

ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ
ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ
ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ
ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

Διερεύνηση και σύγκριση προσομοιωτών για μελέτη απόδοσης
τοπολογιών NoC
(Network on Chip).

Survey and comparison of Simulators for NoC (Network on chip)
topologies performance study.

Μεταπτυχιακή Διατριβή

Αικατερίνη Ι. Νικολαΐδου

Επιβλέποντες Καθηγητές :

Σταμούλης Γεώργιος
Ευμορφόπουλος Νέστωρ
Δημητρίου Γεώργιος

Βόλος, Φεβρουάριος 2019



ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ
ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ
ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ
ΥΠΟΛΟΓΙΣΤΩΝ

Διερεύνηση και σύγκριση προσομοιωτών για μελέτη απόδοσης
τοπολογιών NoC
(Network on Chip).

Μεταπτυχιακή Διατριβή

Αικατερίνη Ι. Νικολαΐδου

Επιβλέποντες :

Σταμούλης Γεώργιος
Ευμορφόπουλος Νέστωρ
Δημητρίου Γεώργιος

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 10^η Φεβρουαρίου
2019

.....

.....

.....

Μεταπτυχιακή Διατριβή για την απόκτηση του Μεταπτυχιακού διπλώματος Ειδίκευσης «Επιστήμη και Τεχνολογία Υπολογιστών, Τηλεπικοινωνιών και Δικτύων», στα πλαίσια του Προγράμματος Μεταπτυχιακών Σπουδών του Τμήματος Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών του Πανεπιστημίου Θεσσαλίας.

.....
Αικατερίνη Ι. Νικολαΐδου

Διπλωματούχος Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών
Πανεπιστημίου Θεσσαλίας

Copyright © Nikolaidou Aikaterini, 2019
Με επιφύλαξη παντός δικαιώματος. All rights reserved.

Απαγορεύεται η αντιγραφή, αποθήκευση και διανομή της παρούσας εργασίας, εξ ολοκλήρου ή τμήματος αυτής, για εμπορικό σκοπό. Επιτρέπεται η ανατύπωση, αποθήκευση και διανομή για σκοπό μη κερδοσκοπικό, εκπαιδευτικής ή ερευνητικής φύσης, υπό την προϋπόθεση να αναφέρεται η πηγή προέλευσης και να διατηρείται το παρόν μήνυμα. Ερωτήματα που αφορούν τη χρήση της εργασίας για κερδοσκοπικό σκοπό πρέπει να απευθύνονται προς τον συγγραφέα.

To my family

Ευχαριστίες

Με την περάτωση της παρούσας εργασίας, θα ήθελα να ευχαριστήσω θερμά τους επιβλέποντες της διπλωματικής εργασίας για την εμπιστοσύνη που επέδειξαν στο πρόσωπό μου, την άριστη συνεργασία, την συνεχή καθοδήγηση και τις ουσιώδεις υποδείξεις και παρεμβάσεις, που διευκόλυναν την εκπόνηση της μεταπτυχιακής αυτής διατριβής. Επίσης, θα ήθελα να ευχαριστήσω θερμά τον Απόστολο Ξενάκη για την υποστήριξη, για τις εύστοχες υποδείξεις του και την συνεχή στήριξή του. Τέλος, οφείλω ένα μεγάλο ευχαριστώ στην οικογένειά μου για την αμέριστη υποστήριξη και την ανεκτίμητη βοήθεια που μου παρείχαν τόσο κατά την διάρκεια των σπουδών μου όσο και κατά την εκπόνηση της μεταπτυχιακής αυτής διατριβής.

Νικολαΐδου Αικατερίνη

Βόλος, 2019

Contents

LIST OF FIGURES.....	VI
LIST OF TABLES	VIII
LIST OF ACRONYMS	X
ABSTRACT	XIII
ΠΕΡΙΛΗΨΗ	XV
INTRODUCTION	17
1.1 THESIS MOTIVATION AND OBJECTIVES	17
1.2 THESIS OUTLINE.....	17
NETWORK ON CHIP AND RELATED WORK.....	19
2.1 FROM BUS AND CROSSBAR BASED INTERCONNECT ARCHITECTURES TO NOC.....	19
2.2 NOC ARCHITECTURE AND DESIGN SPACE	29
2.2.1 Introduction	29
2.2.2 Links	30
2.2.3 Routers.....	30
2.2.4 Network Interface	36
2.2.4 NoC Topologies	36
2.3 NOC PERFORMANCE EVALUATION	40
NOC SIMULATORS	41
3.1 INTRODUCTION	41
3.2 COMPARISON OF ACADEMIC NOCS	41
ACCESS NOXIM SIMULATOR	44
4.1 INTRODUCTION	44
4.1.1 SystemC.....	44
4.1.2 Hotspot.....	44
4.1.3 Noxim	45
4.2 ACCESS NOXIM PARAMETERS AND USAGE.....	45
4.3 NOXIM EXPLORER.....	49
SIMULATION AND RESULTS	52
5.1 INTRODUCTION	52
5.1.1 PIR vs Latency.....	52
5.1.2 PIR vs Throughput	53
5.1.3 Simulation Cycle vs Latency	53
5.1.4 Simulation Cycle vs Throughput.....	53
5.2 ROUTING ALGORITHM EXPERIMENTS.....	54
5.2.1 Introduction	54
5.2.2 Experiment 1: Different routing algorithms under Random traffic pattern	54
5.2.3 Experiment 2: Different routing algorithms under Transpose 1 traffic pattern	59
5.3 SELECTION FUNCTION EXPERIMENTS	63

5.3.1 Introduction	63
5.3.2 Experiment 3: Different selection functions for Adaptive Routing algorithm	63
CONCLUSION AND FUTURE WORK.....	70
BIBLIOGRAPHY.....	71

List of Figures

Figure 1 Example of an on chip bus	22
Figure 2 Bus-based SoC communication model	22
Figure 3 Examples of advanced bus architectures.....	23
Figure 4 Packet and flit format used in NoC	25
Figure 5 Communication mechanism in a NoC.....	26
Figure 6 An example NoC interconnect	27
Figure 7 Interface views of a typical router (a) functional view,	31
Figure 8 Architecture of a typical router	35
Figure 9 NoCs with direct regular topologies (a) 2-D grid, (b) 2-D torus, (c) 3-D hypercube, (d) octagon..	37
Figure 10 NoCs with indirect topologies (a) fat-tree, (b) three-stage butterfly	38
Figure 11 NoCs with irregular topologies (a) reduced mesh, (b) cluster-based hybrid topology	39
Figure 12 PIR v.s. Avg. Latency example	51
Figure 13 XYZ Routing.....	55
Figure 14 Odd-even Routing.....	55
Figure 15 Latency for XYZ (left) vs oddeven (right) under random traffic	57
Figure 16 Throughput for XYZ (left) vs oddeven (right) under random traffic.....	58
Figure 17 Max Delay for XYZ (left) vs oddeven (right) under random traffic.....	58
Figure 18 Total Energy for XYZ (left) vs oddeven (right) under random traffic.....	59
Figure 19 Latency for XYZ (left) vs oddeven (right) under transpose1 traffic	61
Figure 20 Throughput for XYZ (left) vs oddeven (right) under transpose1 traffic	61
Figure 21 Max Delay for XYZ (left) vs oddeven (right) under transpose1 traffic.....	62
Figure 22 Total Energy for XYZ (left) vs oddeven (right) under transpose1 traffic	62
Figure 23 Latency for adaptive routing under bufferlevel selection (top-left) vs nop selection (top-right) vs random selection(bottom)	66
Figure 24 Throughput for adaptive routing under bufferlevel selection (top-left) vs nop selection (top-right) vs random selection (bottom)	67
Figure 25 Max Delay for adaptive routing under bufferlevel selection (top-left) vs nop selection (top-right) vs random selection (bottom)	68
Figure 26 Total Energy for adaptive routing under bufferlevel selection (top-left) vs nop selection (top-right) vs random selection (bottom)	69

List of Tables

Table 1 Comparison of NoC and bus architecture	28
Table 2 Access Noxim parameters and explanation.	47
Table 3 Access Noxim default parameters	48
Table 4 PIR vs Latency	52
Table 5 PIR where Average Latency is below 100.....	52
Table 6 PIR vs Throughput.....	53
Table 7 Simulation Cycle vs Latency	53
Table 8 Simulation Cycle vs Throughput.....	54
Table 9 Arguments for xyz routing and random traffic simulation.	56
Table 10 Arguments for odd-even routing and random traffic simulation.....	57
Table 11 Arguments for XZY routing and transpose1 traffic simulation	60
Table 12 Arguments for odd-even routing and transpose1 traffic simulation	60
Table 13 Arguments for fullyadaptive routing and random selection simulation	64
Table 14 Arguments for fullyadaptive routing and nop selection simulation.....	65
Table 15 Arguments for fullyadaptive routing and bufferlevel selection simulation.....	65

List of Acronyms

SoC	System On Chip
NoC	Network On Chip
IC	Integrated Circuit
GPU	Graphical Processing Unit
CPU	Central Processing Unit
RAM	Random Access Memory
DSP	Digital Signal Processor
ASIP	Application-Specific Instruction set Processor
RISC	Reduced Instruction Set Computer
CISC	Complex Instruction Set Computer
IP	Intellectual Property
ROM	Read-Only Memory
EEPROM	Electrically Erasable Programmable ROM
SRAM	Static RAM
DRAM	Dynamic RAM
USART	Universal Synchronous and Asynchronous Receiver-Transmitter
SPI	Serial Peripheral Interface
HDMI	High-Definition Multimedia Interface
I²C	Inter-Integrated Circuit
6LoWPAN	IPv6 over Low-Power Wireless Personal Area Networks
VLIW	Very Long Instruction Word
SIMD	Single instruction, multiple data
QoS	Quality of Service
AMBA	Advanced Microcontroller Bus Architecture
MPSoC	Multiprocessor System-On-Chip

NA	Network Adapter
NI	Network Interface
FIFO	First in first out
GALS	Globally Asynchronous Locally Synchronous
Phit	Physical unit
RASOC	Router Architecture for SoC
TDM	Time Division Multiplexing
VCs	Virtual channels
NF	Negative First
WF	West First
2D	Two-Dimensional
3D	Three-dimensional
OCP	Open Core Protocol
VCI	Virtual component interface
AXI	Advanced Extensible Interface
DTL	Device Transaction Level
XML	eXtensible Markup Language
VHDL	VHSIC Hardware Description Language
MIT	Massachusetts Institute of Technology
MIPS	Microprocessor Without Interlocked Pipeline Stages
pir	Packet Injection Rate
NoP	Neighbours-On-Path
Flits	Flow Control Units

Abstract

In the recent years, NoC has emerged as a new communication infrastructure to decrease communication complexity of the current SoCs [43]. The advantages of NoC include high bandwidth, low latency, low power consumption and scalability. The interconnection architecture has a significant impact on the performance of networks in terms of point-to-point delay, throughput, and loss rate. A basic NoC architecture consists of various techniques and blocks connected together to form interconnect architecture for a SoC. Main aspects of the NoC architecture are: Links, Routers, Network Interface and NoC Topology. The performance of a network-on-chip can be evaluated by three parameters: Latency, Throughput and Power/Energy Consumption.

To facilitate the development of embedded systems containing a network on chip, several dedicated tools have been proposed. These initiatives are often presented by the scientific community through research teams. The Access Noxim is a co-simulation platform for 3D NoC system that couples the network model, power model and thermal model. There are many parameters that can be set in Access Noxim and Noxim Explorer is a tool for running batch simulations. Experiments have been set, to evaluate the performance of NoC using Access Noxim for different PIRs and Simulation Cycles, for different Routing Algorithms under the same traffic pattern and for different Selection functions under Adaptive routing.

Περίληψη

Τα τελευταία χρόνια, ο NoC προέκυψε ως νέα υποδομή επικοινωνίας για τη μείωση της πολυπλοκότητας της επικοινωνίας των σημερινών SoCs [43]. Τα πλεονεκτήματα του NoC περιλαμβάνουν υψηλό bandwidth, χαμηλό latency, χαμηλό power consumption και scalability. Η αρχιτεκτονική διασύνδεσης έχει σημαντικό αντίκτυπο στην απόδοση των δικτύων όσον αφορά την point-to-point delay, το throughput, και το loss rate. Μια βασική αρχιτεκτονική NoC αποτελείται από διάφορες τεχνικές και block που συνδέονται μαζί για να σχηματίσουν μια αρχιτεκτονική διασύνδεσης για ένα SoC. Κύριες πτυχές της αρχιτεκτονικής ενός NoC είναι: τα Links, οι Routers, το Network Interface and το NoC Topology. Η απόδοση ενός NoC μπορεί να αξιολογηθεί με τρεις παραμέτρους: Latency, Throughput και Power/Energy Consumption.

Για να διευκολυνθεί η ανάπτυξη ενσωματωμένων συστημάτων που περιέχουν ένα NoC, έχουν προταθεί διάφορα ειδικά εργαλεία. Οι πρωτοβουλίες αυτές παρουσιάζονται συχνά από την επιστημονική κοινότητα μέσω ερευνητικών ομάδων. Το Access Noxim είναι μια πλατφόρμα συν-προσομοίωσης για 3D NoC που συνδυάζει το network model, το power model and το thermal model. Υπάρχουν πολλές παράμετροι που μπορούν να ρυθμιστούν στο Access Noxim, και το Noxim Explorer είναι ένα εργαλείο για την εκτέλεση batch προσομοιώσεων. Έχουν οριστεί πειράματα για την αξιολόγηση της απόδοσης του NoC χρησιμοποιώντας τον Access Noxim για διαφορετικά PIRs και Simulation Cycles, για διαφορετικούς Routing Algorithms με το ίδιο traffic pattern και για διαφορετικές Selection functions για Adaptive routing.

