levels within a few nanoseconds [2], enabling fast fine grain DVFS. However, on-chip SVRs can suffer a considerable efficiency loss of 15∼20% [53] [2]. To improve the conversion efficiency of SVRs, researcher propose integrating on-chip low-drop-out (LDO) voltage regulators [54]. LDOs have two advantages over

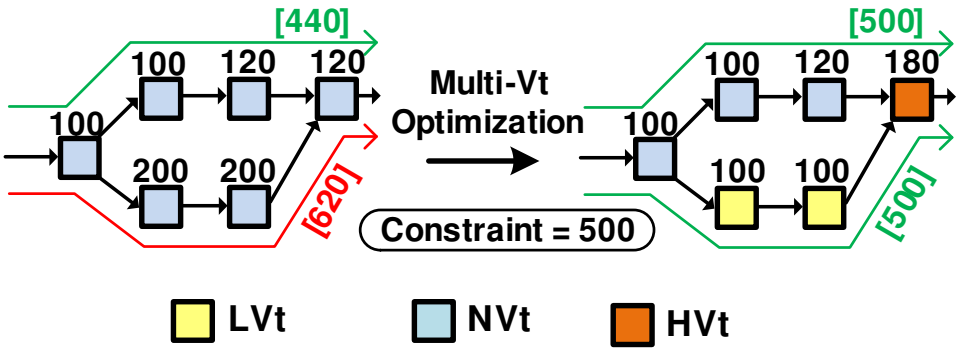


Figure 2.6: An example of multi-Vt circuit optimisation.

on-chip SVRs: 1) LDOs consume considerably less silicon area, as they do not use complex inductor and capacitors circuits, and, 2) because of low area cost, multiple LDOs can be used for different parts of the chip, and therefore a finer grain DVFS scheme can be used [55, 56].

2.1.2.6 Multi-Vt Optimisation

Commercial fabrication foundries such as TSMC and GlobalFoundaries provide cell libraries with various gate threshold voltages (Vt). These library packages contain cells with normal Vt (NVt), Low Vt (LVt), and High Vt (HVt). The availability of heterogenous Vt provides an opportunity for designers to optimize the the energy-delay characteristics of a circuit under a given synthesis latency constraint. For example, use of Low Vt (LVt) cells results in lower circuit latency, whereas a circuit made up of High Vt (HVt) cells has a high latency. However, the high performance of LVt cells comes at the cost of increased power consumption. For example, LVt cells can be up to 5× leakier than their HVt counterparts. In multi-vt optimisation, CAD tools exploit the energy-delay characteristics of the cells by inserting low Vt cells on circuit logic paths with negative slack to meet the latency constraint and replacing normal cells with high Vt cells on paths with positive slack to save energy. For example, in Figure 2.6, the circuit on the left contains only normal Vt cells and one of the logic paths has latency higher than the synthesis constraint. By using

multi-Vt circuit optimisation, some of the cells are replaced with low Vt and high Vt cells to ensure that the circuit meets the latency constraint while keeping the leakage power of the circuit to minimal.

2.1.2.7 Unconventional Semiconductor Devices

DVFS is a useful scheme for saving dynamic and leakage power in multicore chips. However, due to the dark silicon phenomenon, it is expected that a large number of cores in future many-core chips will operate at very low frequency due to power or thermal constraints, and to achieve energy efficiency [57]. Such a scheme is also called *near threshold computing*. However, scaling the supply voltage close to V-th of the CMOS circuit create problems. First, DVFS does not significantly affect the leakage power of a CMOS circuit, and this phenomenon becomes even worse for smaller node sizes. Second, reducing the supply voltage combined with process variation can result in unreliable operation due to uncertain transistor delays. Process variation can also introduce bit errors in on-chip SRAM-based devices used in *near threshold computing*.

Researchers argue that a key breakthrough in energy efficient computing architecture would be to use unconvential CMOS devices for low frequency operation [58] [59] [60]. One such promising semiconductor technology is *tunnel field effect transistor* (TFET) [59]. TFET has a lower threshold voltage than MOSFET and therefore a TEFT can be switched on at a lower gate voltage than a MOSFET [61]. Therefore, for the same voltage, TFET can be operated at a higher frequency. Moreover, TFETs have lower leakage power than MOSFET [61]. However TFET has higher delay for higher voltages, therefore TFET cannot be used in high frequency architectures. Based on these characteristics, Swaminathan et al. [28] proposed a heterogeneous multicore architecture where high frequency cores are fabricated using CMOS technology, but the low frequency cores are fabricated using TFETs.

Similarly, researchers have investigated architectures with STT-RAM based memories that use Magnetic Tunnel Junction (MTJ) transistors for low power architectures [62] [63]. STT-RAM has an advantage over standard SRAM because of near-zero leakage power and non-volatility. However, the main disadvantage of STT-RAM is higher write latency than SRAM [63].

2.2 Multiprocessor System on Chip (MPSoC)

The current advancement in fabrication technology has provided an opportunity to integrate multiple computing resources into a single chip; the term coined being Multiprocessor System on Chip(MPSoC) [64]. According to Wolf [64], MPSoCs have given a new direction to the field of Embedded System Design. MPSoC are targeted for embedded system, in contrast to homogenous CMP (chip multiprocessors), which are targeted towards general computing applications. This is why the main design objective, unique to embedded systems, is to maximise the computing power while keeping the energy consumption as low as possible [64]. However, some innovative cluster computing infrastructures have been built by integrating small scale MPSoC based clusters [65].

Over the last decade, many MPSoC and CMP architectures have been purposed. Unarguably, RAW Architecture [66] from MIT was the most influential work in the field of homogenous CMPs. The 72 core GX [67] series processor from Tilera and Intel 48-core Single Chip Cloud Computer [51] are the latest example of many core CMP designs. Intel IXP Network Processor [68], IBM Cell BE [69], Texas Instrument OMAP Architecture [70], NVIDIA Tegra Chipset [71], Qualcomm Snapdragon [72] and NXP Nexperia Architecture [73] are examples of widely used commercial MPSoC architectures in embedded system products. To maximize the performance, most of these MPSoCs have integrated heterogeneous processing elements like domain specific accelerators and Application Specific Instruction Processors (ASIPs). These architectures employ some proprietary or open source on-chip

interconnects standards. These architectures are usually tailor-made to meet the bandwidth requirement for the specific SoC component. ASIPs are being studied extensively for applications where general purpose processors fail to meet the performance requirements for embedded systems [74]. Moreover, ASIPs provide a better performance per watt ratio compared to general purpose CMPs.

2.3 Bus-Based SoC Architectures

The Bus-based architecture is perhaps the oldest on-chip interconnect standard in the computer industry, and is still used in many SoC applications [12]. The simplicity of protocol and hence low gate cost is possibly the main reason that bus-based architectures have dominated all other available on-chip interconnect options. In bus-based architectures, multiple components interact using a single data and control bus, hence providing a simple master-slave connection. Arbitration is required when multiple masters try to communicate with a single slave, giving rise to resource contention. Hence the scalability of bus-based architecture in terms of performance is questionable in a large SoC based designs [75]. Some classic design techniques for bus-based SoC proposed in [76, 77] use worst-case bus traffic to design optimal architecture. Kumar et al. [78] have given a detailed study of the scalability and performance of shared bus based chip multiprocessor(CMP) architectures. They concluded that a bus-based interconnect network can significantly affect the performance of cache-coherent CMPs.

Several improvements to traditional bus based interconnect architectures have been proposed. ARM Ltd., AMBA Architecture [79], IBM CoreConnect Architecture [80] and Tensilica PIF Interface [81] are few a examples of the widely used advance bus-based communication medium. All of these architectures provide several advanced functionalities like burst-based data transfers, multi-master arbitration, multiple outstanding transactions, bus locking and simultaneous asynchronous and synchronous communication. However, Rashid et al. [82] have analytically showed

that even advanced bus based architectures like AMBA are outperformed by modern NoC based communication architectures in terms of performance. However, the same study shows that designers are still inclined towards AMBA-based on-chip interconnects due to the area and energy overhead of modern NoC designs. SoC designers have a keen interest in apple-to-apple comparison of various commercial bus architectures, however the performance of on-chip interconnects greatly depends on each application's traffic pattern, bus microarchitecture and system parameters [83].

The simplicity of the bus-based architecture design, predictable access latency and low area overhead are the key selling points. However, beyond small number of cores, the performance of the bus interconnect degrades significantly [13].

2.4 Crossbar On-chip Interconnect

Single shred bus architecture is evidently slower in the case of multiple master-slave data transactions. The prime bottleneck is the single shared medium and latency due to arbitration among many master interfaces. Therefore, the first approach to design a scalable on-chip interconnect was adaption of crossbar topology. A crossbar is a matrix switch fabric that connects all the inputs with all the outputs enabling multiple communication connections. The idea has been borrowed from the Telecommunication industry where such architectures have been successfully used for four decades in telephony applications [84].

The same concept of multiple communication channels was implemented in the SoC design industry by combining multiple shared buses to form an all-input-all-output connection matrix. The concept is also known as Hierarchal Bus or Multi-layer Bus. STBus [43] is perhaps the best known commercial bus architecture that inherently support crossbar architectures. A design methodology for AMBA-based cascaded bus architecture is provided by Yoo [85]. Yoo et al. have experimented with integrating 90 IP blocks in a single crossbar based SoC design. Similarly,

authors in [86] have provided a complete design methodology for designing an application specific crossbar architecture using the STBus protocol. They claim significant performance improvements over standard single bus architectures. The most interesting crossbar implementation is the interconnect system for IBM Cyclops64 Architecture. Each Cyclops64 crossbar connects 80 custom processors and about 160 memory banks. With single transaction latency of 7 cycles and bandwidth comparable to state of the art NoC architecture, Cyclops64 interconnects is perhaps the most advanced practical crossbar design for the SoC domain.

Researchers have been arguing over the scalability of crossbar-based interconnect architecture due to the non-linear relation between number of the ports and latency and wire cost [87]. However, recent experiments from [84] show that a 128 x 128 port crossbar in a 90nm technology is feasible. They have benchmarked their crossbar design against state of the art mesh-based NoC design and concluded that the crossbar design matched NoC architectures in terms of latency, bandwidth and power consumption. However, the design complexity is prohibitively high due to complex wire layouts.

2.5 Network-on-Chip (NoC) Interconnect

Following Moore's Law of available on-chip transistor resources, we are looking beyond having thousands of cores on a single chip. It has been predicted that the performance of such kilo-core MPSoCs will be communication architecture dependent [88]. Traditional bus-based architectures cannot scale beyond a few tens of IP blocks and there is a need to provide a more scalable and protocol invariant communication architecture.

The solution to the scalability problem of bus based architectures was found in the form of Network-on-Chip architectures [14, 89]. NoC inherently supports the general trend of highly integrated SoC design and provides a new defacto standard of on-chip communication design. The basic idea of NoC has been adapted from the

well established structure of computer networks. In particular, the layered service architecture of computer networks has been well adapted in NoC to provide a scalable solution. In a NoC architecture, data is converted into packets and these packets are traversed through number of hops (switches or routers) based on a predefined routing technique.

NoC-based MPSoC designs have attracted attention of researchers for the last decade. The defining features of NoC design are router design, routing algorithms, buffer sizing, flow control and network topology. We will discuss them in more detail.

2.5.1 Topology

A NoC consists of routers and communication channels. NoC topology defines the physical layout of the routers and how these routers are connected with each other using the communication channels. The selection of NoC topology can have a significant effect on multicore performance, power budget, reliability and overall design complexity. For example, if the average number of hops between nodes is high, packets have to travel longer and hence network latency will be high. Similarly, if a topology requires very long physical links, designers have to make an effort to ensure timing closure for longer links. Moreover, topologies that allow diverse paths between nodes can potentially provide higher bandwidth and also prove to be more reliable in the case of faulty links. Therefore, when designing NoC based multicores, the first decision is to choose the NoC topology [13].

NoC topologies can be classified as *direct* and *indirect* topologies [13]. In direct topologies, each on-chip components such as core or memory, is connected with a router, and therefore each router is used to inject or eject traffic from the network. In indirect topologies, the on-chip components are connected only to *terminal* routers whereas the other routers only forward the traffic [13]. The *degree* parameter of a router defines the number of neighbouring routers and on-chip component to which

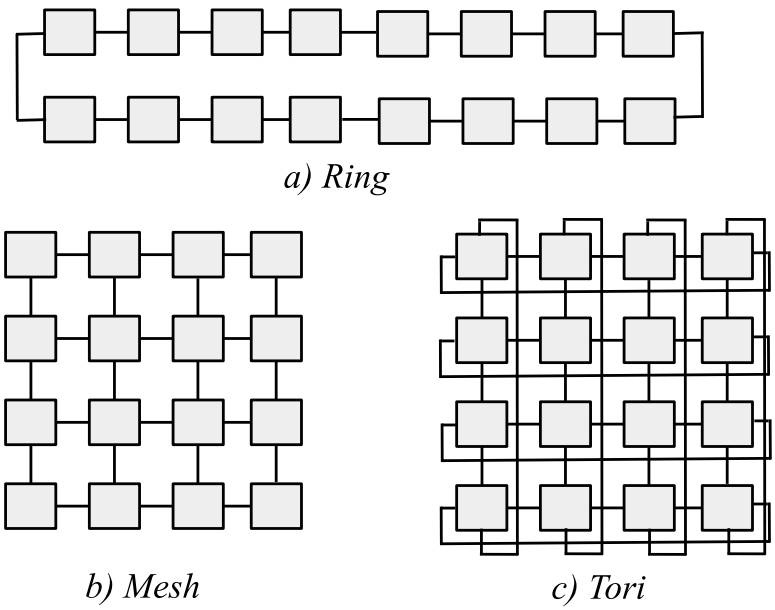


Figure 2.7: Well known direct topologies

the router has links. The degree parameter defines the number of input/output ports in each router. Note that the complexity of router microarchitecture increases with an increase in the degree of router.

Figure 2.7 shows three commonly used direct topologies *ring*, *mesh* and *tori*. The ring is the simplest topology to implement in terms of silicon area and design complexity. The degree of each router in ring interconnect is 3 (2 neighbour router + 1 local resource (core, memory, etc.)). The drawback of ring topology is performance scalability. The number of hops between two nodes in worst case scenarios is proportional to N: the number of nodes in the topology. Furthermore, rings provide limited bandwidth and are less reliable due to poor path diversity. Therefore, rings become impractical for multicore chips with more than 8-16 nodes [90].

Mesh and Tori solve the scalability problems of ring topology, albeit at a cost of higher degree routers and possibly more complex VLSI layout. Each router in a mesh topology has a degree of 5, except for the routers on the border. Tori can be classified as an enhanced form of mesh with wrap around links between border routers. These links use router ports that are not required to implement mesh

topology. The wrap-around links in tori reduce the average number of hops and provide better bisection bandwidth. The worst case number of hopes is $\sqrt{N}+1$ for tori and $2\sqrt{N}-1$ for mesh. In mesh, all communication links are short and equal in size. However, for tori wrap around-links are considerably longer and need special attention for timing closure.

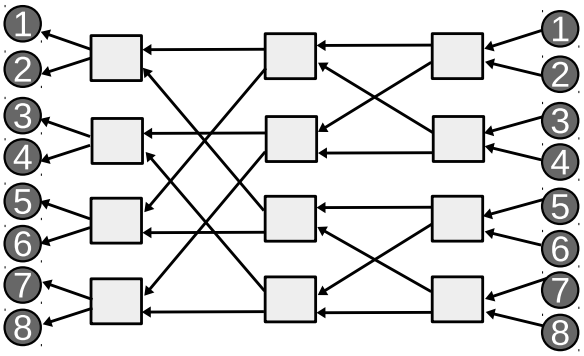


Figure 2.8: 2-ary 3-fly Butterfly Topology

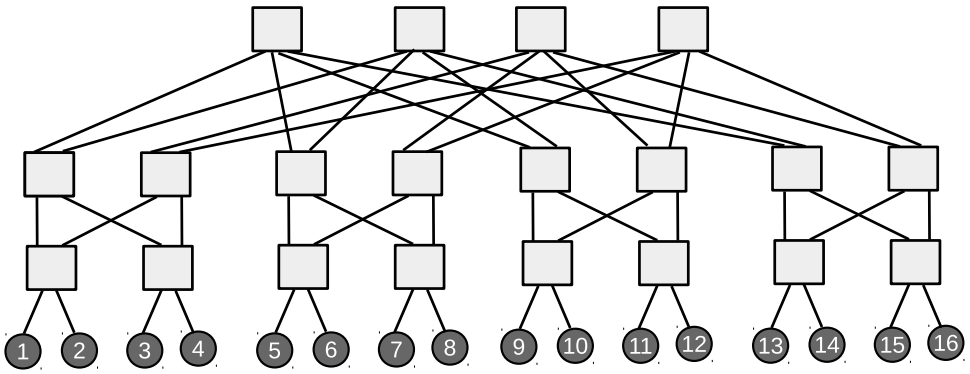


Figure 2.9: 3 Tier Fat Tree topology

Two classical examples of indirect networks, *Butterfly* and *Fat Tree* are shown in Figure 2.8 and Figure 2.9, respectively. The important feature of butterfly topology is that the hop distance between any source-destination node pair is fixed (3 in the topology shown in Figure 2.8). The router has degree 2 (2 input and 2 output ports), resulting in low cost routers. However, the number of routers is greater than the number of SoC components. The two main disadvantages of butterfly topologies are single communication path between a given source-destination pair resulting

in low bandwidth and low link fault tolerance, and more complex wire layout due to uneven links lengths. The Fat Tree topology provides higher bandwidth and excellent diversity of possible routing paths. However, these qualities come at a cost of silicon area (more routers) and complex wire layout.

In addition to these regular topologies, application specific MPSoCs are often designed on top of NoCs with customised topologies [91]. The data traffic patterns are often known at design time and therefore a communication graph can be extracted from the application specifications [92] [93]. The communication graph combined with knowledge of the physical mapping of SoC components can be used to create a topology that meets certain performance, energy or area constraints [94] [95] [91].

2.5.2 Routing

The routing algorithm defines the sequence of routers between soruce and destination nodes that a data packet will traverse. The quality of the routing algorithm is determined by the average packet latency and power consumption. A good routing algorithm evenly distributes the traffic across all the routers and maximise the saturation throughput of the network. Power can be optimised by keeping the routing circuit simple and keeping the number of hops travelled by data packets low [13].

Deterministic Routing is the simplest routing scheme and is widely used in NoCs. For a given source and destination pair, data packets always travel through a predefined set of routers. XY Dimension Ordered Routing (DOR) for mesh is a common example of deterministic routing. In DOR XY routing for mesh, depending on the physical location of the source and destination pair, the packet always travels first in the X (horizontal) direction and then in the Y (vertical) direction. However, deterministic routing can cause traffic hotspots in the case of an asymmetrical communication pattern between nodes [96] [13].

Oblivious Routing is superior to deterministic routing in terms of path selection. For a given source-destination pair, oblivious routing can select one of many possible

routes. However, this decision is taken without any knowledge of the network's current traffic condition. One example for oblivious routing is ROMM [97]. In the ROMM routing scheme for mesh, an intermediate node is selected at random on minimal paths between source and destination, and the data packet is first sent to the intermediate node and from there to the destination using a deterministic routing algorithm.

Adaptive Routing is the most robust routing scheme, as it uses global knowledge about the current network traffic state to select the optimal path [96]. Adaptive routing distributes the traffic node across different network routers and hence maximally utilise the network bandwidth. However, implementing adaptive routing increase the design complexity of the routers [13]. Moreover, there is always a limitation on how much global knowledge can be forwarded to each router, hence limiting the effectiveness of the routing scheme [96].

In addition to standard routing schemes, MPSoC designers often use application-specific routing schemes for NoCs [98] [99] [100]. Application communication graphs can be analysed to extract information about data volume and criticality. This information can be used to design routing algorithms that minimise the communication latency [100].

2.5.3 Flow Control

Flow control determines how data is transferred between different routers in a NoC. Specifically, flow control dictates the buffer and links allocation schemes. The design objective for flow control architecture is to minimise the buffer size and hence silicon area and power of routers, and to keep the network latency low. In packet-switched NoCs, a data message is broken into a predefined *packet* format. A network packet size can be further broken and serialised into multiple *flits*. The size of flit is normally equal to the physical channel width [13]. Additional information is added to each flit to indicate *header*, *body* and tail flit. The routing and other control information

can either be added only to the header flit or it can be added to each flit depending on implementation.

In *store-and-forward* flow control [101], before forwarding the packet to the next node, the router waits until the whole packet has been transmitted into its local buffer. This means that the input buffer must have enough space to store the whole packet, which can increase router area and power consumption. This scheme also increases the communication latency, as packets spend a long time at each node just waiting for buffering, although the output port might be free.

Virtual cut-through [102] improves on store-and-forward flow control by allowing a packet to be routed to the next router even before the whole packet arrives at the current router. However, the packet is only forwarded if the downstream router has enough buffer space to store the complete packet. This means that buffer size remains the same as in the case of store-and-forward flow control with improvement in per hop latency.

Wormhole routing [103] is a more robust scheme, as it allocates buffer space at the granularity of flit, opposed to the virtual cut-through and store-and-forward scheme which allocates buffers at the granularity of packet. As soon as flit of a packet arrives at an input port, it can be forwarded even if only one flit space is available in the input port of the next router (and output channel is not allocated). The wormhole flow control scheme results in low-area routers and it is therefore widely used in most on-chip networks [13]. The term *wormhole* implies that a single packet can span multiple routers at the same time. The main downside of this scheme is that the multiple links can be blocked at the same time in case the header flit of a multiple flit packet is blocked in one of the routers on the communication path.

2.5.4 Router Microarchitecture

The key building block of NoC is the router. The router's microarchitecture dictates the silicon area, power and, most importantly, the performance of the NoC. The maximum frequency at which a router can operate depends on the complexity of the logic used in the router microarchitecture, which in turn translates into higher level performance metrics such as network latency and maximum achievable bandwidth. The complexity of the router's microarchitecture depends on the network topology (degree), flow control method and routing algorithms. For example, a complex adaptive routing algorithm can be used to improve the worst case network bandwidth but it will result in increased area and power.

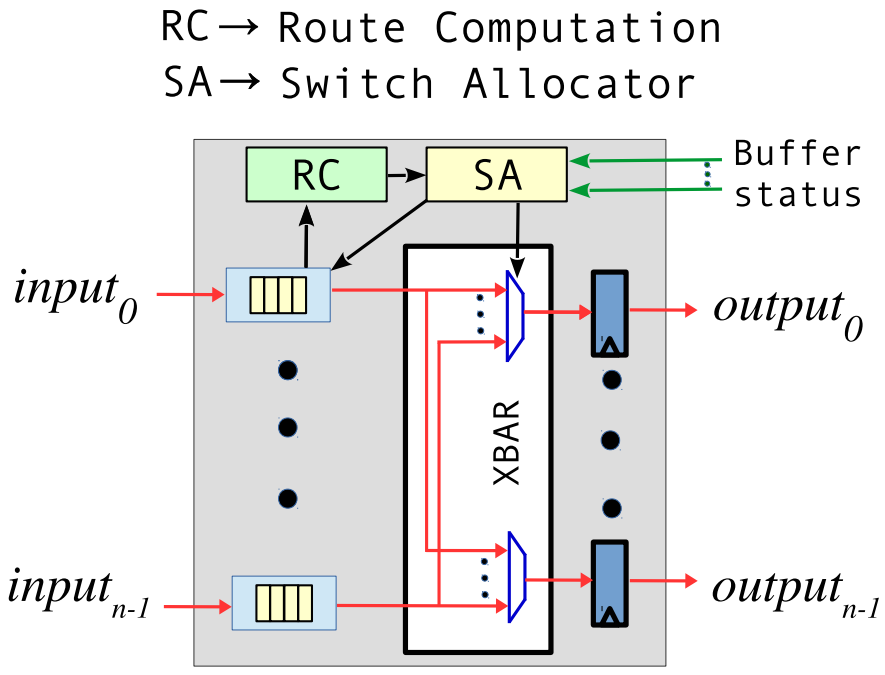


Figure 2.10: Wormhole Router Architecture

Figure 2.10 shows the overall architecture of a packet switch wormhole router [104]. The basic building blocks of a wormhole router are *input buffer*, *route computation*, *switch allocators* and *crossbar switch*. Input buffers store flits when they arrive in the router and keep them stored until they are forwarded to the next router.

The route computation unit calculates the output port for the head flit stored at the head of each input buffer, depending on the routing algorithm. The switch allocator arbitrates between different packets contending for the same output ports. The crossbar switch is logically a set of muxes that route flits from the input port to the output port. The data from the crossbar is stored in output buffers which can be as simple as flip-flops. The input buffers also propagate the buffer occupancy status to neighbouring routers to implement flow control [104].

2.5.4.1 Progress of a Packet in a Wormhole Router

The incoming flit is first stored in the input buffer (Buffer Write (BW) stage). If the flit at the front of the input buffer is the head flit, the route computation unit calculates the output port required to send the packet to its destination and asserts the port request signal to the switch allocator (Route Computation (RC) stage). In modern routers, the switch allocator consists of multiple port arbiters, one for each output port. Each output arbiter choose one of the multiple input requests for a given outport port. When an output port is allocated to an input port, the specific output port is locked until the whole packet (tail flit) has crossed the crossbar switch. Before doing anything further, the switch allocator checks if the downstream router has enough space to store the outgoing flit. If the buffers are full at the downstream routers, the switch allocator blocks the packet transmission. However, if buffer space is available, switch allocator sets the proper select lines for the crossbar switch and also instructs the input buffer to remove the flit at the front. This whole process is called the switch allocation (SA) stage. On getting a valid signal and output port selection from the switch allocator, the crossbar switch transfers the flit from the input port to the output port (Switch Traversal ST stage). The flit from the output port of the router then travels over wire links to get latched in the input buffer of the downstream router (Link Traversal (LT) stage). Note that the head flit of a packet goes through all stages discussed here. The body and tail flit, however, skip the RC and SA stages, as the output had already been reserved by the head flit.

2.5.4.2 Optimisation and Logic Synthesis of Routers

Executing all router stages in a single cycle can be achieved at a lower frequency because the cumulative logic delay of stages can be long. Single cycle operation might require a higher supply voltage depending on the target frequency which can increase the power consumption. Therefore, most of the high performance routers are often pipelined [104] [101]. However, increasing the number of pipeline stages increases the per-hop latency and hence the overall network latency. The number of pipeline stages also depends on the sophistication of the RC, SA and ST stages.

In commonly used pipelined routers, the LT and BW are done in one cycle, and RC, SA and ST are executed in the next cycle. However in the case of more complex router architectures, the second pipeline stage can be further divided into RC+SA and ST pipeline stages. The pipeline stages can affect the area and power consumption [105]. Using fewer pipeline stages results in more stringent latency constraints for logic synthesis and hence the synthesis tool has to insert larger gates with lower delays. Larger logic gates have higher dynamic and leakage power. Pipeline stages on the other hand can reduce the gate sizes; however the overall area may increase due to addition of the pipeline flip-flops [105]. Therefore, the logic synthesis of routers is a classic power-performance-area tradeoff problem [104] [105] [106].

2.6 Overview of Recent Academic and Commercial NoCs

The most interesting NoC design currently available is the Tilera iMesh on-chip interconnect network [107]. The iMesh interconnect architecture has been used in the commercial TilePro64 multicore chip. The latest 72 core Tilera GX chip [67] also uses the same NoC design. The proposed Network on Chip (NoC) architecture is different from other academic and commercially available on-chip architectures in terms of the number of physical NoC. iMesh provides five different physical NoC

channels: two of these networks are used for memory access and management tasks while the rest are user accessible. The motivation is that future integrated circuits will have enough silicon resources to integrate more than one NoC per chip.

The next generation SPARC M7 processor combines three different NoC topologies for the on-chip interconnect [108], ring based request network, point-to-point network for response messages, and mesh for data packets. The M7 processor uses a shared-distributed cache model. The request ring network is used to broadcast the cache requests to all cores in the system. The point-to-point network is only used when adjacent cores share data. The mesh data network is built using 10 port routers and is used primarily to stream data from memory controllers to local cache.

Anton2 is a special-purpose supercomputer for performing molecular dynamic simulations. The building block of the supercomputer is Anton ASIC which contain a number of special purpose hardware units for numerical calculations [4]. Anton2 ASIC uses a 4×4 2D mesh for connecting on-chip components, whereas the chips are connected with each other using a 3D Torus. The novel feature of the communication architecture is that the same set of routers are used for both intra and inter-chip communication. About 10% of the total ASIC area is dedicated to on-chip communication.

SMART (Single-cycle Multi-hop Asynchronous Repeated Traversal) NoC from MIT is another interesting low latency on-chip communication architecture [109]. Authors observed that a data bit can travel 9-11 hops in a single clock cycle at 1 GHz for 45nm silicon technology. Based on this observation they propose a NoC architecture where data can be transferred across physically distant mesh nodes in a single cycle by bypassing multiple routers. This reduces the latency of multi-hop mesh NoC. The proposed architecture is reported to improve the performance of PARSEC and SPLASH-2 benchmarks by an average of 26% for private L2 cache and 49% for shared L2 cache architectures.

Intel used a mesh-based NoC for an 80-core TeraFlop experimental chip [31]. The mesh NoC uses five stage pipelined routers designed for 5 GHz frequency. This

results in $1ns$ per-hop latency. According to experiments conducted on the research chip, the NoC consumed about 28% of total chip power although it consumed 17% of total chip area.

Intel has also introduced a 48-core mesh NoC based multicore chip called Single Chip Cloud Computer (SCC) [110]. The target frequency for the NoC was set at $2GHz$ with $1.1V$ supply voltage. The router is four stage pipelined and uses virtual cut-through flow control. To mitigate the problem with higher NoC power from the previous 80-core chip, Intel opted for a DVFS scheme for NoC. The NoC was organised as a 6×6 mesh so that two compute cores share a single router. These techniques helped to reduced the share of NoC power to 10%.

There are two well known commercial NoC IP providers, Sonics [111] and Arteris [112]. Both Sonics and Arteris provide various predesigned NoC IPs for commonly used processor IPs such as ARM and Xtensa. Furthermore, both of them provide design tools that can be used to optimize the NoC IP for various design objectives such as power, area, performance, Quality-of-Service (QoS) and reliability. It is anticipated that hardware developers will be using 3rd party NoCs to reduce both design cost and development time [91].

2.7 NoC Power Optimisation

As more cores are integrated on a chip, the on-chip interconnect's complexity increases and so does its power. Computer architects always want to keep the on-chip interconnect's power low so that processing elements and memory hierarchy can use a larger share of power [6]. Although NoC provides the scalable communication for multicore chips, it can consume valuable power. For example Daya et al. [8] report that NoC in their 36 core SCORPIO experimental chip consumes 19% of the total chip power. Similarly NoC consumes 28% of total power in Intel's TeraFlop chip [31]. With limited power budgets constrainting multicore scaling [5], employing power saving techniques for NoCs is an active research topic.

Over the years, various voltage-scaling based solutions have been investigated for reducing NoC dynamic power. Shang et al. [113] presented the pioneering work on DVFS for on-chip links. The scheme uses usage history of links to predict their future utilisation. Based on this prediction, a certain voltage and frequency (VF) level is selected for each link. Bogdan et al. [114] proposed a DVFS scheme for spatially and temporally varying workloads. The proposed DVFS scheme worked at the granularity of individual routers. Based on fractal modelling of workloads, the scheme selected an appropriate VF level for each router. Mishra et al. [115] proposed per router frequency tuning in response to changes in network workload to manage power and performance. Based on the optimisation goal, the frequency of the routers is increased to relieve network congestions and improve performance, or the frequency is decreased to meet certain network power constraints. Ogras et al. [116] described a framework for fine grain VF selection for VF island based NoC. The scheme first statically allocates VF level to each NoC router and then uses the runtime network information for fine tuning the assigned VF level. All these works base their DVFS schemes on network metrics such as packet latency, injection rate, queue occupancy and network throughput. Researchers [117] [118] argue that by neglecting higher-level performance metrics such as application execution time, these schemes can result in non-optimal results. Therefore, work by Chen et al. [117] and Hesse and Enright [117] based their DVFS schemes on the actual execution delay faced by applications due to DVFS for NoCs. Zhan et al. [119] explored the problem of per-router DVFS schemes for streaming applications. They developed an analytical model to statically predict the effect of VF scaling on application throughput. Depending on application to core mapping, routers on the critical communication path are operated at lower frequency.

Since the ratio of leakage power to total chip power is increasing with transistor scaling, researchers have been exploring leakage power techniques. Through experiments with realistic benchmarks, Chen [120] reported that the router's static power share is 17.9% at 65nm, 35.4% at 45nm, and 47.7% at 32nm. As discussed earlier

in this chapter, power gating is often used to save the static power of on-chip components [39]. Soteriou and Peh [121] proposed power gating at the link level. They used the channel utilisation history to select links that can be switched off with minimal impact on performance. As switching off some links results in irregular topologies, they proposed an adaptive routing algorithm to avoid the switched off links. Matsutani et al. [122] proposed adding low frequency virtual channel buffers that can be power gated at runtime to save leakage power. An ultra fine grain power gating for NoC routers is proposed in [123]. They used special cell libraries where each CMOS gate has a power gate signal. Therefore, the power for each router component such as input buffer, switch allocator and crossbar, can be individually turned on or off, based on the network activity. This helped to save leakage power by 78.9%. However, the main drawback of this technique is the complexity of including special power gate circuitry in each CMOS gate. The additional power circuitry also increases the router area. Kim et al. [124] proposed a fine grain buffer management scheme for NoC routers. In the scheme, depending on network traffic load, the size of the input buffers was increased or decreased at runtime. The unused buffer slots were power gated to save leakage power. However, this scheme only targeted buffer leakage power and neglected other router components. Parikh et al. [125] proposed performing power gating at the granularity of the *datapath segment* which consists of an input buffer, crossbar mux and output link. In the Router Parking [126] scheme, routers for cores that are not under use are switched-off and the traffic is routed around the switched off routers through new routing paths that are calculated at runtime.

The main problem with the previous proposals for power gating is the performance overhead due to wakeup latency. Chen and Pinkston [127] proposed overlaying a standard mesh NoC with a low cost ring network. If a packet arrives at a router and the router is switched off, the packet is routed through the ring network which is always switched on. Das et al. [128] proposed to divide wider mesh NoCs into four smaller NoCs for efficiency. Additional NoCs are only switched on if the

network load exceeds a certain limit and the unused NoC planes remain switched off. In this network connectivity is ensured.

As buffers in routers consume a considerable share of dynamic and leakage power, researchers have also explored using bufferless NoCs. The first effort in this direction was the BLESS router [129]. The idea is that instead of storing the flit in router, the flit is routed even in a direction that does not result in a minimal path. This means that even with some misrouting, the flit will eventually reach its direction. Therefore this scheme is applicable for routers with the number of output ports equal or higher than the number of input ports. The deadlock and livelock condition is avoided by allocating a channel to the oldest flit first. To improve some of the shortcomings of BLESS router, CHIPER [130] was introduced. Both BLESS and CHIPPER show potential in reducing network power. However there are two concerns with bufferless NoCs. First, the deflection routing causes packets to take non-minimal routes resulting in longer packet delays. The effect of network delays is more significant for memory-intensive applications. Secondly, the bufferless routing result in out of order arrival of the flits which increases the complexity of the network interface.

2.8 Dark Silicon Aware Multicore Design

Dark Silicon is seen as an inevitable threat to Moore's Law, as ever increasing number of transistors cannot be switched on, due to power and thermal budget constraints [5, 19]. Interestingly, researchers have been approaching the dark silicon problem from different angles. One perspective is that dark silicon can be used to improve the energy efficiency of existing computing architectures [24] [18] and other system metrics such as reliability [29]. Another perspective is that improved thermal and power management schemes for multicore architectures can be used to mitigate the effects of dark silicon to some extent [19, 26, 131]. We will now discuss both perspectives in more detail.

Researchers believe that dark silicon can be used to design heterogeneous computing system with high energy effeciency. Esmaeilzadeh [18] brought the issue of dark silicon into the limelight in his PhD thesis. He used a combination of analytical and experiment data to predict that approximately 50% of transistors in 8nm chips will remain dark due to a combination of a limited power budget and limited application parallelism [5]. He based his research on the argument that traditional computing systems are highly energy-inefficient for emerging applications such as image processing, sensor data fusion and other similar signal processing applications. To mitigate the energy inefficiency of general-purpose cores, Esmaeilzadeh et al. [30] proposed integrating application-specific *neural processing units (NPUs)* for signal processing applications. The NPUs work on the principal of *approximate computing* by exchanging the energy efficiency for output error. The NPU are tightly coupled with the execution stage of the standard processor. Compared to general purpose x64 architecture, NPU accelerators are reported to be 2.3× faster and use 2.3× less energy at a cost of 9.3% loss in quality. Venkatesh et al. [23] proposed exchanging dark silicon for energy efficiency by using *quasi-specific (QS)* cores. The idea behind QS core architecture is to identify the code hotspots in commonly used software libraries and design application specific cores (QS cores) and integrate them with general-purpose cores. At runtime, whenever the general purpose cores come across a piece of code that be run on a QS core, the general purpose core is put in the low energy mode and the QS core is activated. The reported results show that QS cores can be up to 18.4× more energy efficient than general purpose cores. Similar design philosophy has been used behind GreenDroid [22] to extend the idea to well known Android operating system libraries. Cong et al. [132] discuss various architectural issues regarding the integration of such application specific accelerators with general purpose cores.