Chapter 1

Introduction

1.1 Thesis Motivation and Objectives

In the recent years, NoC has emerged as a new communication infrastructure to decrease communication complexity of the current SoCs [43]. The advantages of NoC include high bandwidth, low latency, low power consumption and scalability. The interconnection architecture has a significant impact on the performance of networks in terms of point-to-point delay, throughput, and loss rate. The proposed activity is building on the learnings from both academic and industrial attempts to evaluate the performance of NoC in order to provide the required QoS in terms of predictable guaranteed service and best effort services. After presenting the NoC Architecture and design space and comparing some academic NoCs, we run a set of experiments to evaluate the performance of NoC.

1.2 Thesis Outline

The thesis has been organized into multiple chapters to explain the motivation, design and simulation results of NoCs.

In Chapter 2, we explain the reason to use a NoC over bus based interconnect architecture for a MPSoC. After that we explain the techniques and architecture used inside the NoC which are router architecture, network interface, routing algorithms, flow control and others. Different topologies are compared as well. We also provide methods and parameters to assess the performance of a NoC.

In Chapter 3, we present related work on Academic NoC simulators. We explain capabilities and limitations of each NoC simulator.

In Chapter 4, we focus on Access Noxim simulator. Its architecture, parameters and usage are presented along with Noxim Explorer, a tool for running batch simulations.

A set of experiments along with their results are analyzed in Chapter 5. The experiments focus on PIR, Simulation Cycle, Routing algorithms, Selection functions and other.

Conclusion and recommendations for future work are provided in Chapter 6.

Chapter 2

Network on Chip and Related Work

2.1 From Bus and Crossbar based interconnect architectures to NoC

A SoC combines the required electronic circuits of various computer components onto a single, IC. SoC is a complete electronic substrate system that may contain analog, digital, mixed-signal or radio frequency functions. Its components usually include a GPU, a CPU that may be multi-core, and RAM. Because SoC includes both the hardware and software, it uses less power, has better performance, requires less space and is more reliable than multi-chip systems. Most system-on-chips today come inside mobile devices like smartphones and tablets.

More precisely, a typical SoC consists of:

- At least one processor core, but will typically have more. Processor cores can be a microcontroller, microprocessor, DSP or ASIP core. Multiprocessor SoCs have more than one processor core. ASIPs have instruction sets that are customized for an application domain and designed to be more efficient than general-purpose instructions for a specific type of workload. Multiprocessor SoCs have more than one processor core by definition. Whether single-core, multi-core or manycore, SoC processor cores typically use RISC instruction set architectures. RISC architectures are advantageous over CISC processors for systems-on-chip because they require less digital logic, and therefore less power and area on board, and in the embedded and mobile computing markets these are often highly constrained. In particular, SoC processor cores often use the ARM architecture because it is a soft processor specified as an IP core and more power efficient than x86.
- Semiconductor memory blocks to perform their computation, as do microcontrollers and other embedded systems. Depending on the application, SoC memory may form a memory hierarchy and cache hierarchy. In the mobile computing market, this is common, but in many low-power embedded microcontrollers this is not necessary. Memory technologies for SoCs include ROM, RAM, EEPROM and flash memory. As in other computer systems, RAM can be subdivided into relatively faster but more expensive SRAM and the slower but cheaper DRAM. When a SoC has a cache hierarchy, SRAM will usually be used to implement processor registers and cores' L1

caches whereas DRAM will be used for lower levels of the cache hierarchy including main memory. "Main memory" may be specific to a single processor (which can be multi-core) when the SoC has multiple processors, in which case it is distributed memory and must be sent via Intermodule communication on-chip to be accessed by a different processor.

- External interfaces, typically for communication protocols. These are often based upon industry standards such as USB, FireWire, Ethernet, USART, SPI, HDMI, I²C, etc. These interfaces will differ according to the intended application. Wireless networking protocols such as Wi-Fi, Bluetooth, 6LoWPAN and near-field communication may also be supported. When needed, SoCs include analog interfaces including analog-to-digital and digital-to-analog converters, often for signal processing. These may be able to interface with different types of sensors or actuators, including smart transducers. They may interface with application-specific modules or shields. Or they may be internal to the SoC, such as if an analog sensor is built in to the SoC and its readings must be converted to digital signals for mathematical processing.
- DSP cores are often included on systems-on-chip. They perform signal processing operations in systems-on-chip for sensors, actuators, data collection, data analysis and multimedia processing. DSP cores typically feature VLIW and SIMD instruction set architectures, and are therefore highly amenable to exploiting instruction-level parallelism through parallel processing and superscalar execution. DSP cores most often feature application-specific instructions, and as such are typically ASIP. Such application-specific instructions correspond to dedicated hardware functional units that compute those instructions. Typical DSP instructions include multiply-accumulate, Fast Fourier transform, fused multiply-add, and convolutions.
- Timing sources to generate clock signals, control execution of SoC functions and provide time context to signal processing applications of the SoC, if needed. Popular time sources are crystal oscillators and phase-locked loops. System-on-chip peripherals including counter-timers, real-time timers and power-on reset generators. SoCs also include voltage regulators and power management circuits.

One can summarize the communication issues of a complex SoC in terms of six major requirements for the communication infrastructure:

1. Performance:

It should meet different performance levels of throughput, latency, wire delay, and synchronization.

2. Scalability:

It should make it easier the inclusion of additional functional units.

3. Parallelism:

It should provide parallel communication between sets of cores, and the sub-set of cores communicating in parallel may change over the system lifetime.

4. Reusability:

The defined infrastructure should be easily reusable in new designs to reduce design time.

5. Quality of Service:

QoS should provide, when necessary, guarantees of (at least) some of the services provided, not only in terms of performance, but also in terms of reliability.

6. Reliability and Fault Tolerance:

It should provide detection and recovery schemes for manufacturing and operational faults to increase not only system reliability and dependability, but also yield and system lifetime. [5]

The aforementioned large number of computing and storage cores (components) that a SoC consists of, also called "blocks", were traditionally interconnected by means of single or multiple layers of shared buses or crossbar structures as shown in Figure 1 and Figure 2. A very common bus for system-on-chip communications is ARM's royalty-free AMBA standard. Direct memory access controllers route data directly between external interfaces and SoC memory, bypassing the CPU or control unit, thereby increasing the data throughput of the system-on-chip. This is similar to some device drivers of peripherals on component-based multi-chip module PC architectures. Major characteristics of the bus architecture are

- a) One transaction at a time
- b) Central arbiter
- c) Limited bandwidth
- d) Synchronous
- e) Low cost [1].

The advantages of the shared bus architecture are simple topology, low area cost, and extensibility.

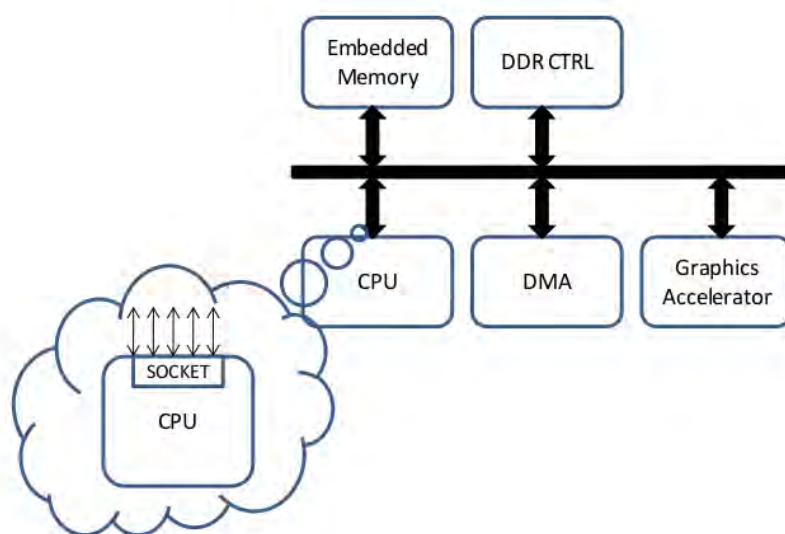


Figure 1 Example of an on-chip bus

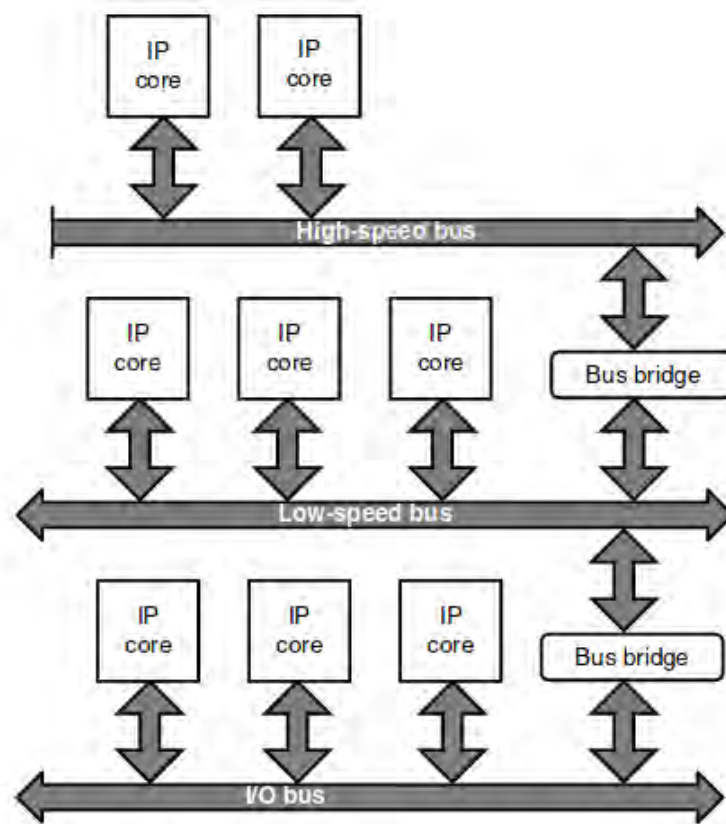


Figure 2 Bus-based SoC communication model

They support a modular design approach that uses standard interfaces and allows for IP re-use, but as the number of IP modules in SoCs increases, simple bus-based interconnection architectures may prevent these systems to meet the performance required by many applications. For systems with intensive parallel communication requirements simple buses may not provide the required bandwidth, latency, and power consumption. Hence, a few advanced bus architectures like segmented bus, pipeline packetized multistage crossbar has been mentioned in [1] and as shown in Figure 3. some of the key features of the advanced bus architecture are a) versatile compared to simple bus architecture b) pipeline capability c) burst transfer d) split transactions e) overlapped arbitration f) transaction preemption and resumption g) transaction reordering.

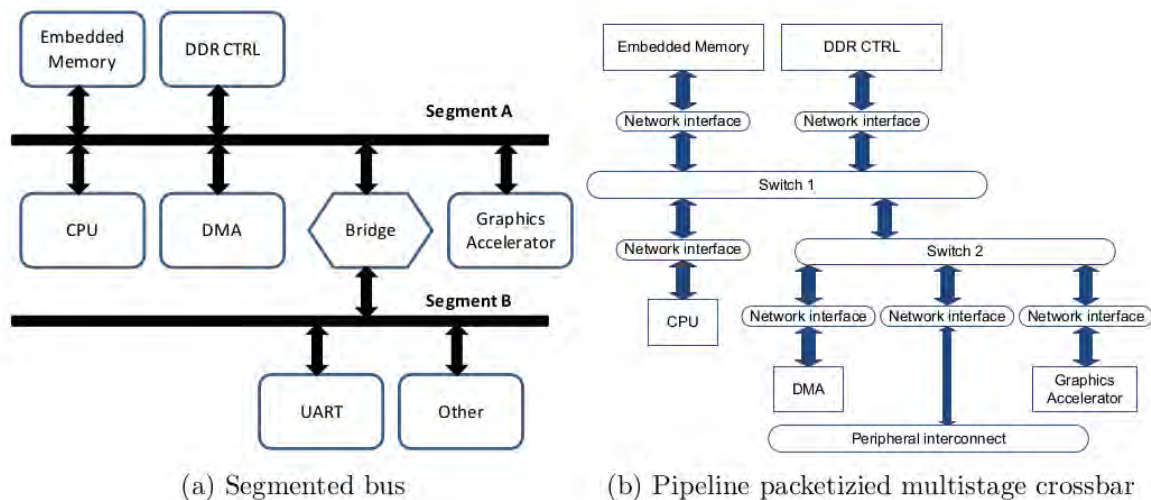


Figure 3 Examples of advanced bus architectures

The next generation MPSoCs keep pushing for higher operating frequency, increase in integration of memory, cores and IP's which are called nodes. Hence it would increase bandwidth required for communication between the nodes. However, such approach has several shortcomings which will limit its use in future SoCs. To start with, non-scalability is one of the major drawbacks, since bandwidth is fixed at design time and shared which means throughput decreases with increase in number of nodes. Implementation of pipelining is complex and central arbitration per layer or bus is required. Also, the non-predictable wire delay, power consumption and its complication of the design process become communication bottleneck in the MPSoCs. Finally, in SoC bus architecture there are conflicting tradeoffs between compatibility requirements, driven by IP blocks reuse strategies, and the necessary bus evolutions driven by technology changes: In many cases, introducing new features has required many changes in the bus implementation but more importantly in the bus interfaces with major impacts on IP reusability and new IP design [2].

In search of a proven solution to scalability worries, researchers turned to wide area networks to get inspiration. In the late 2010s, a trend of systems-on-chip implementing communications subsystems in terms of a network-like topology instead of bus-based protocols has emerged. A trend towards more processor cores on SoCs has caused on-chip communication efficiency to become one of the key factors in determining the overall system performance and cost. This has led to the emergence of interconnection networks with router-based packet switching known as "networks on chip" (NoCs) to overcome the bottlenecks of bus-based networks [3]. NoCs design space is considerably larger when compared to a bus-based solution, as different routing and arbitration strategies can be implemented as well as different organizations of the communication infrastructure. In addition, NoCs have an inherent redundancy that helps tolerate faults and deal with communication bottlenecks. This enables the SoC designer to find suitable solutions for different system characteristics and constraints. Some major advantages of NoCs are the

following: energy efficiency and reliability, scalability of bandwidth, reusability and distributed routing decisions.

A NoC is composed of three main building blocks. The first and most important one are the links that physically connect the nodes and actually implement the communication. The second block is the router, which implements the communication protocol (the decentralized logic behind the communication protocol). One can see the NoC as an evolution of the segmented busses where the router plays the role of a “much smarter buffer” [4]. The router basically receives packets from the shared links and, according to the address informed in each packet, it forwards the packet to the core attached to it or to another shared link. The protocol itself consists of a set of policies defined during the design (and implemented within the router) to handle common situations during the transmission of a packet, such as, having two or more packets arriving at the same time or disputing the same channel, avoiding deadlock and livelock situations, reducing the communication latency, increasing the throughput, etc. The last building block is the NA or NI. This block makes the logic connection between the IP cores and the network, since each IP may have a distinct interface protocol with respect to the network.

Networks-on-Chip are based on the interconnection networks largely used in parallel computers. NoCs can be defined as a structured set of routers and point-to-point channels interconnecting the processing cores of a SoC in order to support communication among them. Such a structure can be described as a graph with routers on the nodes and channels on the arcs. The NoC interconnect model can be viewed as an evolution of the segmented bus structure where wires are connected through a control logic (the router) which implements the communication control in a distributed model, opposed to the centralized control of the bus-based solution. In this model, the segmented wires are “public” and shared by all embedded cores.

NoCs typically use the message-passing communication model, and the processing cores attached to the network communicate by sending and receiving request and response messages. A message forwards from a sender to a receiver by requesting and reserving resources of the network in order to establish a route between the sender and the receiver. Depending on the network implementation, messages can be split into smaller structures named packets, which have the same format of a message and are individually routed. Packet-based networks present a better resource utilization, because packets are shorter and reserve a smaller number of channels during their transfer.

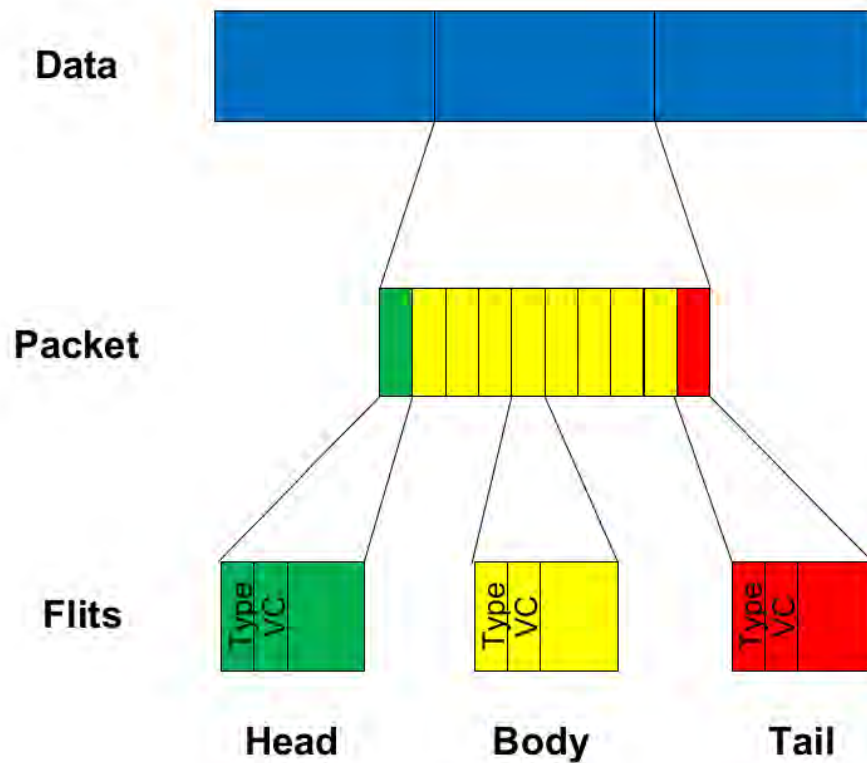


Figure 4 Packet and flit format used in NoC

NoC uses packets to route data from source to destination via a network fabric as shown in Figure 4. Notice that the message may have different formats depending on the protocol implemented by the network, but three blocks can be identified in the message (or packet) independent of the implemented protocol: the first information of the message is called the header which contains the data about the target node for the message. The header establishes the path between the source and the target node, according to the network routing algorithm. The second part of the message is called the payload and is composed of the actual data that needs to be sent to the target node. Finally, the end of the packet is indicated in the last word of the packet, which is called tail. [5] Figure 5 depicts the basic message format and transmission along the network.

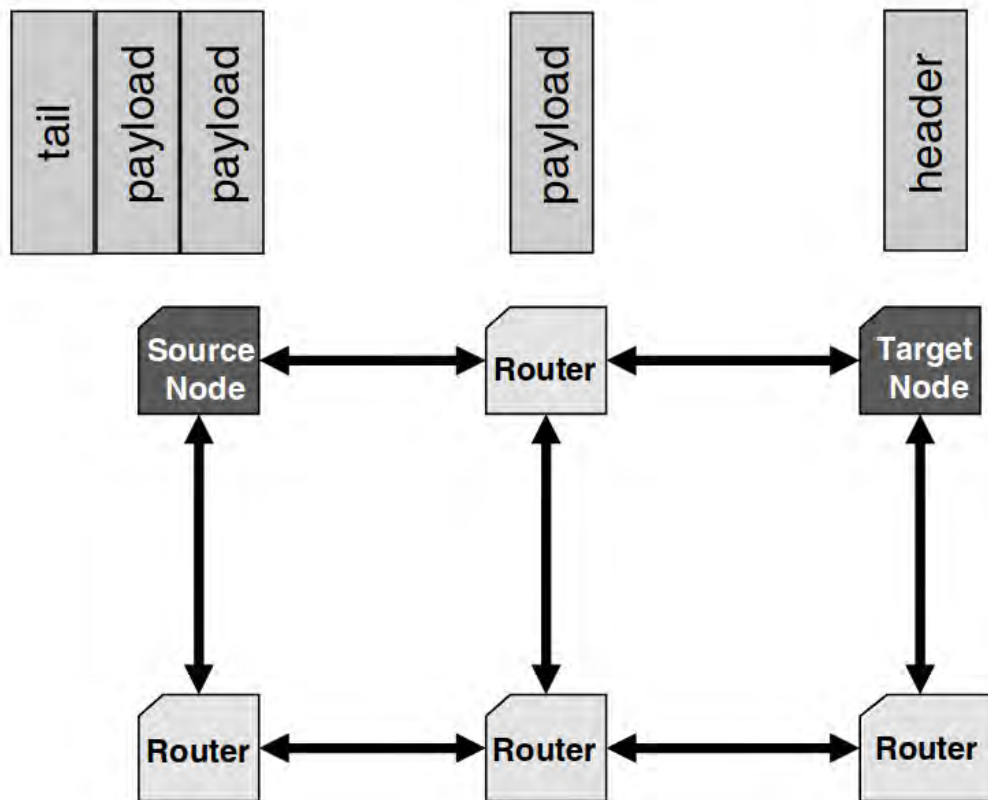


Figure 5 Communication mechanism in a NoC

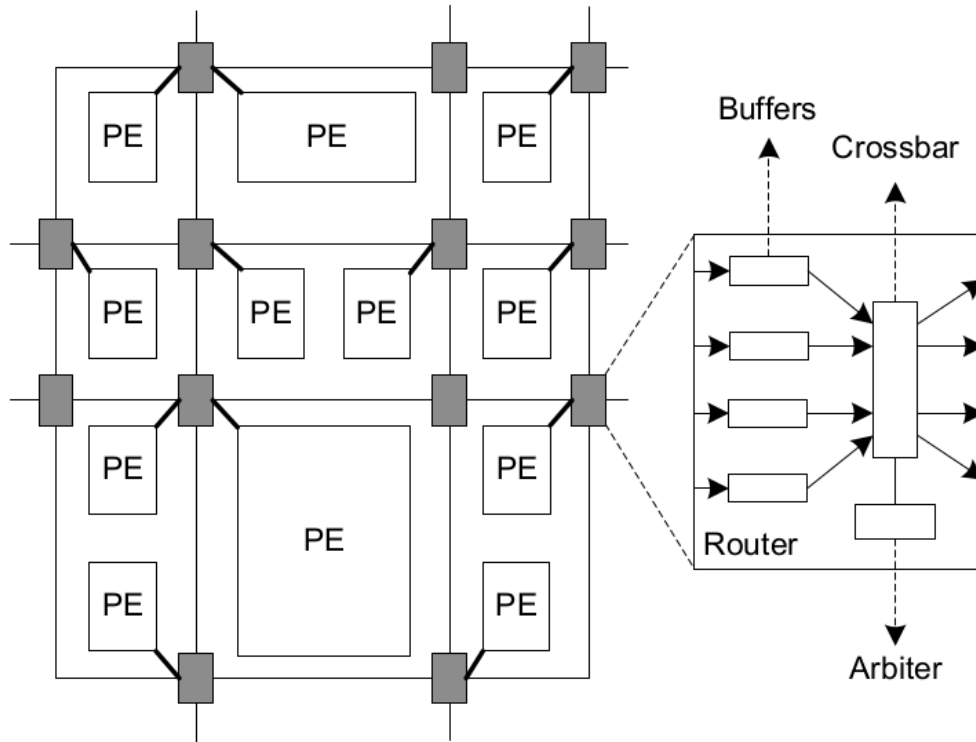


Figure 6 An example NoC interconnect

The network fabric consists of routers and interconnection links (wires) as shown in Figure 6. A NoC interconnection architecture consists several nodes connected together by network interface, routers and wires. A network interface is at boundary of node connected to a router. It converts the data generated by the node into flits and packets. A router accepts the packets generated by network interface or other routers connected to it. It has buffers at the input or output based on the router architecture to store the packets received. The packets are transported to destination link via the crossbar switch based on address mentioned in the packets. An arbiter is used to determine priority for a packet to be serviced if multiple packets from different source requires same output link. Thus, packets traverse multiple links and hop multiple routers in the NoC from source to destination node. The data is extracted for received packets at destination node by network interface.

With increase in nodes in the network, NoC link speed does not get affected. And there is aggregate growth in the bandwidth due to inherent structure and design of the NoC. It has built-in capability to pipeline transfers of packets in network interface and routers. The arbitration of packets is distributed across network interfaces and routers which can be classified into various levels of abstraction layers. In Table 1 we summarize main features of NoC over bus architecture. But disadvantage of NoC over bus architecture is overhead of area, power and delay in routers [1].

NoC	Bus
Aggregate growth in bandwidth	Bandwidth is limited, shared
Link speed unaffected by number of nodes	Speed goes down as nodes increases
Built in pipeline	Pipelining is tough
Distributed arbitration	Central arbitration per layer
Separate abstraction layers	No layers of abstraction
Performance guarantee is complex to assess	Fairly simple implementation
Extra delay in routers	
Area and power overhead	

Table 1 Comparison of NoC and bus architecture

2.2 NoC Architecture and Design Space

2.2.1 Introduction

A basic NoC architecture consists of various techniques and blocks connected to form interconnect architecture for a SoC.

Main aspects of the NoC architecture are:

- Links
- Routers
 - Flow Control
 - Routing Algorithm
 - Arbitration Logic
 - Switching
 - Buffering
- Network Interface
- NoC Topology

A NoC can be described by its topology (the organization of the cores and routers) and the approaches used to implement the mechanisms for flow control, routing, arbitration, switching and buffering, as follows. The flow control deals with data traffic on the channels and inside the routers. Routing is the mechanism that defines the path a message takes from the sender to the receiver. The arbitration establishes priority rules when two or more messages request the same resource. Switching is the mechanism that takes an incoming message of a router and puts it in an output port of the router. Finally, buffering is the strategy used to store messages when a requested output channel is busy. Current cores usually need to use wrappers to adapt their interfaces and protocols to the ones of the target NoC. Such wrappers pack and unpack data exchanged by the processing cores with the network. In the design of the NoC the most essential decisions are choosing a network topology, switching technique, and a routing algorithm [5].

Peh et al. [6], Dally et al. [7] and others have written in detail about these parts and properties used in the NoC. In next section it is briefly discussed these parts and properties which will be relevant to this thesis.

2.2.2 Links

A communication link is composed of a set of wires and connects two routers in the network. Links may consist of one or more logical or physical channels and each channel is composed of a set of wires. In the remaining chapters, unless stated otherwise, the words net, wire, and line mean a single wire interconnecting two entities (routers and/or IP cores). The words channel and link mean a group of wires connecting two entities. Typically, a NoC link has two physical channels making a full-duplex connection between the routers (two unidirectional channels in opposite directions). The number of wires per channel is uniform throughout the network and is known as the channel bitwidth. The implementation of a link includes the definition of the synchronization protocol between source and target nodes. This protocol can be implemented by dedicated wires set during the communication or through other approaches such as FIFOs [8]. Asynchronous links are also an interesting option to implement GALS systems where local handshake protocols are assumed [4]. The links ultimately define the raw performance (due to link delays) and power consumption in an NoC and designers are supposed to provide fast, reliable, and low-power interconnects between nodes in the network. The concept of flits is defined at the link level. Flits are the atomic units that form packets and streams. In most cases, a flit corresponds to a phit, which is the minimum amount of data that is transmitted in one link transaction. In this case, the flit width matches the width of the channel. However, when highly serialized links are used, a flit may be composed of several phits [5].