Allred et al. [25] proposed designing multicores with architecturally homogeneous cores that have been designed for different VF levels. They show that running cores that have designed cores for specific VF levels is more energy efficient than applying

DVFS on cores that have been designed for higher VF levels, due to optimized gate sizing at the time of circuit synthesis. Turakhia et al. [133] proposed a design flow, *HaDeS*, to design a general-purpose heterogeneous CMP for a given set of multithreaded applications under a given dark silicon budget. They start with a library of cores with different architectural parameters, such as L1 and L2 cache size, multi-issue dispatch width, and re-order buffer size. The proposed solution then designs a CMP containing some heterogeneous cores from a library such that the average runtime of all applications is minimised while abiding by certain power and area constraints.

Recently researchers have investigated exploiting dark silicon for system reliability. Kriebel et al. [29] departed from the general trend of exchanging dark silicon for energy efficiency by using dark silicon to design reliability heterogeneous CMPs. At design time, cores with varying protection against soft and hard errors are integrated into systems. At runtime, depending on the application's susceptibility to soft errors and power budget, applications are mapped to certain cores and unused cores are switched off. Gnad et al. [134] proposed runtime management of ageing induced variability of dark silicon through intelligent application to core mapping.

Researchers have proposed techniques to mitigate the amount of dark silicon by improving system-level thermal and power management schemes [135]. Muthukaruppan [136] has proposed hierarchical power management for dark silicon inspired heterogeneous asymmetric multicores, specially ARM's Big.LITTLE architecture. The proposed framework is based on multi-agent control theory and consists of three different types of controller: per-task resource share controller, per-cluster DVFS controller, and per-task QoS controller. If the chip temperature exceeds a given thermal constraint, the framework uses DVFS to selectively throttle some of the tasks in a way that QoS is degraded gracefully. They reported better performance than the standard Linux task scheduler. Pagani et al. [137] proposed a framework to efficiently manage the switched-on cores in tiled CMP to maximise performance while abiding by thermal design power (TDP) constraints. They observed that the

ambient temperature of a core depends on the power dissipation of the core itself and the power dissipation of other active cores physically nearby. They proposed a polynomial time algorithm for calculating the thermal safe power (TSP) for a given application to core mapping. TSP is the power budget for each core that ensure that overall chip temperature will remain below a given TDP constraint. They reported a significant reduction in the amount of dark silicon and improvement in system performance compared to budgeting the TDP at chip level. Khdr et al. [138] also proposed a resource management scheme to maximise the performance of TDP constraint multicore. The proposed framework selects the optimum number of cores, the location of those cores within the chip, and the VF level for the cores under given TDP constraints. In addition to static core allocation, the proposed scheme also dynamically adapts to changing TDP or power budgets at runtime.

Another interesting idea is to convert *dark silicon* into *dim silicon*. The term *dim silicon* implies that instead of turning on a few cores at high frequency and turning off the other cores (dark silicon), we can turn on all cores at relatively lower frequency without exceeding the thermal or power budget [24, 139]. Such schemes are shown to be useful for scalable multithreaded applications as the net performance using few high frequency cores is similar to using a higher number of lower frequency cores [26]. *Near threshold computing (NTC)* is based on principals of integrating a large number of cores such that power supply is close to the CMOS transistor threshold voltage. Lowering the chip supply voltage close to threshold voltage significantly decreases the maximum operating frequency but also decreases the energy consumption at the same time. The basic idea of NTC is to find the minima of energy-delay product (EDP) of such multicore architecture for a given multithreaded application [26, 57].

It is clear from our discussion that all the prior research concentrates on the interaction of dark silicon and computational resources, completely neglecting the role of on-chip communication architecture. A recent study shows that NoC will consume 18∼24% of total power in 8nm dark silicon chips [1], yet there is no prior

research on using dark silicon to improve NoC energy efficiency. This thesis fills this gap by proposing different techniques to improve energy efficiency, reliability and performance of NoCs by exploiting abundant transistors.

2.9 Communication-Aware Mapping

While designing an MPSoC based system, it is not uncommon to have fixed specifications for the underlying hardware. Therefore, in such cases, it is the responsibility of the system programmer to use the available hardware resources efficiently. The situation can be more complicated when we have a heterogeneous MPSoC architecture. Over the years, many techniques have been proposed for effectively mapping a given application on a target MPSoC architecture.

Embedded system designers are expected to tweak the application mapping to maximize the performance, while meeting the hard time-to-market limits [140] [141]. There is no single technique to map applications on a given MPSoC. The mapping problem is further complicated due to presence of many different MPSoC architectures. Therefore, the common practice is to rebuild the mapping strategy for every application-architecture combination.

Application mapping was an area of interest for researchers working in parallel computing and supercomputing [142]. Their idea was to map the applications that share the data as close as possible to reduce the network latency. This naive idea is also applicable in the case of chip multiprocessors (CMPs) where mapping the application has to take into account the underlying on-chip interconnect architecture. Similarly, the choice of shared memory architecture and message passing interface heavily influence the mapping algorithms.

The mapping solutions proposed in the literature can be broadly divided into two categories 1) Static Mapping, and, 2) Dynamic Runtime Mapping. Each class of mapping algorithm has its advantages and disadvantages. Static mapping is useful when applications to be executed are known at design time [143,144]. System

designers can formulate an optimal or near optimal solution for such scenarios. On the other hand, runtime mapping algorithms allocate resources to application on the fly. These algorithms give a better mapping solution in cases where the application behaviour is unknown at design time [145] [146].

In the case of embedded systems applications to be executed are known at design time. Therefore, static mapping techniques result in better system performance [147]. Another interesting area of research is studying the mapping problem for NoC-based architectures. The unique data communication capabilities of NoC-based architectures demand a smart communication-aware mapping strategy [99,148–150]. It is important to analyse the network contention while estimating the runtime of a given application. Unfortunately, most of the mapping problem proves to be a NP-complete problem with only near-optimal solution. However, heuristic-based algorithms have also been proposed, which reduce the time required to solve these NP-complete problems. Time division multiplexed access (TDMA) based NoCs are used for predictable systems. The expected traffic for a given link can be estimated using analytical modelling or simulations. Through proposed algorithms, it is then possible to map a given processor-processor communication in an allotted time slot. This not only improves the network congestion, but also makes the system predictable [151].

Some researchers have provided solutions where the mapping and partitioning problem of NoC-based embedded systems are dealt with as a combined problem. Although these solutions require a very strict analytical model for application, the final software and hardware is highly optimised in area footprint, power and performance [149,152]. It has been observed that a generalised MPSoC architecture might eventually fail to cover the design requirements of a particular real time embedded system. Therefore, the option of highly customised hardware for a particular set of applications has also been widely explored.

2.10 Application Specific Communication Architecture for MPSoCs

Over the years, researchers have analysed that flat communication and memory architectures are not sufficient for designing high performance application specific MPSoC architectures. Moreover, the quest for low-power, low-cost embedded system has forced designers to optimize the system. Therefore, it is anticipated that highly optimised hardware and software designs will include customised memory and on-chip network designs in the future [141].

Customizing the communication architectures at system level and gate level not only improves the system performance, it also results in energy efficient design. By eliminating the logic overhead, the leakage and static power loss of the system can be significantly reduced [86]. It has been shown that some MPSoC applications actually do not need highly complex network architectures like NoC, and simple bus-based architecture can essentially meet the performance requirements. Therefore, it is important to avoid over-kill by intelligently designing communication architectures that give optimal solutions without incurring area overhead [153].

There have been several studies on designing application specific on-chip communication architectures for bus, crossbar and NoC based embedded MPSoC. Some classic studies for highly optimised bus-based system synthesise have been presented in [76]. Further study on the same topic was carried out by Daveau [77]. At the time of research presented in [76,77], the amount of logic resources that were present on the chip was limited and the main objective was to maximise the utilisation of these resources. Another interesting area of interest has been to form combined design space exploration techniques for both processor and communication optimisation [154]. Moreover, recent advancements in fabrication technology mean that we have plenty of wires and logic resources at our disposal [5]. Therefore, design for low power has been the key objective in recent research.

The selection of optimal on-chip communication architectures is non-trivial.

Therefore, an effort has been made to introduce library-oriented design space exploration where the designer has several configurations to choose from. Designers can then use analytical modelling or simulations to select the best communication architectures [155]. Similarly, the option to integrate different communication architectures in a single design has also been studied extensively. Murali et al. [86] proposed a complete design method for application-specific crossbar synthesis. They use the traffic pattern of the application to design a communication architecture that is a combination of crossbars and bus. Bus-based architectures are simple but offer less performance. The crossbar based architectures are more complex but provide better throughput. Therefore, Murali et al. combined the two communication architectures while keeping the gate cost low and fulfilling the performance requirements.

The idea of application specific communication architectures is more interesting in the case of NoC. Inherently, NoC architectures are highly customisable and designers can tweak a variety of parameters of the architecture to meet performance, cost and energy constraints. X-pipes is perhaps the first project that provided a library-based approach to NoC design [156]. The same idea has also been explored by Jeremy & Parameswaran [157]. Another interesting project that was targeted towards low-power NoC based embedded MPSoC was Aethereal Network on Chip [158]. The significance of these projects is the flexibility provided to the designer to quickly explore the various possible optimisations for a given application. These optimisations can provide up to twelve times improvement in performance while reducing the area cost by 25 times when compared with flat communication architectures [153].

All the techniques referred to above improve the performance of on-chip communication in terms of aggregate on-chip bandwidth and average data latency, which are not suitable metrics for streaming applications executing on MPSoCs. In the case of streaming MPSoC, application-level throughput or latency of the system is the most important performance metric [159]. Thus, it is important to incorporate system level performance constraints in the design flow for an application-specific

on-chip network [92].

Chapter 3

darkNoC: Energy Efficient NoC for Dark Silicon Chips

3.1 Introduction

Network-on-Chips (NoCs) provide a scalable communication fabric for connecting large number of resources within a chip. NoC can contribute significantly to the total chip power, e.g., up to 18% and 33% in Intel SCC [51] and RAW [160] architectures, respectively. To address this issue, various power saving techniques for NoC at system-, architecture- and circuit-level have emerged, among which Dynamic Voltage and Frequency Scaling (DVFS) is very prominent [114, 115]. System-level DVFS managers apply reduced voltage and frequency (VF) levels in a NoC to save power. According to several recent studies [17, 27], the energy saving potential of DVFS has been diminishing for two reasons:

- With shrinking node size, the difference between nominal voltage and the threshold voltage is decreasing. As the nominal V approaches threshold voltage, the transistor delay increases exponentially, resulting in huge performance penalties [27]. This limits the factor by which V can be scaled down. Transistors with lower threshold voltages can be used in the circuit to increase the gap

between the nominal and threshold voltages, but that results in an increased leakage power.

- DVFS does not influence the intrinsic physical properties of the circuit such as gate sizes, capacitances, etc. which are fixed at design time.

With the help of the following NoC synthesis case study, we show how the problem of diminishing returns of DVFS can be alleviated.

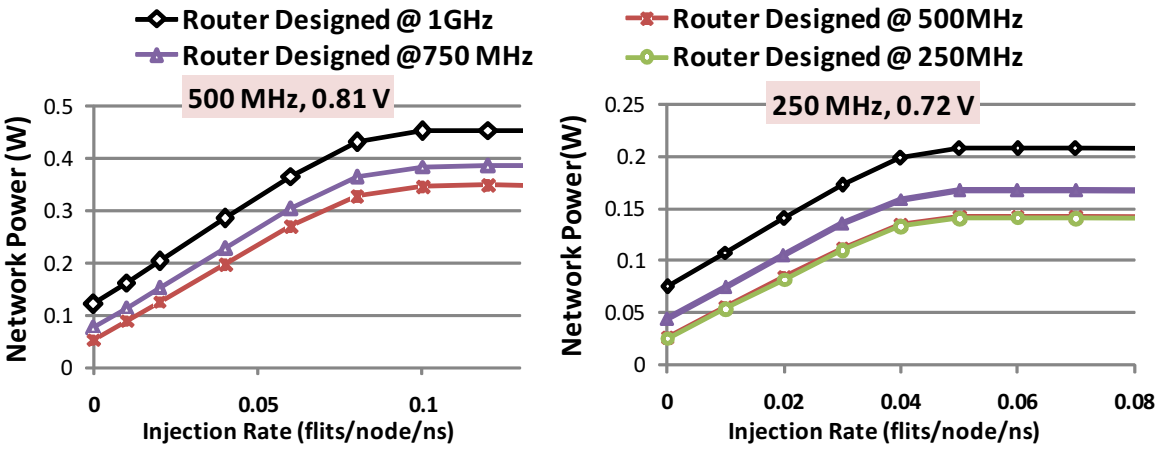


Figure 3.1: NoC Power for Transpose Traffic for a)(left)[500 MHz, 0.81V] VF level, b)(right) [250 MHz, 0.72V] VF level

Motivational Example: We exploited the multi-Vt circuit optimisation available in CAD tools to synthesis architecturally identical NoC routers for a set of target VF levels: [1GHz, 0.9V], [750 MHz, 0.81V], [500 MHz, 0.81V] and [250 MHz, 0.72V]. Figure 3.1 reports the network power for operation at [500 MHz, 0.81V] and [250 MHz, 0.72V] (details of our experiments are presented in Section 3.4). We can observe that for operation at [500 MHz, 0.81V], the NoC designed particularly for [500 MHz, 0.81V] VF level is on average 35% and 16% more power efficient than applying DVFS on a NoC designed for [1GHz, 0.9V] and [750 MHz, 0.81V], respectively. Similarly, for operation at [250 MHz, 0.72V], the NoC designed particularly for [250 MHz, 0.72V] VF level is on average 42% and 22.7% more power efficient

than DVFS applied to NoC designed for [1GHz, 0.9V] and [750 MHz, 0.81V], respectively. This shows that, unlike traditional NoC with a single layer of routers, it may be beneficial in terms of power to have multiple layers of routers in a NoC such that each layer is optimised for a particular VF level.

The provision of multiple layers of routers in a NoC may cost a significant amount of silicon; however, we can leverage *dark silicon* [5] [161] to alleviate this overhead. As discussed in Chapter 2, several researchers have proposed to leverage dark silicon to provide extra application-specific accelerators [22], extra low-power processors [133], etc. for better energy efficiency. In fact, the authors of [24] state that *"in eight years, we will be faced with designs that are 93.75% dark!"*, and, *"we will spend exponentially increasing amounts of silicon area to buy energy efficiency"*. Therefore, even with the addition of extra accelerators [22] and processors [133], *we believe that there will still be dark silicon available which could be exploited for the energy-efficient design of other system components such as NoC.* Given the availability of dark silicon, three research problems need to be addressed for the realization of a NoC with multiple network layers:

- What kind of architectural support is required to integrate multiple VF-optimized network layers in a NoC?
- How many network layers (and their VF levels) should be used for a set of applications under a dark silicon budget?
- How to enhance system-level DVFS managers to exploit the provision of multiple VF-optimized network layers?

In this chapter, we focus on the architectural details of our darkNoC architecture and study the interplay of application characteristics, architecture and dark silicon budget. We discuss the fine-grain details of a NoC architecture, where architecturally homogenous routers which have been optimised specifically for particular VF ranges, are seamlessly integrated at the architecture-level to complement system-level DVFS

managers, and exchange dark silicon for energy efficiency in NoC. In particular, our key contributions are:

- We propose a novel NoC architecture, named *darkNoC*, where multiple network layers consisting of architecturally identical routers, but optimised to operate in different voltage and frequency ranges during synthesis are used. Only one network layer is illuminated (active) at a given time while the rest of the network layers are dark (deactivated).

- We propose an efficient hardware-based mechanism to transition from one network layer to another. Our network layer switch-over mechanism preserves the lossless communication property of a packet-switched buffered NoC, and is transparent to the software. Further, we investigate and report the factors that can potentially affect the time and energy overhead of the switch-over mechanism.

- Finally, to illustrate the potential of our darkNoC architecture, we show how a state-of-the-art DVFS manager [118] is deployed. Further, we compare our darkNoC architecture having a different number of network layers (due to dark silicon budget) with a traditional single network layer NoC, where both the architectures have the same DVFS manager (our results show savings of up to 56% in energy-delay product). We also discuss how this energy saving is correlated with application characteristics.

3.2 Related Work

Various system-level DVFS techniques have been proposed for NoC power management to strike a balance between performance and power, such as at the granularity of communication links [113], routers [114, 115] and regions [118]. Run-time power gating has also been proposed to reduce leakage power of NoCs. Matsutani et al. [123] proposed an ultra fine-grained power gating technique for routers. Similar

techniques at other granularities have been explored in [124, 126, 128]. All of these techniques are orthogonal to our work and can be tweaked to exploit the provision of multiple network layers in a NoC. In the experimental results in this chapter, we use the DVFS technique of [118] as the state-of-the-art to compare our darkNoC architecture with the traditional single network layer NoC.

Pullini et al. [162] investigated the advantages of using multi-Vt circuit optimisation in terms of performance and power for a single layer NoC architecture. In contrast, we exploit the multi-Vt circuit optimisation in the context of multiple network layers and DVFS. Various NoCs with multiple network layers have been proposed in [11, 107, 163]. However, all of these works customised the architecture of the network layers or the routers, in contrast to our work where we integrate architecturally homogenous yet VF-optimised network layers. Our work is inspired by [164,165], where the authors exploited multi-Vt circuit optimisation for designing processors in the context of dark silicon. However, one cannot infer from their work that multi-Vt circuit optimisation will be beneficial for NoCs, and what should be the architecture of a NoC with multiple VF-optimised network layers.

In summary, to the best of our knowledge, this is the first work that proposes a NoC architecture which exploits multi-Vt circuit optimisation for multiple network layers in the context of system-level DVFS and dark silicon.

3.3 darkNoC Architecture

We consider NoCs with predefined topologies as shown in Figure 3.2(a). The NoC consists of n routers, which are divided into regions. Each region has m routers, and separate voltage and frequency (VF) rails with their own VF regulators. The value of m can vary from 1 to n, depending on the granularity of the regions. For communication between VF regions, bi-synchronous links [166] must be used. However, in this work we assume that VF regions can only communicate if they are operating at the same VF level. A designer can opt to use the darkNoC architecture

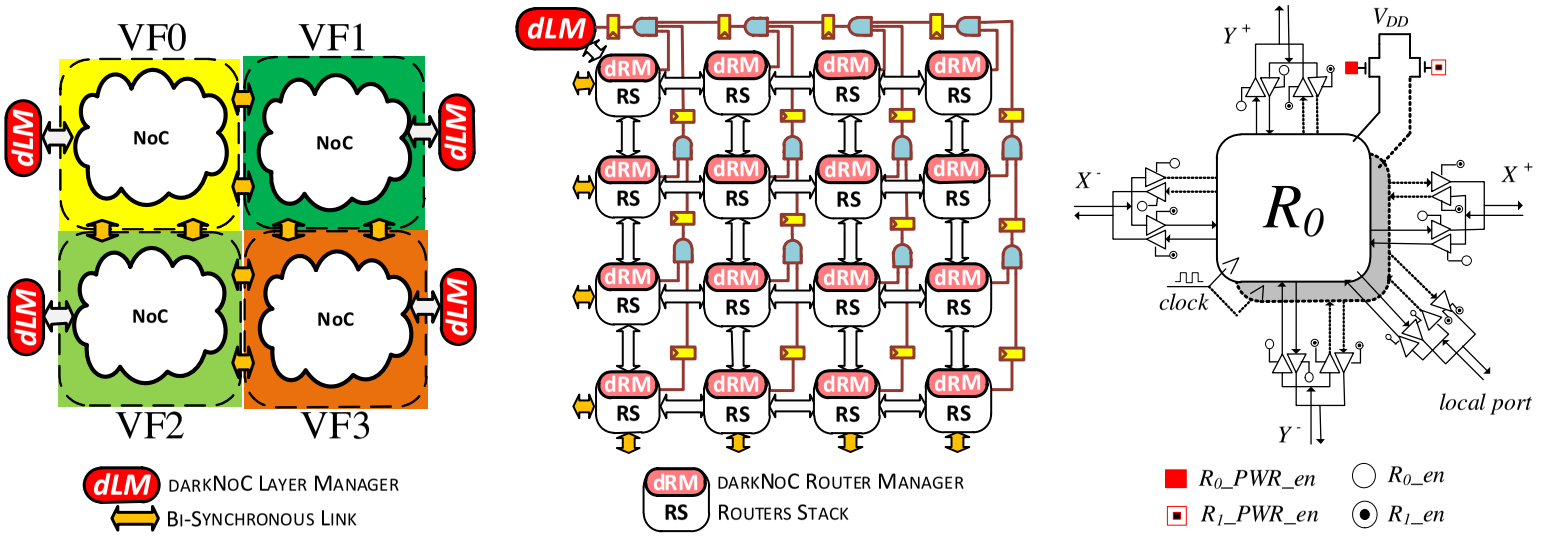


Figure 3.2: darkNoC architecture: **(a) (left)** NoC divided into VF regions **(b) (middle)** a single VF region **(c) (right)** a stack of (two) VF optimized routers.

in any of the VF regions based upon their requirements. The following paragraphs explain the darkNoC architecture from the perspective of a single VF region.

3.3.1 darkNoC Layers

A region contains l network layers, where a network layer consists of m routers as shown in Figure 3.2(b). 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 any given time, only m routers are active. Figure 3.3 shows an example of transitioning from network layer 0 to 3 in a region with 4 network layers.

Each network layer is optimised at design-time to operate in a certain VF range. That is, multi-Vt circuit optimisation 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*) as shown in Figure 3.2(b). The function of the *dLM* is to switch between network layers when directed by the system-level DVFS manager.

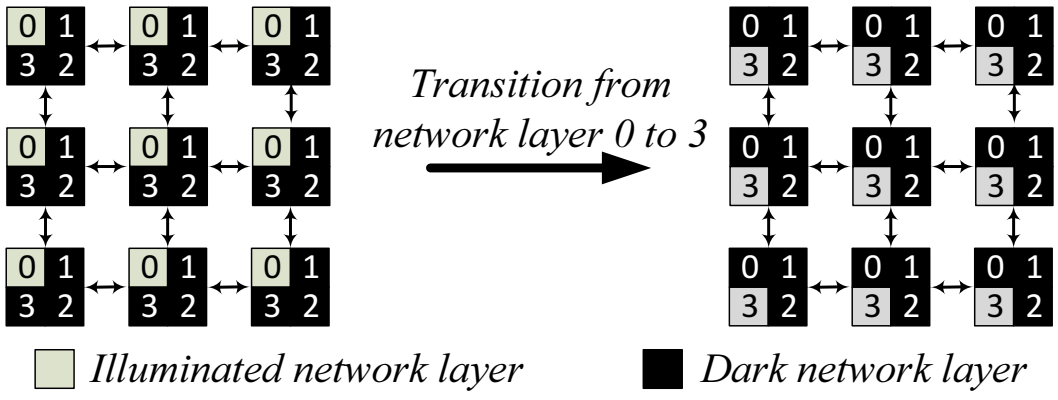


Figure 3.3: An example darkNoC architecture with four network layers.

3.3.2 darkNoC Routers Stack

Each node in Figure 3.2(b) represents a stack of l routers to realise l network layers of the region. For example, for the 4 network layers in Figure 3.3, each router stack has 4 routers and all the routers marked 2 belong to the network layer 2. As shown in Figure 3.2(b), all the routers in a stack share the same set of link wires with the neighbouring stacks of routers, *which reduces verification costs.*

Each router in a stack has separate controls for power-gating and enabling input/output ports. For example, a stack of two routers is shown in Figure 3.2(c), where $R_0\_PWR\_en$ and R_0_en controls are used to power-on or off and enable or disable input/output ports of router 0. Thus, a router in a stack is illuminated in two steps: powering it on and enabling its input/output ports. The local port of a router stack is connected to a tile (which may consist of a processor, memory, etc.) through a network interface (NI). The NI has the capability to stop injection of packets from its tile, which we exploit during the switching of network layers. To manage a router stack, we use a hardware-based darkRouter Manager (dRM) as shown in Figure 3.2(b). The function of dRM is to switch between routers when directed by the dLM.

3.3.3 darkNoC Layer Switch-Over Mechanism

The switch-over between network layers is an important design requirement for our darkNoC architecture. The main challenges are: a) the lossless data communication property of packet-switched buffered NoC should be preserved, b) the switch-over mechanism should be transparent to software, and c) the switch-over mechanism should be efficient in terms of time and energy overhead. In our solution, the darkNoC Layer Manager (dLM) and the darkNoC Router Managers ($dRMs$) autonomously coordinate with each other to realise a switch-over mechanism with the aforementioned requirements.

At a high level, the dLM ensures correct switch-over between two network layers

Algorithm 3.1: dLM Function

```
1  nodes = m; // number of routers in a region
2  WaitForSwitchOverCommand();
3  for i=nodes to 0 do
4      SendDisableInjControlFlitTodRM(i);
5  WaitForNetworkFlush();
6  for i=nodes to 0 do
7      SendSwitchRouterControlFlitTodRM(i);
8  WaitForNetworkFlush();
9  for i=nodes to 0 do
10     SendEnableInjControlFlitTodRM(i);
```

Algorithm 3.2: dRM Function

```
   // Phase 1
1  WaitForDisableInjControlFlitFromdLM();
2  SwitchOnTargetRouter();
3  DisableInjectionFromNI();
4  if RouterLocatedAtRegionBorder then
5      SetStopFlagForNeighbourRouters();
6  WaitForRouterFlush();
7  SetEmptyFlag();
   // Phase 2
8  WaitForSwitchRouterControlFlitFromdLM();
9  DisableActiveRouterPorts();
10 EnableTargetRouterPorts();
11 PowerOffActiveRouter();
12 SetEmptyFlag();
   // Phase 3
13 WaitForEnableInjControlFlitFromdLM();
14 EnableInjectionFromNI();
15 if RouterLocatedAtRegionBorder then
16     ResetStopFlagForNeighbourRouters();
```

in the region. The *dLM* uses the injection channel of the NI of the router stack it is attached to, and sends different types of control flits to *dRMs* through the NoC channels. Further, the *dLM* gathers the buffer occupancy information from the routers stacks using a single bit AND-gate network as shown in Figure 3.2(b).

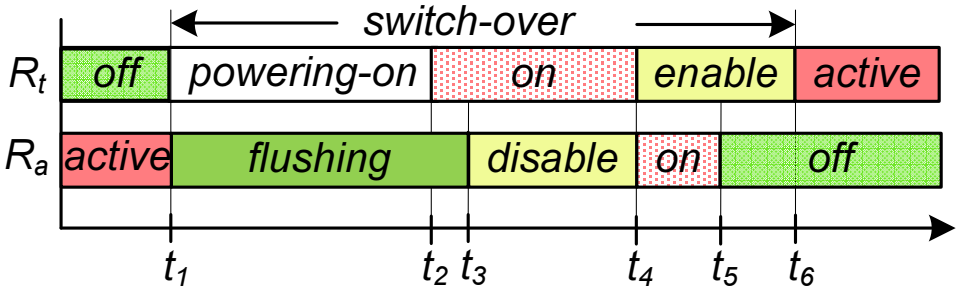


Figure 3.4: Timing diagram for switch-over from router R_a to R_t.

Similar AND-gate networks have been used in NoC for congestion detection [96]. The *dRM* of a router stack is attached to the ejection port of the NI, and receives control flits from the *dLM*. Based upon the control flits from the *dLM*, a *dRM* can: (a) power-on or off a router, (b) enable or disable input/output ports of a router, or (c) enable or disable injection of packets into the NI. Further, a *dRM* generates a single bit flag by analysing the buffer occupancies of its router to indicate the presence or absence of in-transit packets, which is propagated through the AND-gate network to the *dLM*.

Details of dLM and dRM: Algorithms 3.1 and 3.2 report the functionalities of *dLM* and *dRM*, respectively, during switch-over of a network layer. Figure 3.4 illustrates the switch-over mechanism for a router stack (other stacks similarly switch-over their routers in parallel). On detecting a new switch-over command (from a system-level DVFS manager), the *dLM* starts sending control flits to all the *dRMs* to initiate the switch-over mechanism (Algorithm 3.1, lines 2–4). A *dRM* enters into its first phase on receiving the control flits from the *dLM* (Algorithm 3.2, lines 2-5; Figure 3.4 @t_1). The *dRM*: (1) powers-on the target router, and (2) stops NI from injecting any new packets. If the *dRM* is controlling one of the border routers, then it also sets a flag to stop neighbouring routers of other regions from injecting any new packets. The reason behind stopping the injection of packets is to ensure that all the in-transit packets reach their destination before the actual switch-over starts. Once the router buffers are flushed and no packets are injected by the neighbouring routers, the *dRM* sets the empty flag, which is propagated through the AND-gate

network. It is important to note that the powering-on of the target router and the flushing of the currently active router (Figure 3.4, $t_1 - t_3$) is simultaneous to speed up the switch-over mechanism.

Once the empty flags from all the *dRMs* have been propagated to the *dLM* through the AND-gate network, the *dLM* concludes the flushing of the currently active network layer. Then it starts another round of control flits to all the *dRMs* to start the actual switch-over of routers (Algorithm 3.1, lines 5 – 7). On receiving the control flit, a *dRM* enters into its second phase (Algorithm 3.2, lines 8 – 12), where it: (1) disables the input/output ports of the active router (Figure 3.4, @t_3), (2) enables the input/output ports of the target router (Figure 3.4, @t_4), and (3) powers-off the (previously) active router (Figure 3.4, @t_5). Note that the order of (1) and (2) is essential to avoid short-circuit conditions. At the end, the *dRM* sets the empty flag, which is propagated through the AND-gate network.

Once the *dLM* detects (from the AND-gate network) that the network is empty again, it starts the last round of control flits to all the *dRMs* (Algorithm 3.1, lines 8 – 10). This control flit forces a *dRM* to enter its third phase (Algorithm 3.2, lines 13 – 16), where the injection of packets into NI is enabled (Figure 3.4, @t_6). Further, if the *dRM* is controlling one of the border routers, then it enables injection of packets from the neighboring routers of other regions. With this, the network layer switch-over mechanism is complete.

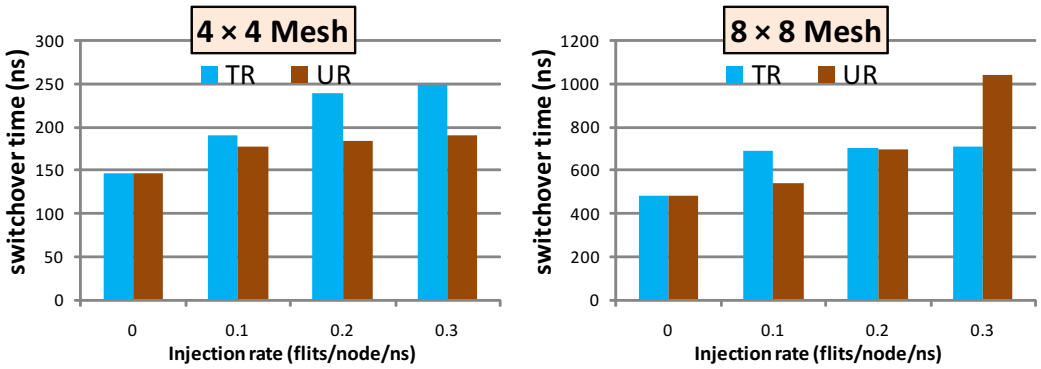


Figure 3.5: Time overhead of switch-over mechanism for NoC running @ 500MHz

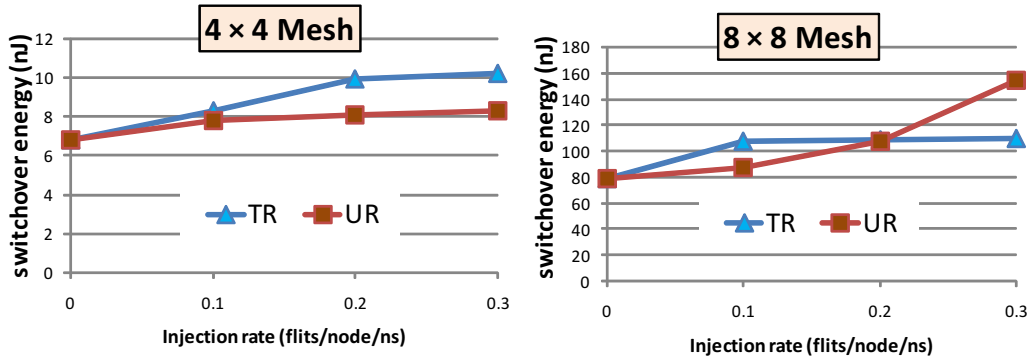


Figure 3.6: Energy overhead of switch-over from [500MHz,0.81V] to [750MHz,0.81V] layer

Time and energy overhead. Now we present an analysis of the time and energy overhead of switch-over mechanism in darkNoC architecture. For this analysis, we assume that the transition of VF level is done after the switch-over from an active network layer to a target network layer. We do not include the time and energy overhead of VF scaling because this overhead is present in the traditional single layer NoC as well. The time overhead in clock cycles of switch-over can be estimated as:

$$\begin{aligned} T_{so} &= \textit{in-transit-packets}(m,\ \textit{load}) + \textit{no-load}(m) \\ &\quad + \textit{control-flits}(m,\ \textit{load}) \end{aligned}$$

The *in-transit-packets* term represents the time to flush the active network layer which depends on the number of routers m and the network load. The *no-load* term captures the time spent in switch-over of all the routers which only depends on m, while the term *control-flits* represents the time spent by *dLM* in sending control flits to *dRMs* and is dependent on both m and the network load. Since the whole switch-over mechanism occurs at the frequency of the active network layer, it can be used to convert the switch-over clock cycles to actual time.

The energy overhead of switch-over mechanism can be estimated as:

$$\begin{aligned} E_{so} &= \textit{power-on-energy}(L_t,\ m) + \textit{control-flits}(L_t,\ L_a,\ m) \\ &\quad + \textit{leakage-power}(L_t,\ L_a,\ m) \times T_{so} \end{aligned}$$

where L_a and L_t refer to the active and target network layers respectively. The first term *power-on-energy* is the energy overhead of powering-on a network layer, which depends on the type of the target network layer and the number of routers m in it. During a typical switch-over, the control flits from the *dLM* use both the active and target network layers, and their energy consumption is captured in the *control-flits* term. The last term includes the leakage energy of the active and target layers, since both the layers consume leakage power for almost the whole duration of the switch-over mechanism.

We experimented with 4×4 and 8×8 mesh sizes, transpose random (TR) and uniform random (UR) traffic patterns and varying traffic injection rates (details are in Section 3.4). The results for time and energy overhead are presented in Figures 3.5 and 3.6 respectively. Three trends are evident. The time and energy overhead: (1) increases with an increase in mesh size due to a higher number of control flits and hops, and longer delays in the AND-gate network, (2) increases with an increase in the traffic injection rates due to longer flushing times of in-transit packets and delays of control flits, and (3) heavily depends on the network load as trends are different for TR and UR traffic patterns. It is important to note that the time and energy overhead of our switch-over mechanism is not significant compared to the total execution time of typical applications and energy consumption of the NoC (see Section 3.4).

Topology	4×4 mesh
Router Architecture	5 port router (4 neighbors + 1 local), 3 stage pipeline [LT+BW, RC+SA, ST]
Input Buffer	8 flit deep (no virtual channel)
Routing	Dimension Order XY
Link Flow Control	On/Off
Link Width	64 bit data, 8 bit control
Switch Arbiter	Matrix arbiter

Table 3.1: NoC architectural details.

Frequency Voltage	1 GHz 0.9 V	750 MHz 0.81 V	500 MHz 0.81 V	250 MHz 0.72 V
Area [μm^2]	49200	46313	45750	45225
HVt Cells [%]	44.7	60.5	90.7	92.8
NVt Cells [%]	12.5	13.6	4.3	3.9
LVt Cells [%]	42.8	25.9	5.0	3.3

Table 3.2: Synthesis results for a single router.

3.4 Experiments and Results

3.4.1 NoC Synthesis

We use a packet-switched wormhole NoC architecture for experiments. The details of the NoC architecture are reported in Table 3.1. The proposed router is implemented in Verilog RTL, and synthesised using Synopsys Design Compiler for commercial TSMC 45nm libraries characterized for HVt, NVt and LVt cells. We enabled leakage and dynamic power optimisation in Synopsys Design Compiler, which automatically enabled multi-Vt optimisation. We explored the following four VF levels: [1GHz, 0.9V], [750MHz, 0.81V], [500 MHz, 0.81V] and [250 MHz, 0.72V], and the synthesis results are reported in the last four columns of Table 3.2.

We can observe that the target VF level significantly affects the type and number of cells used during optimisation. For example, the router optimised for [1GHz, 0.9V] contains 42.8% LVt cells, which decreases to 5.0% and 3.3% for optimisation

Topology	4×4 mesh
Processors	Tensilica in-order LX4 @ 1GHz
L1 I/D Caches	4KB, 2-way set associative, 16 byte line size
DRAM	2 memory controllers, 40 ns latency
NoC VF Levels	[1GHz, 0.9V], [750MHz, 0.81V], [500 MHz, 0.81V], [250 MHz, 0.72V]

Table 3.3: MPSoC architectural details.

AM-1	mpeg2enc, sha, mpeg2dec, g721enc
AM-2	h264enc, mpeg2dec, sha, jpeg_dec
AM-3	jpeg_enc, sha, g721enc, g721dec
AM-4	g721enc, mpeg2enc, g721dec, mpeg2dec

Table 3.4: Details of application mixes with four copies of each application.

at [500 MHz, 0.81V] and [250 MHz, 0.72V], respectively. Moreover, the cells are sized according to the target VF level, which has resulted in different silicon areas for different target VF levels (row 2). These results corroborate our motivational example that it is worthwhile to explore multi-Vt optimisation in NoCs for energy efficiency.

3.4.2 Experimental Setup

MPSoC and Applications. We used a 16-node MPSoC laid out as a 4×4 2D-mesh, with whose architectural details are in Table 3.3. Four VF levels are assumed for the NoC, whose architectural details are in Table 3.1. Note that the VF level of processors is not changed in our experiments. We used eight applications from the mediabench suite and created diverse multi-programmed application mixes (AM) whose details are in Table 3.4. For an application mix, four copies of each application in that mix were used. We also used synthetic traffic injection models consisting of Uniform Random (UR), Bit Complement (BC) and Transpose (TR).

Simulation Setup. We used two-step system simulation methodology where

Name	**Configuration**	**Relative Chip Area**
baselineNoC	Traditional NoC with [1Ghz, 0.9V] layer	1 ×
darkNoC1	[1GHz, 0.9V] + [750MHz, 0.81V]	1.13×
darkNoC2	[1GHz, 0.9V] + [500MHz, 0.81V]	1.13×
darkNoC3	[1GHz, 0.9V] + [750MHz, 0.81V] + [500MHz, 0.81V]	1.26×

Table 3.5: NoC configurations used in experiments.

the memory access trace of each application executing on a processor is collected from the Xtensa instruction set simulator. These memory access traces are then simulated through a closed-loop cycle-accurate NoC and DRAM simulator. Our NoC simulator also modelled different VF levels accurately for the NoC. We also modelled the *dLM*, *dRM* and AND-gate network in the NoC simulator. For application mixes, each processor was executed for at least 2 million instructions. For synthetic traffic injection models, the NoC was warmed up for 100,000 clock cycles, followed by collection of statistics for 80,000 clock cycles. The injection rates were measured in $flits/node/ns$.

NoC Power Estimation. We used Synopsys PrimeTime tool to analyse the netlist generated by Synopsys Design Compiler to compute leakage and dynamic power of a NoC. We first apply this analysis to compute power values at the nominal VF level of the NoC. Then, scaled VF conditions are applied to compute power values at VF levels lower than the nominal VF level. These computed power values are used in the NoC simulator to compute the total NoC energy consumption. This methodology is also used for all the network layers in a darkNoC.

NoC Configurations. We used different NoC configurations in our experiments, which are listed in Table 3.5. The **baselineNoC** represents the traditional single layer NoC designed for the highest VF level, [1GHz, 0.9V]. *For fairness of comparison, we synthesised baselineNoC with multi-Vt optimisation.* The **darkNoC1** and **darkNoC2** configurations have 2 VF-optimised network layers each, while **darkNoC3** configuration is a superset of darkNoC1 and darkNoC2. In Table 3.2, the difference in silicon area and distribution of multi-Vt cells between [500 MHz, 0.81V] and [250 MHz, 0.72V] optimised routers (columns 4 and 5) is small. This means that two network layers optimised for [500 MHz, 0.81V] and [250 MHz, 0.72V] will have nearly identical properties, and thus similar energy efficiencies. Therefore, a designer can use synthesis results for early elimination of network layers in a darkNoC. That is why we did not include [250 MHz, 0.72V] optimised network layer in our experiments.

The last column of Table 3.5 reports the relative chip area of NoC configurations, measured as a *sum of the area of the processors, caches, network layers, and the dLM and dRMs*. A designer can use darkNoC1 or darkNoC2 if they have dark silicon budget of a single network layer. Likewise, they can use darkNoC3 if the dark silicon budget permits the inclusion of two network layers.

System-level DVFS Manager. We used the state-of-the-art memory latency based NoC DVFS technique introduced by Chen et al. [118] for the baselineNoC. We used a control interval of $50K$ clock cycles for the DVFS. For the darkNoC configurations, we modified their technique as follows. If the DVFS manager decides to switch to the *i-th* VF level and there is a network layer optimised for the *i-th* VF level, then the DVFS manager requests the *dLM* to switch-over to that particular network layer. On the other hand, if there is no network layer optimised for *i-th* VF level, then the DVFS manager requests the *dLM* to switch-over to the network layer optimised for the closest yet higher VF level, and will scale the VF of the selected network layer. For example, in darkNoC1, if the system-level DVFS manager decides to operate NoC at [250 MHz, 0.72V], then the [750 MHz, 0.81V]

network layer will be scaled down to operate at [250 MHz, 0.72V] rather than the [1 GHz, 0.9V] network layer. We created two flavours of DVFS managers based upon application requirements: **DVFS-1 with a target performance loss of 15%** and **DVFS-2 with a target performance loss of 10%.** Note that our NoC simulator *included the time and energy overhead of network layer switch-over* for the darkNoC configurations.

3.4.3 Results and Discussion

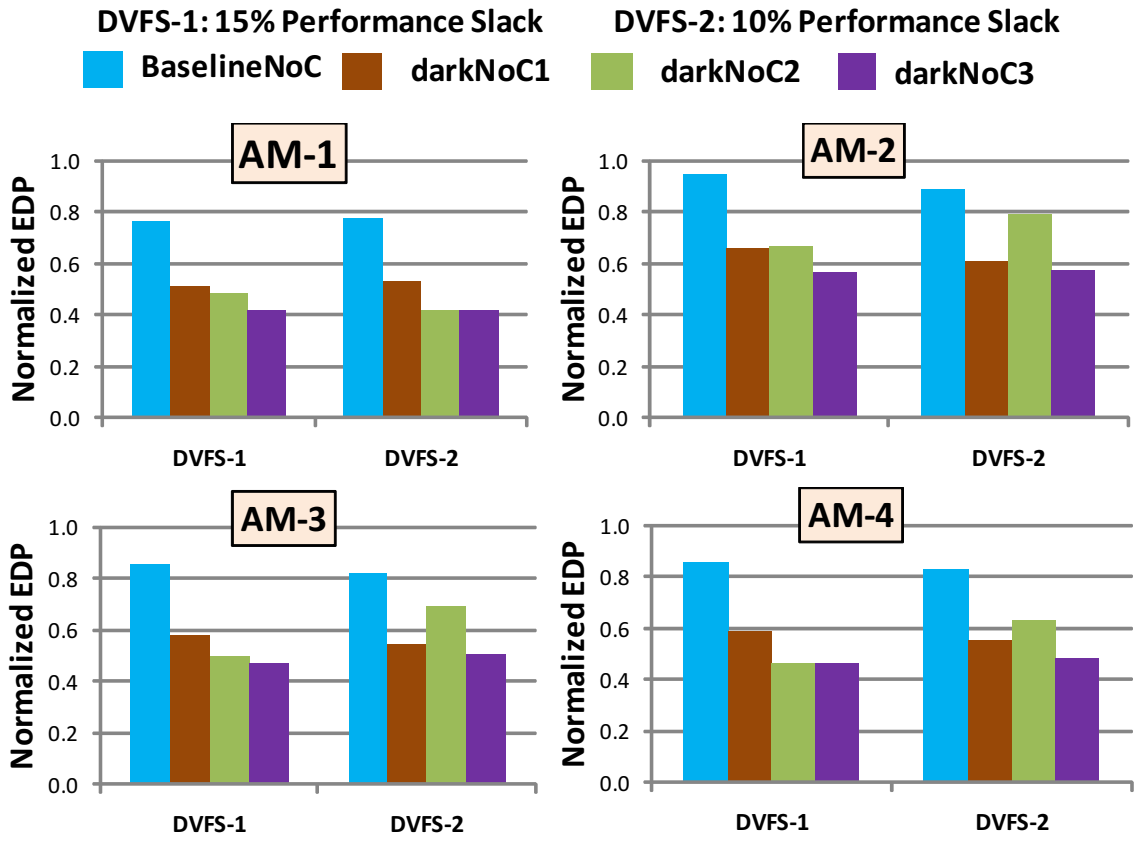


Figure 3.7: NoC Energy-Delay Product (EDP) normalised w.r.t traditional NoC operating at highest VF level.

Overall Trend in NoC Energy-Delay Product (EDP) Savings. Figure 3.7 shows the savings in NoC EDP for the four application mixes of Table 3.4,

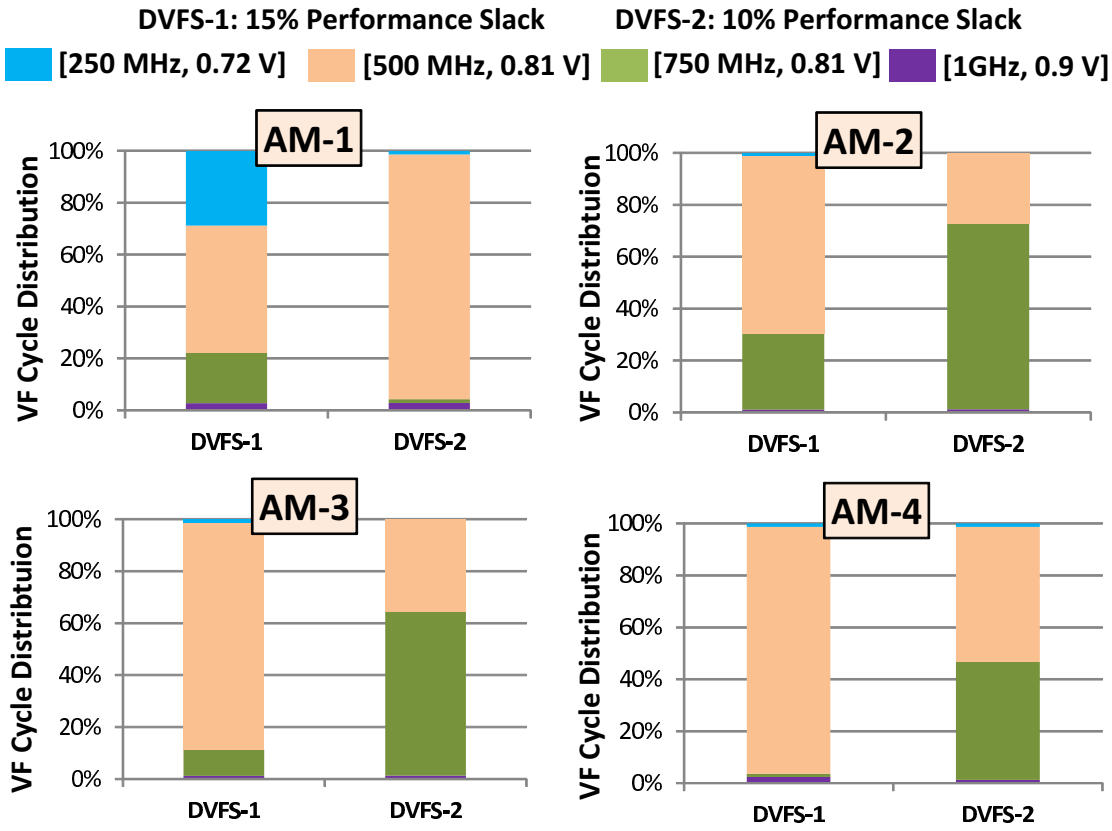


Figure 3.8: Distribution of NoC VF cycles.

four NoC configurations of Table 3.5 and two DVFS managers. Overall, darkNoC configurations shows an improvement in EDP over baselineNoC.

For AM-2, the DVFS-1 reduced EDP by only 5% in baselineNoC. On the other hand, EDP is reduced by 34% and 44% by DVFS-1 in darkNoC1 and darkNoC3, respectively.

Likewise, for AM-3, DVFS-1 in baselineNoC saved 15% EDP, whereas DVFS-1 in darkNoC1 and darkNoC3 reduced EDP by 45% and 52%, respectively. In summary, darkNoC provides up to 56% EDP savings (AM-1, darkNoC3, DVFS-2) over baselineNoC. These results show that DVFS alone can be ineffective in reducing NoC energy due to increasing leakage power to dynamic power ratio. Thus, our darkNoC architecture complements system-level DVFS managers.

Interplay of Applications, DVFS Manager and darkNoC Configurations. EDP savings vary across DVFS managers and darkNoC configurations. For example, in AM-2, AM-3 and AM-4, use of DVFS-2 with darkNoC1 is far more beneficial than DVFS-2 with darkNoC2. Likewise, for AM-3 and AM-4, DVFS-1 with darkNoC2 results in more EDP savings than DVFS-1 with darkNoC1.

To further explain these results, in Figure 3.8, we report the distribution of NoC VF cycles, which is the percentage of total execution cycles the NoC operates at a particular VF level. For AM-2, DVFS-2 operates the NoC at [750MHz, 0.81V] and [500MHz, 0.81V] levels for 71.4% and 27.4% of the time, respectively. Thus, darkNoC1 results in better EDP savings because it has a [750MHz, 0.81V] optimised network layer. In the case of AM-4, DVFS-1 with darkNoC3 provided only 1% EDP improvement over darkNoC2, at the expense of an additional network layer. This is because the NoC is operated at [500MHz, 0.81V] level for 95% of the time, and thus a darkNoC with [500MHz, 0.81V] optimised network layer is more than sufficient.

Insights. From these results, we can conclude that savings in energy-delay product depends on:

- darkNoC configuration
- System-level DVFS manager
- Applications

There is a complex interaction between these factors. For example, the energy efficiency of a darkNoC configuration (number and type of network layers) heavily depends upon the dark silicon budget and distribution of VF cycles, which in turn depends upon the target performance loss set in the system-level DVFS manager and the executing applications. Taking it one level further, the decisions taken by the system-level DVFS manager depend on the number and type of network layers in the darkNoC configuration, which results in a cyclic dependency between these factors. Thus, the problems (highlighted in the Introduction) of: (1) determining the number and type of network layers, and (2) enhancement of the system-level DVFS

manager to exploit multiple network layers, are interdependent. Consequently, these problems result in a multidimensional design space exploration problem.

3.4.4 Results for Static VF Selection

In addition to using DVFS, we also explored the case where a static VF (SVFS) is applied to the NoC throughout the AM execution. These experiments were repeated with the same set of **darkNoC Configurations**. We present our results for the following SVFS schemes:

- SVFS-250: Static VF level of [250 MHz, 0.72 V]
- SVFS-500: Static VF level of [500 MHz, 0.81 V]
- SVFS-750: Static VF level of [750 MHz, 0.81 V]

The results for the experiments with SVFS are presented in Figure 3.9 which are similar to our results with the DVFS-based scenario. As expected running the NoC continuously at [750MHz, 0.81V] for darkNoC2 configuration does not result in any energy saving compared to baseline configuration as there is no NoC plane designed for that particular VF level. For VF levels [250 MHz, 0.72 V] and [500 MHz, 0.81 V], all darkNoC configurations give lower EDP than the baseline. However, the darkNoC2 and darkNoC3 configurations give the best result as both configurations contain a NoC designed for [500 MHz, 0.81 V].

3.4.5 Results with Synthetic Traffic

Figure 3.10(b) shows the network power for three VF levels for the UR traffic model. An overall trend is that it becomes power inefficient to apply DVFS on a network layer designed for a higher VF level as the difference between the VF level for which the network layer was designed, and the VF level at which the network layer is being operated increases. This is the reason that, for operation at [250 MHz, 0.72V] VF

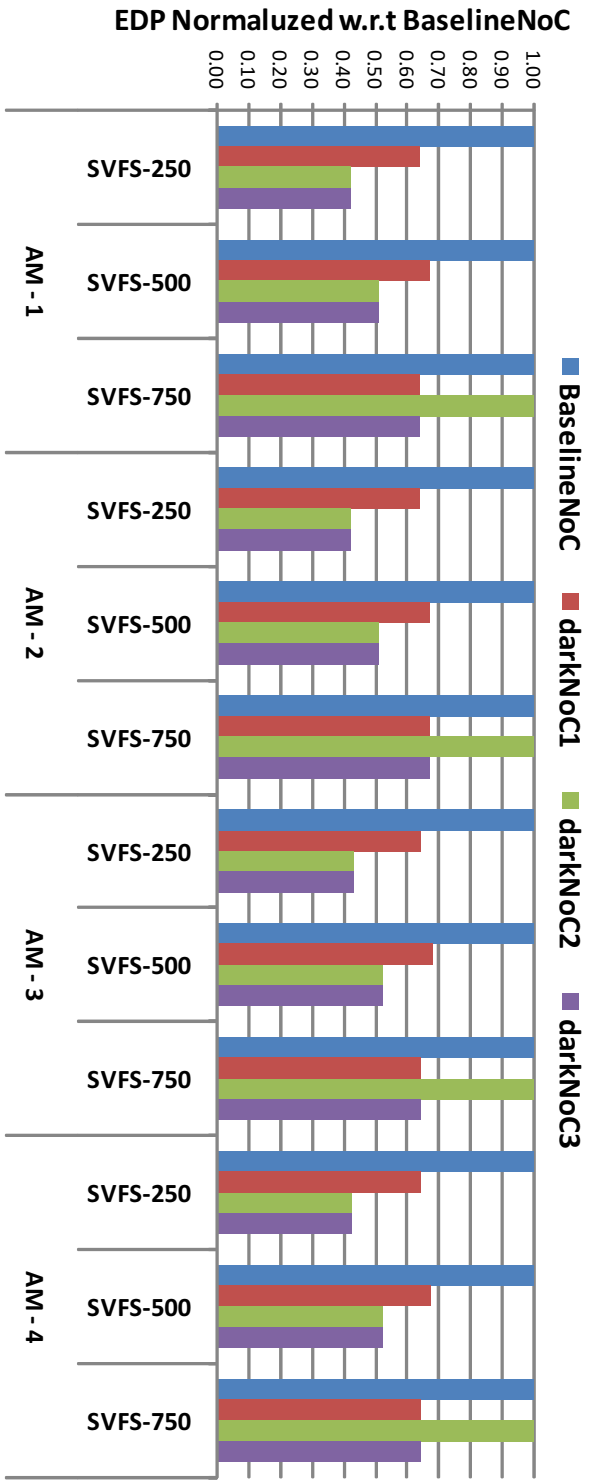


Figure 3.9: Results with Static VF Scaling. Results are normalised w.r.t Baseline NoC

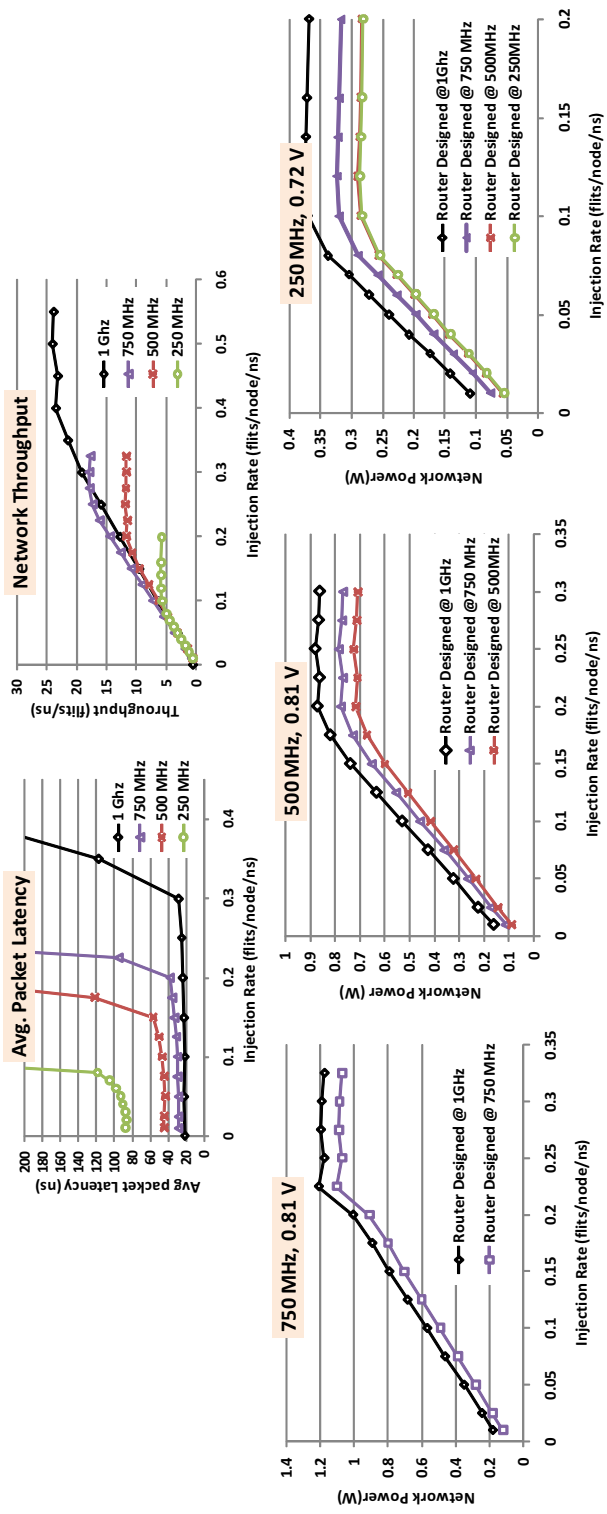


Figure 3.10: Results for UR Traffic a)(top) Avg. Packet Latency and Network Throughput. b)(bottom) Network power results for different VF levels

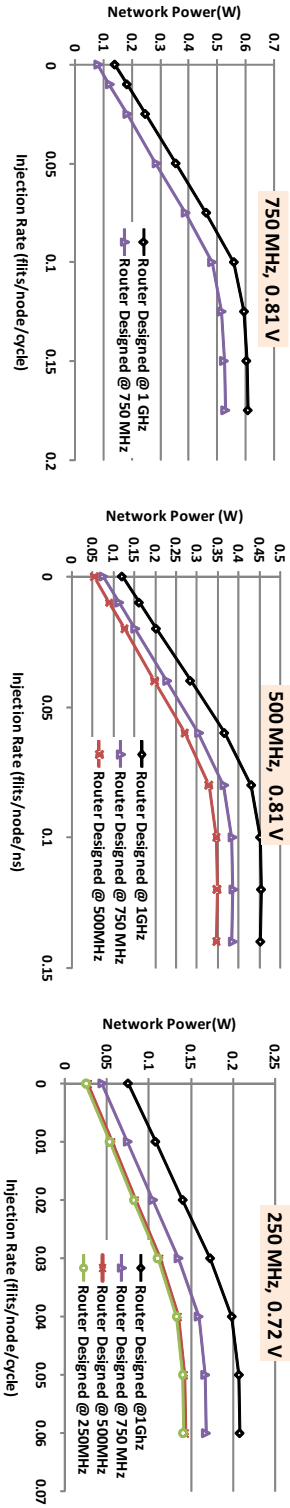


Figure 3.11: Network Power for TR traffic for different VF levels

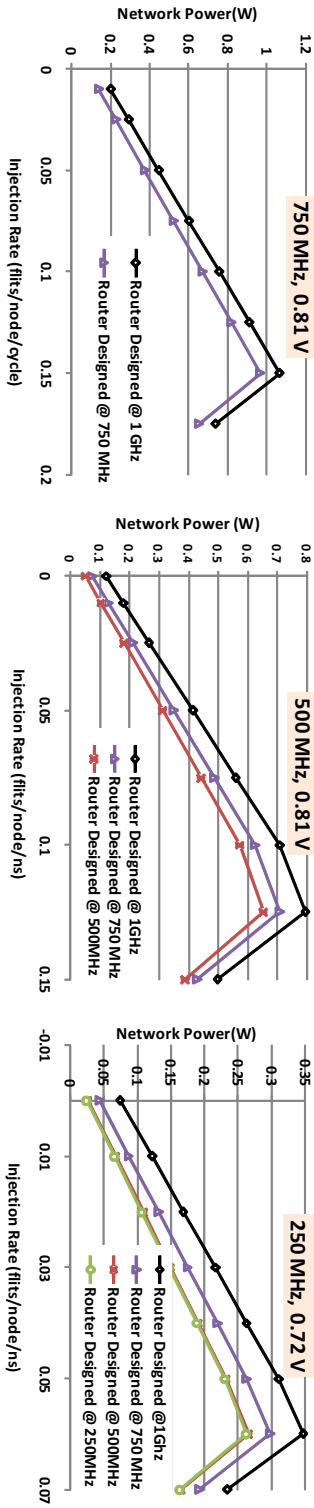


Figure 3.12: Network Power for BC traffic for different VF levels

level, the NoC designed for the particular VF level can save on average 37% more power than DVFS applied to NoC designed for [1GHz, 0.9V] and 20% more power than DVFS applied to NoC designed for [750 MHz, 0.81V].

Results for TR and BC traffic are presented in s 3.11 and 3.12. Combining the result from the three synthetic traffic patterns we can observe that usefulness of darkNoC architecture is dependent on relative ratio of dynamic and leakage power. The UR traffic has the highest saturation throughput whereas the TR has the lowest saturation throughput. The saturation throughput directly translates into dynamic power as network can sustain higher injection rate. Therefore, we can observe that the darkNoC provides better energy efficiency for TR traffic than for UR traffic.

3.5 Conclusion

In this chapter we presented darkNoC, a novel NoC architecture consisting of multiple network layers to complement system-level DVFS managers. Each network layer is optimised at design-time to operate within a voltage-frequency range using multi-Vt circuit optimisation. We utilise dark silicon that will be available in future chips to realise the extra network layers. Compared to a traditional single layer NoC, darkNoC enabled system-level DVFS to achieve up to 56% better energy-delay product. We believe that darkNoC will open new research avenues in energy-efficient NoC design for the dark silicon era.

Chapter 4

Malleable NoC: Dark Silicon Inspired Adaptable NoC

4.1 Introduction

Network-on-Chips (NoCs) are used in multi-core chips as a scalable communication fabric. As technology nodes shrink and the number of cores in chips increase, designers continue to face the challenging task of balancing NoC power and performance. NoC, being a shared resource in a chip, has a strong role to play in the application performance and energy consumption. Thus, numerous researchers have introduced power saving techniques which minimally affect the application performance. Dynamic voltage and frequency (DVFS) [114,115] is a prominent technique which exploits the application execution phase to reduce the voltage and frequency of NoC.

Three aspects motivate this work:

• The first aspect is that general-purpose multi-cores execute a diverse set of applications. Each application interacts with cache architecture uniquely, hence the NoC traffic injection rates can vary across all applications. The behaviour of two such applications is presented in Fig 4.1. In this architecture, L1 cache is

the last level cache (before the misses are transported on the NoC). It is evident from Fig 4.1 that misses per kilo instruction (MPKI) is very different for the two applications *H264*, and *MPEG2ENC*. On average the MPKI for *H264* is 40× that of *MPEG2ENC*. Due to a higher cache miss rate, *H264* has a higher dependence on NoC latency. Therefore, the NoC may have to be clocked at a higher frequency. On the other hand, *MPEG2ENC* has a very low cache miss rate, providing an opportunity to save energy by clocking the NoC at a lower frequency without any significant system performance degradation. However, simultaneous execution of the two applications will require the NoC to be executed at a higher frequency to support the worst-case application.

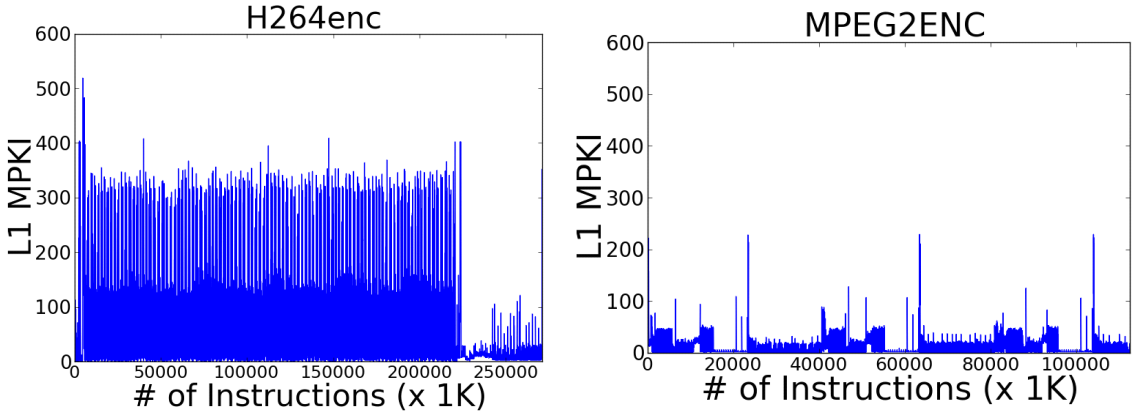


Figure 4.1: L1 MPKI for 2 different applications

• The second aspect is that in addition to the applicable MPKIs, the application to core mapping in mesh NoC based multi-core also has a considerable impact on performance and energy. An example 4×4 mesh is shown in Figure 4.2. Cache load and store misses from the cores get serviced by the memory controller. The cache load time penalty is directly proportional to the hop distance between the core and memory controller. The two applications mentioned earlier can be mapped to either core A or core B. If *H264* is mapped to core B, NoC frequency will not have any significant effect on system performance. However, if *H264* is mapped to core A, NoC frequency could play a vital role since *H264* has a higher MPKI. On the

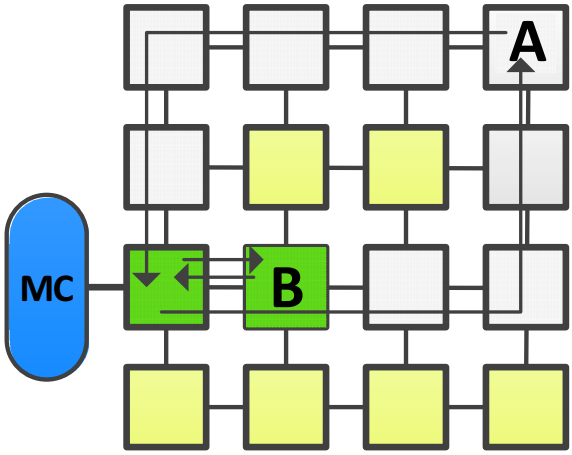


Figure 4.2: Application Mapping for 4×4 Mesh NoC (MC: Memory Controller)

other hand, due to lower MPKI of *MPEG2ENC*, mapping *MPEG2ENC* to core A or B doesn't matter, and irrespective of mapping, NoC can be operated at a lower frequency. Intuitively, if the two application must be mapped at the same time, *H264* to core B and *MPEG2ENC* to core A. Given such mapping, the grey routers can be clocked at a lower frequency, while green routers can be clocked at a higher frequency. In this discussion, the voltage can be adjusted with the frequency to save power as needed.

Per router DVFS (where certain routers work faster, while others run slower at the same time) makes sense in scenarios where applications with markedly differing MPKIs are executed simultaneously.

• Although DVFS is a useful technique, its effectiveness has been reduced with shrinking node size/technology. Leakage power is a major contributor towards total NoC power (and is expected to grow in proportion in the future). Leakage power remains largely unaffected with reduction in operating voltage. To counter this phenomenon, *darkNoC* (proposed in Chapter 3), which uses dark silicon to complement the system level DVFS by the use of muti-Vt based multiple routers instead of a single router optimised for the highest frequency. The proposed architecture consists of a multi-layer NoC, where each NoC layer is optimised for a particular voltage and frequency(VF). Using darkNoC, with the associated pre-set VF transitions, system

switches between different NoC layers. However, the architecture proposed in Chapter 3 is limited to cases where all routers lie in a single VF island. Therefore, the darkNoC architecture cannot be used with per-router DVFS.

Thus the third and final aspect is the need for a NoC where the routers can work simultaneously at differing frequencies and utilise the dark silicon area to save energy.

Based on our observations, we introduce a new and novel low power NoC for dark silicon chips. We call this architecture *Malleable NoC*. At each mesh node (where traditionally only one router lies), we integrate *architecturally homogenous, yet circuit-wise heterogeneous routers* designed for different frequencies and voltages that support pre-set VF switching of each router. Given such a NoC, at runtime, depending on the dynamic profiling of the applications, we choose a different router from each node (hence the term *Malleable*), and connect these routers to create a low power NoC fabric. The composition of this NoC fabric adapts to the dynamic changes in the application phase. Routers that are not used in a particular phase remain switched off (*dark*). We further explore the fact that intelligent mapping of applications to cores can be exploited for efficient per-node frequency and router selection.

4.1.1 Contributions:

We make the following contribution in this chapter:

- We introduce a dark silicon-inspired NoC architecture called *Malleable NoCs*. We discuss the technical challenges that are required to realize such an architecture with per router VF switching control.

- We introduce a novel per-router VF selection algorithm. This algorithm uses application profiling and mapping information for runtime management of *Malleable NoC* resources.

- We use full system simulations with diverse applications to show the efficacy of the proposed architecture. We show that our architecture is more useful when applications have diverse MPKIs.

4.2 Related Work

Recently various heterogenous NoCs have been proposed for general purpose multi-core chips. Mishra et al. [11] proposed integrating two architecturally heterogenous NoCs in a chip. Each NoC is designed using a different operating frequency, data width and Virtual Channel (VC) configuration. One network is optimised for low latency while the other network is optimised for bandwidth. At runtime, processors inject traffic in one of the networks based on application profiling. In contrast, our proposed architecture creates only one NoC fabric out of various per-node routers. Bakhoda et al. [167] proposed NoC with heterogenous routers using *checkerboard* configuration. In this proposed architecture, some of the regular routers are replaced by a router with limited connectivity. Mishra et al. [168] also proposed heterogenous NoC at router granularity by integrating *big* and *small* routers. Big and small routers differ in terms of channel width and number of VCs. In contrast, we introduce heterogeneity by integrating architecturally homogenous but VF-optimised routers at each node and switching on one router at each node depending on the application profile. This allows us to create virtual layers at runtime, making our NoC architecture malleable.

Over the years various system-level power management techniques have been introduced. DVFS for on-chip networks has been proposed at various levels of granularity such as link-level [113], router-level [114, 115] and region-level [118, 169, 170]. The DVFS algorithms used in [113–115] are oblivious of applications' mapping and performance, and use NoC queue occupancy or network congestion to select a frequency. Chen et al. [118] presented some DVFS algorithms based on application profiling. However, these algorithms do not take into account scenarios

where potentially heterogenous application are executed. Furthermore, all of these research studies assume fast on-chip voltage regulators which are impractical for per router DVFS due to low voltage conversion efficiency and design complexities [46]. In contrast to all these previous studies, we consider both application to core mapping and application profiling to decide the optimum frequency at the granularity of a router. To reduce the design complexity and improve efficiency, we explore the use of multiple supply voltage rails. Nakamura et al. [56] used multiple voltage supply rails to propose a variable pipeline router in which number of pipeline stages are changed in response to voltage change, while keeping frequency constant. In contrast, in our proposed architecture we switch between the VF-optimised routers and change the operating frequency in response to voltage changes.

Pullini et al. [162] explored the usefulness of multi-vt optimisation technique for NoC synthesis for leakage power saving. Bokhari et al. [170] used multi-vt optimisation and exploited dark silicon to provide a multi-layer NoC, where each NoC layer is optimised for a given VF. This scheme considerably improved the efficiency of system-level DVFS schemes. In this work we show that in addition to multi-vt optimisation, application profiling and mapping, and per-router VF selection can further reduce the NoC energy.

4.3 Malleable NoC Architecture

4.3.1 NoC Architecture:

Our target architecture is a general-purpose tiled multi-core chip as shown in Figure 4.3, which closely matches commercial SoC chips such as Intel Single Chip Cloud Computer(SCC) [110] architecture. We assume a processing core with private last level cache. Cache load and store requests are injected into the NoC by cores. Memory controllers are connected to border routers as shown in Figure 4.3 to serve these

load and store requests. We target a N×N Mesh topology based NoC architecture in this work as it is commonly used by commercial and academic multi-core chips. Each core runs a different program, therefore, at a given time, up to N×N applications are executed in parallel.

4.3.1.1 Per Router VF Selection:

Normally high efficiency off-chip voltage regulators are used to scale high supply voltage to low voltage for the purpose of DVFS. This scheme has been used for region-based DVFS for NoC in Intel SCC [110]. However, the voltage transition takes considerable time, which can be as high as 1 millisecond [110]. This limits the usefulness of DVFS algorithms for NoCs. Off-chip voltage is also an impractical option for fine-grain per-router DVFS due to limited chip pins and a large number of voltage regulators are required for large mesh sizes such as 8×8. There are two different approaches to support fast per-router DVFS: 1) use high switching frequency on-chip voltage regulators [2] for fine-grain DVFS, or 2) use multi-voltage rails [46] for coarse-grain VF selection. On-chip voltage regulators provide very fast voltage switching. Analysis by Kim et al. [2] show that voltage transition can take place within 100-200 ns. However on-chip voltage regulators suffer from low conversion efficiency of about 75-80%.

Multi-voltage rail (MVR) design enables fast coarse-grain switching between different voltages without compromising efficiency. The voltages are generated through high efficiency off-chip voltage regulators. Mathematical modelling performed by Miller et al. [46] show that voltages can be switched within 7-10 ns at runtime. This can reduce the performance overhead of DVFS schemes. However, it limits the possible VFs for routers.