2.2.3 Routers

The design and implementation of a router requires the definition of a set of policies to deal with packet collision, the routing itself, and so on. A NoC router is composed of several input ports (connected to shared NoC channels), a number of output ports (connected to possibly other shared channels), a switching matrix connecting the input ports to the output ports, and a local port to access the IP core connected to this router. As an example, the interface of the RaSoC router [10] is presented in Figure 7. Herein, we use the terms router and switch as synonymous, but the term switch can also mean the internal switch matrix that actually connects the router inputs to its outputs. In addition to this physical connection infrastructure, the router also contains a logic block that implements the flow control policies (routing, arbiter, etc.) and defines the overall strategy for moving data through the NoC.

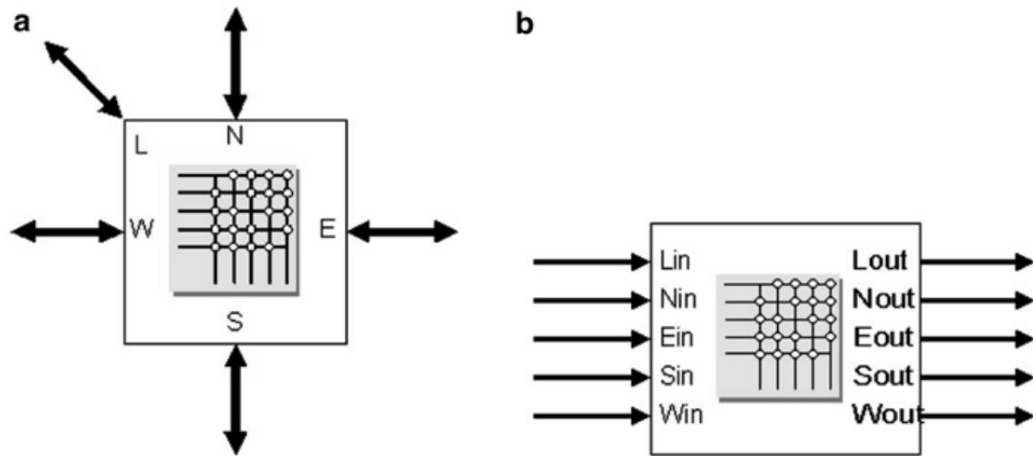


Figure 7 Interface views of a typical router (a) functional view, (b) architectural view

2.2.3.1 FLOW CONTROL

A flow control policy characterizes the packet movement along the NoC and as such it involves both global (NoC-level) and local (router-level) issues. One can ensure a deadlock-free routing, for instance, by taking specific measures in the flow control policy (by avoiding certain paths within the NoC for example). Also, the optimization of the NoC resources usage (channels, bandwidth, etc.) and the guarantees on the communication can be ensured as part of the flow control policy (for instance, by choosing a routing algorithm that minimizes the path or by implementing virtual channels to reduce congestion, etc.). Guarantees on the communication performance and quality are known as “quality-of-service” and will be detailed later on. Control can be centralized or distributed. In centralized control, routing decisions are made globally and applied to all nodes, with a strategy that guarantees no traffic contention. This approach avoids the need for an arbitration unit but requires that all nodes share a common sense of time. A possible implementation of this approach is the use of TDM mechanisms where each packet is associated to a time frame [11], [12]. However, NoCs typically use a distributed control, where each router makes decisions locally. VCs are an important concept related to the flow control. VCs implement the concept of multiplexing a single physical channel over several logically separate channels with individual and independent buffer queues. The main goal of a VC implementation is to improve performance by avoiding deadlocks, optimizing wire usage and providing some traffic guarantees [4]. Deadlock occurs when network resources are fully occupied and waiting for each other to be released to proceed with the communication, that is, when two paths are blocked in a cyclic fashion [7]. Livelock occurs when the status of the resources keep changing (there is no deadlock) but the communication is not completed.

2.2.3.2 ROUTING ALGORITHM

The routing algorithm is the logic that selects one output port to forward a packet that arrives at the router input. This port is selected according to the routing information available in the packet header. There are several possible routing algorithms that can be used in a NoC, each one leading to different trade-offs between performance and cost. For instance, in a deterministic routing a packet always uses the same path between two specific nodes. Common deterministic routing schemes are source routing and XY routing. In source routing, the source core specifies the route to the destination. In XY routing, the packet follows the rows first, then moves along the columns toward the destination or vice versa [4]. In the adaptive routing, alternative paths between two nodes may be used if the original path or a local link is congested. This involves a dynamic evaluation of the link load and implies a dynamic load balancing strategy. Negative First and West First algorithms proposed by [9] are examples of adaptive routing algorithms. In the static routing, paths between cores are defined at compilation time (of the application), while in the dynamic routing the path is defined at run-time. A unicast routing indicates that a packet has a single target whereas in the multicast routing a packet can be sent to several nodes in the NoC simultaneously (like a bus) or several slaves of a master node. Similarly, a broadcast communication targets all nodes whereas a narrowcast communication initiated by a master is related to a single slave associated to it. A routing algorithm can also be classified as minimal or non-minimal. A minimal routing guarantees that the shortest path to destination is always chosen. Minimal routing algorithms are those where a bounding box is virtually present and implies that only decreasing distances from source to destination are valid. On the other hand, non-minimal routing algorithms allow increasing the distance from source to destination. Routing algorithms can lead to or avoid the occurrence of deadlocks and livelocks. For instance, the turn model [9] is a routing algorithm that prohibits certain turns that could lead to a cycle in the network and to a risk of deadlock. The odd-even turn model [13] restricts the locations in the network where some types of turns can be taken. Another routing algorithm worth mentioning is the hot potato routing. In this algorithm, the packet is immediately forwarded towards the path with the lowest delay (instead, for example, of the shortest or minimal path). This routing scheme is also called deflective routing because if a packet cannot be accepted by the target node, it is deflected into the network, to return at a later time. The packet is not stored in a buffer (buffer-less approach) and each packet has a set of preferred outputs that will be used whenever possible in the forwarding operation.

2.2.3.3 ARBITRATION LOGIC

While the routing algorithm selects an output port for a packet, the arbitration logic implemented in the router selects one input port when multiple packets arrive at the router simultaneously requesting the same output port. Again, one has several options to implement the arbiter: it can be distributed (one per port) or centralized (one per router), it can be based on static (fixed) or dynamic (variable) priorities among ports. A centralized arbiter optimizes the use of the router switching matrix but may lead to higher latency whereas the distributed approach optimizes the latency. Arbitration logic also defines whether the network assumes a delay or a loss communication model. In the delay model, packets can be delayed, but never dropped. In the loss model a packet can be dropped as a solution, for instance, to a congestion situation. In this case, retransmission logic must be implemented as well [4].

2.2.3.4 SWITCHING

The switching defines how the data is transmitted from the source node to the target one. In the circuit switching approach the whole path (including routers and channels) from source to node is previously established (by the header) and reserved for the transmission of the whole packet. The payload is not sent until the whole path has been reserved. This can increase latency, but once the path is defined, this approach can give some guaranteed throughput, for example. In the packet-based switching approach on the other hand, all flits of the packet are sent as the header establishes the connection between routers. Still in this model the designer can choose between different buffering and forward strategies that impact the overall NoC traffic (storing the whole packet in each router before establishing the connection to the next router or sending the flits in a pipeline mode for instance). There are several switching techniques among them store-and-forward, virtual cut-through and wormhole.

In the store-and-forward strategy, the node stores the complete packet before forwarding it to the next node in the path. In this strategy one must ensure that the buffer size at each node is enough to store the whole packet or the packet can be stalled.

In the wormhole strategy, on the other hand, the node makes the routing decision and forwards the packet as soon as the header arrives. The subsequent flits follow the header as they arrive. This reduces the latency within the router, but in case of packet stalling, many links risk to be locked at once.

The virtual-cut-through mechanism is like the wormhole approach but, before forwarding the first data flit to the next node in the path, the node waits for a confirmation that the whole packet can be accepted by the next node. Thus, in case of stalling, no links are affected, only the current node.

2.2.3.5 BUFFERING

The Buffering policy is the strategy used to store information in the router when there is congestion in the network and a packet cannot be forwarded right away. The buffering strategy (number, location and size of the buffers), has an important impact on the network traffic and, therefore, on the NoC performance. In addition, the buffers are responsible for a large portion of the router area. One can have a single buffer in the router, shared by all input ports, or one can have one buffer per port (input or output). The main advantage of the first approach is the area optimization, but the control can be more complex and additional care must be taken to deal with buffer overflow. In the distributed approach, each input port has its own buffer and the most common implementation is in the form of a FIFO, although other implementations are also possible. Distributed output buffers are also possible, but they tend to be less efficient because several input ports may need to store data in a single structure.

2.2.3.6 TYPICAL ROUTER ARCHITECTURE

Figure 8 depicts a typical router architecture (used in 2D NoCs) with the above-mentioned elements identified. One can observe that the NoC designer has several possible strategies to implement a router (and, therefore, the network communication protocol) leading to a large design space. This is the main advantage of the NoC approach and explains why this platform has more chances to meet the system communication requirements. Different from a bus structure, one can tailor the NoC according to the specific requirements of the application and issues such as guaranteed performance can be naturally implemented as part of the network flow control.

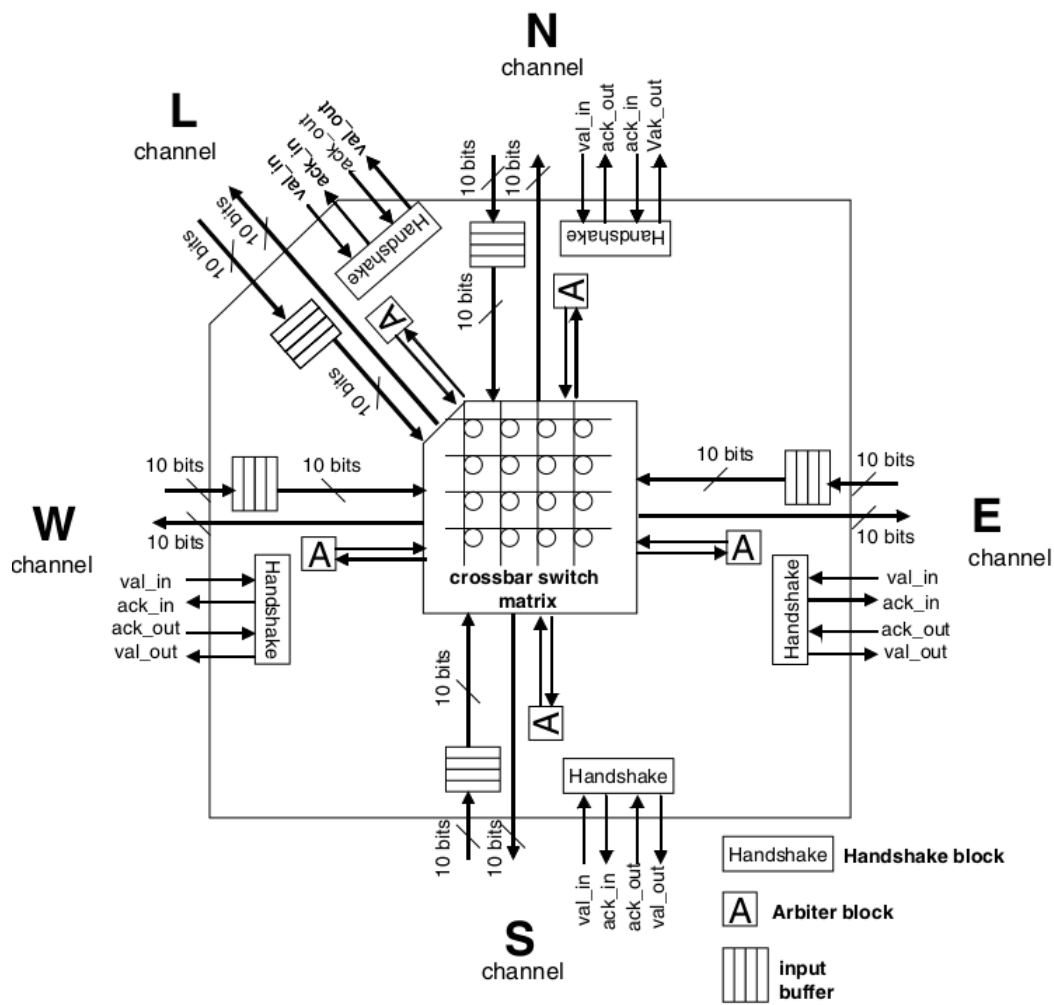


Figure 8 Architecture of a typical router

2.2.4 Network Interface

The third NoC building block is the NA or NI. This block since each IP may have a distinct interface protocol with respect to the network. This block is important because it allows the separation between computation and communication. This allows the reuse of both, core and communication infrastructure independent of each other [4]. The adapter can be divided into two parts: a front end and a back end. The frontend handles the core requests and is ideally unaware of the NoC. This part is usually implemented as a socket – OCP [14], VCI [15], AXI [16], DTL [17], etc. The back-end part handles the network protocol (assembles and disassembles the packet, reorder buffers, implement synchronization protocols, helps the router in terms of storage, etc.).

2.2.4 NoC Topologies

A NoC can be characterized by the structure of the router's connections. This structure or organization is called topology and is represented by a graph $G(N, C)$ where N is the set of routers and C is the set of communication channels. The routers can be connected in direct or indirect topologies.

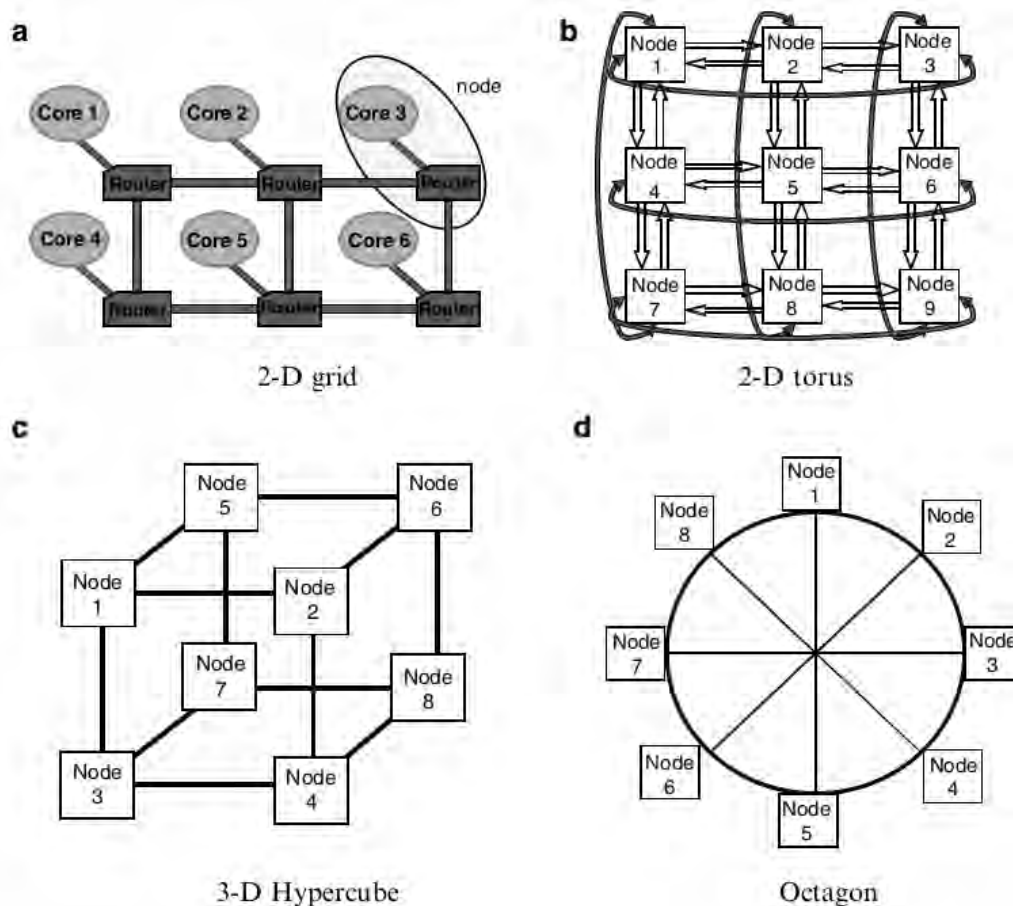


Figure 9 NoCs with direct regular topologies (a) 2-D grid, (b) 2-D torus, (c) 3-D hypercube, (d) octagon

In the direct topologies, each router is associated to a processor and this pair can be seen as a single element in the system (so-called a node in the network). In this topology, each node is directly connected to a fixed number of neighbor nodes and a message between two nodes goes through one or more intermediate nodes. Only the routers are involved in the communication in a direct topology and the communication is based on the routing algorithm implemented by the routers. Most NoC implementations are based on orthogonal arrangements of the routers in a direct topology. In this arrangement, nodes are distributed in a n -dimensional space and the packet moves in one dimension at a time. These arrangements are the ones that present the best trade-off between cost and performance, and also present good scalability. The most common direct topologies are the n -dimensional grid or mesh, torus (or k -ary n -cube) and the hypercube, as shown in Figure 9.

In an indirect topology not all routers are connected to processing units as in the direct model. Instead, some routers are used only to propagate the messages through the network, while other routers are connected to the logic and only those can be source and/or target of a message. Some topologies of indirect networks stand out: the crossbar, and the multi-stage. The multi-stage topology is a regular NoC, where routers are identical and organized in stages. Input and output stages are connected to the functional units in one side and to the

internal nodes in another side. For instance, Figure 10 shows two examples of indirect networks.

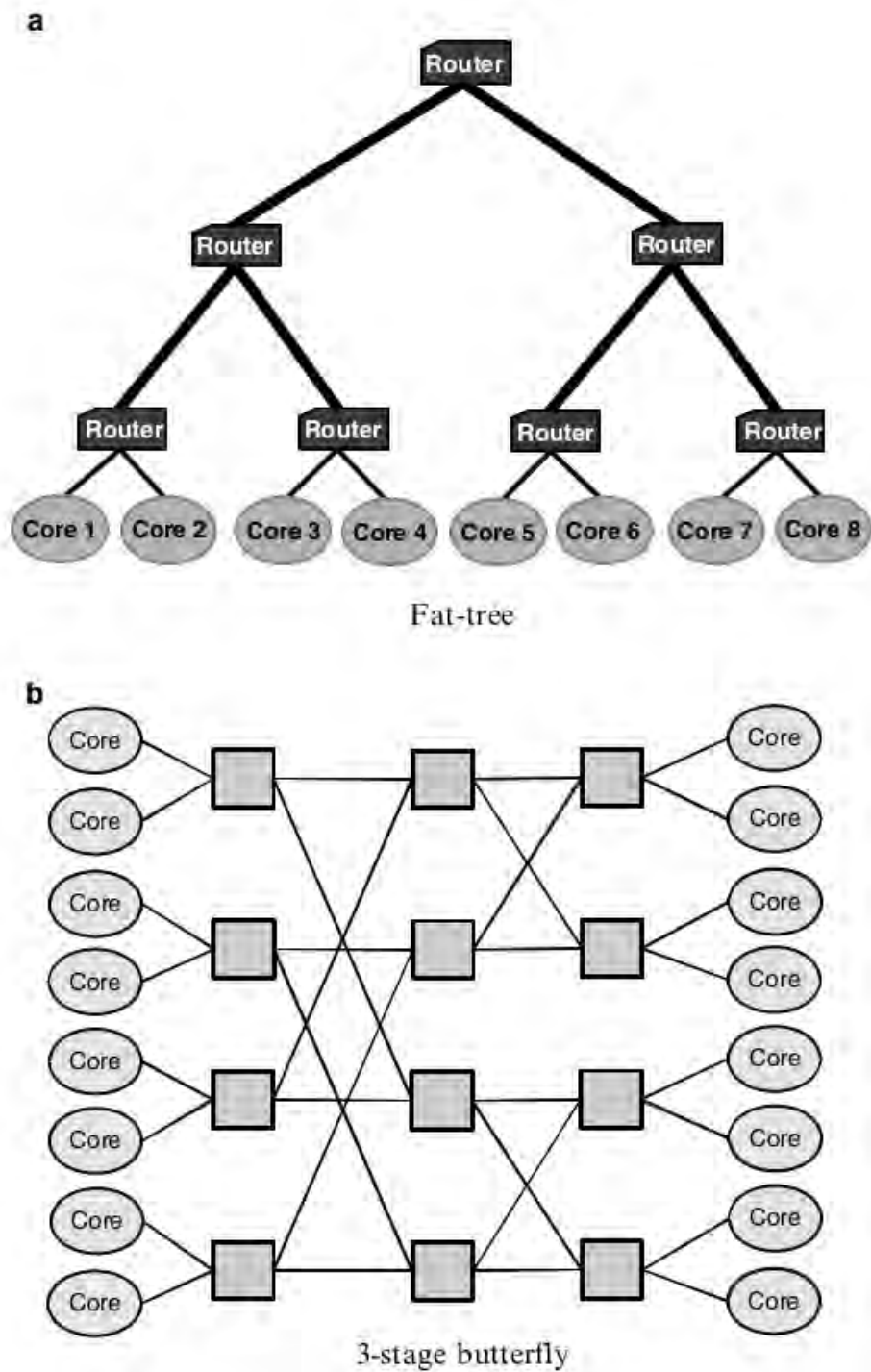


Figure 10 NoCs with indirect topologies (a) fat-tree, (b) three-stage butterfly

Another possible classification for network topologies is related to the regularity of the connections between routers. In regular networks, all routers are identical in terms of number of ports connecting to other routers or elements in the network. For instance, in a regular grid topology presented in Figure 9a all routers have five ports, one local port connecting to the functional unit and another four ports connecting to neighbor routers. In irregular topologies the routers may present different connection patterns, usually defined according to the application [18] , as depicted in Figure 11.

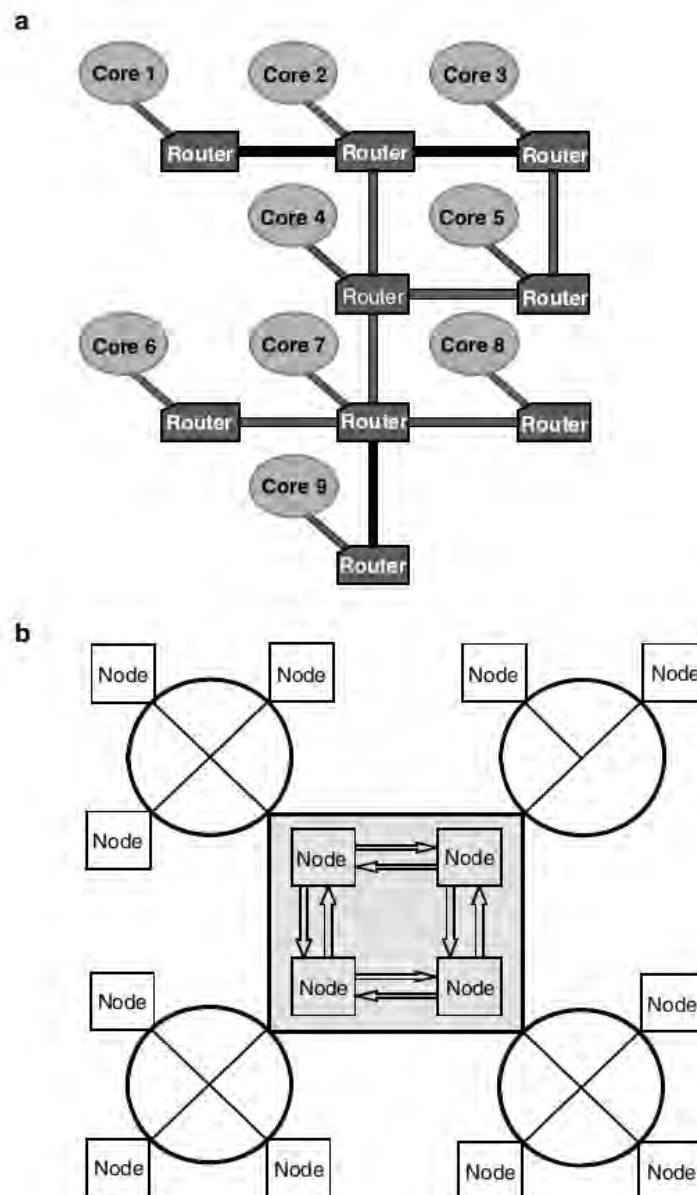


Figure 11 NoCs with irregular topologies (a) reduced mesh, (b) cluster-based hybrid topology

2.3 NoC Performance Evaluation

The performance of a network-on-chip can be evaluated by three parameters:

- Latency
- Throughput
- Power/Energy Consumption

The throughput measure is the parameter used for evaluating the bandwidth of data transfer between IP cores. It can be used to estimate the quality of service the network can achieve [20].

Throughput is messages per second or messages per clock cycle. One can have a normalized throughput (independently from the size of the messages and of the network) by dividing it by the size of the messages and by the size of the network. As a result, the unit of the normalized throughput is bits per node per clock cycle (or per second). The throughput of the system may follow the formula [21] :

$$T_{avg} = \frac{1}{N(T_{sim} - T_{warm})} \sum N_i$$

Latency is one of the most important parameters to estimate the performance of system. It is the time difference (in clock cycles) between when a packet gets delivered to its destination IP and when it was sent [20]. Latency is measured in time units and mostly used as comparison basis among different design choices. In this case, latency can also be expressed in terms of simulator clock cycles. Normally, the latency of a single packet is not meaningful [18] and one uses the average latency to evaluate the network performance. On the other hand, when some messages present a much higher latency than the average, this may be important.

Therefore, the standard deviation may be an interesting measure as well. The latency of the system may follow the formula [21] :

$$L_{avg} = \frac{1}{1} \sum \frac{1}{N_i} \sum L_{ij}$$

Power/Energy Consumption is also a vital parameter that needs to be taken into consideration especially in the area of developing an integrated circuits[20] .

Chapter 3

NoC Simulators

3.1 Introduction

To facilitate the development of embedded systems containing a network on chip, several dedicated tools have been proposed. These initiatives are often presented by the scientific community through research teams. Each tool developed tries to cover one or more aspects of NoC design space exploration like:

- Configuration of nodes
- Configuration of the NoC like topology, routing algorithms, virtual channels and others.
- Data communication requirements.
- Benchmarking and analysis of results.

In order to demonstrate how different communication solutions can be more naturally implemented using a NoC, academic NoC implementations with their basic characteristics are mentioned to show how even a small difference in the design can create a different communication structure that fits better one application.

3.2 Comparison of Academic NoCs

- **Booksim:** Booksim [22] is an open source simulator developed using C++. It is a simulator for network interconnect which supports virtual channel (VC) routers. It supports 10 topologies like mesh, torus, cmesh, fat tree and others. Twenty routing algorithms can be configured to direct packets for the supported topologies. Links between the routers and nodes is set to one, hence multiple links cannot be configured. Packet injection into the network can be configured. Allocators and virtual channels configuration are parametrized based on configuration file. The router design is based on event driven hence it is not cycle accurate. Area, power, hot spot analysis cannot be performed in Booksim. It support various synthetic traffic generator but we cannot apply custom traffic over the NoC. Support for mixed language simulation is not available.