In our architecture, we employ MVR for each router to realise per-router voltage scaling. For frequency scaling, a global clock is generated and distributed using a normal clock tree. At each node a scaled frequency is generated using local clock dividers. In this architecture the master frequency is scaled down in powers of 2 only

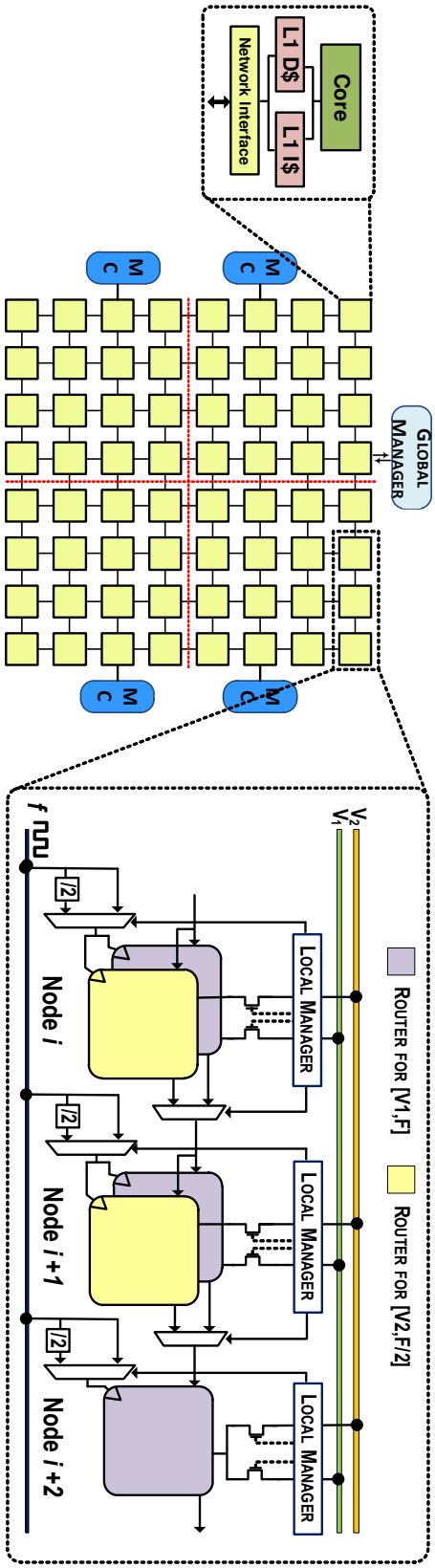


Figure 4.3: Malleable NoC

i.e., f , $f/2$ and $f/4$. This keeps the NoC routers ratio-synchronous and reduces the design complexity and performance penalty associated with bisynchronous FIFOs [166]. We implemented the per-router VF switching by combining fast voltage switching using MVR and local frequency division.

4.3.1.2 Dark Silicon Inspired Multiple Router with Multi-Vt Optimisation:

From our discussion in previous chapters, we know that we have more than enough silicon resources available due to *dark silicon* phenomenon. In our Chapter 3, proposed using dark silicon for improving efficiency of DVFS for NoC. This is enabled using a well known circuit synthesis technique called multi-vt optimisation. In *darkNoC* architecture, all of the routers are placed in a single VF island. The *Malleable NoC* differs from *darkNoC* in three aspects: a) *Malleable NoC* uses per router VF selection, b) *Malleable NoC* architecture does not put a constraint of replicating router at every node, and, c) *Malleable NoC* framework integrate application-to-core mapping information in the VF selection algorithm.

We use the multi-vt optimisation technique to design routers for different VFs and integrate them at each mesh node. Our novel *Malleable NoC* is a flexible NoC fabric that enables combining one router from each node to create an adaptable NoC at runtime. Instead of chip-wide budget, we explore a per node dark silicon budget based design paradigm. In this way we can have a partial *Malleable NoC* architecture where only a limited number of nodes contain multiple routers. Such architectures are useful where NoC has to compete with other dark silicon inspired components such as application accelerators [22]. Therefore, we also explore designs where 25%, 50% or 75% of nodes contain multiple routers.

An example template for *Malleable NoC* architecture is shown in Figure 4.3. On the right, routers for three consecutive nodes are shown. As discussed earlier, we use MVR for per-router VF selection. In the architecture shown in Figure 4.3, we have 2 different voltage rails $V1$ and $V2$, where $V1 > V2$. These voltages are

associated with two frequencies f and $f/2$. At each node we can have maximum of two different routers that are optimized for 2 VF pairs. Similarly, there can be nodes which only contain s router for maximum VF. At each node, we can have following configuration in the architecture shown in Figure 4.3:

1. Routers for [f,$V1$]
2. Routers for [f,$V1$], [$f/2$,$V2$]

Each router is optimised for a particular VF using multi-vt optimisation. The single router in the Configuration 1 is connected to both voltage rails. Whereas, for Configuration 2 the router designed for [f,$V1$] is attached to voltage rail $V1$ and the router designed for [$f/2$,$V2$] is connected to $V2$. If there is no router available for a target VF, the router designed for the closest VF is used instead. Therefore, if a node needs to be operated at [$f/2$,$V2$] and there is no router designed for [$f/2$,$V2$] integrated at that node, the router designed for [f,$V1$] is used by connecting it to the $V2$ rail. Multiplexers are used to connect one of the routers at a node to upstream routers. These multiplexers add minimum hardware and do not affect the critical path of the router.

4.3.1.3 Malleable NoC Management:

Routers at each node are managed by a Local Manager (LM). The LM controls the power gating signal and also selects the clock for each node. The LM can be addressed by sending a single flit packet over NoC. The LM also monitors whether all the buffers in the routers are empty. All the LMs are managed by one Global Manager (GM) on the chip. This GM can be a part of existing NoC power controllers. At runtime, the LM and GM coordinate autonomously to implement the VF selection scheme as shown in Algorithm 4.1 and Algorithm 4.2.

The $RunVFselectAlgo()$ function called in Algorithm 4.1 is the actual per router DVFS selection algorithm which will be explained in details later in this section. The $RunVFselectAlgo()$ function returns the frequency setting for each router in

Algorithm 4.1: Runtime NoC Management by GM

```
1  nodes = m; // number of routers in chip
2  MPKI = [ ]
3  for i=nodes to 0 do
4      InstCount,CacheMiss = recvProfileData();
5      MPKI[i] = CacheMiss / InstCount;
6  FrequnecyMap = RunVFselectAlgo(MPKI[]); // get the new frequency
   for each router
7  for i=nodes to 0 do
8      StopNIsLM(i);
9  WaitForNoCEmpty();
10 for i=nodes to 0 do
11     if FrequnecyMap[i] has changed then
12         SendNewFreqToLM(i);
13 WaitForNoCEmpty();
14 for i=nodes to 0 do
15     EnableNIsLM(i);
```

Algorithm 4.2: Local Manager Operation

```
1  while (True) do
2      command = RecvCommand();
3      if command == StopNI then
4          StopNI();
5      else if command == StartNI then
6          if DifferentRouterWasRequired then
7              SwitchOffOldRouter();
8          StartNI();
9      else if command == ChangeFreq then
10         if DifferentRouterRequired then
11             SwitchOnNewRouter();
12         else
13             SwitchToNewVoltage();
14         ChangeFrequency();
```

the mesh. First, *GM* stops all Network Interfaces (NIs) from injecting new traffic to ensure no data is lost when routers are switched on or off. Once the NoC is

free of any traffic, *GM* sends information for the new frequency to each router. As mentioned earlier, depending on the architectural configuration of routers at each node and the new frequency, *LM* turns on the new router, changes the voltage and selects the appropriate frequency. Once the new frequency setting for each routers has been set, *GM* sends a control flit to each *LM* to enable normal operation.

4.3.2 Application Mapping and Profiling

As discussed in Section 1, we exploit application mapping to perform efficient per router VF selection. Our mapping is based on policies introduced by Das et al. [9]. We divide our multi-core chip into *memory islands* based on the number of memory controllers. For example, in the architecture in Figure 4.3, there are four memory controllers and therefore we have four *memory islands*. Applications mapped in *memory islands* get their L1 cache served by the associated memory Island. Applications that are required to be mapped are sorted on the basis of L1 MPKI. These applications are then allocated to each memory Island in a round robin order. Within each *memory island*, applications with the highest MPKI are mapped closest to the memory controller. Das et al. [9] showed that such mapping significantly improves the overall system throughput. This method ensures that application that requires the least NoC service are mapped to the centre of the mesh while applications with high communication requirements are mapped to the outer periphery of the mesh (closer to memory controller). Moreover, this mapping scheme more or less balances the load across NoC and on each memory controller.

We assume that an average L1 MPKI for each application is known *a priori*. Note that the L1 MPKI is only dependent on ISA and cache architecture and is independent of NoC parameters and mapping. However for the purpose of run-time per-router VF selection, we regularly sample the L1 MPKI after every control interval. We assume that each processing core contains a L1 cache miss counter and an Instruction Counter. Such counters are normally included with the core for

Algorithm 4.3: VF Selection Algorithm

```
1  FrequnecyMap = [ ][ ]; // frequency for each router
2  cores = m; // number of cores in chip
3  for i= 0 to cores − 1 do
4      SenstivityMetric[i] = getMPKI(i) ∗ getDistranceMC(i);
5  for i= 0 to cores − 1 do
6      if (SenstivityMetric[i] > Threshold1) then
7          SetRoutersInPath(i,FrequnecyMap,MaxFrequnecy);
8      else
9          SetRoutersInPath(i,FrequnecyMap,LowFrequnecy);
10 return FrequnecyMap;
```

performance measurement. The number of instructions executed and number of cache misses are collected at each core and sent over NoC to GM after each control interval.

4.3.3 VF Selection Algorithm

We now present our novel VF selection algorithm. We made two key observations. The first observation is that applications have different L1 MPKIs and this value can change over time, providing an opportunity for run-time NoC VF switching. Moreover, as the MPKI increases, the application spends more time waiting for data to be fetched from DRAM and therefore the NoC latency plays a vital role in application execution time. The second observation is that the execution time of the applications increases when they are mapped further away from the memory controller, as it takes longer for cache read misses to be served. Therefore VF selection algorithms for NoC should take into account both the application's MPKI and its distance from the memory controller. These observations can be summarized formally as:

$$\textit{Execution Time} \propto \textit{L1 MPKI} \times \textit{Distance from MC}$$

We base our VF selection algorithm on this equation, as shown in Algorithm 4.3. This algorithm is executed by GM after each control interval. The applications are initially mapped according to the procedure mentioned earlier in this section. *Note that, in this work, we assume that application mapping is not changed at runtime.* At runtime, each application's L1 MPKI is monitored and transferred to GM. These MPKI values are then multiplied by the distance of the core from the memory controller node. We call the resultant product $Senstivity\ Metric$. Then, the value of $Sensitivity\ Metric$ is compared with already set threshold values. In Algorithm 4.3, we assume that there are two possible frequencies for VF selection. Based on $Sensitivity\ Metric$ and threshold values, we select the new frequency. This frequency is selected for all the routers that lie on the request path from core to memory controller and the reply path from memory controller to core. We assume a static routing algorithm such as dimension order XY routing. For example, in Figure 4.2, all the routers marked grey and green mark the request and reply paths for core A. These steps are repeated for all the cores in the system. The output of the VF selection algorithm is a 2D frequency map containing a frequency setting for each router in the mesh.

Designers can select threshold values depending on the available application performance slack. A high threshold value results in reduced performance and relatively lower NoC power, whereas, a lower threshold value results in higher performance and relatively higher NoC power. We propose performing a generic system profiling to select an appropriate threshold value.

The time complexity of the VF selection algorithm is $O(n)$. However, it is worth mentioning that the runtime of this algorithm does not affect the application execution time because NoC operates normally while the new frequency map is being calculated. The normal NoC operation is only suspended once the new frequency map is calculated.

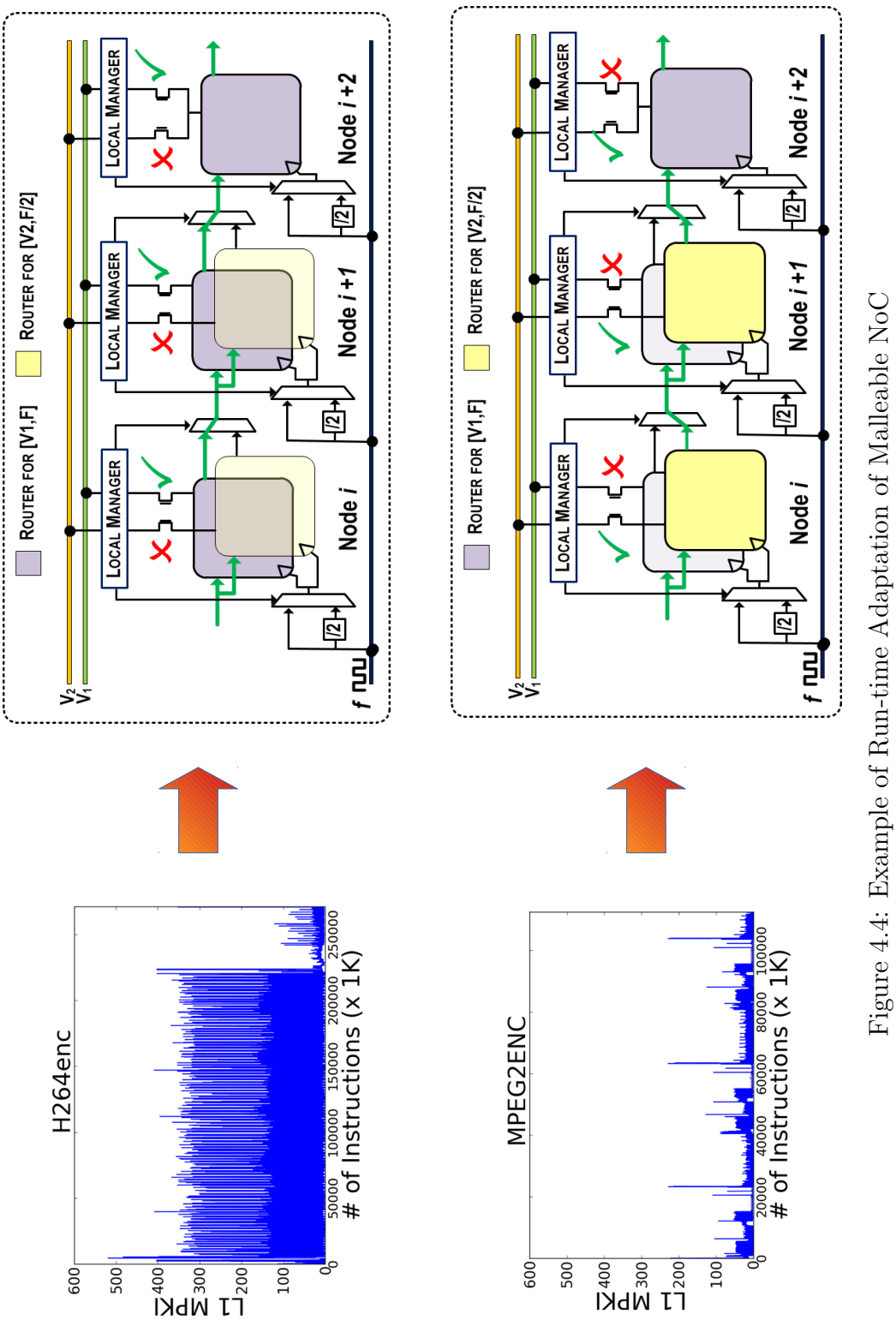


Figure 4.4: Example of Run-time Adaptation of Malleable NoC

Topology	8×8 mesh
Router Architecture	5 port router (4 neighbours + 1 local), 2 stage pipeline [LT+BW, RC+SA+ST]
Input Buffer	8 flit deep (no virtual channel)
Routing	Dimension Order XY
Link Flow Control	Wormhole switching with On/Off flow control
Link Width	64 bit data, 8 bit control
Switch Arbiter	Matrix arbiter

Table 4.1: NoC architectural details.

Frequency	1 GHz	500 MHz
Voltage	0.9 V	0.81 V
Area [μm^2]	33398	28792
HVt Cells [%]	12.7	43.76
NVt Cells [%]	3.6	7.28
LVt Cells [%]	83.7	48.9

Table 4.2: Synthesis results for a single router using Multi-Vt optimisation.

4.3.3.1 Runtime Configuration Example

Figure 4.4 show a visual example of how Malleable NoC is configured at runtime. In the figure, node i and node $i+1$ have routers for both high and low VF whereas node $i+2$ only has a high VF router. When a high MPKI application, *H264* in this case, is executing, the high VF routers are switched on from all the nodes. When a low MPKI application, *mpeg2* in this case, is executing, low VF routers from node i and $i+1$ are switched on. As this example architecture does not contain a low VF router at node $i+2$, a router designed for high VF is connected to lower supply voltage.

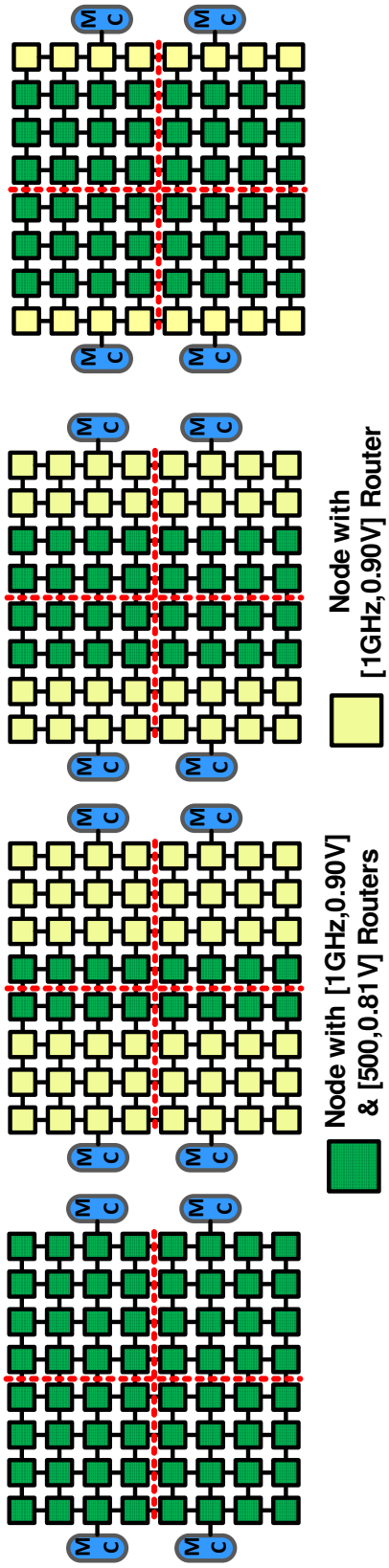


Figure 4.5: Malleable NoC Configurations

4.4 Experiment and Results

4.4.1 NoC Synthesis

Architectural details of the NoC router used in our study are reported in Table 4.1. We use a two stage pipelined wormhole switched router. In the first stage, flit travels on the link and get written in the input buffer. The second stage consist of route computation, switch arbitration and switch transversal. The router architecture RTL is written in Verilog. For input buffers, we changed the RTL style to ensure proper read/write pointer update. As mentioned earlier, the frequencies are in ratios of powers of 2, which ensure the design remain ratio-synchronous, thus reducing the design complexity associated with asynchronous designs.

We synthesised the RTL using Synopsys Design Compiler version *H-2013.03-SP4*. We used commercial TSMC 45nm libraries that are characterised for Low Vt(LVt), Normal Vt(NVt) and High Vt(HVt). We enable leakage and dynamic power optimisation options which automatically invoke multi-vt optimisation. We target two VF levels for the router design in experiments, [1GHz, 0.9V] and [500MHz, 0.81V]. The results for the router synthesis are reported in Table 4.2. We can observe that the target frequency affected the type and number of cells. As expected, the number of LVt cells used for [500MHz, 0.81V] is considerably lower than the LVt cells used for [1GHz, 0.9V].

4.4.2 Simulation Setup

MPSoC Details: We target a 64-core general-purpose MPSoC organised in an 8×8 2D-mesh. Architectural details for the MPSoC are included in Table 5.5. The processor is an in-order Tensilica Xtensa core with private L1 instruction and data cache. The processor stalls when there is a outstanding cache load miss. The frequency for the cores is fixed at 1GHz. For the baseline system, we assume that there are four memory controllers placed at the boundaries, as shown in Figure 4.3.

Topology	8×8 mesh
Processors	Tensilica in-order LX4 @ 1GHz
L1 I/D Caches	16KB, 2-way set associative, 16 byte line size
DRAM	4 memory controllers, 40 ns access latency
NoC VF Levels	[1GHz, 0.9V], [500 MHz, 0.81V]

Table 4.3: MPSoC Configuration.

Workload	**Applications**	**Min MPKI**	**Max MPKI**	**Avg MPKI**
W-1	mpeg2enc, sha, mpeg2dec, jpeg_dec	0.5	6	2.13
W-2	mpeg2enc, jpeg_dec, mpeg2dec, jpeg_enc	0.5	9	4.25
W-3	h264enc, sha, jpeg_enc, jpeg_dec	0.5	40	14
W-4	dijkstra, mpeg2enc, sha, jpeg_enc	0.5	16	6.63
W-5	dijkstra, h264enc, jpeg_enc, mpeg2enc	1	40	14
W-6	bzip2, lbm, dijkstra, sha	0.5	16	10.5
W-7	bzip2, lbm, soplex, sha	0.5	18	11
W-8	h264enc, dijkstra, soplex, mpeg2enc	1	40	18.75

Table 4.4: Details of application mixes with 16 copies of each application.

Benchmark Application: We use a diverse set of benchmark application from MediaBench and SPEC2006 packages to evaluate our proposed scheme. We create multiprogrammed workloads using the applications from these two packages as listed in Table 4.4. In a given workload, 16 copies of each application are executed. We also list the details of MPKIs for each workload in Table 4.4. For example workloads $W1$, $W2$ and $W4$ have lower MPKIs while other workloads have moderate to high MPKIs.

Simulator: We use an in-house cycle accurate simulator for our study. The simulation is carried out in two steps: First, the applications are executed on Xtensa

instruction set simulator and information about cache load and store memory traces is collected along with timing information. In the full system simulation, these memory traces are replayed in a closed loop simulator, which accurately models the NoC latency effect on application execution. Our simulator accurately models the low level hardware details for NoC VF switching, operation of LM and GM, and memory controller queuing delays.

VF Switching Setup: We explore designs with a dual voltage coarse-grain VF selection scheme. Two possible VF levels for NoC are [1GHz, 0.9V] and [500MHz, 0.81V]. We assume that it takes 10ns to switch between two voltages and switch on a router. We used a control interval of $50K$ clock cycles for the VF selection algorithm based on analysis presented by Chen et al. [118]. The threshold value for VF selection algorithm is set to 150 based on prior evaluation.

For power characterisation of routers, we passed the netlist generated by synthesis tool to Synopsys PrimeTime. The netlist generated for the [1GHz, 0.9V] router is evaluated at [1GHz, 0.9V] and [500MHz, 0.81V], while the netlist for [500MHz, 0.81V] router is evaluated at the design VF only. The energy values extracted from PrimeTime tool are fed into the simulator tor calculate the NoC energy.

Malleable NoC Configurations: We explored different configurations for *Malleable NoC*, as shown in Figure 4.5. In all configurations, the NoC contain routers designed for [1GHz, 0.9V]. In the baseline configuration, all the nodes contain routers for both [1GHz, 0.9V] and [500MHz, 0.81V]. We then explore designers where the percentage of nodes with [500MHz, 0.81V] routers is 75%, 50% and 25%.

4.4.3 Experiment Results and Discussion

4.4.3.1 Efficacy of Malleable NoC Architecture:

Figure 4.6 shows the normalised NoC energy delay product (EDP) of *Malleable NoC*. We also compare our proposed architecture with darkNoC architecture which was proposed in Chapter 3. All results are normalised to the EDP of *Baseline*

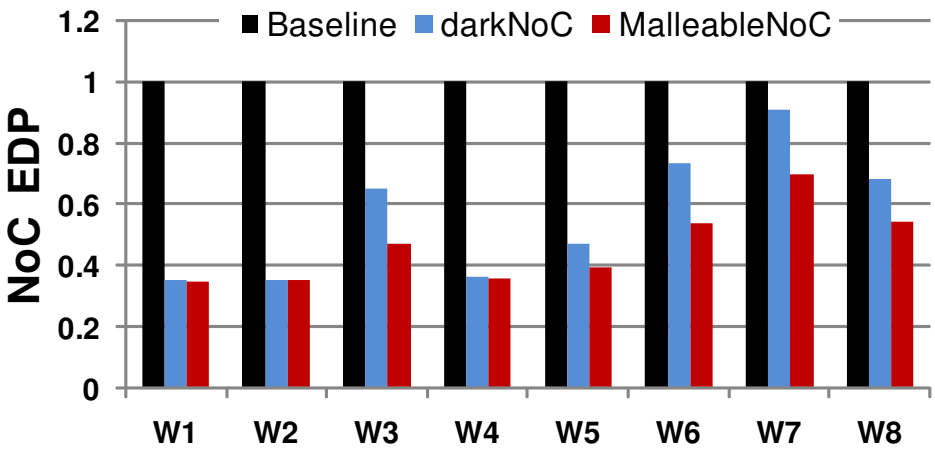


Figure 4.6: Efficacy of Malleable NoC

architecture, where no DVFS is used and NoC is run at maximum VF. Overall, *Malleable NoC* resulted in the lowest NoC EDP for all the workloads. For workloads $W1$, $W2$ and $W4$, *Malleable NoC* and darkNoC performed equally well. Referring to Table 4.4, these workloads have very low MPKIs and it is expected that the VF selection algorithm will operate all routers in the mesh at lower frequency. Therefore, per-router VF selection provided by *Malleable NoC* performs similarly to single VF domain switching used in darkNoC.

On the other hand, for workloads $W3$, $W5$, $W6$, $W7$ and $W8$, *Malleable NoC* provided considerably lower EDP than darkNoC. For example, for workload $W7$, *Malleable NoC* resulted in a relative EDP saving of 30%, while darkNoC only provided an EDP savings of 9%. Similarly, for workload $W6$, *Malleable NoC* resulted in a relative EDP saving of 46%, whereas darkNoC only provided an EDP saving of 26.4%. From information given in Table 4.4, it can observed that workloads $W3$, $W5$, $W6$, $W7$, $W8$ have very higher diversity in terms of MPKI. Therefore, we conclude that *Malleable NoC* can provide superior NoC EDP savings for chips that are expected to run applications with a mix of high and lower MPKIs.

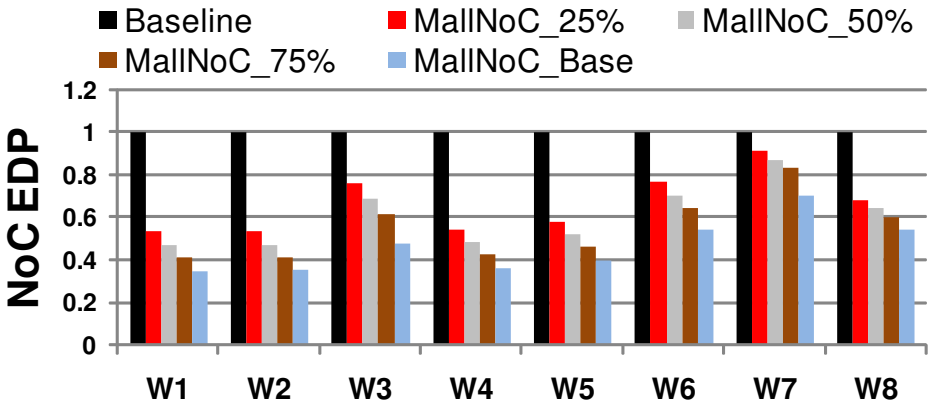


Figure 4.7: Analysis of different Configurations of Malleable NoC

4.4.3.2 Analysis of Malleable NoC Configurations:

In Figure 4.7 we report the normalized EDP of different configurations of *Malleable NoC*. The *MallNoC_Base* is the *Malleable NoC* configuration in which every node contains routers for [1GHz, 0.9V] and [500MHz, 0.81V]. As expected, *MallNoC_Base* produced the lowest EDP. For $W7$ and $W8$, the use of *MallNoC_25%* resulted in 30% and 25% increase in EDP over *MallNoC_Base*, respectively. Whereas for both $W1$ and $W2$, the use of *MallNoC_25%* resulted in a 52% increase in EDP over *MallNoC_Base*. From this result, we can deduce that workloads with diverse MPKIs (such as $W7$ and $W8$) can potentially use a partial *Malleable NoC* configuration, thus saving dark silicon area and using it for other lower power components. The reason is that in workloads with high and low MPKI applications, it is expected that routers on the outer periphery of the mesh will be operated at high VF levels, and therefore such applications are the least affected by the absence of low VF routers on the mesh outer periphery.

4.4.3.3 Importance of Mapping:

The graph in Figure 4.8 shows the importance of using an intelligent application-to-core mapping scheme [9]. We randomly mapped the applications to cores and then used the *Malleable NoC* for the evaluation. For workloads $W3$, $W6$, $W7$ and

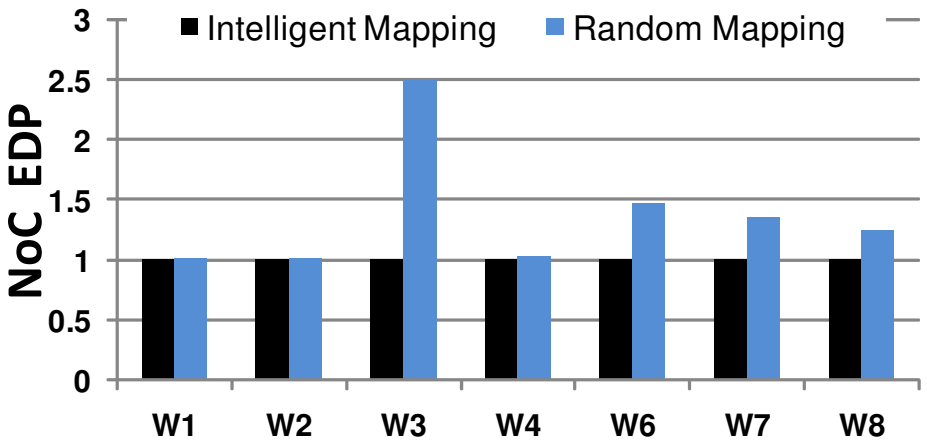


Figure 4.8: Effect of Mapping on Malleable NoC

$W8$ the EDP with respect to intelligent mapping increased by 150%, 48%, 35% and 25%, respectively. This shows that it is important to use intelligent mapping with *Malleable NoC* to get the best results for workloads containing applications with heterogenous MPKI. However, for workloads with lower MPKIs, i.e., $W1$ $W2$ and $W4$, the EDP is reduced only negligibly.

4.5 Conclusion

In this chapter we presented a low-power NoC fabric called *Malleable NoC* which contain multiple VF optimised routers for each node. At runtime, depending on the workload, a router from each mesh node is combined to form a NoC which can adapt to heterogenous application requirements. We show that for a variety of multi program benchmarks executing on *Malleable NoC*, Energy Delay product (EDP) can be reduced by up to 46% for widely differing workloads. *Malleable NoC* is superior to *darkNoC* for executing heterogeneous applications and for design scenarios where NoC has to compete with other design flows for dark silicon and hence VF optimised routers cannot be replicated at every node

Chapter 5

SuperNet: Multimode Interconnect Architecture for Manycore Chips

5.1 Introduction

The cost of designing a SoC has increased dramatically with shrinking node sizes as the laws of physics are challenged [171–173]. The cost trend of chip design is shown in Figure 5.1. A significant amount of financial and human resources are required to design and verify chips designed in smaller nodes. Therefore, according to the laws of economy, the only way to achieve profitability and beat the initial design cost is to increase the number of units sold. To increase the sales volume, a chip has to target a wider set of applications. This means the SoC has to fulfill the design requirements for different usage scenarios. For example, designing a multimedia SoC for a battery-powered smart phone requires a special emphasis on low energy operation. On the other hand, using the same SoC for a home entertainment system requires special attention to high performance. A similar SoC used for engine management or a braking system in a car has to operate reliably throughout the car's lifetime. This

poses a serious challenge to designers who must account for various operational scenarios in a single chip.

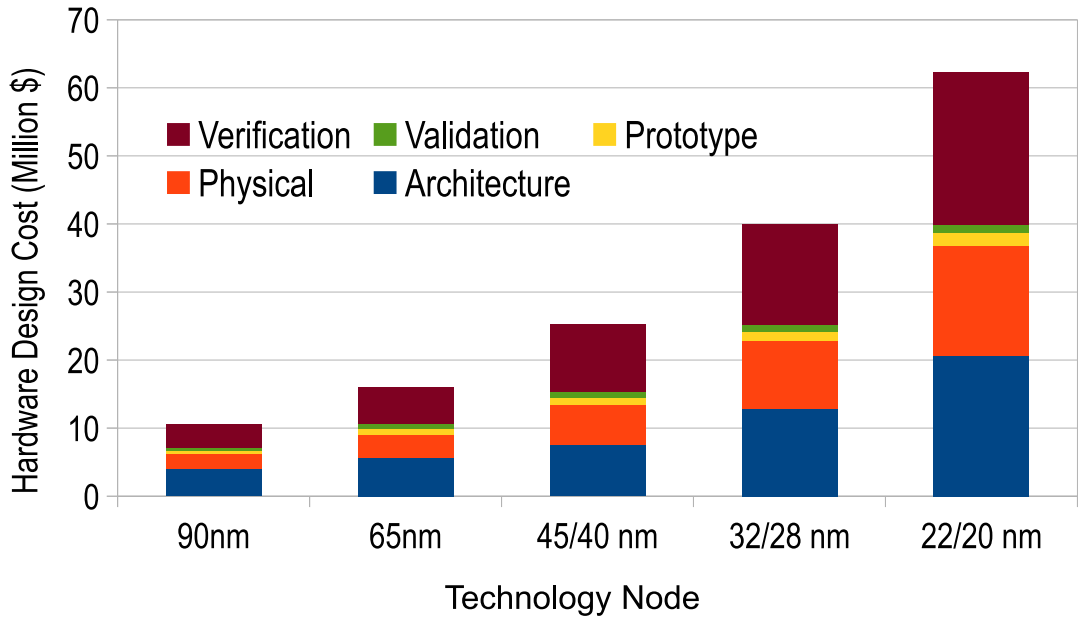


Figure 5.1: Hardware Design Cost Trend (Source: Semico Research)

It is important to reconsider the on-chip interconnect in the context of such complex multipurpose SoC designs. Packet-switched Network-on-Chips (NoCs) are seen as a scalable interconnect to support the growing communication demand of SoC components. Like other on-chip components, it is essential to study the effects of downscaling trends on NoCs. Although NoCs provide a scalable communication backbone for tens of on-chip components, they also consume a considerable share of chip power (e.g., up to 18% and 33% in Intel SCC [51] and RAW [160] architectures, respectively). Research studies show that leakage power is the major share of total NoC power, and this is expected to increase for smaller nodes [170]. On the other hand, deep transistor integration adversely affects the reliability of digital circuits. A soft error in NoC can have adverse effects, such as partial disconnection of SoC component, erroneous execution of application or system level deadlock [174]. To maintain high performance, NoCs are designed either for worst-case or average case

scenarios. However, a design approach for worst-case delay can lead to wasted energy for less demanding applications [11].

The above mentioned design problem becomes more complex when NoCs must be designed for multi-purpose SoCs. Traditionally NoCs are designed with a specific goal of either low-power operation, high performance [11] or high reliability [175]. For example, a NoC for a high performance chip can be designed with a wider channel width and higher clock frequency to fulfil the requirement of high bandwidth and low latency [11]. Such a high performance NoC will be impractical for low power battery powered devices. Similarly, low-power NoC can be designed with narrow channels and lowest possible voltage guard-bands. However this NoC will neither be good enough for application with high bandwidth required to support high performance applications, nor operate reliably under harsh operational conditions. Therefore, to our best of our knowledge, there is no single design proposal that covers all three possible operational requirements discussed earlier.

A very naive solution to the above stated design problem is to integrate a NoC for each possible design goal and use those NoCs depending on the usage scenario. Such designs are not a suitable choice for two reasons. The first reason is that having multiple mutually exclusive NoCs will carry a higher silicon cost. Despite the fact that a large amount of *dark silicon* is available in future chips due to power budget limitations, NoCs still need to compete for this *dark silicon* [131] area with other on chip components such as application accelerators etc. The second reason is that every on-chip component has to be extensively verified before fabrication [171] and therefore naively adding more NoCs can potentially increase the chip design cost. In this chapter we show that in place of such inefficient NoC architecture, we can design a system with the minimal number of NoCs that can be used to achieve the same goal through the adroit reuse of one set of NoCs for different operational modes.

Our solution to the above stated problem is the SuperNet NoC architecture. The architecture consists of parallel architecturally homogenous, but voltage-frequency (VF) optimized heterogeneous NoCs. For energy efficient mode, we leverage the two NoCs for dynamic voltage and frequency scaling (DVFS) by switching on only one NoC depending on application requirements or static configuration. In performance mode, we classify the network packets under the category of critical and non-critical packets, and steer the packets accordingly into either low VF or high VF NoC. In the reliability mode, the two NoCs work in dual lock step mode where one NoC carries the actual data while the other NoC carries the error correction codes. This scheme improves the reliability by mitigating the effects of *soft errors* in the control and data path.

We make the following contributions in this chapter:

- We introduce a novel multimode NoC SuperNet for future multipurpose SoCs. SuperNet uses a simple control interface for switching between three different modes, i.e, low power mode, high performance mode and reliability mode.

- We describe the circuit design techniques and architectural innovation used to realize the SuperNet architecture. The process of switching between modes is completely autonomous with minimal software intervention.

- Using real applications and a cycle-accurate simulation setup, we explore the efficacy of different operational modes of SuperNet.

In Section 5.2 we provide a brief background of previous NoC proposals and compare them with the SuperNet architecture. Section 5.3 provides details about the SuperNet architecture and the three operational modes. In Section 5.4 we present our results from experiments with SuperNet architecture. We present conclusions in Section 5.5.

5.2 Background and Related Work

Muliple NoCs: There are several commercial and academic chips that leverage multiple NoCs. Tilera's iMesh architecture [107] uses five parallel NoCs for different class of traffic, i.e, cache coherence messages, memory controller access, I/O port access, and user space message passing. TRIPS [163] architecture uses multiple networks for operand and data forwarding. The reason for using multiple NoCs in TRIPS and Tilera iMesh architecture is to provide isolation between different classes of traffic. Balfour and Dally [176] showed that adding parallel NoCs can potentially reduce the network latency and improves the bandwidth. Mishra et al. [11] extended the idea by using two heterogeneous network, where one network is used for bandwidth sensitive applications and another network is used for latency sensitive applications. In contrast, in performance mode for SuperNet, we divide the traffic from all the applications based on packet type instead of the application to which the traffic belongs. In addition to the performance improvement, SuperNet can also be used to provide reliability or energy efficiency by using the same set of NoCs.

System Level Power Saving Techniques: Over the years various system level power management techniques have been introduced. DVFS for on-chip networks has been proposed at various granularities — such as link-level [113], router-level [114, 115] and region-level [118, 169]. However the advantage of the DVFS scheme has been reducing due to NoC's high leakage power in smaller nodes and diminishing room for reducing supply voltages [17]. To improve the efficiency of DVFS schemes, in Chapter 3, we proposed integrating multiple NoCs, where each NoC is optimised using multi-vt circuit technique. At runtime, along with changing VF levels, a different NoC is activated. We use a similar technique for our energy-efficient modes. However we also use these multiple NoCs to improve system performance and reliability . Das et al. [128] proposed integrating four architecturally homogenous NoCs for the purpose of energy efficiency. Depending on network traffic, a

subset of these NoC are activated at runtime while unused NoC are power gated. Both Das et al. [128] and Bokhari et al. [170] did not address the issue of reliability and performance in their architecture.

Fault Tolerance in NoC: Many researchers have studied the topic of soft error detection and correction for NoCs[1]. The soft errors are caused by random logic flips on the input/output of a logic gate. This leads to temporary glitches in a NoC router's data path and/or control path [175]. BulletProof router [177] uses a Cyclic Redundancy Check (CRC) to detect and correct the soft errors in data path, and selective triple modular redundancy (TMR) for detecting faults in the control paths of the routers. Prodromou et al. [174] proposed a scheme based on hardware invariance checking to detect faults in router control path. Park et al. [178] explored the use of hop-by-hop error detection and correction by using Single Error Correction - Double Error Detection (SECDED) codes. In the case of errors which were not correctable, flits are retransmitted from downstream routers. Commercial NoC vendor, Arteris, has also introduced an error resilience solution called FlexNoC Resilience Package [179] that combines end-to-end error data correction and port checking. Murali et al. [180] analyzed different error detection and correction schemes and concluded that end-to-end fault detection,correction and retransmission based on time-out is a good tradeoff between performance,energy and area overhead. None of the techniques discussed above studied the soft error resilience in the context of reusing the existing multiple NoCs. In contrast to the previous solutions, we use the additional NoC to transmit a strong byte-based SECDED code for end-to-end error detection and correction, and use the control path of the two NoCs for lock-step dual modulo redundancy (DMR) protection in the control path.

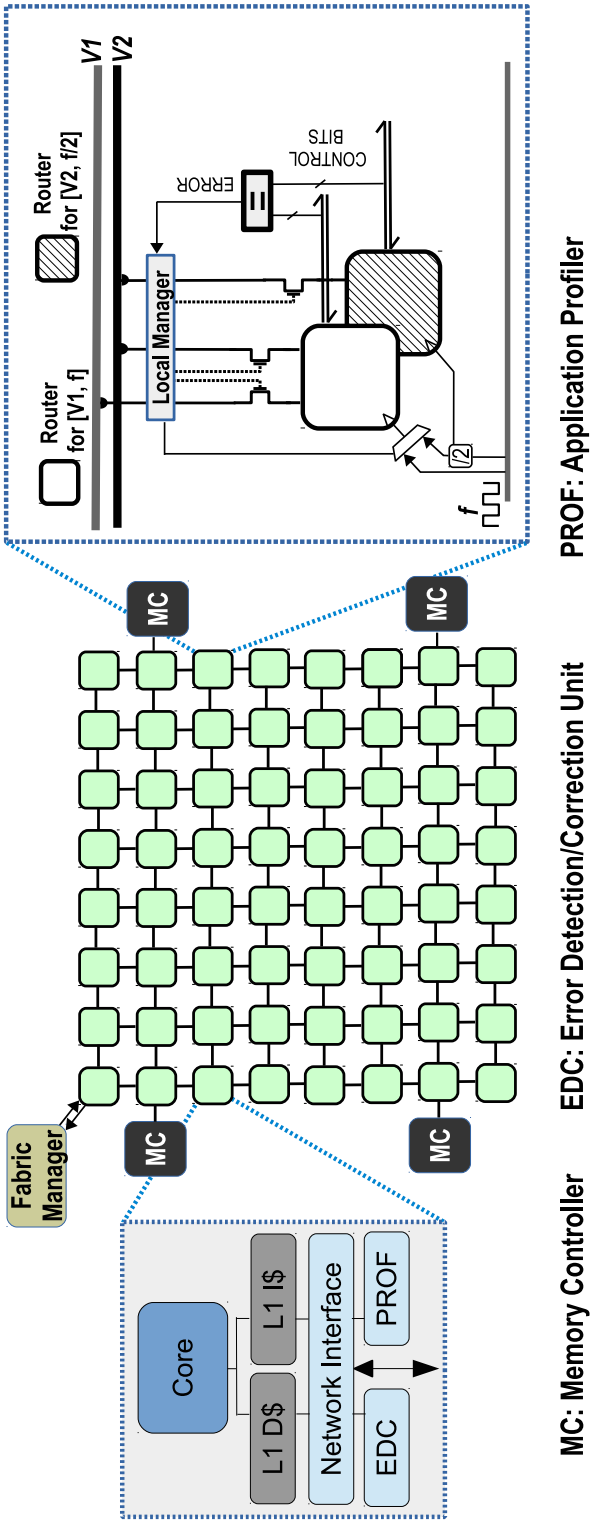


Figure 5.2: Overview of MPSoC with SuperNet Interconnect

5.3 SuperNet Architecture

5.3.1 Architecture

Our target architecture is a general-purpose tiled multi-core chip as shown in Figure 5.2, which closely matches commercial SoC chips such as Intel Single Chip Cloud Computer (SCC) [51] architecture. The processing core is a simple in-order core with private L1 Data and Instruction caches. Cache load and store requests are injected into the NoC by cores. Memory controllers are placed at the border routers as shown in Figure 5.2 to serve cache load and store requests. We target a $N \times N$ Mesh topologybased NoC architecture in this Chapter as it is commonly used by commercial and academic multi-core chips. Each core runs a different application, therefore at a given time up to N^2 applications are executed in parallel.

5.3.1.1 Multiple VF Optimized NoCs

SuperNet architecture consists of two parallel NoCs that are optimized for different VF levels using *multi-vt optimisation* [170]. Commercial fabrication foundries such as TSMC and Global Foundaries provide cell libraries with various gate threshold voltages (Vt). These library packages contain cells with normal Vt (NVt), Low Vt (LVt), and High Vt (HVt). The availability of heterogeneous Vt provides an opportunity for designers to optimize the energy-delay characteristics of circuits below a given synthesis latency constraint. CAD tools exploit the energy-delay characteristics of the cells by inserting low-vt cells (high leakage, faster switching) on circuit logic paths with negative slack to meet the latency constraint and replacing normal cells with high-vt cells (low leakage, slower switching) on paths with positive slack to save energy.

We assume that two voltage supplies, $V1$ and $V2$ are available, where $V1 > V2$. These voltages are associated with two frequencies f and $f/2$. At each node

[1] We are limiting our discussion to mitigating soft errors. An overview of fault tolerance schemes for NoCs is presented by Radetzki et al. [175]

we integrate two parallel routers as shown in Figure 5.2. Although both routers are architecturally homogeneous, they are optimised for a VF level using *multi-vt optimisation*. One of the routers is designed for VF level $[V1,f]$ whereas the other router is designed for $[V2,f/2]$. Using these routers, two parallel NoCs are realized as follows:

- **NoC A:** This NoC consists of routers designed for $[V1,f]$. *NoC A* can be operated at either $[V1,f]$ or $[V2,f/2]$ depending on the operating mode.
- **NoC B:** This NoC consists of routers designed for $[V2,f/2]$. *NoC B* can only be operated at $[V2,f/2]$.

5.3.1.2 Description of NoC Components

Fabric Manager: Fabric Manager (*FM*) is responsible for managing all the NoC components in the MPSoC. The *FM* uses the NoC channels for communication with Local Managers and reads the response through a 3-wire AND network as shown in Figure 5.3(a). The *FM* can be implemented as a specialized hardware or it can be a part of existing chip management firmware. *FM* autonomously co-ordinates with Local Managers in the NoC to implement different operational modes. This eases the burden of writing software routines to manage resources.

Local Manager: Local Managers (*LMs*) are responsible to managie the routers at mesh node and to forward information from the local mesh node to *FM*. *LM* drives the power gating enable and disable signals for each router. *LM* also enables and disables the traffic injection from the local core's port. *LM* monitors the buffer empty condition, application profiler status and error status from the lock-step comparator (more on this in Sections 5.3.3, 5.3.4 and 5.3.5). This information is communicated back to the *FM* through a simple 3-wire AND network as shown in Figure 5.3(a). *LMs* receive their commands from *FM* through the NoC for setting up the operational mode.

Application Profiler Unit: Each core contains an Application Profiler Unit.

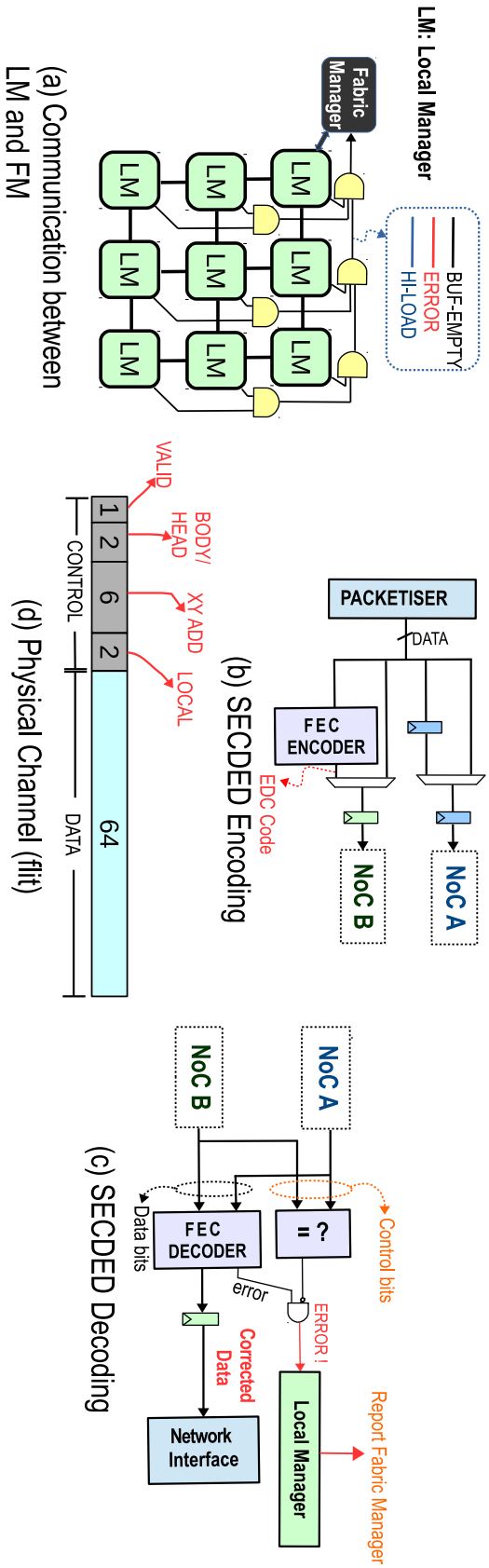


Figure 5.3: Details of SuperNet Components

This profiler unit (marked PROF in Figure 5.2) keeps track of vital information on application behavior. Each profiler unit is augmented with two saturating counters, an instruction counter and a L1 cache miss counter. The profiler unit resets the counters after every preset control interval and saves the value from the previous interval. The counter information is used by *FM* to implement either the Energy Efficient Mode or the Performance Mode.

Error Detection and Correction Unit: Every Network Interface (NI) is augmented with an Error Detection and Correction Unit (marked EDC in Figure 5.2) for use in the Reliability Mode. The EDC unit is responsible for encoding the outgoing data from NI with a forward error correction code (FEC) (Figure 5.3(b)), and detecting and correcting any errors in the incoming data (Figure 5.3(c)). We use SECDED codes for each 8-bit data, which results in a 5-bit codeword. In our architecture each flit contains 64 bits of data as shown in Figure 5.3(d). Therefore for every 64-bit flit, we generate a 40-bit codeword. Similarly on the receiver side, a decoder checks and tries to correct errors in the flit data. Therefore, through EDC unit, we can correct up to eight single bit errors (as long as they occur in separate data bytes) and detect up to 16 bit errors (maximum of two error in each byte). To save power, these units can be power gated by *LM* in case the Reliability Mode is not activated.

Equivalence Checker: Equivalence Checker (blocks with '=' sign in Figure 5.2 and Figure 5.3(a)) are used for implementing the Reliability Mode. The checker circuit asserts that the *control bits* (Figure 5.3(d)) of the two NoC channels are equal. These checkers are placed at two locations: (1) each of the five outgoing router channels, (2) incoming channel of each Error Detection and Correction Unit. In case the control parts of the channels are not equal, the checker unit raises an error signal which is read by the *LM*.

5.3.2 Mode Selection

When the chip is switched on, *FM* by default initializes the SuperNet in the normal mode (i.e, only *NoC A* is powered on with $[V1,f]$). Depending on various factors such as power budget and reliability requirement, the chip user can decide to activate a certain SuperNet mode. Therefore at runtime the user can execute the chip's firmware or OS system call to write the mode change command to the *FM*'s setting register. Once this happens, the *FM* autonomously implements the desired operational mode.

5.3.3 Energy Efficient Mode

We base our Energy Efficient Mode on the fact that not all applications need a low latency NoC. One important metric (to measure the dependence of applications on NoC) is cache misses per kilo instructions (MPKI) [11]. Applications with higher MPKI values spend much time waiting for the cache load miss to be served from the memory controller through the NoC. The graph for MPKIs for two applications *h*264*enc* and *mpeg*2*enc* are shown in Figure 5.4. In our experiments we found that the MPKI value for *h*264*enc* is 40× the MPKI of *mpeg*2*enc*. Therefore, for an MPSoC with *h*264*enc* being executed, NoC has to be clocked at a higher frequency. On the other hand, for low MPKI applications such as *mpeg*2*enc*, NoC can be

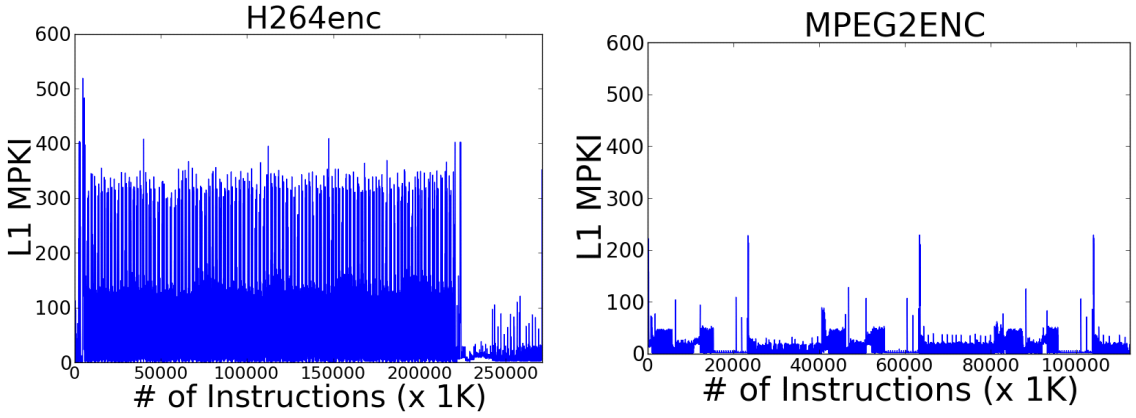


Figure 5.4: MPKI for *h*264*enc* and *mpeg*2*enc*

clocked at a lower frequency to save energy. Furthermore, the MPKI values can also fluctuate, depending on the application behaviour. Therefore, depending on an application's current MPKI, the NoC can be clocked at different frequencies.

VF Selection Scheme: SuperNet implements a coarse grain DVFS method with two VF levels. When $[V1,f]$ is used, *NoC A* is operational and *NoC B* is power gated. Whereas, when $[V2,f/2]$ is used, *NoC B* is used and *NoC A* is switched off. Bokhari et al. [170] showed that using NoC that is optimised for a specific VF level is more energy efficient than using a scaled VF on a NoC that has been designed for a higher VF level. MPKIs are calculated for each application using instruction and cache miss counters in the profiling unit. If the MPKI of the application falls below a pre-set *threshold* value, the profiling unit de-asserts the HI-LOAD signal, which is sensed by *LM* and broadcasted to *FM*. If all applications in the system de-assert the signal, *FM* decides to lower the NoC frequency to $f/2$. The value of threshold can be chosen to mark a trade-off between performance and power. On the other hand, if any application in the system asserts HI-LOAD signal (Figure 5.3(a)), the *FM* switches back to higher frequency (f). Another possible mode that can be used by *FM* is to keep the NoC running at low frequency irrespective of application characteristics (i.e, static VF selection) in case of low power requirements, or if the performance mode has been executed for too long and the chip is too hot.

Mode Setup: To start the Energy Efficient Mode, the *FM* sends control flits to each *LM* to stop any further transaction in the NoC and informs the *LM* about initiation of the mode along with the *threshold value* to be used. *FM* then waits for all existing transactions to finish by monitoring the global *BUFF-EMPTY* signal (Figure 5.3(a)). At the start of the mode, *LM*s switch on the router for *NoC A*.

VF Switching Procedure: Whenever the *FM* decides to switch between VF levels, the currently active NoC has to be switched off after ensuring that there are no in-flight packets in the system. *FM* starts the procedure by sending a *Stop NI* command to all *LMs* to stop any further injection of packets. The global *BUFF-EMPTY* signal (Figure 5.3(a)) is raised when there are no more packets in the NoC.

Then *FM* sends the next batch of control flits to *LM* to activate the required NoC and power gate the existing NoC. *FM* sends the final batch of control flits to LMs to resume normal operation. This concludes the VF switch-over process.

5.3.4 Perfomance Mode

The Performance Mode for SuperNet is based on the fact that not all packets in the NoC are critical to the performance of the application. We have classified the possible packet types in Table 5.1. The *LoadRequest* packet is generated by the core in case a memory load instruction causes a cache miss. The memory controller replies to the request by sending the required cache line through *LoadRequestReply* packet. In case a cache line is evicted, the core generates a *StoreRequest* packet for the memory controller. In response to the *StoreRequest*, the memory controller sends the *StoreRequestAck* packet as an acknowledgement. We assume an MPSoC with an in-order core and therefore the core stalls in case of a cache load miss. Thus, it is critical to deliver the *LoadRequest* and *LoadRequestReply* as quickly as possible. The process of sending the evicted cache line to the memory controller happens in parallel to the execution, and therefore the *StoreRequest* and *StoreRequestAck* packets are not critical for application performance. When only one NoC is used, all packets compete for shared channel resources. Therefore due to contention, packets delivery can be delayed. This delay can possibly affect the system performance if delay for critical packets increase. The situation becomes even worse when applications have high MPKIs.

Packet Steering Based on Criticality: We leverage the two NoCs available in

Type	Length	Critical?
Load Request	$1flit$	yes
Load Request Reply	$(cache\ line\ size)/8 + 1flits$	yes
Store Request	$(cache\ line\ size)/8 + 1flits$	no
Store Request Ack	$1flit$	no

Table 5.1: Packet Classification

SuperNet to implement the performance mode. *FM* switch on both *NoC A*(with [$V1$,f]) and *NoC B* at the same time. At runtime, the cores inject *LoadRequest* packets in the *NoC A* and *StoreRequest* in the *NoC B*. The memory controller injects the *LoadRequestReply* packets in the *NoC A* and *StoreRequestAck* packet in the *NoC B*. The intuition behind the steering scheme is to reduce the contention between critical and non-critical packets by physical separation. However, this scheme is implemented at the cost of increased NoC power compared to normal operation because two NoCs are used instead of one NoC. Moreover the performance mode is only useful for applications where network load is high due to a high application MPKI. Therefore, *FM* can enable or disable the performance mode based on the power budget and application characteristics.

Mode Setup: To start the Performance Mode, the *FM* sends control flits to each *LM* to stop any further transaction in the NoC and informs the *LM* about initiation of the mode. *FM* then waits for all the existing transactions to finish by monitoring the global *BUFF-EMPTY* signal (Figure 5.3(a)). In parallel, *LM* also switches on *NoC A* and *NoC B* routers in case one of them was not already switched on. *FM* sends control flits to each *LM* in response to which NI is enabled and configured to steer traffic into a specific NoC based on traffic classification.

5.3.5 Reliability Mode

The Reliability Mode for SuperNet enables the detection and correction of soft errors in the NoC. The scheme can provide protection against soft errors in both the data path and control path. In this mode both NoCs are used at the same time. Reliability Mode uses a *NoC A* and *NoC B* in parallel by operating them at [$V2$, $f/2$].

Data Path Protection: The router's data path is protected through the use of strong single error correction and double error detection codes (SECDED) in a forward error correction mode (FEC). The Error Detection and Correction Unit

(EDC) is used to generate a (13,8) BCH code (8-bit data generating 5-bit code). In SuperNet we only encode the data at the time of packet injection and only try to check and correct the data at the receiver node. A more error resilient approach is to perform error encoding and decoding at each router in the path of the packet. However this scheme would increase the energy cost due to additional hardware at each router. Moreover, per-router encoding and decoding will increase the router's pipeline stages, which will eventually result in an increase in network latency and hence performance penalty. The actual packets are injected in *NoC A* and the error correction codes are injected in *NoC B* simultaneously as shown in Figure 5.3(b).

Control Path Protection: In SuperNet, we employ the idea of dual modular redundancy (DMR) and lock step processing to protect the router control path against the soft errors. Both NoCs are architecturally homogenous and are operated at the same frequency. Furthermore, during the setup of Reliability Mode, switch allocators for all the routers are reset to a known state. Therefore, routers from different NoCs at a given mesh node are expected to generate the same output. Each NoC channel consists of control bits and data bits as shown in Figure 5.3(d). We perform equivalence check on the control bit output on outgoing router channels in a given direction as shown in Figure 5.2 and Figure 5.3(c). Any soft error in the control circuit in either of the routers will result in in-correct data written on a port or no output at all. For example, an error in routing logic will route the flit to the wrong port. If the equivalence check fails, an error is raised for *LM*.

Mode Setup: To start reliability mode, the *FM* sends control flits to each *LM* to stop any further transaction in the NoC. The *FM* then waits for all the existing transactions to finish by monitoring the global *BUFF-EMPTY* signal (Figure 5.3(a)). Then, *FM* sends control flits to *LM* to switch on the routers for *NoC A*(at VF $[V2, f/2]$) and *NoC B*. The last set of control flits are sent to *LM* to reset the router switch allocators to a known state and continue normal operation.

Runtime Fault Recovery: At runtime, whenever a non-correctable error occurs in the data path or lock step router execution fails, an error is raised by the

LM for *FM* as shown in Figure 5.3(c). In response to the error, *FM* sends control flits to each *LM* to stop any new transaction from the NI. Once the network is free of any traffic, *FM* sends control flits to *LM*s to reset the switch arbiter to a known state. *FM* sends the last set of control flits to *LM*s to continue normal operation with these conditions:

- If the core has sent a *LoadRequest* packet and the memory controller has not sent a reply through *LoadRequestReply* packet, the local NI again sends the *LoadRequest* (Table 5.1).

- If the core has sent a *StoreRequest* packet and the memory controller has not sent a reply through *StoreRequestAck* packet, the *StoreRequest* packet is sent again.

Performance Overhead: The main performance overhead of SuperNet's Reliability Mode is due to the fact that NoC is operated at a reduced frequency of $f/2$ which causes an increase in the network latency. The performance penalty of increased NoC latency will be more significant for applications that have high MPKIs. Furthermore, a performance overhead is incurred when there is a non-correctable fault (data path or control) which requires retransmission from source.

5.4 Experiment and Results

5.4.1 Experiment Setup

NoC Synthesis: For experiments, we use a two stage pipelined wormhole switched router with 64-bit data channel width. Architectural details of the NoC router used in our study are given in Table 5.2. In the first stage, flits travel on links and get written in the input buffer. The second stage consists of route computation, switch arbitration and switch transversal. The router description is written at the

Topology	8×8 mesh
Router Architecture	5 port router (4 neighbors + 1 local), 2 stage pipeline [LT+BW, RC+SA+ST]
Input Buffer	8 flit deep (no virtual channel)
Routing	Dimension Order XY
Link Flow Control	Wormhole switching with On/Off flow control
Link Width	64 bit data
Switch Arbiter	Matrix arbiter

Table 5.2: NoC architectural details.

Frequency	1 GHz	500 MHz
Voltage	0.9 V	0.81 V
Area [μm^2]	33041	28976
HVt Cells [%]	45	82
NVt Cells [%]	7	6
LVt Cells [%]	48	12

Table 5.3: Synthesis results for a single router using Multi-Vt optimisation.

	ECC Decode	ECC Encode	Equivalence
Area [μm^2]	1814	760	46

Table 5.4: Synthesis results for Error Detection & Correction Units

RTL (Verilog). We synthesised the RTL using Synopsys Design Compiler version *H-2013.03-SP4*. We used commercial TSMC 45nm libraries that are characterised for Low Vt(LVt), Normal Vt(NVt) and High Vt(HVt). We enable leakage and dynamic power optimisation which automatically invokes multi-Vt optimisation. We target two VF levels for the router design for experiments, i.e, [1GHz, 0.9V] and [500MHz, 0.81V]. We report the synthesis results for NoC in Table 5.3. We also synthesized the error detection and correction hardware, results of which are shown in Table 5.4.

MPSoC Details: We target a 64-core general-purpose MPSoC organized in a 8×8 2D-mesh. Architectural details for the MPSoC are included in Table 5.5. The processors are an in-order Tensilica Xtensa core with private L1 instruction and data

Topology	8×8 mesh
Processors	Tensilica in-order LX4 @ 1GHz
L1 I/D Caches	16KB, 2-way set associative, 16 byte line size
Memory Controller	4 memory controllers with perfect L2 cache
NoC VF Levels	[1GHz, 0.9V], [500 MHz, 0.81V]

Table 5.5: MPSoC Configuration

caches. The core stalls when there is an outstanding cache load miss. The frequency for the cores is fixed at 1GHz. For the baseline system, we assume that there are four memory controllers placed at the boundaries as shown in Figure 5.2. To stress the NoC, we assume that the memory controllers are augmented with perfect L2 caches. This technique has been used previously for testing NoC architectures [130, 181].

Benchmark Applications: We use a diverse set of benchmark applications from MediaBench and SPEC2006 packages to evaluate our proposed scheme. We created multi-programmed workloads using the applications from these two suites by running a copy of the selected application on each core. Each core executes 50 million instructions on average for experimentation.

Simulator: We use an in-house cycle-accurate simulator for our study. The simulation is carried out in two steps: First, applications are executed on the Xtensa instruction set simulator which generates time-stamped cache load/store miss traces. In the full system simulation, these memory traces are replayed in a closed loop simulator which accurately models the NoC latency effect on application execution. Our simulator accurately models the low-level hardware details for NoC VF switching, operation of LM and FM, and memory controller queuing delays.

VF Switching Setup: We explore designs with dual voltage coarse-grain VF selection scheme. Two possible VF levels for NoC are [1GHz, 0.9V] and [500MHz, 0.81V]. For Energy Efficient Mode, we used a control interval of $50K$ clock cycles for the VF selection algorithm based on analysis presented by Chen et al. [118]. For power characterization of routers, we passed the netlist generated by synthesis

tool to Synopsys PrimeTime. The netlist generated for [1GHz, 0.9V] router was evaluated at [1GHz, 0.9V] and [500MHz, 0.81V], while the netlist for [500MHz, 0.81V] router was evaluated at the design VF only. The energy values extracted from PrimeTime tool are fed into the simulator to calculate the NoC energy.

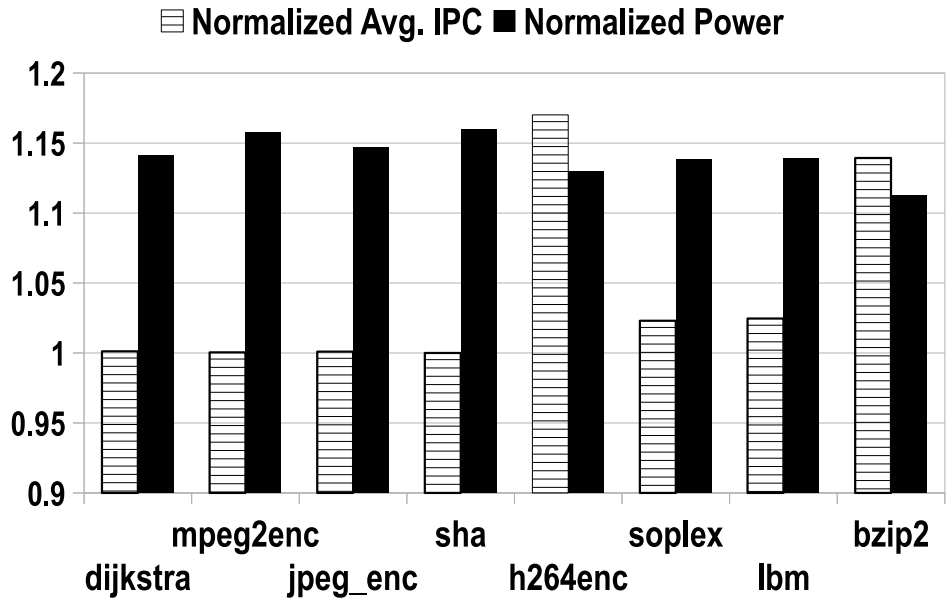


Figure 5.5: Results for Performance Mode

5.4.2 Experiment Results

Performance Mode: The results for experiments with the performance mode are shown in Figure 5.5. We report the values for *NoC Power* and *Average IPC*. All the results are normalized to a system where only [1GHz, 0.9V] NoC is used. The performance mode for SuperNet is clearly useful for applications with high MPKI. For the application *h*264*enc*, the average IPC is improved by 17%. Similarly for another application *bzip*2, the average IPC is improved by 14%. This improvement comes at a cost of increase in NoC power by 13% and 11% for *h*264*enc* and *bzip*2, respectively. A small improvement of 3% is observed for applications *lbm* and *soplex* at a cost of 14% increase in NoC power. No improvement was observed for

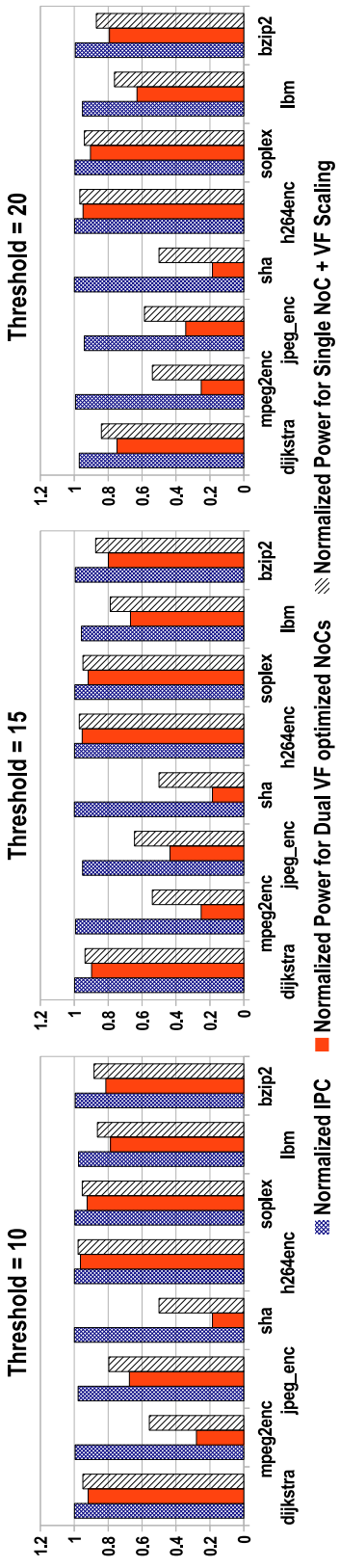


Figure 5.6: Results for Energy Efficient Mode with Different Thresholds

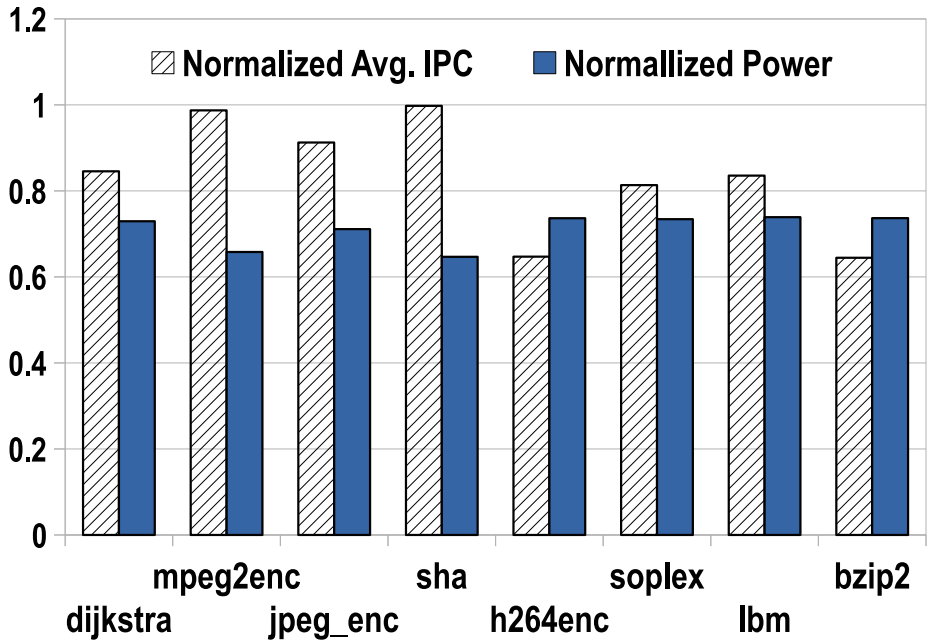


Figure 5.7: Results for Reliability Mode

other applications (i.e, *mpeg2enc*, *dijkstra*, *jpeg_enc* and *sha*) as the MPKIs for these applications are moderate to low.

Based on these results we advocate the judicious use of SuperNet's Performance Mode for applications with very high MPKIs. Therefore *FM* can be enhanced to use the already existing application profiling information to selectively switch on the performance mode for applications with high MPKIs.

Energy Efficient Mode: The results for experiments with the energy efficient mode are shown in Figure 5.6. We report the values for the *NoC Power* and the *Average IPC* for all the cores. All results are normalised to a system where only [1GHz, 0.9V] NoC is used with a fixed VF level. For comparison, we also report results for architecture where a single VF-optimised NoC is used with DVFS. From Figure 5.6, it is evident that using multiple multi-vt optimised NoCs provide better power savings than using DVFS with a single NoC. We performed a parameter sweep on the values of *threshold* for the VF selection routine. From the results in Figure 5.6, a *threshold* value of 20 gives the best results across all benchmarks.

For example, by using energy efficient mode of SuperNet we are able to save NoC power by 75% for *mpeg2enc*, 65% for *jpeg_enc*, and 81.5% for *sha* applications. The power saving comes at a cost of a small decrease in IPC by 0.9%, 6% and 0.1% for *mpeg2enc*, *jpeg_enc*, and *sha* applications, respectively. This is because NoC is sometimes running at a lower frequency, which increases the application execution time. Note that these three applications have low MPKIs. Even with applications with high MPKIs such as *dijkstra*, *lbm* and *bzip2*, the NoC power is saved by 25%, 37% and 21%, respectively. This NoC power saving comes at a cost of reduction of IPC by 3%, 4% and 0.6% for *dijkstra*, *lbm* and *bzip2* applications, respectively. A small saving in NoC power of 5% and 9% is observed for applications *h264enc* and *soplex* application, respectively,

It is very important to select an appropriate value for *threshold*. We can observe from Figure 5.6 that choosing a low *threshold* value results in low power savings, whereas choosing a high *threshold* value results in low power savings and considerable degradation in IPC. Therefore, it is expected that designers will profile the applications of interest to choose a suitable *threshold* value.

Reliability Mode: The results for experiments with the reliability mode are shown in Figure 5.7. We report the values for *NoC Power* and *Average IPC*. All results are normalized to a system where only [1GHz, 0.9V] NoC is used without activating any reliability features. Running SuperNet in reliability mode can cause some degradation in chip performance for applications with moderate to high MPKIs. For example, for applications *mpeg2enc*, *jpeg_enc* and *sha*, the IPC is reduced by 1.3%, 9% and 0.3%, respectively. On the other hand, the performance of high MPKI applications *h264* and *bzip2* is strongly affected by the reliability mode, as the IPC is dropped by 35% for both applications. The reason for the drop in performance is the fact that the NoC is operated at a lower frequency of $500MHz$. Overall, the relative NoC power is low due to two reasons: (1) reduced VF level, and (2) longer execution times.