- **Noxim:** Noxim [23] is a NoC simulator developed at the University of Catania (Italy) using SystemC. It has command line interface to parametrize various components of NoC. In Noxim we can customize network size, buffer size, packet size, routing algorithm, injection rate and traffic pattern. Noxim only supports mesh topology with wormhole routers over synthetic traffic patterns. Support for mapping custom application on the NoC is not available. Evaluation of NoC is done by producing results in terms of throughput, delay and power consumption. Detailed evaluation metrics can be analyzed like total number packets or flits sent or received, global average throughput, maximum and minimum global delay, total energy consumption etc. Like Booksim it does not support area, power, hot spot analysis and mixed language simulation.
- **NoCTweak:** NoCTweak [24] is like Noxim simulator, developed for cycle accuracy simulation of NoC. The simulator has been developed using SystemC. It supports only 2D mesh topology with each node consisting of core and network interface. For traffic generation it has support for synthetic traffic and embedded application traces with capability to map the application on each node. Router settings are parametrized with various options on virtual channel, buffer depth, routing algorithm, switch arbitration etc. It has support to read power models for early assessment of power consumption in the NoC. It generates statistic outputs like network latency, network throughput, power consumption etc. for evaluation of NoC.
- **NoCBench and NoCSim:** NoCBench [24] is a benchmarking platform to evaluate NoC. The core engine of the simulator is based on NoCSim NoC simulator. It provides an integrated simulation environment with set of standard NoC components and cores. Its main feature is to be able integrated different embedded cores and network components. NoCSim simulator is developed using SystemC. The NoC has to be modeled using manual and XML based configuration. The simulator would connect components from set of core library, simulate and generate reports.
- **Atlas:** Atlas [25] is a framework which automates various process related to design flow of NoC. The NoC components are described in VHDL and testbench has been developed using SystemC. The tool can be configured to parametrize network dimension, communication channel width, buffer depth, flow control, virtual channel selection and routing algorithms. It supports only synthetic traffic application and mapping on NoC. It has predefined power models for each component and can provide early power estimation of the NoC.
- **ORION 2.0:** ORION 2.0 is the successor of the version proposed by a team from Princeton University in 2003 [26]. It is a simulator dedicated primarily to the estimation of power and space for NoCs architectures. Among the improvements compared to the first

version we find the support for new semiconductor technology through models of transistors and capacitances upgraded from industry [27] [28].

- **DARSIM:** DARSIM is a NoC simulator which was developed at the MIT. This tool allows the simulation of mesh NoC architectures of 2 and 3 dimensions. It offers a multitude of NoC simulation configurations with various parameters. This includes two generation modes of data generation:
 - Trace-driven injection which involves the injection of packets into the network and monitors their spatial and temporal evolution. Each injection contains the tracing parameters that are time constraints, the identifier of the stream, the packet size and possibly the injection frequency.
 - MIPS Simulation mode: each node can be configured as a MIPS with its own memory. These are connected to the NoC MIPS and receiving/sending data from/to the network can be simulated cycle by cycle. Using these methods and a more detailed explanation of the simulator are presented in this reference [29].
- **NS-2:** NS-2 was first developed for prototyping and simulating ordinary computer networks. However, since NoCs shares many characteristics with classic networks, NS-2 was widely used by many NoC researchers to simulate NoCs [30] [31]. Many NoC studies have used NS-2 as a simulation tool making it a reliable reference especially when comparing the performances of two different architectures [32] [33]. Finally, NS-2 is an open source, discrete event driven simulator and developed in C++ and OTcl. These modularity and availability have facilitated its spreading between researchers.

The list above is by no means exhaustive and several other implementations can be listed. In the next chapter investigation and experiments will be performed using Access Noxim, a NoC simulator built on top of Noxim that integrates HotSpot.

Chapter 4

Access Noxim Simulator

4.1 Introduction

The Access Noxim is a co-simulation platform for 3D NoC system that couples the network model, power model and thermal model. Noxim and HotSpot are integrated, and adopt the power model of Intel's 80-core processor. Noxim is a cycle-accurate SystemC NoC simulator, and HotSpot provides the architecture-level thermal model. To coupling with HotSpot, the NoC simulator should convert its architecture-level floorplan and power trace to chip-level physical floorplan and power trace. The model of basic 3D router and the vertical crossbar router are first added, and Noxim is extended to be able to generate 3D architectures of NoC based on user-defined parameters of dimension. Then a module is inserted for automatically converting the architecture-level floorplan to physical floorplan. During network traffic simulation, a power trace is generated based on the power model of the NoC. The power trace and physical floorplan are used as inputs of the thermal simulation. In the Access Noxim simulator, the tile geometry and power model are based on Intel's 80-core chip [32].

4.1.1 SystemC

Noxim is a cycle-accurate SystemC NoC simulator. SystemC is a set of C++ classes and macros which provide an event-driven simulation kernel in C++ (see also discrete event simulation). These facilities enable a designer to simulate concurrent processes, each described using plain C++ syntax. SystemC processes can communicate in a simulated real-time environment, using signals of all the data types offered by C++, some additional ones offered by the SystemC library, as well as user defined.

In certain respects, SystemC deliberately mimics the hardware description languages VHDL and Verilog but is more aptly described as a system-level modeling language [35].

4.1.2 Hotspot

HotSpot provides the architecture-level thermal model. HotSpot is an accurate and fast thermal model suitable for use in architectural studies. It is based on an equivalent circuit of thermal resistances and capacitances that correspond to microarchitecture blocks and essential aspects of the thermal package. The model has been validated using finite element simulation. HotSpot has a simple set of interfaces and hence can be integrated with most power-performance simulators like Wattch. The chief advantage of HotSpot is that it is compatible with the kinds of power/performance models used in the computer-architecture community, requiring no detailed design or synthesis description. HotSpot makes it possible to study thermal evolution over long periods of real, full-length applications [36] .

4.1.3 Noxim

Noxim, the Network-on-Chip Simulator developed at the University of Catania (Italy) is developed using SystemC. Noxim has a command line interface for defining several parameters of a NoC. The user can customize the network size, buffer size, packet size distribution, routing algorithm, selection strategy, packet injection rate, traffic time distribution, traffic pattern, hot-spot traffic distribution. The simulator allows NoC evaluation in terms of throughput, delay and power consumption. This information is delivered to the user both in terms of average and per-communication results. In detail, the user can collect different evaluation metrics including the total number of received packets/flits, global average throughput, max/min global delay, total energy consumption, per-communications delay/throughput/energy etc. [37] .

4.2 Access Noxim Parameters and Usage

There are many parameters that can be set in Access Noxim. The following command shows all parameters.

```
% ./noxim -help
```

The arguments are listed to the Table 2 below:

Option	Sub-Option	Explanation
--------	------------	-------------

-help		Show this help and exit.
-verbose	N	Verbosity level (1=low, 2=medium, 3=high, default off)
-trace	FILENAME	Trace signals to a VCD file named 'FILENAME.vcd' (default off)
-dimx	N	Set the mesh X dimension to the specified integer value (default 8)
-dimy	N	Set the mesh Y dimension to the specified integer value (default 8)
-dimz	N	Set the mesh Z dimension to the specified integer value (default 4)
-buffer	N	Set the buffer depth of each channel of the router to the specified integer value [flits] (default 16)
-size	Nmin Nmax	Set the minimum and maximum packet size to the specified integer values [flits] (default min=8, max=8)
-routing	TYPE	Set the routing algorithm to TYPE where TYPE is one of the following (default 0): <ul style="list-style-type: none"> • <u>xyz</u>: XYZ routing algorithm • <u>westfirst</u>: West-First routing algorithm • <u>northlast</u>: North-Last routing algorithm • <u>negativefirst</u>: Negative-First routing algorithm • <u>oddeven</u>: Odd-Even routing algorithm • <u>dyad T</u>: DyAD routing algorithm with threshold T • <u>fullyadaptive</u>: Fully Adaptive routing algorithm • <u>table FILENAME</u>: Routing Table Based routing algorithm with table in the specified file
-sel	TYPE	Set the selection strategy to TYPE where TYPE is one of the following (default 0): <ul style="list-style-type: none"> • <u>random</u>: Random selection strategy • <u>bufferlevel</u>: Buffer-Level Based selection strategy • <u>nop</u>: Neighbors-on-Path selection strategy
-pir	R TYPE	Set the packet injection rate to the specified real value [0..1] (default 0.01) and the time

		<p>distribution of traffic to TYPE where TYPE is one of the following:</p> <ul style="list-style-type: none"> • <u>poisson</u>: Memory-less Poisson distribution (default) • <u>burst R</u>: Burst distribution with given real burstness • <u>pareto on off r</u>: Self-similar Pareto distribution with given real parameters (alfa-on alfa-off r) • <u>custom R</u>: Custom distribution with given real probability of retransmission
-traffic	TYPE	<p>Set the spatial distribution of traffic to TYPE where TYPE is one of the following (default 0):</p> <ul style="list-style-type: none"> • <u>random</u>: Random traffic distribution • <u>transpose1</u>: Transpose matrix 1 traffic distribution • <u>transpose2</u>: Transpose matrix 2 traffic distribution • <u>bitreversal</u>: Bit-reversal traffic distribution • <u>butterfly</u>: Butterfly traffic distribution • <u>shuffle</u>: Shuffle traffic distribution • <u>table FILENAME</u>: Traffic Table Based traffic distribution with table in the specified file
-hs	ID P	Add node ID to hotspot nodes, with percentage P (0..1) (Only for 'random' traffic)
-warmup	N	Start to collect statistics after N cycles (default 1)
-seed	N	Set the seed of the random generator (default time())
-detailed		Show detailed statistics
-volume	N	Stop the simulation when either the maximum number of cycles has been reached or N flits have been delivered
-sim	N	Run for the specified simulation time [cycles] (default 10000)

Table 2 Access Noxim parameters and explanation.

The usage of Access Noxim is the following:

```
% ./noxim [options]
```

Where options are one or more of the aforementioned ones.

The default parameters are indicated in the Table 3 below:

Option	Sub-Option	Value
-verbose	N	Off
-trace	FILENAME	Off
-dimx	N	8
-dimy	N	8
-dimz	N	4
-buffer	N	16
-size	Nmin Nmax	min=8, max=8
-routing	TYPE	xyz
-sel	TYPE	random
-pir	R TYPE	R=0.01, TYPE=poisson
-traffic	TYPE	random
-warmup	N	1
-seed	N	time()
-sim	N	10000

Table 3 Access Noxim default parameters

A command example to run a single simulation that topology of NoC is 6 by 6 by 4(H), the router buffer size is 8 and packet size is (2,8), simulation cycle as 50,000 with warm up cycle 20,000 with other parameters in default value is the following:

```
% ./noxim -dimx 6 -dimy 6 -dimz 4 -buffer 8 -size 2 8 -sim 50000 -warmup 20000
```

4.3 Noxim Explorer

In NoC simulation, it is needed to set different parameters to explore the huge design space. Simulation time is quite long and need a set of experiment. Thus it can be used Noxim explorer to explore each configuration of the design space generated by configuration file and exports results in MATLAB format.

For example, the procedure is the following:

- 1) Switch the directory to /other, use vim to open the configuration file “sim.cfg”

```
% cd /other  
% vim sim.cfg
```

- 2) Modify the topology first, set topology as an 8 by 8, 4-layer NoC.

```
[topology]  
8x8x4  
[/topology]
```

- 3) Leave the routing and selection tag untouched. Routing or selection function can be modified here.

```
[routing]  
xyz  
[/routing]  
[sel]  
random  
[/sel]
```

- 4) Modify the exploration interval of PIR. While the first parameter means start point, second represent end point, third is the step size and the last is the distribution pattern. We leave distribution pattern untouched and modify the interval to explorer the design space.

```
[pir]  
0.001 0.050 0.005 poisson  
[/pir]
```

- 5) There are some default settings, you can change the simulation time, warm-up time, packet size and buffer size here. If you want to add some self-definition parameters, please add your parameters here. Leave the aggregation tag untouched, pir is the main design space index we use.

```
[default]
-sim 100000
-warmup 10000
-size 8 8
-buffer 4
[/default]
[aggregation]
pir
[/aggregation]
```

- 6) In default case, the path of noxim is “../bin/Noxim”. If execution path changed, please modify here.

```
[explorer]
simulator ../bin/noxim repetitions 1
[/explorer]
```

- 7) Now, save the configuration file and then compile the Noxim explorer to explore the design space of NoC! After simulation ends, use vim to open the log file.

```
[shift] + [;], wq + [enter]
%make
%noxim_explorer sim.cfg
% vim routing_xyz_sel_random_topology_8x8x4_.m
```

- 8) If you have MATLAB, you can execute the function directly.

```
>> routing_xyz_sel_random_topology_8x8x4_('r-x')
```

- 9) Or you can use other software to plot the PIR vs Avg. Latency figure.

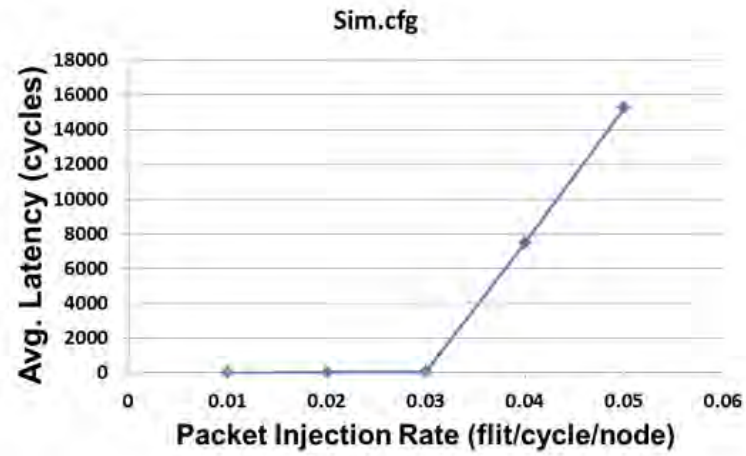


Figure 12 PIR v.s. Avg. Latency example

Chapter 5

Simulation and Results

5.1 Introduction

The rate that packets are injected into the network by a node is termed as a packet injection rate $\text{pir}()$ (packet/cycle/IP). Pir is restricted between 0 and 1($0 < \text{Pir} \leq 1$), for Instance when $\text{pir} = .3$ it means in IP sends 3 packets every 10 cycles [38].