5.4.3 Discussion

In our experiments, the SuperNet architecture is operated exclusively in one of three possible modes. One way to reduce the performance overheard of the reliability mode would be to virtually partition the SuperNet into reliable and non-reliable regions [179] and map only applications with high susceptibility [182] to errors in the reliable NoC region. Similarly, regions where reliability mode is not activated can be operated in either the performance mode or the energy efficient mode depending on power budget and application characteristics.

5.5 Conclusion

In this chapter we presented a novel multimode on-chip interconnect, SuperNet, for many core SoC chips. The design of SuperNet is based on the motivation that future SoC chips will be used in various scenarios with different performance, power and reliability requirements. SuperNet contains two parallel NoCs that can be configured at runtime to operate in either *Reliability Mode*, *Performance Mode* or *Energy Efficient Mode*. To the best of our knowledge, there is no pervious proposal for NoC that achieved such multimode operational diversity.

Chapter 6

Communication Optimisation Using Express Links for Throughput Constrained Streaming MPSoC

6.1 Introduction

Many applications, which demand high computation and on-chip communication resources, are emerging in the field of multimedia, such as H.264 and Multi View Coding (MVC). Such applications are generally referred to as streaming applications. These applications have to adhere to stringent performance constraints, and thus are often implemented on Multi Processor System on Chip (MPSoC) platforms where both the processing elements and on-chip communication architecture are (highly) customised [43]. In this chapter we focus on customisation of the on-chip communication architecture of an MPSoC, where pipeline-level parallelism of streaming applications (represented as Kahn Process Networks (KPN) [183]) is engaged for high performance [159, 184].

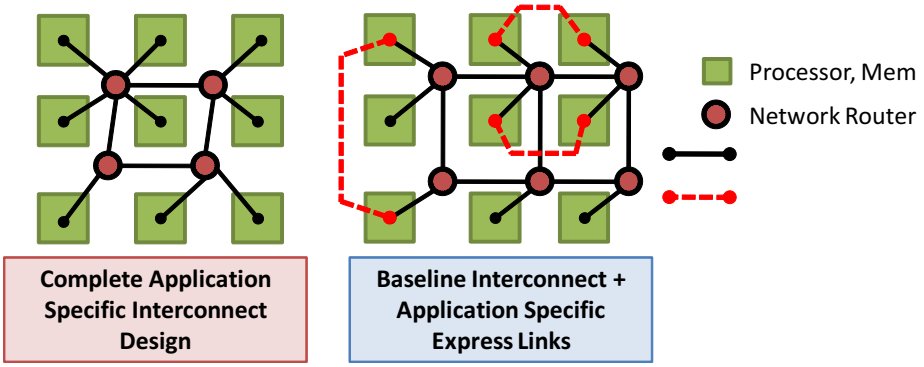


Figure 6.1: Two approaches for custom on-chip interconnect design

The design of custom on-chip interconnect can be classified into two categories (Figure 6.1): 1) Complete application-specific; and, 2) Baseline with application specific express links. The former approach designs on-chip communication architectures from scratch according to an application's needs, and thus results in highly optimised designs in terms of performance, area and energy. However, such designs are complex, and each design instance requires synthesis and floor planning for timing verification. These factors lead to longer time-to-design and time-to-market for the MPSoC. Additionally, use of dedicated, point-to-point links for all communication falls into this category; however, such a design might be overkill because of the need for instantiation of multiple point-to-point links for shared data.

The latter approach augments an off-the-shelf baseline on-chip interconnect with dedicated high bandwidth express links. The express links provide better bandwidth than the baseline interconnect, and thus reduce traffic congestion. This approach avoids the complexity of designing the entire interconnect, and is thus more feasible when lower time-to-design and time-to-market is intended. Additionally, this technique enables re-usability of interconnects from previous MPSoC designs (or when necessary tools to design a complete interconnect are not available), and lets a designer choose the actual implementation of the express links. For example, many modern embedded processors support direct point-to-point links such as Tensilica

	Link $a \rightarrow$ BI	Link $a \rightarrow$ EL
Link $a \rightarrow$ BI	1/1450 (Case 1)	1/1350 (Case 2)
Link $b \rightarrow$ EL	1/1450 (Case 3)	1/1350 (Case 4)

Table 6.1: MPSoC throughput (BI: Baseline Interconnect, EL: Express Link)

queue interface [81], Xilinx MicroBlaze FSL [185], Xilinx Mailbox IP [186] and Altera NIOS II external interface [187]. The express links can also be instantiated between network interfaces (NIs) of the communicating processors. Alternatively, express links can be embedded in the baseline communication architecture itself by modifying the routers [188], or by overlaying the underlying architecture with these express links [189, 190]. Therefore, the original characteristics of the baseline communication architecture are largely preserved [188–190], which eliminates the need for full re-verification of timing after the insertion of the express links. Hence, in this work we focus on customising a given baseline interconnect using express links such that the application's throughput constraint is met.

Several techniques for insertion of express links have been proposed in the literature [188–191]. Those techniques use only the information of traffic volume between communicating processing elements. In addition, they evaluate the performance of the customised communication architecture in terms of average packet latency and on-chip bandwidth rather than the throughput of the MPSoC, which is more important (and practical) in the context of streaming applications. Therefore, traditional techniques might result in over-design due to unnecessary insertion of express links. We explain this fact with an example.

6.1.1 Motivational Example

Let us assume a simple three-processor MPSoC as shown in Figure 6.2. The three processes of the application KPN, namely P1, P2 and P3, are mapped to CPU1, CPU2 and CPU3, respectively, to exploit pipeline-level parallelism. In pipelined execution, the different processing elements of an MPSoC are logically organised

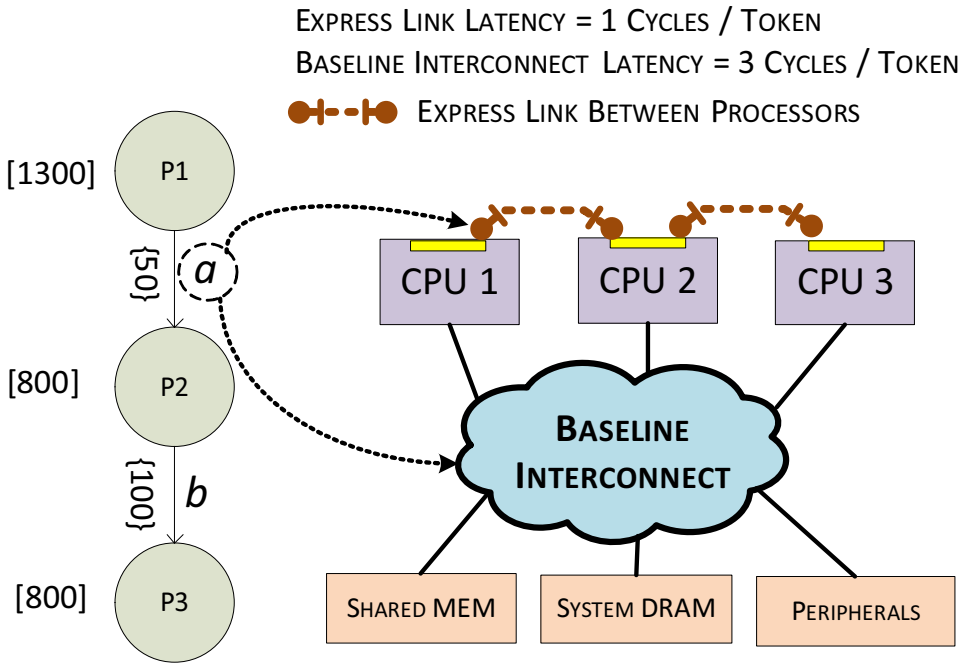


Figure 6.2: an example KPN and the target MPSoC

in a pipeline. Each processing element in the pipeline performs some computation on the incoming data and then transfers it to the next processing element. In this way, multiple processing elements process different sets of input data in a pipelined fashion. For example, when CPU1 is processing data block i, then CPU2 will be processing data block $i-1$. An iteration of the application is defined as the processing of one data block by all the processing elements, whereas throughput of the MPSoC is governed by the latency of the slowest stage.

An analysis, [192], of on-chip traffic in a pseudo-balanced pipeline shows that inter-processor communication periodically takes place at the boundaries of the iterations, resulting in a bursty traffic pattern. Additionally, temporal and spatial overlap of these traffic bursts results in collisions due to arbitration for the on-chip communication architecture. This effect leads to the formation of traffic hot spots, causing high communication delays. In such a scenario, even high-end communication architectures like Network-on-Chip (NoC) fail to sustain the required

throughput [98]. Moreover, the bandwidth offered by the network will not be sufficient for meeting the application's performance constraint. The addition of express links here provides alternative routes, which isolates some of the overlapping traffic, and hence reduces congestion in the baseline interconnect. Since traditional techniques for insertion of express links use only the information of traffic volume, they overlook the problem of bursty traffic patterns which is governed by the latency of the critical stage in a logical pipeline.

Let us assume the execution time for one iteration of process P1 is 1300 cycles in Figure 6.2, while the other two processes consume 800 cycles each. Process P1 transfers 50 tokens of data to P2 and P2 transfers 100 tokens of data to P3 during their respective iterations. The processors can communicate with each other through either on-chip baseline communication interconnect (BI) or express links (EL). Assume that the communication latencies for baseline interconnect and express links are 3 and 1 *cycle/token* respectively. The processor with the highest latency is the critical processor and the inverse of its latency is the MPSoC's throughput. For example, if both channels a and b are mapped to BI, the latencies of the processors are $1,450$ $(1,300 + 503)$, $1,250$ $(800 + 503 + 1003)$ and $1,100$ $(800 + 1003)$, and therefore MPSoC throughput is $1/1,450$. Table I report the MPSoC throughput for all the possible combinations of BI and EL for links a and b, as described below:

It is interesting to note that both cases 1 and 3 have the same throughput even with an additional express link in case 3. Likewise, cases 2 and 4 have the same throughput, but case 2 uses one fewer express link. *Therefore, the fact that non-critical processors in a logical pipeline can tolerate communication delays without affecting the application throughput can be exploited to reduce the number of express links.* If a traffic volume-based algorithm is used, then channel b will be mapped to an express link first, and channel a will be mapped to another express link. This will result in case 4 being the final MPSoC design which is a non-optimal solution. On the other hand, if the channel of the critical processor (CPU1 is critical when both the channels are mapped to baseline interconnect) is considered first, then

only channel a will be mapped to an express link, resulting in the optimal design. *Therefore, we propose that both the processor latencies and traffic volume should be used to decide the number of express links to avoid the unnecessary addition of express links.*

6.1.2 Novel Contribution

This work proposes a framework that adds an optimal or near optimal number of express links to the baseline interconnect of an MPSoC for inter-processor communication under a given throughput constraint. The addition of express links is oblivious to the baseline interconnect, and thus can be used with bus or NoC based communication architectures. For experiments, we applied our framework to two different MPSoC interconnect topologies: crossbar and mesh NoC, and compared our heuristic to a traditional iterative heuristic, adapted from [188]. The main contributions of this work are:

- A framework for selective addition of a minimal number of express links to guarantee throughput of a streaming application on an MPSoC with the baseline communication architecture.

- A novel heuristic which considers both processor latencies and traffic volume to efficiently determine the number of express links. The heuristic is designed so as to efficiently prune the design space and reach the solution quickly.

The rest of the chapter is organised as follows: Section 6.2 encompasses the state-of-the-art while Section 6.3 formalise the problem and Section 6.4 describe the proposed framework. The experiment results for crossbar NoC based MPSoC are discussed in Section 6.5 and results for mesh NoC based MPSoC are discussed in Section 6.6.

6.2 Related Work

Over the past few years researchers have expressed a keen interest in designing application-specific MPSoCs for streaming applications. Authors of [159] and [193] explore MPSoC architectures for throughput-constrained streaming applications which use hardware queues for all interprocessor communication. Similarly, the architecture proposed in [194] uses high speed memory buffers for all the communicating processors. We argue that mapping every processor-to-processor communication to a separate direct link may result in overdesign, and hence a better technique for selective addition of such links is needed. Complete synthesis of custom communication architectures based on NoCs and crossbars are discussed in [86] and [92] respectively. Author of [195] have explored the use of either a crossbar or P2P network for MPSoC synthesis. Recent work by Castrillon et al. [148] considers the option of having multiple on-chip communication primitives for MPSoCs such as direct links or shared memory. However, they consider a predefined MPSoC platform with a fixed number of communication primitives. Thus, they focused on mapping the KPN of an application to the MPSoC to maximise its throughput under the constraints on the number and the memory size of the communication primitives. Unlike [148], we minimise the area of express links under a throughput constraint, which is typical of real-time streaming applications [159].

Several techniques have been proposed to improve the performance of baseline communication architecture by the addition of express links. Some classical works in this area add random links to improve the overall system performance [196, 197]. Chan et al. [190] introduced the idea of synthesising a NoC-based communication architecture that is a mix of a packet switch network and direct point-to-point links. Another well-known work presented in [188] added a few long-range links within an NoC to improve the MPSoC performance, measured in terms of the "critical load point" of the system. Other similar techniques have been proposed in [191, 198]. Some futuristic works are proposed in [199] and [189], where the authors suggest

the use of overlay point-to-point wireless links and high frequency RF links for the most frequently communicating tiles in a mesh based NoC architecture. All the techniques referred to improve the performance of on-chip communication in terms of aggregate on-chip bandwidth and average data latency, which are not suitable metrics for streaming applications executing on MPSoCs. Our framework uses both the latencies of processors and the traffic volume to evaluate the need for additional express links under a throughput constraint, as throughput is the most important performance metric for streaming applications [159, 193].

6.3 Problem Definition

We target MPSoC architectures similar to the one presented in Figure 6.2. We assume that mapping of a streaming application's KPN has already been performed by using one of the tools proposed in the literature [200]. The communication in the MPSoC is represented as a directed graph $G(P, C)$, where vertices P represent the processors and edges C describe the inter-processor communication. The total number of processors is $|P|$ while the total number of edges is $|C|$. The edges of the graph are annotated with the amount of data exchanged between processors during an iteration of the application. The MPSoC topology along with the baseline communication architecture, MT, is provided by the designer, where all the communication edges are initially mapped to the baseline interconnect.

Our goal is to achieve a particular throughput for the application (provided by the designer) through the addition of a minimal number of express links, C_{EL}. Since express links can be realised differently, we abstract their details with the use of a cost function $CF(C)$, provided by the designer, which will return the gate cost of implementing express link for a particular communication edge C. In this framework, we add an extra port to the processor's interface for communicating through the express link. Such a facility is provided in some of the more modern processor IPs, such as Tensilica's Xtensa processor range. The size of an express

link's buffer is equal to the volume of the data transferred between processors in a single iteration [159], thus ensuring quality of service (QoS) on that particular express link under a bursty communication pattern. Therefore, the cost function is:

$$CF(C) = PortAreaCost + BufferAreaCost(V) + WiringCost(1)$$

where V is the volume of data for communication edge C and $BufferAreaCost(V)$ is the function that returns the area footprint express link's buffer.

Now, we can formally define our problem:

Given:

$G(P, C)$ Communication Model of MPSoC

$CF(C)$ Express Link cost Model

M_T MPSoC Topology with baseline interconnect

T_H Applicatoion throughput constraint

Return:

MPSoC topology $M_T^{/}$, which is as enhanced version of M_T after and addition of express links set C_{EL}.

Such That:

$Application\ Throughput \geq\ T_H$, and,

$$Minimize(\ \sum CF(C_{EL})\ )$$

The throughput of the application, T, on the MPSoC is the inverse of the latency of the critical processor in the logical pipeline. That is,

$$T = 1/L_C$$

where Lc is the latency of the critical processor. The latency of a processor P, L_P, is calculated by subtracting the waiting time spent in a blocking read from or a blocking write to the communication buffer (baseline interconnect or express link), from the total time spent by the processor in an iteration. The latency L_P covers both the net computation and the communication time of the processor P [159]. Formally:

$$L_p = IterationTime - BlockingReadDelays - BlockingWriteDelays$$

The inter-processor communication time, CT, for a particular communication edge can be formally defined as:

$$CT(V, BW_B) = V/BW_B + H_B + \beta_B + d(s,t)$$

where BW_B and H_B are the bandwidth and intrinsic hop delay of the baseline communication architecture, and β_B is the software overhead of managing communication buffers. Note that V, BW_B, H_B and β_B are constants for a given application and communication architecture. The factor $d(s,t)$ captures the delays incurred due to arbitration for shared resources of the baseline architecture (for example, routers, bus, memories), and is a function of time t and number of hops s. The variable $d(s,t)$ can be further decomposed into sub-factors, representing each hop resource i within the communication path:

$$d(s,t) = \sum_{i=0}^{n} d(s_i, t)$$

The variance of $d(s,t)$ under different traffic conditions makes the communication time unpredictable. Introducing express links for a particular communication edge removes the contention and arbitration overhead ($d(s,t) \rightarrow 0$), and hence reduces CT to:

$$CT(V, BW_{EL}) = V/BW_{EL} + H_{EL} + \beta_{EL}$$

where BW_{EL}, H_{EL} and β_{EL} represent the bandwidth, hop/pipelined delay and software overhead of the express links.

In this framework we assume that there exist only two possible inter-processor communication primitives: baseline communication architecture and express links. Moreover, we assume that the data transfer through express links is always faster than the baseline interconnect.

6.4 Framework Overview

Figure 6.3 shows a typical design flow [195] for creation of customised MPSoCs, and how our framework fits in it. A designer provides the baseline MPSoC architecture and the application KPN. The second phase involves mapping of the KPN onto the MPSoC, including selection of appropriate processing elements (which mimics customisation of the processing elements). Then the communication architecture of the MPSoC is customised. Note that customisation of the processing elements and communication architecture should be done simultaneously; however, such a customisation results in a prohibitively large design space often with impractical exploration times [201]. Finally, the customised MPSoC is synthesised.

Our framework takes the baseline MPSoC architecture, throughput constraints and cost function for express link as the input. The framework then simulates the

MPSoC by mapping all the communication edges to the baseline interconnect. If the required throughput is achieved, the framework exits with a solution that does not contain any express links. However, if the throughput achieved in the previous step is less than the required throughput; then all the communication edges are mapped to individual express links and the MPSoC is simulated again. Under the assumption that express links always provide faster communication, if the MPSoC still fails to meet the throughput constraint, the framework exits with a fail report. On the other hand, if the throughput constraint is met by the MPSoC, then we can infer that there exists a solution where the required throughput can be achieved with the number of express links less than or equal to $|C|$. Here, the framework runs our novel heuristic for insertion of a minimal number of express links.

We use cycle-accurate simulations of the MPSoC in our framework. We argue that on-chip shared resources such as communication architecture, and associated routers and memories exhibit a complex, dynamic behaviour under different workload as described in Equation 1. Moreover, the data traffic patterns for various processors can interact in a non-deterministic manner [202]. Therefore, it is hard to develop a useful performance analytical model of the MPSoC targeted in this work, which has also been mentioned in [203]. Even though some of the frameworks presented in literature do use analytical models, we advocate the use of real simulations for accurate computation of the MPSoC throughput. However, an analytical model can be incorporated in our framework as long as it can predict the MPSoC throughput with high accuracy.

6.4.1 Express Link Insertion Algorithms

6.4.1.1 Exhaustive Algorithm

The problem of insertion of express links can be naively solved by using an exhaustive algorithm, where all the possible additions of express links (design space)

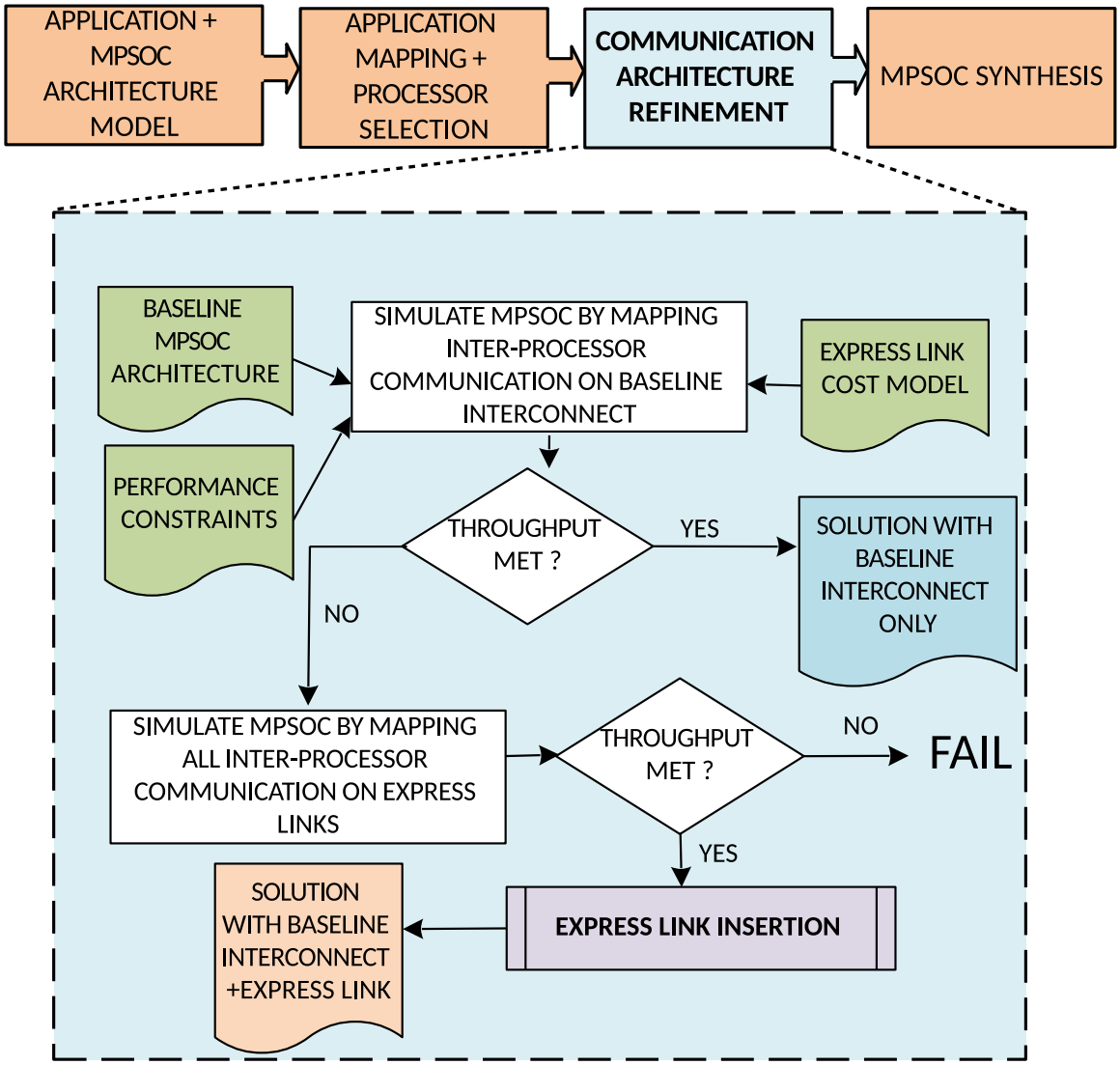


Figure 6.3: Framework overview

are explored. For example, the exhaustive algorithm will simulate all the four design points (cases) reported in Figure 2. Then, those design points that violate the throughput constraint are deleted. From the remaining design points, the algorithm chooses the one with minimal cost computed using the express link cost function provided by the designer. The time complexity of this algorithm is given by $O(2^{|C|})$, where $|C|$ is the total number of communication edges in the application communication graph. This algorithm has an exponential run time, which makes it impractical when there are more than a few communication edges.

Algorithm 6.1: xLink Heuristic

```
1  Map alledges in G(P, C) to BI
2  T_H ← Application throughput constraint
3  T ← MPSoC throughput after simulation
4  if T < T_H then
5      if First Algorithm Iteration then
6          μP ← 0
7      else
8          L_1 ← Latency improvement for P_CH
9          L_2 ← Latency improvement for P_C
10         V ← Volume(Edge between P_CH and P_C)
11         μP ← min(L_1, L_2)
12     R_H ← list of all P where L_P > (1/T_H)
13     Sort R_H in descending order of L_P
14     P_C ← First processor in list R_H
15     R_L ← list of all P where L_P ≤ (1/T_H)
16     R_HC ← list of all member of R_H having BI edge with P_C
17     if R_HC ≠ ∅ then
18         P_CH ← Processor in R_HC with highest latency
19         Map edge between P_CH and P_C to EL
20         goto Step3
21     else
22         C_CL ← All BI edges from P_C to R_L
23         S_F ← (T_H − T)/μP
24         C'_CL ← Edges in C_CL where Volume(C_CL) ≥ S_F
25         Sort C'_CL in ascending order of data volume
26         Pick the first edge in C'_CL and map it to EL
27         P_CH ← Processor connected to P_C through edge selected in Step 26
28         goto Step3
29 else
30     return Solution
```

6.4.1.2 Express Link Insertion Heuristic (xLink)

We now introduce our heuristic, $\times Link$, for the insertion of express links that efficiently explore the design space and gives an optimal or near-optimal solution (Algorithm 6.1).

Our heuristic uses the latencies of the processors in the MPSoC for efficient

pruning of the design space. The algorithm starts by mapping all communication edges to baseline interconnect (BI). After simulating the MPSoC, the algorithm checks whether the required throughput has been achieved or not. If the throughput has been achieved, then the algorithm exits and returns the new MPSoC topology with the current number of express links. However, if the throughput is not met, then the algorithm has to find a candidate link that is currently mapped to baseline interconnect. At this stage, the algorithm divides all the processors into two sets: R_H contains processors that have latency higher than $1/T_H$ (Step 12), where T_H is the required throughput; and R_L contains the processors with latency less than or equal to $1/T_H$ (Step 15). *Here the heuristic uses the intuition that processors in the set R_L are non-critical, and the addition of an express link between them will not improve the MPSoC throughput.* Therefore, preference is given to processors in set R_H. Additionally, selection of a BI edge that will reduce the latencies of more than one processor from set R_H at the same time will be more beneficial. Let processor P_C be the critical processor in the MPSoC with the highest latency (Steps 13-14). Then, the set R_{HC} is populated containing all the processors from R_H that have a BI edge with processor P_C (Step 16). Let processor P_{CH} be the one that has the highest latency within the set R_{HC}. The edge between P_C and P_{CH} then represents the edge between the two most critical processors which is still mapped to BI, and hence the heuristic maps this edge to an EL (Steps 18-19). *The steps 18-19 are based on the intuition that the processor with latency close to the latency of the critical processor has the potential to become the next critical processor.*

It is possible that processor P_C does not have any BI edge with the processors in the set R_H (i.e. $R_{HC} = \emptyset$). In this case, the heuristic looks for a BI edge between processor PC and processors in the set R_L (Step 22). The intuition is to change an edge from BI to EL so that the throughput of P_C goes just below the throughput constraint. Therefore, we introduce a **prediction factor** μP, which predicts the approximate improvement in $latency/unit\_data\_volume$ of a processor when one of its edges is changed from BI to EL. Generally speaking, factor μP helps the heuristic

predict the volume of candidate edge that should be changed to EL, and thus will lead to a more optimal solution rather than picking the edge with the largest data volume or smallest data volume (Steps 22-26). The value of μP is set using the following method:

- For the first iteration of the algorithm, an initial value is selected based on short pre-framework profiling of the specific application (Steps 5-6) (Algorithm 1)
- At the start of each subsequent iteration, a new value of μP is calculated based on the improvement in the latency of the processors from the previous iteration, when an edge between these processors was mapped from BI to EL. (Steps 8-11) (Algorithm 1)

The prediction factor μP can be time-dependent. Authors of [188] observed that adding the long-range links in the network eased some of the paths in the network due to isolation of data streams, hence reducing the average packet latency. However, the complex interconnect behaviour can introduce errors in this estimation, which can affect the optimality of the heuristic.

Our proposed heuristic has a linear run time $O(|C|)$ because it changes only one communication edge during each iteration.

6.4.1.3 Iterative Algorithm for Comparison

For the sake of comparison, we also present a heuristic (Algorithm 6.2), that is adapted from [188]. We modified their heuristic by changing the objective function from maximising the critical load to minimising the express link area footprint and area constraint to throughput constraint. This heuristic replaces all the BI edges with EL edges one by one and simulates the MPSoC to observe the throughput improvement (Steps 4-8). When none of the individual EL edges meet the throughput constraint, the edge with the highest *improvement/volume* ratio is permanently

Algorithm 6.2: Iterative Heuristic adapted from [188]

```
1  Map alledges in G(P, C) to BI
2  T_H ← Application throughput constraint
3  T ← MPSoC throughput after simulation
4  if T < T_H then
5      L_BI ← all C where C is mapped on BI
6      for each edge C_BI in L_BI do
7          Map C_BI to EL and simulate the MPSoC
8          Record improvement in throughput and map C_BI back to BI
9      if no edge changeover guarantees T < T_H then
10         Map link with highest (improvement/volume) ratio to an EL//
11     else
12         Select edge that guarantee T < T_H AND has smallest volume and
           map it to Express Link
13         goto Step2
14 else
15     return Solution
```

changed to an EL (Step 10). Otherwise, the BI edge with the smallest volume is changed to an EL (Step 12). This process is repeated iteratively until the throughput constraint is met. The worst-case run time of this heuristics is $O(|C|^2)$ because, for each iteration, the heuristic cycles through all the edges that are mapped on BI.

Note that a branch and bound algorithm proposed in [191] could also have been used for comparison. The algorithm in [191] limits the searchable design space by the number of ELs to be inserted (S) and the number of ways of inserting those ELs (B). The optimal solution is sought when these factors are set to $|C|$, in which case the complexity of the algorithm in [191] is same as the exhaustive search. Since a designer cannot predetermine the best values of these factors, we believe that the use of arbitrary values will lead to an unfair comparison. Therefore, we did not compare xLink with [191].

	JPEG	H.264	Media1	Media2	FFT	TDE
# of Processor	6	6	8	9	12	13
# of Edges	5	8	10	12	12	12
Node Max Fan in/out	1/1	2/2	2/2	2/2	2/2	1/1

Table 6.2: Benchmarks for Crossbar NoC based MPSoC

6.5 Case Study: Crossbar NoC Based MPSoC

6.5.1 Experiment Setup

Our first case study uses crossbar NoC based MPSoC. To verify our proposed framework, we used a commercial design environment from Tensilica [81]. XTSC, a SystemC based cycle-accurate simulator, is employed to simulate the MPSoC shown in Figure 6.4. SystemC based simulation models for Xtensa LX4 processors are used in this experiment. The baseline interconnect is a non-blocking crossbar NoC which is organised in a master-slave router configuration. The flit buffers for routers are sized to store one cache line, while the hop latency of each router is two cycles. The slave routers follow round-robin arbitration for serving the incoming requests. A number of shared on-chip memory banks are distributed across the NoC to implement software managed FIFO buffers for interprocessor communication. The DRAM is used to serve instruction and data cache. Further details of the MPSoC are included in Table 6.3. For experiments, we used two streaming applications: $JPEG$ *encode*r and $H.264$ *encoder*, FFT and TDE signal processing applications from StreamIt [204], and two synthetic streaming applications: $Media1$ and $Media2$ which are derived from the original $H.264$ application. The details of these applications, when mapped on the MPSoC, are provided in Table 6.2. All these benchmarks are a mixture of linear pipeline graphs and complex graphs as depicted by the node's maximum fan in/out values. For simulations, we used an Intel Xeon based server, with 256GB RAM, and Ubuntu Server 10.10 OS.

We used hardware queues to realise express links between processors in our experiments. Xtensa LX4 provides queue ports for high bandwidth point-to-point

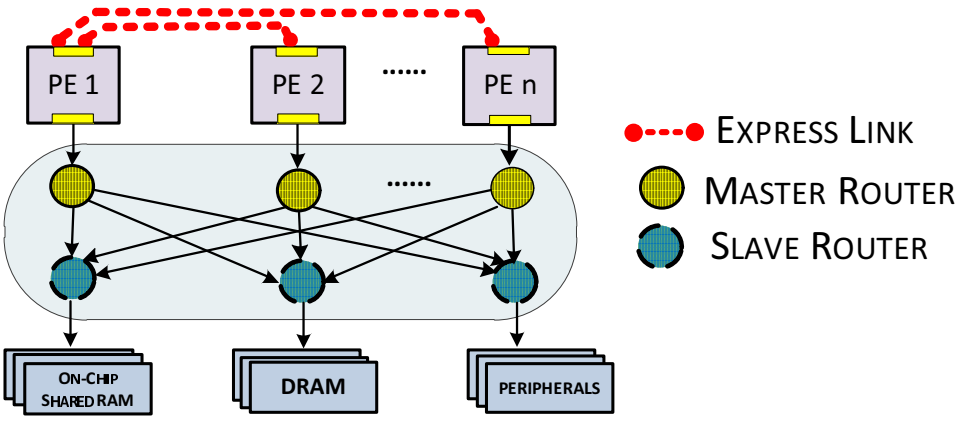


Figure 6.4: Crossbar NoC based MPSoC for Experiments

communication. These ports are directly interfaced with the ALU of the processor and therefore can be read and written using standard register access instructions. We used Tensilica's area tool to estimate the area footprint of queue ports, and CACTI [205] for the area footprint of the queue buffer for a given 45nm technology.

	Benchmarks					
	JPEG	H.264	Media1	Media2	FFT	TDE
Processor	Xtensa LX4 @ 1GHz					
I/D Cache Size	1,4 Kb	2,4,6,8,16 Kbytes				
Cache Ways	1,2 Ways					
Custom Instruction	No	Yes				No
Base Communication	Pipelined Crossbar with Master-Slave Router					
Data Width	32	128 bit				
On-Chip SM Latency	4 cycles					
DRAM Latency	20 Cycles					

Table 6.3: Architectural Details for Crossbar NoC based MPSoC

	Throughput Constraint	Area Overhead (mm^2) for Express Links			Design Points Explored			Framework Runtime		
		Optimal	% Error for xLink	% Error for Iterative	Exhaustive	xLink	Iterative	Exhaustive	xLink	Iterative
JPEG	#1	0.035	0%	0%	32	3	9	3h	15m	45m
	#2	0.052	0%	0%		4	12		20m	1h
	#3	0.052	0%	0%		4	12		20m	1h
H264	#1	0.052	0%	0%	256	3	15	1d	17m	1h 22m
	#2	0.104	0%	0%		5	27		28m	2h 28m
	#3	0.032	0%	0%		2	8		11m	44m
	#4	0.084	0%	0%		4	21		22m	1h 55m
	#5	0.125	0%	0%		5	31		28m	2h 50m
Media 1	#1	0.052	0%	0%	1024	3	19	4d	18m	1h 54m
	#2	0.0324	0%	0%		1	10		6m	1h
	#3	0.101	0%	20%		5	40		30m	4h
	#4	0.122	0%	14%		6	49		36m	4h 45m
	#5	0.084	0%	45%		4	40		24m	4h
Media 2	#1	0.094	0%	0%	4096	3	19	16d	30m	2h 18m
	#2	0.154	0%	24%		7	68		42m	6h48m
	#3	0.057	0%	0%		3	13		18m	2h18mh
	#4	0.126	6%	16%		5	27		30m	5h 42m
	#5	0.171	0%	12%		8	63		48m	6h 18m
FFT	#1	0.091	0%	11%	4096	9	75	17h	2.5m	19m
	#2	0.03	0%	0%		3	33		1m	8m
	#3	0.051	20%	20%		6	57		2m	14m
	#4	0.126	0%	29%		7	72		11m	1h 45m
TDE	#1	0.5445	0%	0%	4096	6	57	5d	11m	1h 45m
	#2	0.2178	0%	0%		3	33		5m	55m
	#3	0.4356	0%	0%		5	50		9m	1h30m
	#4	0.3267	6%	16%		4	42		8m	1h 16m

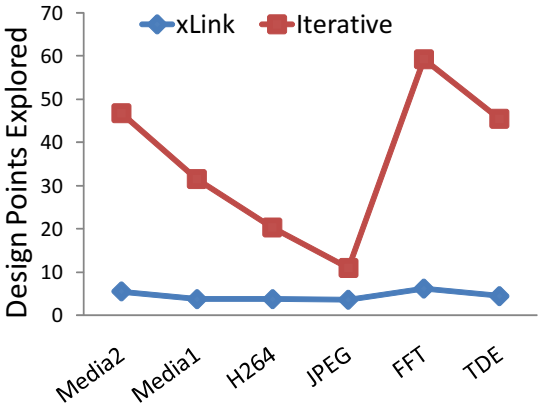
Table 6.4: Detailed Experiment Results for Crossbar NoC based MPSoC

6.5.2 Results and Discussion

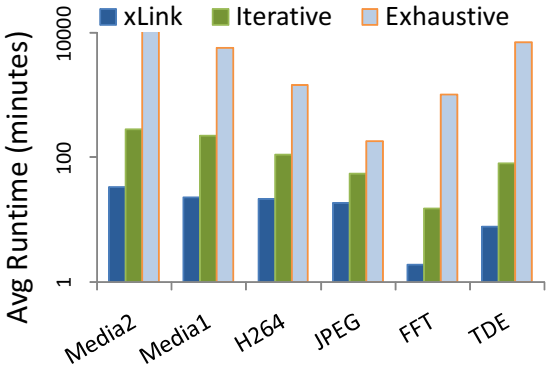
The summary of results for both the xLink and Iterative (adapted from [188]) heuristics for crossbar-NoC based MPSoC are reported in Figure 6.5. We provided different throughput constraints for each application and recorded the area footprint of the express links and the runtime of the framework. The area overhead of express links encompasses the cost of adding a port to processor and the buffer cost associated with the express link. The quality of the solutions generated by the heuristics was compared to the optimal solutions from the exhaustive algorithm. Detailed results are included in Table 6.4. Major column 1 lists the area of the express links, while major columns 2 and 3 list design points explored and framework runtime respectively.

Figure 6.5c and 6.5d compare the average and maximum % error in the area footprint results generated by the two heuristics. For $JPEG$, TDE and $H.264$ applications, both the heuristics generated optimal solutions. However, for the other three benchmarks, the maximum and average errors for the Iterative heuristic are higher. The iterative heuristic generated a maximum error of 45% for $Media1$, 29% for FFT and 25% for $Media2$ applications. In the case of xLink, there was a maximum error of 20% for the FFT benchmark. The source of this error is the condition when a critical processor has only BI links to non-critical processors, and xLink has to predict a link for changeover to EL (Steps 13-20, Algoithm 6.1). Such conditions are rare and only occur in MPSoCs with sparse communication patterns. Despite this, our results show that heuristics which do not take into account the latency information of the processors are more prone to reaching the local minima.

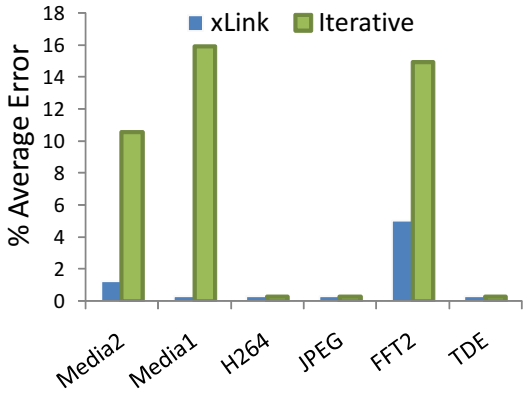
The xLink heuristic also performed better than the Iterative heuristic in terms of design space explored. On average, xLink explored 7 less design points than the Iterative heuristic (Figure 6.5a). Given the quality of the solutions generated by xLink, this is a significant improvement. Similarly, the xLink heuristic has a considerably lower runtime than both the Iterative heuristic and exhaustive algorithm for



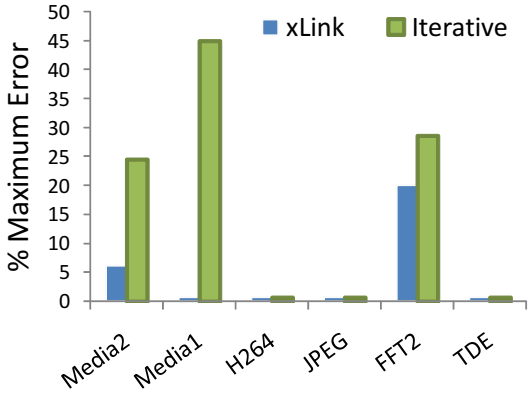
(a) Average Design points explored



(b) Average Framework Runtime



(c) Average error in Express Link Area



(d) Max error in Express Link Are

Figure 6.5: Summary of Results for Crossbar NoC based MPSoC.

	VPOD	TGFF-1	MWD
# of Processor	12	16	12
# of Edges	13	19	12
Node Max Fan in/out	2/2	3/2	2/2

Table 6.5: Benchmarks for Mesh NoC based MPSoC

all applications, as shown in Figure 6.5b (note that the y-axis scale is logarithmic). For example, the average runtime of xLink for the TDE application was 8 mins. The iterative heuristic consumed 1h20m, while the exhaustive algorithm consumed 5 days for the same application. Similarly, for other applications, the runtime of the xLink heuristic is in minutes, compared to hours for the iterative heuristic and days for the exhaustive algorithm. This result signifies that the use of processor latencies can help in efficient pruning of the design space, and hence lead to the solution quickly. Thus, the xLink heuristic is more scalable than the iterative heuristic and exhaustive algorithms.

6.6 Case Study: Mesh NoC Based MPSoC

6.6.1 Experiment Setup

Our second case study is based on a 2D mesh NoC-based MPSoC as shown in Figure 6.6. The mesh NoC architecture is modelled in Verilog HDL, while the processing

	Benchmarks		
	VOPD	TGFF-1	MWD
Processor	Xtensa LX4 @ 1GHz		
Memories	Private Instruction and Data Memories		
Network Topology	4×3 2D Mesh	4×4 2D Mesh	4×3 2D Mesh
Network Router	2 Cycle Latency, 4 flit Buffer, Matrix Arbiter		
Channel Width	32 bit		
Network Interface	Bi-directional, 128 flit deep receiver FIFO		
End-to-End Flow Control	Software managed credit based		

Table 6.6: Architectural Details for Mesh NoC based MPSoC

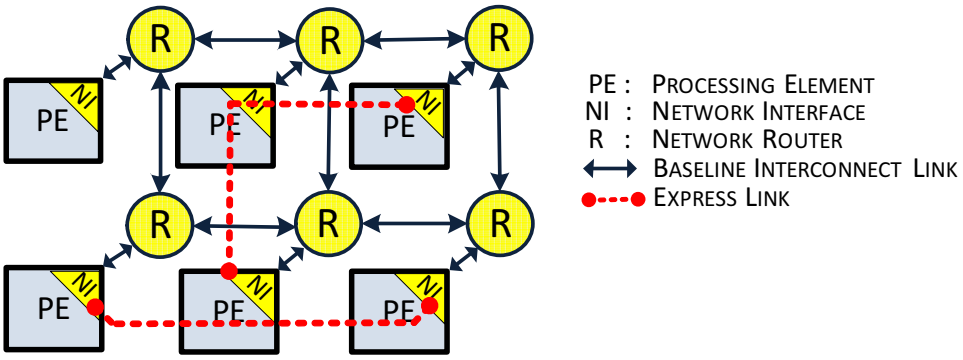


Figure 6.6: Mesh NoC based MPSoC

nodes are modelled in SystemC. We used Tensilica's SystemC tool, XTSC, and ModelSim for co-simulation of processors and interconnect. Processing elements in the MPSoC are Xtensa LX4 processors with local instruction and data memory. The Network Interface (NI) and express links are directly interfaced with the ALU of the Xtensa processor for low-latency network access. Each network interface (NI) has a 128 flit deep incoming data FIFO. We used a software-based credit exchange system for NI buffer management to avoid application-level deadlock [206]. The baseline interconnect is modelled to as 2D mesh-tiled architecture where each processing element is connected to a unique network router. The routers are connected with other neighbour routers using bi-directional links. The network router has two cycle hop latency and has a 4 flit buffer at each input port. We employ XY dimension routing as it is inherently deadlock free. Processing elements directly communicate with each other using a Message Passing Interface (MPI) like software API over NoC. Other architectural details are included in Table 6.6. For benchmarking we used three application graphs: 1) Video Object Plane Decoder ($VOPD$) from [206], 2) $TGFF-1$ application task graph generated from the TGFF tool [207], and, 3) Multi Window Display (MWD) from [206]. The details of these applications, when mapped on the MPSoC, are provided in Table 6.5. We used signal processing kernels for the applications that mimic the behaviour of the original streaming application. All these benchmarks consist of complex graphs with series and parallel pipeline stages as depicted by the node fan-in/out values. An Intel Quad-core machine, with

8 GB of RAM and Ubuntu 10.10 operating system was used for simulations in this case study.

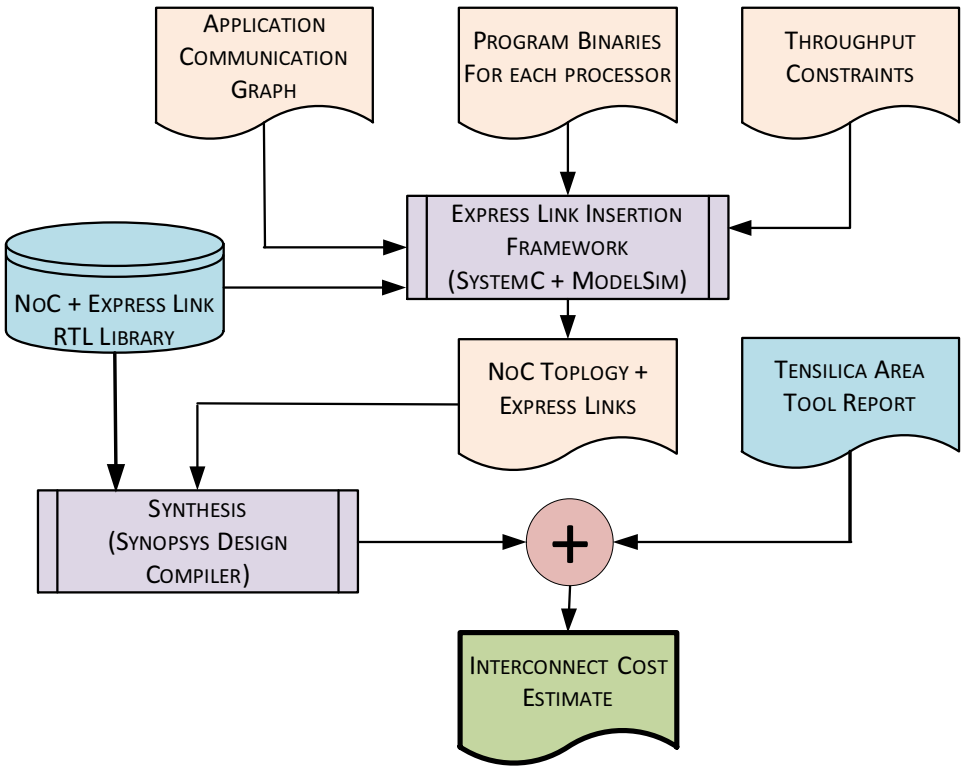


Figure 6.7: Experiment Tool Flow for Mesh NoC based MPSoC

The Verilog RTL for complete interconnect (baseline interconnect and express links) was synthesised using Synopsys Design Compiler tool (details in Figure 6.7). We used TSMC's 65-nm cell library for the NoC and express link synthesis, and Tensilica's area tool to estimate the area footprint of hardware queue ports.

6.6.2 Results and Discussion

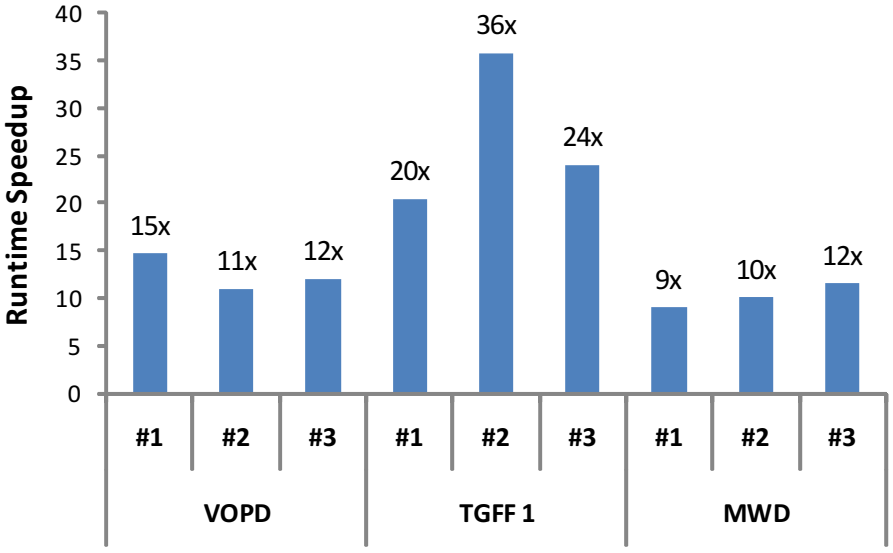


Figure 6.8: Interconnect Area for Mesh NoC + Express Links using xLink (normalized w.r.t Iterative)

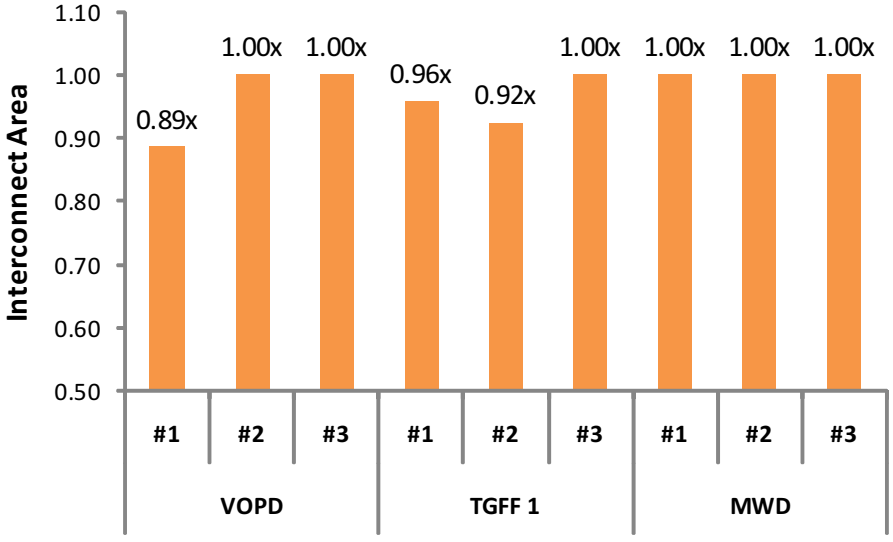


Figure 6.9: Framework Runtime Speedup for NoC based MPSoC using xLink (normalized w.r.t Iterative)

	Throughput Constraint	Total Interconnect Area (K Gates)		Design Points Explored			Framework Runtime	
		xLink,	Iterative	Exhaustive	xLink	Iterative	xLink	Iterative
VOPD	#1	1087.9	1229.1	8192	6	88	12m	2h 65m
	#2	966.5	966.5		5	55	10m	1h 50m
	#3	869.7	869.7		3	36	6m	1h 12m
TGFF 1	#1	1118.3	11167.1	524288	7	143	7m	2h 23m
	#2	1055.9	1142.1		4	143	4m	2h 23m
	#3	1031.1	1030.8		2	48	2m	48m
MWD	#1	860.8	860.8	4096	7	63	14m	2h 6m
	#2	817.1	817.1		5	50	10m	1h 40m
	#3	773.6	772.6		2	23	4m	23m

Table 6.7: Detailed Experiment Results for Mesh NoC based MPSoC

The results of our experiments with mesh NoC-based MPSoC are summarised in Figure 6.8 and Figure 6.9. For each benchmark, we provided different throughput constraints to the framework and recorded the performance of the xLink and iterative heuristic in terms of framework runtime and reported NoC area footprint (baseline mesh NoC and express links). The exhaustive algorithm did not complete in time, therefore, for the mesh NoC, we compare only xLink and iterative heuristics. Detailed results are included in Table 6.7. Major column 1 lists the total area of the interconnect, while major columns 2 and 3 list design points explored and framework runtime, respectively.

Figure 6.8 reports the final interconnect area for xLink, which includes the cost of the baseline interconnect and the express links. The data has been normalised with respect to data reported by the iterative heuristic. In 3 out of 9 cases the xLink heuristic reported lower interconnect area than Iterative heuristic. For example, the xLink heuristic saved 11% and 8% interconnect area for VOPD constraint 1 and $TGFF-1$ constraint 2 respectively, compared to the iterative heuristic. Our results signify that heuristics which do not take into account the latency information of processor latencies are more prone to yield an over designed interconnect. Apart from the design cost, a relatively larger silicon area also results in higher power consumption of the MPSoC.

Figure 6.9 summarises the runtime speedup for xLink with respect to the iterative heuristic. The xLink heuristic performed better than the iterative heuristic in framework runtime for mesh NoC based MPSoC. On average, xLink was 27× faster than the iterative algorithm in the case of $TGFF-1$, 13× faster in case of VOPD and 10× faster in the case of MWD. For example, for constraint 1 in $VOPD$, the framework runtime for xLink and the iterative algorithm was 12m and 2h 56m respectively. This is a significant achievement given given the accuracy of the xLink algorithm. The use of processor latencies enabled xLink to visit fewer design points (Table 6.7), which resulted in a shorter runtime.

6.7 Conclusion

In this chapter we proposed a framework for customising on-chip communication of an application-specific MPSoC, by augmenting a baseline interconnect with express links, under a throughput constraint for the executing streaming application. As part of the framework, we proposed a novel heuristic, xLink, that considered both processor latencies and traffic volume to efficiently prune the design space and quickly reach the solution, and compared it to a traditional iterative heuristic [188]. By using case studies of a crossbar and mesh NoC based MPSoC, we showed that xLink heuristic is more accurate and faster than the traditional iterative heuristic. Our novel heuristic is oblivious to the baseline interconnect, which is illustrated by the application of our framework to two different MPSoC interconnects. The low run-time and high accuracy of our framework can enable MPSoC designers to quickly investigate possible designs for the baseline interconnect, along with different implementations of express links. At the end, designers can perform a cost-performance tradeoff analysis to select suitable on-chip communication architecture. Thus, adapting our novel xLink heuristic in an express link insertion framework can help MPSoC designers to reduce the time-to-market without sacrificing accuracy.

Chapter 7

Conclusions and Future Direction

7.1 Thesis Summary

Multicore architectures were introduced as a solution to design problems associated with high frequency single-core monolithic architectures, and Network-on-Chip (NoC) is envisioned as a scalable communication fabric for multicore chip containing hundreds of cores. Under Moore's Law, we continuously added more computing power in the shape of computing cores. However, continuous transistor size scaling and poor power density scaling have resulted in an abundance of transistors that can not be switched on due to power and thermal budget constraints. This phenomenon is called *Dark Silicon*. The previous research on *Dark Silicon* concentrated on improving the energy efficiency of computing cores by introducing heterogeneity at architecture and system level and by better thermal and power management schemes, completely neglecting the effect of dark silicon on NoC design flows. In this thesis, we explore different NoC architectures that exploit dark silicon to improve power, performance and reliability metrics. To the best of out knowledge, this is the first comprehensive study on interplay of dark silicon and various NoC metrics.

In Chapter 2 we introduced the reader to some basic concepts of NoC design. We briefly discussed some current and emerging power saving techniques and we

discussed in detail the application of these techniques to NoCs. We observed that leakage power in NoCs is a growing concern in the design community. Architectural details of some of the recent academic and commercial NoCs are discussed, pointing towards a general trend of integrating multiple NoC to improve different metrics. We outlined some the application mapping techniques for NoC based MPSoC that try to optimize different system parameters such as performance and energy. In the end, we introduced some design flows that have been proposed over the years for designing application-specific communication architectures.

In Chapter 3, a novel NoC architecture called *darkNoC* is introduced. darkNoC consists of multiple NoCs, where each NoC is optimised at design time using multi-vt optimisation for specific VF level. The main purpose of the architecture is to improve the efficiency of the DVFS scheme. We show that instead of applying DVFS to a router that has been designed to operate at a higher VF level, using a router that has been designed for lower VF level is more energy efficient. Based on this observation, we propose using dark silicon to integrate multiple NoCs, and propose a scheme to switch between these NoCs in parallel to switching between different VF levels. The switch-over mechanism is developed using a combination of a global *Region Manager (dRM)* and local *Local Managers (dLMs)* which coordinate autonomously for seamless transition between NoC planes. Through experiments, we show that the switch-over process can take about 400$\sim$1000 ns for a 8$\times$8 mesh depending on various factors. Our experiments with real application show that the darkNoC architecture can provide up to 52% saving in NoC energy delay product (EDP) for certain benchmarks, whereas, a state-of-the-art DVFS scheme only saved 15% EDP. This shows the efficacy of our proposed architecture. We also discuss the affect of application characteristics, dark silicon budget available and DVFS schemes on overall NoC EDP savings.

In Chapter 4, we propose Malleable NoC architecture which improves the energy efficiency of NoC through a combination of multiple VF optimised routers and per-node VF selection. We point out two important facts about the interplay of

application and NoC based multicores. First, modern multicores are expected to execute a diverse set of application and each application is expected to impose different constraints on network and memory latency. Second, most multicore chips use Mesh topology for simplicity, with memory controllers placed on the periphery of the mesh. Therefore, the network latency component of memory accesses varies depending on the physical location of core in the mesh. Based on these observations, we classify our applications on the basis of miss per kilo instructions (MPKI) and map applications with high MPKI closer to memory controllers. To exploit such mapping, we purpose a NoC architecture where multiple VF optimised routers are integrated at each mesh node. At a given time, only one of the routers is switched on, depending on the VF level chosen for that node. At runtime, depending on application's MPKI and the distance of the core from the memory controller, we choose a VF level for each node, and the system autonomously connects routers from each node to realise this VF map. Our experimental results show that a fine-grain VF and router selection scheme gives better EDP saving than a *darkNoC* architecture for workloads with heterogeneous MPKIs. For example, in one of the benchmark applications, darkNoC saved 26.4% NoC EDP whereas Malleable NoC saved 46% EDP. NoCs have to compete for dark silicon with other components such as application specific accelerators. Therefore, we also discussed results for partial Malleable NoC configurations, where only a subset of nodes contain routers for a lower VF level.

In Chapter 3 and Chapter 4 we optimised the energy consumption of NoCs by exploiting dark silicon. In Chapter 5 we presented a novel NoC architecture, *SuperNet*, that optimises energy, performance and reliability of on-chip interconnect in exchange for dark silicon. We are motivated by the fact that chip design cost are rising and it is therefore becoming normal practice to design SoCs for multiple use scenarios. *SuperNet* consists of multiple parallel NoC planes that are optimised for VF levels [$V1$,f] and [$V2$, $f/2$]. At a given time, SuperNet can be configured to run in energy efficient mode, reliability mode or performance mode, depending

on available power and optimisation goals. In energy efficient mode, depending on the MPKI values of all the cores, the DVFS manager decides to run NoC at a low or high VF level and, depending on the VF level, the NoC designed for that VF is switched on while the other NoCs remain switched off. Our experiment shows that the proposed mode can save up to 81.5% power for the *sha* benchmark with a small decrease of 0.1% in average IPC across all cores, whereas a single NoC with DVFS only saved 50% power. The performance mode is based on the fact that not all network traffic is critical for an application's performance. In terms of general-purpose cache memory based cores, the cache loads are on the critical path of the application's execution time as in-order processor stalls in the case of cache load miss, whereas a cache store operation can be executed in parallel to normal execution. In performance mode, depending on the power budget, both NoCs are switched on at the same time. To avoid interference between critical and non-critical traffic, the data packets associated with cache load operations are injected into the high VF NoC, and data packets associated with cache miss operation are injected into the low VF NoC. During our experimental evaluations performance mode improved average IPC for two applications *h264enc* and *bzip2* by 13% and 11%, respectively, at a cost of 14% increase in NoC power. To summarise, energy efficient mode is useful for low MPKI applications, whereas performance mode is more useful for high MPKI applications. The reliability mode defends against soft error in both data path elements and the control circuit of NoCs. Both NoCs are operated at the lower VF level. For data path protection, the data is sent on the main NoC while byte oriented BCH(8,5) data correction codes are sent on the secondary NoC. The correction bits are added at the source node and data is checked for error at the time of receiving at destination node. The routers at each node are architecturally homogeneous and they are reset to a known state at the start of mode. The control information part of every ejected flit from the ports of the two routers is checked for a equivalence, essentially resulting in a *lock step* processing. However this reliability comes at a cost of a drop in performance, as the NoC is always operated at a lower

frequency.

Finally, in Chapter 6, we proposed a design flow for customising the on-chip interconnect for application-specific MPSoCs, targeting streaming applications. Streaming application are normally realised using a set of pipeline processors and data is passed between different pipeline processors using on-chip communication. The streaming applications such *H264* decoder are often constrained by a minimum throughput requirement (e.g., 30 frames/s). We identified that previous studies on this topic only considered data volume and completely neglected the computational latencies of cores. In our framework, we consider both data volume and pipeline latency to insert an optimum number of express communication links between certain cores so that communication latency is reduced and the pipeline throughput meets a given constraint. The express links have a lower latency than baseline interconnect but carry additional area cost. Simulating every possible combination of express link and baseline communication is prohibitively time consuming because of the exponential design space. To reduce the runtime of the framework, we propose a heuristic with linear time complexity. We compared our framework with previous state-of-art heuristic and showed that our framework was up to 27× faster, while our proposed heuristic always resulted in optimal or near optimal solutions in terms of additional area required for express links.

7.2 Future Directions

In the darkNoC architecture, the routers used in all NoCs are architecturally homogenous. However, it will be interesting to study the effect of heterogeneous router microarchitecture. Some of the router parameters can be changed, such as channel width, pipeline stages, complexity of switch allocators and buffer sizing. Heterogeneous NoC architectures can potentially provide better control over the dark silicon area and power saving tradeoff.

Another way of designing heterogeneous darkNoC would to use both a buffered

and bufferless NoC architectures. We discussed in Chapter 2 that bufferless NoCs offer potential energy saving at the cost of a significant performance penalty for network bound applications. However, with the combination of buffered and bufferless NoCs, the bufferless NoC can be intelligently used only for application phases with low traffic injection.

In the *Malleable NoC* architecture, the routers from each node are only switched according to applications' performance profiling. The transistor ageing phenomenon in modern multicores is attracting the attention of researchers [208]. An interesting future research direction would be to add router ageing awareness in the energy optimisation algorithm such that all routers age gracefully. This would require keeping track of the time for which a given router is activated. Chip user can opt for either an energy efficient policy or an ageing-aware policy, depending on use scenarios.

In *SuperNet* architecture, the NoC is exclusively operated in one of the three possible modes. However, *SuperNet* is can be easily modified to operate in multiple modes at the same time by dividing the physical fabric into multiple logical networks and operating each logical network in modes chosen independently. For example, half of the chip can be used to map applications that are safety-critical whereas the other half can be used to run normal applications.

Bibliography

[1] A. Shafaei Bejestan, Y. Wang, S. Ramadurgam, Y. Xue, P. Bogdan, and M. Pedram, "Analyzing the dark silicon phenomenon in a many-core chip multi-processor under deeply-scaled process technologies," in *Proceedings of the 25th edition on Great Lakes Symposium on VLSI*, pp. 127–132, ACM, 2015.

[2] W. Kim, M. Gupta, G.-Y. Wei, and D. Brooks, "System level analysis of fast, per-core dvfs using on-chip switching regulators," in *High Performance Computer Architecture, 2008. HPCA 2008. IEEE 14th International Symposium on*, pp. 123 –134, 2008.

[3] G. E. Moore *et al.*, "Cramming more components onto integrated circuits," *Proceedings of the IEEE*, vol. 86, no. 1, pp. 82–85, 1998.

[4] B. Towles, J. Grossman, B. Greskamp, and D. E. Shaw, "Unifying on-chip and inter-node switching within the anton 2 network," in *Computer Architecture (ISCA), 2014 ACM/IEEE 41st International Symposium on*, pp. 1–12, IEEE, 2014.

[5] H. Esmaeilzadeh, E. Blem, R. St.Amant, K. Sankaralingam, and D. Burger, "Dark silicon and the end of multicore scaling," in *Computer Architecture (ISCA), 2011 38th Annual International Symposium on*, pp. 365–376, 2011.

[6] S. Borkar and A. A. Chien, "The future of microprocessors," *Commun. ACM*, vol. 54, pp. 67–77, May 2011.

[7] R. H. Dennard, V. Rideout, E. Bassous, and A. Leblanc, "Design of ion-implanted mosfet's with very small physical dimensions," *Solid-State Circuits, IEEE Journal of*, vol. 9, no. 5, pp. 256–268, 1974.

[8] B. K. Daya, C.-H. O. Chen, S. Subramanian, W.-C. Kwon, S. Park, T. Krishna, J. Holt, A. P. Chandrakasan, and L.-S. Peh, "Scorpio: a 36-core research chip demonstrating snoopy coherence on a scalable mesh noc with in-network ordering," in *Computer Architecture (ISCA), 2014 ACM/IEEE 41st International Symposium on*, pp. 25–36, IEEE, 2014.

[9] R. Das, R. Ausavarungnirun, O. Mutlu, A. Kumar, and M. Azimi, "Application-to-core mapping policies to reduce memory system interference in multi-core systems," in *High Performance Computer Architecture (HPCA2013), 2013 IEEE 19th International Symposium on*, pp. 107–118, Feb 2013.

[10] R. Das, *Application-aware on-chip networks*. PhD thesis, The Pennsylvania State University, 2010.

[11] A. K. Mishra, O. Mutlu, and C. R. Das, "A heterogeneous multiple network-on-chip design: an application-aware approach," in *Proceedings of the 50th Annual Design Automation Conference*, DAC '13, (New York, NY, USA), pp. 36:1–36:10, ACM, 2013.

[12] E. Salminen, V. Lahtinen, K. Kuusilinna, and T. Hamalainen, "Overview of bus-based system-on-chip interconnections," in *Circuits and Systems, 2002. ISCAS 2002. IEEE International Symposium on*, vol. 2, pp. II–372, IEEE, 2002.

[13] L. Peh and N. Jerger, "On-chip networks (synthesis lectures on computer architecture)," *Morgan and Claypool, San Rafael*, 2009.

[14] W. J. Dally and B. Towles, "Route packets, not wires: on-chip interconnection networks," in *Design Automation Conference, 2001. Proceedings*, pp. 684–689, IEEE, 2001.

[15] S. Borkar, "Design challenges of technology scaling," *Micro, IEEE*, vol. 19, no. 4, pp. 23–29, 1999.

[16] S. Borkar, T. Karnik, S. Narendra, J. Tschanz, A. Keshavarzi, and V. De, "Parameter variations and impact on circuits and microarchitecture," in *Proceedings of the 40th annual Design Automation Conference*, pp. 338–342, ACM, 2003.

[17] S. Garg, D. Marculescu, R. Marculescu, and U. Ogras, "Technology-driven limits on dvfs controllability of multiple voltage-frequency island designs: A system-level perspective," in *Design Automation Conference, 2009. DAC '09. 46th ACM/IEEE*, pp. 818–821, 2009.

[18] H. Esmaeilzadeh, *Approximate Acceleration for a Post Multicore Era*. PhD thesis, 2013.

[19] J. Henkel, H. Khdr, S. Pagani, and M. Shafique, "New trends in dark silicon," in *Design Automation Conference (DAC), 2015 52nd ACM/EDAC/IEEE*, pp. 1–6, IEEE, 2015.

[20] P. Bose, "Is dark silicon real?: technical perspective," *Commun. ACM*, vol. 56, pp. 92–92, Feb. 2013.

[21] M. Shafique, S. Garg, J. Henkel, and D. Marculescu, "The eda challenges in the dark silicon era," in *Design Automation Conference (DAC), 2014 51st ACM/EDAC/IEEE*, pp. 1–6, IEEE, 2014.

[22] N. Goulding-Hotta, J. Sampson, G. Venkatesh, S. Garcia, J. Auricchio, P. Huang, M. Arora, S. Nath, V. Bhatt, J. Babb, S. Swanson, and M. Taylor, "The greendroid mobile application processor: An architecture for silicon's dark future," *Micro, IEEE*, vol. 31, no. 2, pp. 86–95, 2011.

[23] G. Venkatesh, J. Sampson, N. Goulding-Hotta, S. K. Venkata, M. B. Taylor, and S. Swanson, "Qscores: Trading dark silicon for scalable energy efficiency with quasi-specific cores," in *Proceedings of the 44th Annual IEEE/ACM International Symposium on Microarchitecture*, pp. 163–174, ACM, 2011.

[24] M. B. Taylor, "Is dark silicon useful?: Harnessing the four horsemen of the coming dark silicon apocalypse," in *Proceedings of the 49th Annual Design Automation Conference*, DAC '12, (New York, NY, USA), pp. 1131–1136, ACM, 2012.

[25] J. Allred, S. Roy, and K. Chakraborty, "Designing for dark silicon: a methodological perspective on energy efficient systems," in *Proceedings of the 2012 ACM/IEEE international symposium on Low power electronics and design*, ISLPED '12, (New York, NY, USA), pp. 255–260, ACM, 2012.

[26] N. Pinckney, K. Sewell, R. Dreslinski, D. Fick, T. Mudge, D. Sylvester, and D. Blaauw, "Assessing the performance limits of parallelized near-threshold computing," in *Design Automation Conference (DAC), 2012 49th ACM/EDAC/IEEE*, pp. 1143–1148, 2012.

[27] K. Swaminathan, E. Kultursay, V. Saripalli, V. Narayanan, M. Kandemir, and S. Datta, "Steep slope devices: From dark to dim silicon," *IEEE Micro*, vol. 99, no. PrePrints, p. 1, 2013.

[28] K. Swaminathan, E. Kultursay, V. Saripalli, V. Narayanan, M. T. Kandemir, and S. Datta, "Steep-slope devices: From dark to dim silicon," *Micro, IEEE*, vol. 33, no. 5, pp. 50–59, 2013.

[29] F. Kriebel, S. Rehman, D. Sun, M. Shafique, and J. Henkel, "Aser: Adaptive soft error resilience for reliability-heterogeneous processors in the dark silicon era," in *Design Automation Conference (DAC), 2014 51st ACM/EDAC/IEEE*, pp. 1–6, IEEE, 2014.

[30] H. Esmaeilzadeh, A. Sampson, L. Ceze, and D. Burger, "Neural acceleration for general-purpose approximate programs," in *Proceedings of the 2012 45th Annual IEEE/ACM International Symposium on Microarchitecture*, pp. 449–460, IEEE Computer Society, 2012.

[31] Y. Hoskote, S. Vangal, A. Singh, N. Borkar, and S. Borkar, "A 5-ghz mesh interconnect for a teraflops processor," *IEEE Micro*, no. 5, pp. 51–61, 2007.

[32] A. Carroll and G. Heiser, "An analysis of power consumption in a smartphone.," in *USENIX annual technical conference*, vol. 14, 2010.

[33] D. A. Neamen, *Semiconductor physics and devices*. McGraw-Hill Higher Education, 2003.

[34] O. Coudert, "Gate sizing for constrained delay/power/area optimization," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 5, no. 4, pp. 465–472, 1997.

[35] M. R. Berkelaar and J. A. Jess, "Gate sizing in mos digital circuits with linear programming," in *Proceedings of the conference on European design automation*, pp. 217–221, IEEE Computer Society Press, 1990.

[36] T. Reimann, C. C. Sze, and R. Reis, "Gate sizing and threshold voltage assignment for high performance microprocessor designs," in *Design Automation Conference (ASP-DAC), 2015 20th Asia and South Pacific*, pp. 214–219, IEEE, 2015.

[37] M. Donno, A. Ivaldi, L. Benini, and E. Macii, "Clock-tree power optimization based on rtl clock-gating," in *Design Automation Conference, 2003. Proceedings*, pp. 622–627, IEEE, 2003.

[38] Q. Wu, M. Pedram, and X. Wu, "Clock-gating and its application to low power design of sequential circuits," *Circuits and Systems I: Fundamental Theory and Applications, IEEE Transactions on*, vol. 47, no. 3, pp. 415–420, 2000.

[39] Z. Hu, A. Buyuktosunoglu, V. Srinivasan, V. Zyuban, H. Jacobson, and P. Bose, "Microarchitectural techniques for power gating of execution units," in *Proceedings of the 2004 international symposium on Low power electronics and design*, pp. 32–37, ACM, 2004.

[40] A. Lungu, P. Bose, A. Buyuktosunoglu, and D. J. Sorin, "Dynamic power gating with quality guarantees," in *Proceedings of the 2009 ACM/IEEE international symposium on Low power electronics and design*, pp. 377–382, ACM, 2009.

[41] C. J. Akl, R. Ayoubi, M. Bayoumi, *et al.*, "An effective staggered-phase damping technique for suppressing power-gating resonance noise during mode transition," in *Quality of Electronic Design, 2009. ISQED 2009. Quality Electronic Design*, pp. 116–119, IEEE, 2009.

[42] D. Liu and C. Svensson, "Trading speed for low power by choice of supply and threshold voltages," *Solid-State Circuits, IEEE Journal of*, vol. 28, no. 1, pp. 10–17, 1993.

[43] A. Jerraya and W. Wolf, *Multiprocessor Systems-on-chips*. Electronics & Electrical, Morgan Kaufmann, 2005.

[44] S. H. Kulkarni, A. N. Srivastava, and D. Sylvester, "A new algorithm for improved vdd assignment in low power dual vdd systems," in *Low Power Electronics and Design, 2004. ISLPED'04. Proceedings of the 2004 International Symposium on*, pp. 200–205, IEEE, 2004.

[45] B. H. Calhoun and A. P. Chandrakasan, "Ultra-dynamic voltage scaling (udvs) using sub-threshold operation and local voltage dithering," *Solid-State Circuits, IEEE Journal of*, vol. 41, no. 1, pp. 238–245, 2006.

[46] T. Miller, X. Pan, R. Thomas, N. Sedaghati, and R. Teodorescu, "Booster: Reactive core acceleration for mitigating the effects of process variation and application imbalance in low-voltage chips," in *High Performance Computer Architecture (HPCA), 2012 IEEE 18th International Symposium on*, pp. 1–12, Feb 2012.