5.1.1 PIR vs Latency

Setting different PIR) has impact on the Latency. A set of simulations has been performed with different PIR in order to evaluate the impact on the Latency. The PIR is set from 0.01 to 0.05 with step 0.005. The rest of the configuration is the default as it is mentioned in Table 3. The Table 4 below shows the results of the simulations. It is observed that while the PIR increases, the Latency increases as well.

PIR	0.01	0.015	0.02	0.025	0.03	0.035	0.04	0.045	0.05
Latency (cycles)	17.403	19.2732	21.6021	25.6335	61.113	162.026	289.208	300.138	341.005

Table 4 PIR vs Latency

We usually focus on the interval that Average Latency is lower than 100. After running a set of simulations, 20 data points in that interval are found and indicated in Table 5.

PIR where Average Latency < 100	From	To	Step
	0.001	0.0328	0.00159

Table 5 PIR where Average Latency is below 100

5.1.2 PIR vs Throughput

Setting different packet injection rate (PIR) has also impact on the Throughput. A set of simulations has been performed with different PIR in order to evaluate the impact on the Throughput. The PIR is set from 0.01 to 0.05 with step 0.005. The rest of the configuration is the default as it is mentioned in Table 3. The Table 5 below shows the results of the simulations. It is observed that while the PIR increases until 0.045, the Throughput increases as well but there is a decrease at 0.05.

PIR	0.01	0.015	0.02	0.025	0.03	0.035	0.04	0.045	0.05
Throughput (flits/cycle)	0.386241	0.511245	0.604546	0.655728	0.718443	0.755426	0.76927	0.803249	0.7755

Table 6 PIR vs Throughput

5.1.3 Simulation Cycle vs Latency

Setting different Simulation Cycle has impact on the Latency. A set of simulations has been performed with different Simulation Cycles in order to evaluate the impact on the Latency. The Simulation Cycle is set from 10000, which is the default value to 140000 with step 15000. The rest of the configuration is the default as it is mentioned in Table 3. The Table 6 below shows the results of the simulations. It is observed that while the Simulation Cycle increases the Latency remains at the same levels.

Sim. Cycle	10000	25000	40000	55000	70000	85000	100000	125000	140000
Latency (cycles)	17.4348	17.4002	17.4206	17.4691	17.4063	17.4369	17.4102	17.4409	17.4398

Table 7 Simulation Cycle vs Latency

5.1.4 Simulation Cycle vs Throughput

Setting different Simulation Cycle has also impact on the Throughput. A set of simulations has been performed with different Simulation Cycles in order to evaluate the impact on the Throughput. The Simulation Cycle is set from 10000, which is the default value to 140000 with step 15000. The rest of the configuration is the default as it is mentioned in Table 3. The Table 7 below shows the results of the simulations. It is observed that while the Simulation Cycle increases the Throughput decreases.

Sim. Cycle	10000	25000	40000	55000	70000	85000	100000	125000	140000
Throughput (flits/cycle)	0.393734	0.294473	0.226514	0.177584	0.148158	0.13393 1	0.121165	0.104318	0.0998465

Table 8 Simulation Cycle vs Throughput

5.2 Routing Algorithm Experiments

5.2.1 Introduction

In this chapter, we will run some experiments with different routings to observe the properties of NoC on the Access Noxim. The routing algorithm is the logic that selects one output port to forward a packet that arrives at the router input. This port is selected according to the routing information available in the packet header.

Under the same traffic pattern, different routing algorithms have different performance.

5.2.2 Experiment 1: Different routing algorithms under Random traffic pattern

An experiment has been set to show the performance between XYZ and OddEven routing under `random` traffic.

One of the well-known and well used routing schemes in 3D-NoCs is the Dimension Order Routing (DOR) XYZ algorithm. XYZ is a simple algorithm, easy to implement and free of deadlock and lifelock [39]. In XYZ routing algorithm the node will move in x direction as shown in the Figure 13

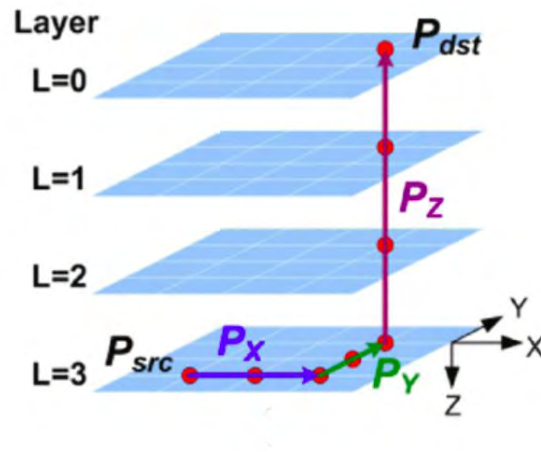


Figure 13 XYZ Routing

Odd Even Routing Algorithm as shown in Figure 14 is distributed adaptive routing algorithm which is based on odd-even turn model. It exerts some restrictions, for avoiding and preventing from deadlock appearance. This routing algorithm combines different turn prohibitions at odd and even columns, thus providing a more balanced distribution of adaptiveness throughout the network in comparison to Turn based models but still does not provide valid solution for failures in general [40].

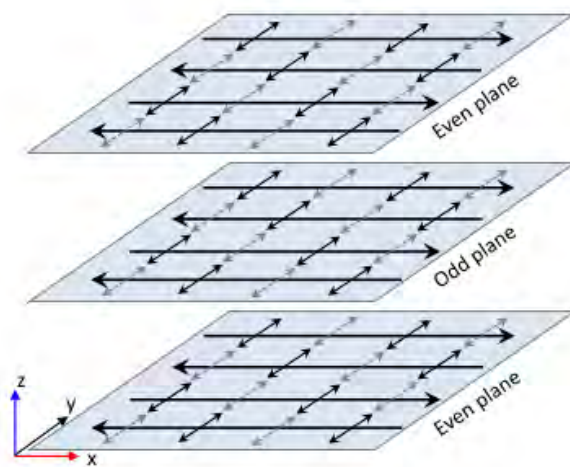


Figure 14 Odd-even Routing

5.2.3.1 Configuration for XYZ Routing - Random Traffic

The arguments of the first simulation are the following as indicated in Table 9:

Traffic			
-traffic	random		
Routing			
-routing	XYZ		
Selection			
-sel	random		
Warmup			
-warmup	10000		
PIR	FROM	TO	STEP
-pir	0.001	0.0328	0.00159
Channel Buffer Depth			
-buffer	4		
Topology			
-dimx	8		
-dimy	8		
-dimz	4		
Packet Size	Min	Max	
-size	8	8	

Table 9 Arguments for xyz routing and random traffic simulation.

The rest of the configuration is the default as it is mentioned in Table 3.

5.2.3.2 Configuration for Odd-even Routing - Random Traffic

The arguments of the first simulation are the following as indicated in Table 10:

Traffic			
-traffic	random		
Routing			
-routing	oe_z		
Selection			
-sel	random		
Warmup			
-warmup	10000		
PIR	FROM	TO	STEP

-pir	0.001	0.0328	0.00159
Channel Buffer Depth			
-buffer	4		
Topology			
-dimx	8		
-dimy	8		
-dimz	4		
Packet Size	Min	Max	
-size	8	8	

Table 10 Arguments for odd-even routing and random traffic simulation.

The rest of the configuration is the default as it is mentioned in Table 3.

5.2.3.3 Performance Results for XYZ vs Odd-even under random traffic

As it is depicted in Figures 15 - 18, under random traffic XYZ routing has better performance than OddEven in terms of Average Latency, Throughput, and Max Delay. OddEven decreases at Total Energy after 0.02 PIR. Maximum Average Latency for XYZ is 75 cycles, maximum Throughput 0.25 flits/cycle and Max Delay 1500 cycles.

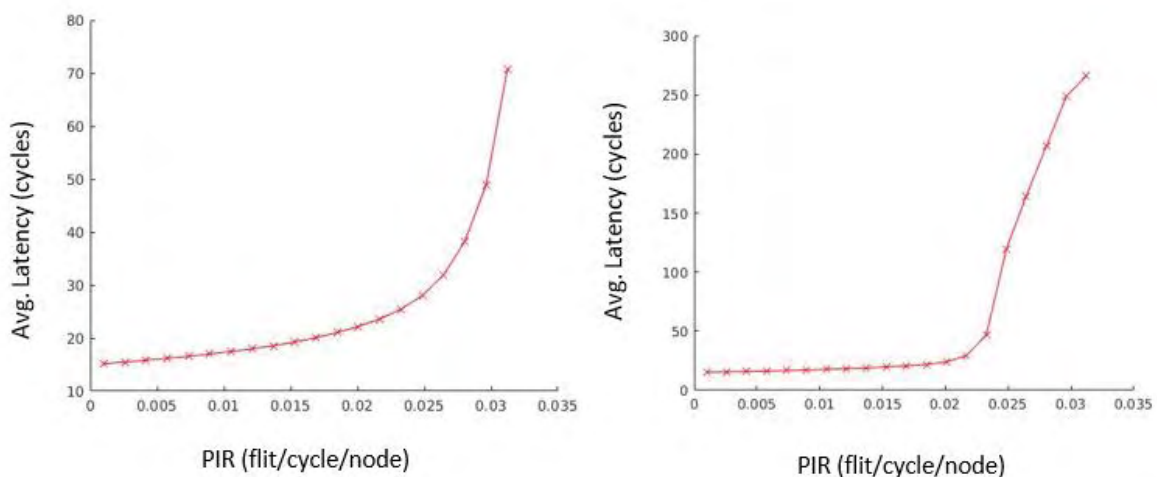


Figure 15 Latency for XYZ (left) vs oddeven (right) under random traffic

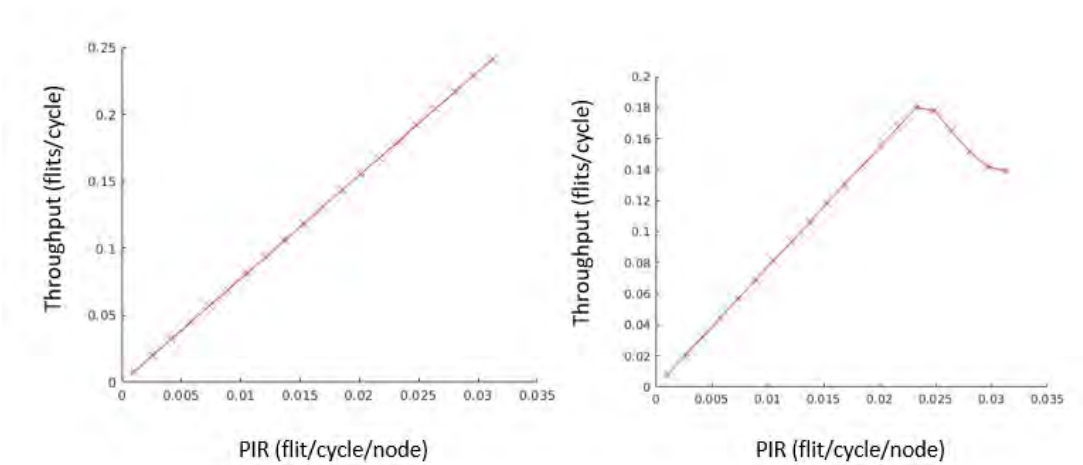


Figure 16 Throughput for XZY (left) vs oddeven (right) under random traffic

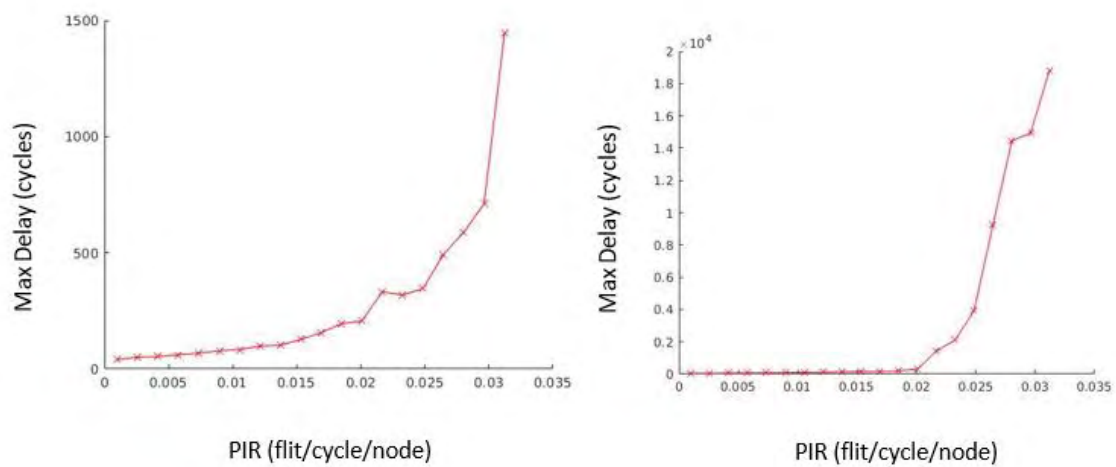


Figure 17 Max Delay for XZY (left) vs oddeven (right) under random traffic

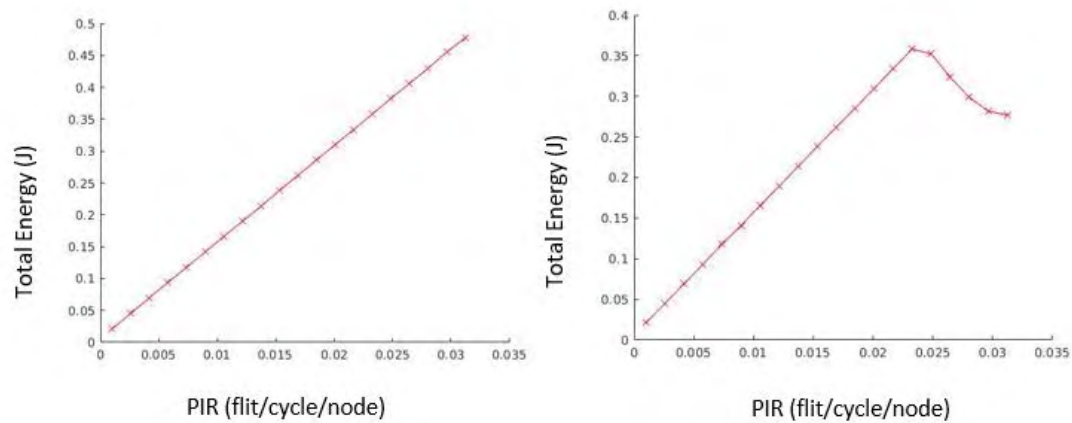


Figure 18 Total Energy for XYZ (left) vs oddeven (right) under random traffic

5.2.3 Experiment 2: Different routing algorithms under Transpose 1 traffic pattern

In transpose1 traffic, the coordinates of a destination node ($\text{meshx} - s.y - 1$, $\text{meshy} - s.x - 1$, $s.z$) transpose x coordinate and y coordinate of the source node ($s.x$, $s.y$, $s.z$). meshx and meshy are the size values of x coordinate and y coordinate from the topology size.