[47] M. Takahashi, M. Hamada, T. Nishikawa, H. Arakida, T. Fujita, F. Hatori, S. Mita, K. Suzuki, A. Chiba, T. Terazawa, *et al.*, "A 60-mw mpeg4 video codec using clustered voltage scaling with variable supply-voltage scheme," *Solid-State Circuits, IEEE Journal of*, vol. 33, no. 11, pp. 1772–1780, 1998.

[48] T. D. Burd, T. Pering, A. J. Stratakos, R. W. Brodersen, *et al.*, "A dynamic voltage scaled microprocessor system," *Solid-State Circuits, IEEE Journal of*, vol. 35, no. 11, pp. 1571–1580, 2000.

[49] T. Kuroda, K. Suzuki, S. Mita, T. Fujita, F. Yamane, F. Sano, A. Chiba, Y. Watanabe, K. Matsuda, T. Maeda, *et al.*, "Variable supply-voltage scheme for low-power high-speed cmos digital design," *Solid-State Circuits, IEEE Journal of*, vol. 33, no. 3, pp. 454–462, 1998.

[50] J. M. D. PEDDERSEN, *RUN-TIME ENERGY-DRIVEN OPTIMISATION OF EMBEDDED SYSTEMS: A COMPLETE SOLUTION*. PhD thesis, UNSW Australia, 2008.

[51] P. Salihundam, S. Jain, T. Jacob, S. Kumar, V. Erraguntla, Y. Hoskote, S. R. Vangal, G. Ruhl, and N. Borkar, "A 2 tb/s 6 x 4 mesh network for a single-chip cloud computer with dvfs in 45 nm cmos.," *J. Solid-State Circuits*, vol. 46, no. 4, pp. 757–766, 2011.

[52] M. Ang, R. Salem, and A. Taylor, "An on-chip voltage regulator using switched decoupling capacitors," in *Solid-State Circuits Conference, 2000. Digest of Technical Papers. ISSCC. 2000 IEEE International*, pp. 438–439, IEEE, 2000.

[53] A. Ansari, A. Mishra, J. Xu, and J. Torrellas, "Tangle: Route-oriented dynamic voltage minimization for variation-afflicted, energy-efficient on-chip networks," *International Symposium on High Performance Computer Architecture*, 2014.

[54] H. R. Ghasemi, A. A. Sinkar, M. J. Schulte, and N. S. Kim, "Cost-effective power delivery to support per-core voltage domains for power-constrained processors," in *Proceedings of the 49th Annual Design Automation Conference*, pp. 56–61, ACM, 2012.

[55] J. Kim, S. Yoo, and C.-M. Kyung, "Program phase-aware dynamic voltage scaling under variable computational workload and memory stall environment," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 30, no. 1, pp. 110–123, 2011.

[56] T. Nakamura, H. Matsutani, M. Koibuchi, K. Usami, and H. Amano, "Fine-grained power control using a multi-voltage variable pipeline router," in *Embedded Multicore Socs (MCSoC), 2012 IEEE 6th International Symposium on*, pp. 59–66, 2012.

[57] R. G. Dreslinski, M. Wieckowski, D. Blaauw, D. Sylvester, and T. Mudge, "Near-threshold computing: Reclaiming moore's law through energy efficient integrated circuits," *Proceedings of the IEEE*, vol. 98, no. 2, pp. 253–266, 2010.

[58] D. E. Nikonov, I. Young, *et al.*, "Overview of beyond-cmos devices and a uniform methodology for their benchmarking," *Proceedings of the IEEE*, vol. 101, no. 12, pp. 2498–2533, 2013.

[59] A. M. Ionescu, L. De Michielis, N. Dagtekin, G. Salvatore, J. Cao, A. Rusu, and S. Bartsch, "Ultra low power: Emerging devices and their benefits for integrated circuits," in *Electron Devices Meeting (IEDM), 2011 IEEE International*, pp. 16–1, IEEE, 2011.

[60] S. Das, "Itrs assessment and benchmarking of emerging logic devices," *Emerging Nanoelectronic Devices*, pp. 405–416, 2014.

[61] A. C. Seabaugh and Q. Zhang, "Low-voltage tunnel transistors for beyond cmos logic," *Proceedings of the IEEE*, vol. 98, no. 12, pp. 2095–2110, 2010.

[62] J. Zhan, J. Ouyang, F. Ge, J. Zhao, and Y. Xie, "Dimnoc: A dim silicon approach towards power-efficient on-chip network," *DAC. ACM*, 2015.

[63] C. W. Smullen, V. Mohan, A. Nigam, S. Gurumurthi, and M. R. Stan, "Relaxing non-volatility for fast and energy-efficient stt-ram caches," in *High Performance Computer Architecture (HPCA), 2011 IEEE 17th International Symposium on*, pp. 50–61, IEEE, 2011.

[64] W. Wolf, "The future of multiprocessor systems-on-chips," in *Design Automation Conference, 2004. Proceedings. 41st*, pp. 681–685, IEEE, 2004.

[65] A. Munir, S. Ranka, and A. Gordon-Ross, "High-performance energy-efficient multicore embedded computing," *Parallel and Distributed Systems, IEEE Transactions on*, vol. 23, no. 4, pp. 684–700, 2012.

[66] E. Waingold, M. Taylor, D. Srikrishna, V. Sarkar, W. Lee, V. Lee, J. Kim, M. Frank, P. Finch, R. Barua, *et al.*, "Baring it all to software: Raw machines," *Computer*, vol. 30, no. 9, pp. 86–93, 1997.

[67] "EZChip TileGX Multicore Architecture." `http://www.tilera.com/products/?ezchip=585&spage=614`.

[68] "Intel IXP4XX Netrwork Processors." `http://www.intel.com/content/www/us/en/intelligent-systems/previous-generation/intel-ixp4xx-intel-network-processor-product-line.html`.

[69] "IBM Cell Architecture Project." `http://researcher.watson.ibm.com/researcher/view_group.php?id=2649`.

[70] "Texas Instruments KeyStone Multicore Architecture." `http://www.ti.com/lsds/ti/arm/keystone/overview.page?paramCriteria=no`.

[71] "Nvidia Tegra Processors." `http://www.nvidia.com/object/tegra.html`.

[72] "Qualcomm Snapdragon Platform." `https://www.qualcomm.com/products/snapdragon`.

[73] "NXP TriMedia Processor." `http://www.nxp.com/products/media_processors/series/PNX17XX_SERIES.html`.

[74] H. Javaid, A. Ignjatovic, and S. Parameswaran, "Rapid design space exploration of application specific heterogeneous pipelined multiprocessor systems," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 29, no. 11, pp. 1777–1789, 2010.

[75] H. G. Lee, N. Chang, U. Y. Ogras, and R. Marculescu, "On-chip communication architecture exploration: A quantitative evaluation of point-to-point, bus, and network-on-chip approaches," *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, vol. 12, no. 3, p. 23, 2007.

[76] M. Gasteier and M. Glesner, "Bus-based communication synthesis on system-level," in *System Synthesis, 1996. Proceedings., 9th International Symposium on*, pp. 65–70, IEEE, 1996.

[77] J.-M. Daveau, T. B. Ismail, and A. A. Jerraya, "Synthesis of system-level communication by an allocation-based approach," in *Proceedings of the 8th international symposium on System synthesis*, pp. 150–155, ACM, 1995.

[78] R. Kumar, V. Zyuban, and D. M. Tullsen, "Interconnections in multi-core architectures: Understanding mechanisms, overheads and scaling," in *Computer Architecture, 2005. ISCA'05. Proceedings. 32nd International Symposium on*, pp. 408–419, IEEE, 2005.

[79] "ARM AMBA Interconnect Specification." `http://www.arm.com/products/system-ip/amba-specifications.php`.

[80] "IBM CoreConnect Bus Technology." `http://www.xilinx.com/products/intellectual-property/dr_pcentral_coreconnect.html`.

[81] "Xtensa Processors." `http://ip.cadence.com/ipportfolio/tensilica-ip/xtensa-customizable`.

[82] R. A. Shafik, P. Rosinger, and B. M. Al-Hashimi, "Mpeg-based performance comparison between network-on-chip and amba mpsoc," 2008.

[83] S. Medardoni, *Driving the Network-on-Chip Revolution to Remove the Interconnect Bottleneck in Nanoscale Multi-Processor Systems-on-Chip*. PhD thesis, Università degli studi di Ferrara, 2009.

[84] G. Passas, M. Katevenis, and 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, 2012.

[85] J. Yoo, D. Lee, S. Yoo, and K. Choi, "Communication architecture synthesis of cascaded bus matrix," in *Design Automation Conference, 2007. ASP-DAC'07. Asia and South Pacific*, pp. 171–177, IEEE, 2007.

[86] S. Murali, L. Benini, and G. De Micheli, "An application-specific design methodology for on-chip crossbar generation," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 26, pp. 1283–1296, July 2007.

[87] 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 *Parallel and Distributed Processing Symposium, 2006. IPDPS 2006. 20th International*, pp. 10–pp, IEEE, 2006.

[88] S. Borkar, "Thousand core chips: a technology perspective," in *Proceedings of the 44th annual Design Automation Conference*, pp. 746–749, ACM, 2007.

[89] S. Kumar, A. Jantsch, J.-P. Soininen, M. Forsell, M. Millberg, J. Oberg, K. Tiensyrja, and A. Hemani, "A network on chip architecture and design methodology," in *VLSI, 2002. Proceedings. IEEE Computer Society Annual Symposium on*, pp. 105–112, IEEE, 2002.

[90] K. Aisopos, "Fault tolerant architectures for on-chip networks," 2012.

[91] J. CHAN, *ENERGY-AWARE SYNTHESIS FOR NETWORKS ON CHIP ARCHITECTURES*. PhD thesis, UNSW Australia, 2007.

[92] R. Beraha, I. Walter, I. Cidon, and A. Kolodny, "Leveraging application-level requirements in the design of a noc for a 4g soc - a case study," in *Design, Automation Test in Europe Conference Exhibition (DATE), 2010*, pp. 1408–1413, March 2010.

[93] C. Seiculescu, S. Murali, L. Benini, and G. De Micheli, "Noc topology synthesis for supporting shutdown of voltage islands in socs," in *Design Automation Conference, 2009. DAC '09. 46th ACM/IEEE*, pp. 822–825, July 2009.

[94] D. Bertozzi, A. Jalabert, S. Murali, R. Tamhankar, S. Stergiou, L. Benini, and G. De Micheli, "Noc synthesis flow for customized domain specific multiprocessor systems-on-chip," *Parallel and Distributed Systems, IEEE Transactions on*, vol. 16, no. 2, pp. 113–129, 2005.

[95] K. Srinivasan and K. S. Chatha, "A low complexity heuristic for design of custom network-on-chip architectures," in *Proceedings of the conference on Design, automation and test in Europe: Proceedings*, pp. 130–135, European Design and Automation Association, 2006.

[96] S. Ma, N. Enright Jerger, and Z. Wang, "Dbar: an efficient routing algorithm to support multiple concurrent applications in networks-on-chip," in *Proceedings of the 38th annual international symposium on Computer architecture*, ISCA '11, (New York, NY, USA), pp. 413–424, ACM, 2011.

[97] T. Nesson and S. L. Johnsson, "Romm routing on mesh and torus networks," in *Proceedings of the seventh annual ACM symposium on Parallel algorithms and architectures*, pp. 275–287, ACM, 1995.

[98] H.-L. Chao, Y.-R. Chen, S.-Y. Tung, P.-A. Hsiung, and S.-J. Chen, "Congestion-aware scheduling for noc-based reconfigurable systems," in *Design, Automation Test in Europe Conference Exhibition (DATE), 2012*, pp. 1561–1566, March 2012.

[99] C.-L. Chou and R. Marculescu, "Contention-aware application mapping for network-on-chip communication architectures," in *Computer Design, 2008. ICCD 2008. IEEE International Conference on*, pp. 164–169, Oct 2008.

[100] M. Palesi, R. Holsmark, S. Kumar, and V. Catania, "A methodology for design of application specific deadlock-free routing algorithms for noc systems," in *Proceedings of the 4th international conference on Hardware/software codesign and system synthesis*, pp. 142–147, ACM, 2006.

[101] W. J. Dally and B. P. Towles, *Principles and practices of interconnection networks*. Elsevier, 2004.

[102] P. Kermani and L. Kleinrock, "Virtual cut-through: A new computer communication switching technique," *Computer Networks (1976)*, vol. 3, no. 4, pp. 267–286, 1979.

[103] C. Seiculescu, S. Murali, L. Benini, and G. De Micheli, "A method to remove deadlocks in networks-on-chips with wormhole flow control," in *Proceedings of the Conference on Design, Automation and Test in Europe*, pp. 1625–1628, European Design and Automation Association, 2010.

[104] G. Dimitrakopoulos, A. Psarras, and I. Seitanidis, *Microarchitecture of Network-on-Chip Routers*. Springer, 2014.

[105] D. U. Becker, *Efficient microarchitecture for network-on-chip routers*. PhD thesis, Stanford University, 2012.

[106] S. Park, T. Krishna, C.-H. Chen, B. Daya, A. Chandrakasan, and L.-S. Peh, "Approaching the theoretical limits of a mesh noc with a 16-node chip prototype in 45nm soi," in *Proceedings of the 49th Annual Design Automation Conference*, DAC '12, (New York, NY, USA), pp. 398–405, ACM, 2012.

[107] D. Wentzlaff, P. Griffin, H. Hoffmann, L. Bao, B. Edwards, C. Ramey, M. Mattina, C.-C. Miao, J. F. Brown III, and A. Agarwal, "On-chip interconnection architecture of the tile processor," *IEEE Micro*, vol. 27, pp. 15–31, Sept. 2007.

[108] S. Phillips, "M7: Next generation sparc," in *Hot Chips: A Symposium on High Performance Chips*, 2014.

[109] T. Krishna, C.-H. O. Chen, W.-C. Kwon, and L.-S. Peh, "Smart: Single-cycle multihop traversals over a shared network on chip," *Micro, IEEE*, vol. 34, no. 3, pp. 43–56, 2014.

[110] J. Howard, S. Dighe, S. Vangal, G. Ruhl, N. Borkar, S. Jain, V. Erraguntla, M. Konow, M. Riepen, M. Gries, G. Droege, T. Lund-Larsen, S. Steibl, S. Borkar, V. De, and R. Van Der Wijngaart, "A 48-core ia-32 processor in 45 nm cmos using on-die message-passing and dvfs for performance and power scaling," *Solid-State Circuits, IEEE Journal of*, vol. 46, no. 1, pp. 173–183, 2011.

[111] "Sonics ." `http://www.sonicsinc.com/`.

[112] "Arteris NoC." `http://www.arteris.com/`, 2015.

[113] L. Shang, L.-S. Peh, and N. Jha, "Dynamic voltage scaling with links for power optimization of interconnection networks," in *High-Performance Computer Architecture, 2003. HPCA-9 2003. Proceedings. The Ninth International Symposium on*, pp. 91–102, 2003.

[114] P. Bogdan, R. Marculescu, S. Jain, and R. Gavila, "An optimal control approach to power management for multi-voltage and frequency islands multiprocessor platforms under highly variable workloads," in *Networks on Chip (NoCS), 2012 Sixth IEEE/ACM International Symposium on*, pp. 35–42, 2012.

[115] A. Mishra, R. Das, S. Eachempati, R. Iyer, N. Vijaykrishnan, and C. Das, "A case for dynamic frequency tuning in on-chip networks," in *Microarchitecture, 2009. MICRO-42. 42nd Annual IEEE/ACM International Symposium on*, pp. 292–303, 2009.

[116] U. Y. Ogras, R. Marculescu, D. Marculescu, and E. G. Jung, "Design and management of voltage-frequency island partitioned networks-on-chip," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 17, no. 3, pp. 330–341, 2009.

[117] R. Hesse and N. E. Jerger, "Improving dvfs in nocs with coherence prediction," in *Networks on Chip (NoCS), 2015 9th IEEE/ACM International Symposium on*, 2015.

[118] X. Chen, Z. Xu, H. Kim, P. V. Gratz, J. Hu, M. Kishinevsky, U. Ogras, and R. Ayoub, "Dynamic voltage and frequency scaling for shared resources in multicore processor designs," in *Proceedings of the 50th Annual Design Automation Conference*, DAC '13, (New York, NY, USA), pp. 114:1–114:7, ACM, 2013.

[119] J. Zhan, N. Stoimenov, J. Ouyang, L. Thiele, V. Narayanan, and Y. Xie, "Designing energy-efficient noc for real-time embedded systems through slack optimization," in *Proceedings of the 50th Annual Design Automation Conference*, p. 37, ACM, 2013.

[120] L. Chen, *Design of low-power and resource-efficient on-chip networks.* PhD thesis, University of Southern California, 2014.

[121] V. Soteriou and L.-S. Peh, "Exploring the design space of self-regulating power-aware on/off interconnection networks," *Parallel and Distributed Systems, IEEE Transactions on*, vol. 18, no. 3, pp. 393–408, 2007.

[122] H. Matsutani, M. Koibuchi, D. Wang, and H. Amano, "Adding slow-silent virtual channels for low-power on-chip networks," in *Networks-on-Chip, 2008. NoCS 2008. Second ACM/IEEE International Symposium on*, pp. 23–32, 2008.

[123] H. Matsutani, M. Koibuchi, D. Ikebuchi, K. Usami, H. Nakamura, and H. Amano, "Performance, area, and power evaluations of ultrafine-grained run-time power-gating routers for cmps," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 30, no. 4, pp. 520–533, 2011.

[124] G. Kim, J. Kim, and S. Yoo, "Flexibuffer: Reducing leakage power in on-chip network routers," in *Design Automation Conference (DAC), 2011 48th ACM/EDAC/IEEE*, pp. 936–941, 2011.

[125] R. Parikh, R. Das, and V. Bertacco, "Power-aware nocs through routing and topology reconfiguration," in *Design Automation Conference (DAC), 2014 51st ACM/EDAC/IEEE*, pp. 1–6, IEEE, 2014.

[126] A. Samih, R. Wang, A. Krishna, C. Maciocco, C. Tai, and Y. Solihin, "Energy-efficient interconnect via router parking," in *High Performance Computer Architecture (HPCA2013), 2013 IEEE 19th International Symposium on*, pp. 508–519, 2013.

[127] L. Chen and T. M. Pinkston, "Nord: Node-router decoupling for effective power-gating of on-chip routers," in *Proceedings of the 2012 45th Annual IEEE/ACM International Symposium on Microarchitecture*, MICRO '12, (Washington, DC, USA), pp. 270–281, IEEE Computer Society, 2012.

[128] 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 '13, (New York, NY, USA), pp. 320–331, ACM, 2013.

[129] T. Moscibroda and O. Mutlu, "A case for bufferless routing in on-chip networks," *SIGARCH Comput. Archit. News*, vol. 37, pp. 196–207, June 2009.

[130] C. Fallin, C. Craik, and O. Mutlu, "Chipper: A low-complexity bufferless deflection router," in *High Performance Computer Architecture (HPCA), 2011 IEEE 17th International Symposium on*, pp. 144–155, Feb 2011.

[131] M. Shafique, S. Garg, T. Mitra, S. Parameswaran, and J. Henkel, "Dark silicon as a challenge for hardware/software co-design: Invited special session paper," in *Proceedings of the 2014 International Conference on Hardware/Software Codesign and System Synthesis*, CODES '14, (New York, NY, USA), pp. 13:1–13:10, ACM, 2014.

[132] J. Cong, M. A. Ghodrat, M. Gill, B. Grigorian, and G. Reinman, "Architecture support for accelerator-rich cmps," in *Proceedings of the 49th Annual Design Automation Conference*, pp. 843–849, ACM, 2012.

[133] Y. Turakhia, B. Raghunathan, S. Garg, and D. Marculescu, "Hades: architectural synthesis for heterogeneous dark silicon chip multi-processors," in *Proceedings of the 50th Annual Design Automation Conference*, DAC '13, (New York, NY, USA), pp. 173:1–173:7, ACM, 2013.

[134] D. Gnad, M. Shafique, F. Kriebel, S. Rehman, D. Sun, and J. Henkel, "Hayat: harnessing dark silicon and variability for aging deceleration and balancing," in *Design Automation Conference (DAC), 2015 52nd ACM/EDAC/IEEE*, pp. 1–6, IEEE, 2015.

[135] M. Shafique, S. Garg, T. Mitra, S. Parameswaran, and J. Henkel, "Dark silicon as a challenge for hardware/software co-design: Invited special session paper," in *Proceedings of the 2014 International Conference on Hardware/Software Codesign and System Synthesis*, p. 13, ACM, 2014.

[136] T. S. Muthukaruppan, M. Pricopi, V. Venkataramani, T. Mitra, and S. Vishin, "Hierarchical power management for asymmetric multi-core in dark silicon era," in *Proceedings of the 50th Annual Design Automation Conference*, p. 174, ACM, 2013.

[137] S. Pagani, H. Khdr, W. Munawar, J.-J. Chen, M. Shafique, M. Li, and J. Henkel, "Tsp: thermal safe power: efficient power budgeting for many-core systems in dark silicon," in *Proceedings of the 2014 International Conference on Hardware/Software Codesign and System Synthesis*, p. 10, ACM, 2014.

[138] H. Khdr, S. Pagani, M. Shafique, and J. Henkel, "Thermal constrained resource management for mixed ilp-tlp workloads in dark silicon chips," in *Proceedings of the 52nd Annual Design Automation Conference*, p. 179, ACM, 2015.

[139] W. Huang, K. Rajamani, M. R. Stan, and K. Skadron, "Scaling with design constraints: Predicting the future of big chips," *IEEE Micro*, no. 4, pp. 16–29, 2011.

[140] L. Thiele, I. Bacivarov, W. Haid, and K. Huang, "Mapping applications to tiled multiprocessor embedded systems," in *Application of Concurrency to System*

Design, 2007. ACSD 2007. Seventh International Conference on, pp. 29–40, July 2007.

[141] J. Teich, "Hardware/software codesign: The past, the present, and predicting the future," *Proceedings of the IEEE*, vol. 100, pp. 1411–1430, May 2012.

[142] S. H. Bokhari, "On the mapping problem," *IEEE Trans. Comput.*, vol. 30, pp. 207–214, Mar. 1981.

[143] S. Stuijk, T. Basten, M. Geilen, and H. Corporaal, "Multiprocessor resource allocation for throughput-constrained synchronous dataflow graphs," in *Design Automation Conference, 2007. DAC '07. 44th ACM/IEEE*, pp. 777–782, June 2007.

[144] J. Zhu, I. Sander, and A. Jantsch, "Constrained global scheduling of streaming applications on mpsocs," in *Design Automation Conference (ASP-DAC), 2010 15th Asia and South Pacific*, pp. 223–228, Jan 2010.

[145] P. K. F. Hölzenspies, J. L. Hurink, J. Kuper, and G. J. M. Smit, "Run-time spatial mapping of streaming applications to a heterogeneous multi-processor system-on-chip (mpsoc)," in *Proceedings of the Conference on Design, Automation and Test in Europe*, DATE '08, (New York, NY, USA), pp. 212–217, ACM, 2008.

[146] M. Al Faruque, R. Krist, and J. Henkel, "Adam: Run-time agent-based distributed application mapping for on-chip communication," in *Design Automation Conference, 2008. DAC 2008. 45th ACM/IEEE*, pp. 760–765, June 2008.

[147] H. Salamy and J. Ramanujam, "An effective solution to task scheduling and memory partitioning for multiprocessor system-on-chip," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 31, pp. 717–725, May 2012.

[148] J. Castrillon, A. Tretter, R. Leupers, and G. Ascheid, "Communication-aware mapping of kpn applications onto heterogeneous mpsocs," in *Proceedings of the 49th Annual Design Automation Conference*, DAC '12, (New York, NY, USA), pp. 1266–1271, ACM, 2012.

[149] S. Le Beux, G. Bois, G. Nicolescu, Y. Bouchebaba, M. Langevin, and P. Paulin, "Combining mapping and partitioning exploration for noc-based embedded systems," *J. Syst. Archit.*, vol. 56, pp. 223–232, July 2010.

[150] S. Murali, M. Coenen, A. Radulescu, K. Goossens, and G. De Micheli, "Mapping and configuration methods for multi-use-case networks on chips," in *Design Automation, 2006. Asia and South Pacific Conference on*, pp. 6 pp.–, Jan 2006.

[151] C. Zimmer and F. Mueller, "Low contention mapping of real-time tasks onto tilepro 64 core processors," in *Proceedings of the 2012 IEEE 18th Real Time and Embedded Technology and Applications Symposium*, RTAS '12, (Washington, DC, USA), pp. 131–140, IEEE Computer Society, 2012.

[152] S. Le Beux, G. Nicolescu, G. Bois, Y. Bouchebaba, M. Langevin, and P. Paulin, "Optimizing configuration and application mapping for mpsoc architectures," in *Adaptive Hardware and Systems, 2009. AHS 2009. NASA/ESA Conference on*, pp. 474–481, July 2009.

[153] Y. Jan and L. Jóźwiak, "Communication and memory architecture design of application-specific high-end multiprocessors," *VLSI Des.*, vol. 2012, pp. 12:12–12:12, Jan. 2012.

[154] K. Lahiri, A. Raghunathan, and S. Dey, "Design space exploration for optimizing on-chip communication architectures," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 23, pp. 952–961, June 2004.

[155] P.-K. Huang, M. Hashemi, and S. Ghiasi, "System-level performance estimation for application-specific mpsoc interconnect synthesis," in *Application Specific Processors, 2008. SASP 2008. Symposium on*, pp. 95–100, June 2008.

[156] S. Stergiou, F. Angiolini, S. Carta, L. Raffo, D. Bertozzi, and G. De Micheli, "Xpipes lite: a synthesis oriented design library for networks on chips," in *Design, Automation and Test in Europe, 2005. Proceedings*, pp. 1188–1193 Vol. 2, March 2005.

[157] J. Chan and S. Parameswaran, "Nocgen:a template based reuse methodology for networks on chip architecture," in *VLSI Design, 2004. Proceedings. 17th International Conference on*, pp. 717–720, 2004.

[158] K. Goossens and A. Hansson, "The aethereal network on chip after ten years: Goals, evolution, lessons, and future," in *Proceedings of the 47th Design Automation Conference*, DAC '10, (New York, NY, USA), pp. 306–311, ACM, 2010.

[159] H. Javaid, X. He, A. Ignjatovic, and S. Parameswaran, "Optimal synthesis of latency and throughput constrained pipelined mpsocs targeting streaming applications," in *Hardware/Software Codesign and System Synthesis (CODES+ISSS), 2010 IEEE/ACM/IFIP International Conference on*, pp. 75–84, Oct 2010.

[160] J. S. Kim, M. B. Taylor, J. Miller, and D. Wentzlaff, "Energy characterization of a tiled architecture processor with on-chip networks," in *Proceedings of the 2003 international symposium on Low power electronics and design*, ISLPED '03, (New York, NY, USA), pp. 424–427, ACM, 2003.

[161] M. Taylor, "A landscape of the new dark silicon design regime," *Micro, IEEE*, vol. PP, no. 99, pp. 1–1, 2013.

[162] A. Pullini, F. Angiolini, S. Murali, D. Atienza, G. De Micheli, and L. Benini, "Bringing nocs to 65 nm," *Micro, IEEE*, vol. 27, no. 5, pp. 75–85, 2007.

[163] K. Sankaralingam, R. Nagarajan, H. Liu, C. Kim, J. Huh, D. Burger, S. Keckler, and C. Moore, "Exploiting ilp, tlp, and dlp with the polymorphous trips architecture," *Micro, IEEE*, vol. 23, no. 6, pp. 46–51, 2003.

[164] J. Allred, S. Roy, and K. Chakraborty, "Dark silicon aware multicore systems: Employing design automation with architectural insight," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. PP, no. 99, pp. 1–1, 2013.

[165] K. Chakraborty and S. Roy, "Architecturally homogeneous power-performance heterogeneous multicore systems," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 21, no. 4, pp. 670–679, 2013.

[166] A. Strano, D. Ludovici, and D. Bertozzi, "A library of dual-clock fifos for cost-effective and flexible mpsoc design," in *Embedded Computer Systems (SAMOS), 2010 International Conference on*, pp. 20–27, 2010.

[167] A. Bakhoda, J. Kim, and T. Aamodt, "Throughput-effective on-chip networks for manycore accelerators," in *Microarchitecture (MICRO), 2010 43rd Annual IEEE/ACM International Symposium on*, pp. 421–432, Dec 2010.

[168] A. K. Mishra, N. Vijaykrishnan, and C. R. Das, "A case for heterogeneous on-chip interconnects for cmps," in *Proceedings of the 38th annual international symposium on Computer architecture*, ISCA '11, (New York, NY, USA), pp. 389–400, ACM, 2011.

[169] G. Liang and A. Jantsch, "Adaptive power management for the on-chip communication network," in *Digital System Design: Architectures, Methods and Tools, 2006. DSD 2006. 9th EUROMICRO Conference on*, pp. 649–656, 2006.

[170] 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," in *Proceedings of the The 51st Annual Design Automation Conference on Design Automation Conference*, DAC '14, (New York, NY, USA), pp. 161:1–161:6, ACM, 2014.

[171] Synopsys, "Design Solutions for 20nm and Beyond, White Paper." `http://www.synopsys.com/`.

[172] J. Warnock, "Circuit design challenges at the 14nm technology node," in *Proceedings of the 48th Design Automation Conference*, DAC '11, (New York, NY, USA), pp. 464–467, ACM, 2011.

[173] "Semico Research - How Reducing Verification and Validation Time Can Improve Profitability." `http://www.semico.com/`.

[174] A. Prodromou, A. Panteli, C. Nicopoulos, and Y. Sazeides, "Nocalert: An on-line and real-time fault detection mechanism for network-on-chip architectures," in *Microarchitecture (MICRO), 2012 45th Annual IEEE/ACM International Symposium on*, pp. 60–71, Dec 2012.

[175] M. Radetzki, C. Feng, X. Zhao, and A. Jantsch, "Methods for fault tolerance in networks-on-chip," *ACM Comput. Surv.*, vol. 46, pp. 8:1–8:38, July 2013.

[176] J. Balfour and W. J. Dally, "Design tradeoffs for tiled cmp on-chip networks," in *Proceedings of the 20th Annual International Conference on Supercomputing*, ICS '06, (New York, NY, USA), pp. 187–198, ACM, 2006.

[177] K. Constantinides, S. Plaza, J. Blome, B. Zhang, V. Bertacco, S. Mahlke, T. Austin, and M. Orshansky, "Bulletproof: a defect-tolerant cmp switch architecture," in *High-Performance Computer Architecture, 2006. The Twelfth International Symposium on*, pp. 5–16, Feb 2006.

[178] D. Park, C. Nicopoulos, J. Kim, N. Vijaykrishnan, and C. Das, "Exploring fault-tolerant network-on-chip architectures," in *Dependable Systems and Networks, 2006. DSN 2006. International Conference on*, pp. 93–104, June 2006.

[179] "Arteris Resilience Package." `http://www.arteris.com/flexnoc-resilience-package`.

[180] S. Murali, T. Theocharides, N. Vijaykrishnan, M. J. Irwin, L. Benini, and G. D. Micheli, "Analysis of error recovery schemes for networks on chips," *IEEE Design; Test of Computers*, vol. 22, no. 5, pp. 434–442, 2005.

[181] T. Moscibroda and O. Mutlu, "A case for bufferless routing in on-chip networks," *SIGARCH Comput. Archit. News*, vol. 37, pp. 196–207, June 2009.

[182] T. Li, M. Shafique, J. A. Ambrose, S. Rehman, J. Henkel, and S. Parameswaran, "Raster: Runtime adaptive spatial/temporal error resiliency for embedded processors," in *Proceedings of the 50th Annual Design Automation Conference*, DAC '13, (New York, NY, USA), pp. 62:1–62:7, ACM, 2013.

[183] G. Kahn, "The semantics of simple language for parallel programming.," in *IFIP Congress*, pp. 471–475, 1974.

[184] A. Alimonda, S. Carta, A. Acquaviva, A. Pisano, and L. Benini, "A feedback-based approach to dvfs in data-flow applications," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 28, pp. 1691–1704, Nov 2009.

[185] "MicroBlaze Processors." http://www.xilinx.com/tools/microblaze.htm.

[186] "Xilinx Embedded Processing Peripheral IP Cores." `http://www.xilinx.com/ise/embedded/edk\_ip.htm`.

[187] "Altera NOIS II Processors." `https://www.altera.com/products/processors`.

[188] U. Ogras and R. Marculescu, ""it's a small world after all": Noc performance optimization via long-range link insertion," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 14, pp. 693–706, July 2006.

[189] M. Chang, J. Cong, A. Kaplan, C. Liu, M. Naik, J. Premkumar, G. Reinman, E. Socher, and S.-W. Tam, "Power reduction of cmp communication networks via rf-interconnects," in *Microarchitecture, 2008. MICRO-41. 2008 41st IEEE/ACM International Symposium on*, pp. 376–387, Nov 2008.

[190] J. Chan and S. Parameswaran, "Nocout: Noc topology generation with mixed packet-switched and point-to-point networks," in *Proceedings of the 2008 Asia and South Pacific Design Automation Conference*, pp. 265–270, IEEE Computer Society Press, 2008.

[191] J. Jiao and Y. Fu, "B2rac: A physical express link addition methodology for network on chip," in *Proceedings of the 4th International Workshop on Network on Chip Architectures*, NoCArc '11, (New York, NY, USA), pp. 17–22, ACM, 2011.

[192] U. Y. Ogras and R. Marculescu, "Prediction-based flow control for network-on-chip traffic," in *Proceedings of the 43rd Annual Design Automation Conference*, DAC '06, (New York, NY, USA), pp. 839–844, ACM, 2006.

[193] S. Meijer, H. Nikolov, and T. Stefanov, "Throughput modeling to evaluate process merging transformations in polyhedral process networks," in *Design, Automation Test in Europe Conference Exhibition (DATE), 2010*, pp. 747–752, March 2010.

[194] T. Kluter, P. Brisk, E. Charbon, and P. Ienne, "Mpsoc design using application-specific architecturally visible communication," in *High Performance Embedded Architectures and Compilers* (A. Seznec, J. Emer, M. O'Boyle, M. Martonosi, and T. Ungerer, eds.), vol. 5409 of *Lecture Notes in Computer Science*, pp. 183–197, Springer Berlin Heidelberg, 2009.

[195] H. Nikolov, T. Stefanov, and E. Deprettere, "Systematic and automated multiprocessor system design, programming, and implementation," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 27, pp. 542–555, March 2008.

[196] W. J. Dally, "Express cubes: Improving the performance of k-ary n-cube interconnection networks," *IEEE Trans. Comput.*, vol. 40, pp. 1016–1023, Sept. 1991.

[197] S. Loucif, L. M. Mackenzie, and M. Ould-Khaoua, "The "express channel" concept in hypermeshes and k-ary n-cubes," in *Proceedings of the 8th IEEE Symposium on Parallel and Distributed Processing (SPDP '96)*, SPDP '96, (Washington, DC, USA), pp. 566–, IEEE Computer Society, 1996.

[198] A. Kumar, L.-S. Peh, P. Kundu, and N. Jha, "Toward ideal on-chip communication using express virtual channels," *Micro, IEEE*, vol. 28, pp. 80–90, Jan 2008.

[199] S. Deb, K. Chang, A. Ganguly, X. Yu, C. Teuscher, P. Pande, D. Heo, and B. Belzer, "Design of an efficient noc architecture using millimeter-wave wireless links," in *Quality Electronic Design (ISQED), 2012 13th International Symposium on*, pp. 165–172, March 2012.

[200] J. Castrillon, R. Leupers, and G. Ascheid, "Maps: Mapping concurrent dataflow applications to heterogeneous mpsocs," *Industrial Informatics, IEEE Transactions on*, vol. 9, pp. 527–545, Feb 2013.

[201] C. Lee, S. Kim, and S. Ha, "A systematic design space exploration of mpsoc based on synchronous data flow specification," *Journal of Signal Processing Systems*, vol. 58, no. 2, pp. 193–213, 2010.

[202] S. Pasricha and N. Dutt, "A framework for cosynthesis of memory and communication architectures for mpsoc," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 26, pp. 408–420, March 2007.

[203] R. Marculescu, U. Y. Ogras, and N. H. Zamora, "Computation and communication refinement for multiprocessor soc design: A system-level perspective," *ACM Trans. Des. Autom. Electron. Syst.*, vol. 11, pp. 564–592, June 2004.

[204] M. I. Gordon, W. Thies, and S. Amarasinghe, "Exploiting coarse-grained task, data, and pipeline parallelism in stream programs," *SIGARCH Comput. Archit. News*, vol. 34, pp. 151–162, Oct. 2006.

[205] "Cacti Memory Tool." `http://www.hpl.hp.com/research/cacti/`.

[206] N. Concer, L. Bononi, M. Soulie, R. Locatelli, and L. Carloni, "Ctc: An end-to-end flow control protocol for multi-core systems-on-chip," in *Networks-on-Chip, 2009. NoCS 2009. 3rd ACM/IEEE International Symposium on*, pp. 193–202, May 2009.

[207] R. Dick, D. Rhodes, and W. Wolf, "Tgff: task graphs for free," in *Hardware/Software Codesign, 1998. (CODES/CASHE '98) Proceedings of the Sixth International Workshop on*, pp. 97–101, Mar 1998.

[208] H. Kim, A. Vitkovskiy, P. V. Gratz, and V. Soteriou, "Use it or lose it: wear-out and lifetime in future chip multiprocessors," in *Proceedings of the 46th Annual IEEE/ACM International Symposium on Microarchitecture*, pp. 136–147, ACM, 2013.