5.2.3.1 Configuration for XYZ Routing - Transpose1 Traffic

The arguments of the first simulation are the following as indicated in Table 11:

Traffic			
-traffic	transpose1		
Routing			
-routing	XYZ		
Selection			
-sel	random		
Warmup			
-warmup	10000		
PIR	FROM	TO	STEP
-pir	0.001	0.0328	0.00159
Channel Buffer Depth			
-buffer	4		
Topology			
-dimx	8		

-dimy	8	
-dimz	4	
Packet Size	Min	Max
-size	8	8

Table 11 Arguments for XZY routing and transpose1 traffic simulation

The rest of the configuration is the default as it is mentioned in Table 3.

5.2.3.2 Configuration for Odd-even Routing - Transpose1 Traffic

The arguments of the first simulation are the following as indicated in Table 12:

Traffic			
-traffic	transpose1		
Routing			
-routing	oe_z		
Selection			
-sel	random		
Warmup			
-warmup	10000		
PIR	FROM	TO	STEP
-pir	0.001	0.0328	0.00159
Channel Buffer Depth			
-buffer	4		
Topology			
-dimx	8		
-dimy	8		
-dimz	4		
Packet Size	Min	Max	
-size	8	8	

Table 12 Arguments for odd-even routing and transpose1 traffic simulation

The rest of the configuration is the default as it is mentioned in Table 3

5.2.3.3 Performance Results for XYZ vs Odd-even under transpose1 traffic

As it is depicted in Figures 19 - 22, under transpose1traffic OddEven has better performance than XYZ in terms of Average Latency, Throughput, and Max Delay. XYZ outcomes slightly OddEven at the energy consumption. Maximum Average Latency for OddEven is 120 cycles, maximum Throughput 0.23 flits/cycle and Max Delay 2500 cycles.

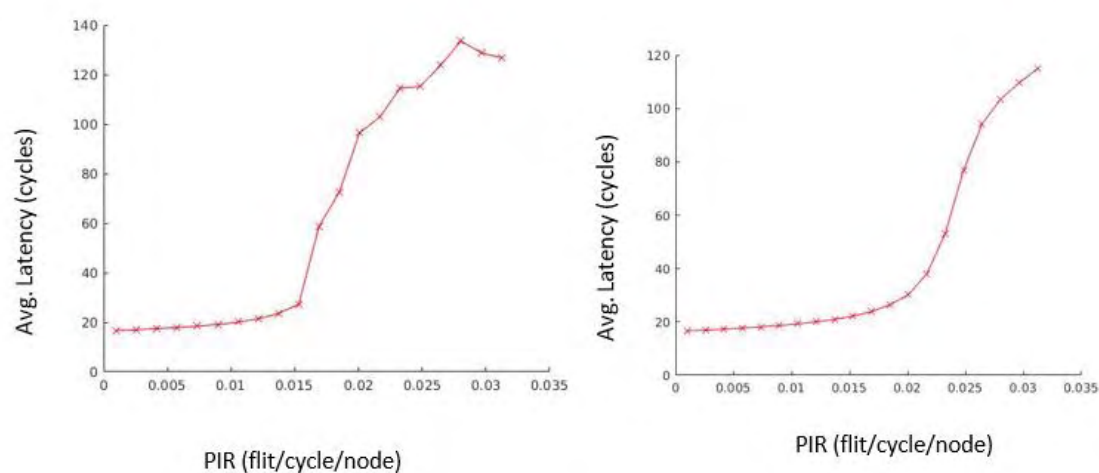


Figure 19 Latency for XYZ (left) vs oddeven (right) under transpose1 traffic

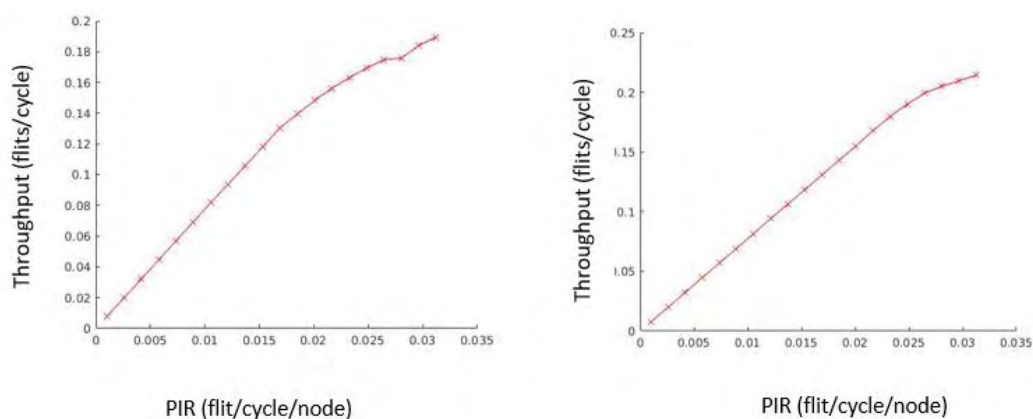


Figure 20 Throughput for XYZ (left) vs oddeven (right) under transpose1 traffic

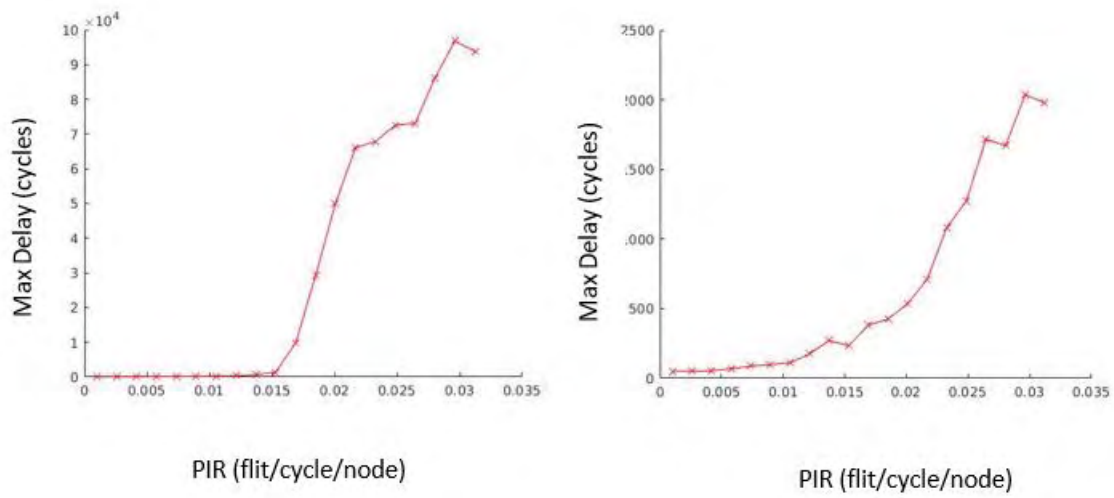


Figure 21 Max Delay for XYZ (left) vs oddeven (right) under transpose1 traffic

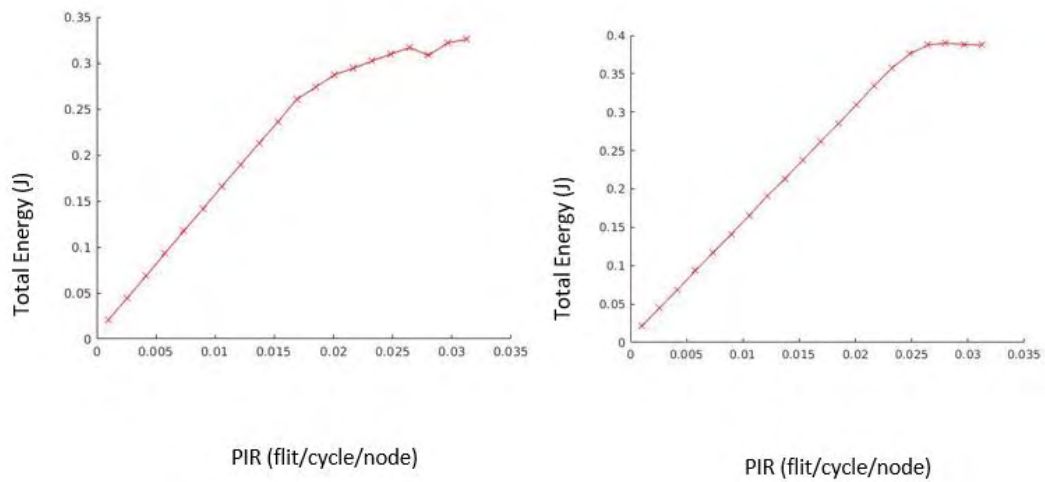


Figure 22 Total Energy for XYZ (left) vs oddeven (right) under transpose1 traffic

5.3 Selection Function Experiments

5.3.1 Introduction

In this chapter, we will run some experiments with different selection functions and adaptive routing, to observe the properties of NoC on the Access Noxim. While we use adaptive routing algorithm, routing function will give us more than one direction to route the packet. We need to decide which way to route to get a better performance. A set of experiments has been designed in order to observe the property of random, buffer-level and NoP selection.

5.3.2 Experiment 3: Different selection functions for Adaptive Routing algorithm

Adaptive routing, also called dynamic routing, is a process for determining the optimal path a data packet should follow through a network to arrive at a specific destination.

Buffer-level is a selection function based on buffer occupancy, which selects the output whose connected input port has the minimum occupied buffer [41].

Another selection function called neighbours-on-path (NoP) is proposed in [42] for NoC architectures. In NoP, each router grasps buffer availability information of every neighbour router and stores this congestion information locally. When a router is selecting output channel for a packet, it queries its adjacent routers stored congestion information and makes the best decision. In this approach, each router also needs the buffer availability information of its neighbour's neighbour to make the routing decision.

The random selection function randomly chooses one output channel from all admissible channels provided by the routing function.

5.3.2.1 Configuration for Adaptive Routing - Random Selection

The arguments of the first simulation are the following as indicated in Table 13 :

Traffic	
---------	--

-traffic	random		
Routing			
-routing	fullyadaptive		
Selection			
-sel	random		
Warmup			
-warmup	10000		
PIR	FROM	TO	STEP
-pir	0.001	0.0328	0.00159
Channel Buffer Depth			
-buffer	4		
Topology			
-dimx	8		
-dimy	8		
-dimz	4		
Packet Size	Min	Max	
-size	8	8	

Table 13 Arguments for fullyadaptive routing and random selection simulation

The rest of the configuration is the default as it is presented at Table 3.

5.3.2.2 Configuration for Adaptive Routing - NoP Selection

The arguments of the first simulation are the following as indicated in Table 14:

Traffic			
-traffic	nop		
Routing			
-routing	fullyadaptive		
Selection			
-sel	random		
Warmup			
-warmup	10000		
PIR	FROM	TO	STEP
-pir	0.001	0.0328	0.00159
Channel Buffer Depth			
-buffer	4		
Topology			
-dimx	8		
-dimy	8		
-dimz	4		
Packet Size	Min	Max	

-size	8	8
-------	---	---

Table 14 Arguments for fullyadaptive routing and nop selection simulation

The rest of the configuration is the default as it is mentioned in Table 3.

5.3.2.3 Configuration for Adaptive Routing - BufferLevel Selection

The arguments of the first simulation are the following as indicated in Table 15:

Traffic			
-traffic	random		
Routing			
-routing	fullyadaptive		
Selection			
-sel	bufferlevel		
Warmup			
-warmup	10000		
PIR	FROM	TO	STEP
-pir	0.001	0.0328	0.00159
Channel Buffer Depth			
-buffer	4		
Topology			
-dimx	8		
-dimy	8		
-dimz	4		
Packet Size	Min	Max	
-size	8	8	

Table 15 Arguments for fullyadaptive routing and bufferlevel selection simulation

The rest of the configuration is the default as it is mentioned in Table 3.

5.3.2.4 Performance Results for Adaptive Routing under NoP, BufferLevel and

Random Selection

As it is depicted in Figures 23-26, adaptive routing has better performance under Random Selection Function in terms of Average Latency, Max Delay, and Total Energy. NoP and Bufferlevel have better Throughput. Maximum Average Latency for Random is 20 cycles, maximum Max Delay 100 cycles and maximum Total Energy 0.17 J. NoP and Bufferlevel have maximum Throughput 0.12 flits/cycle.

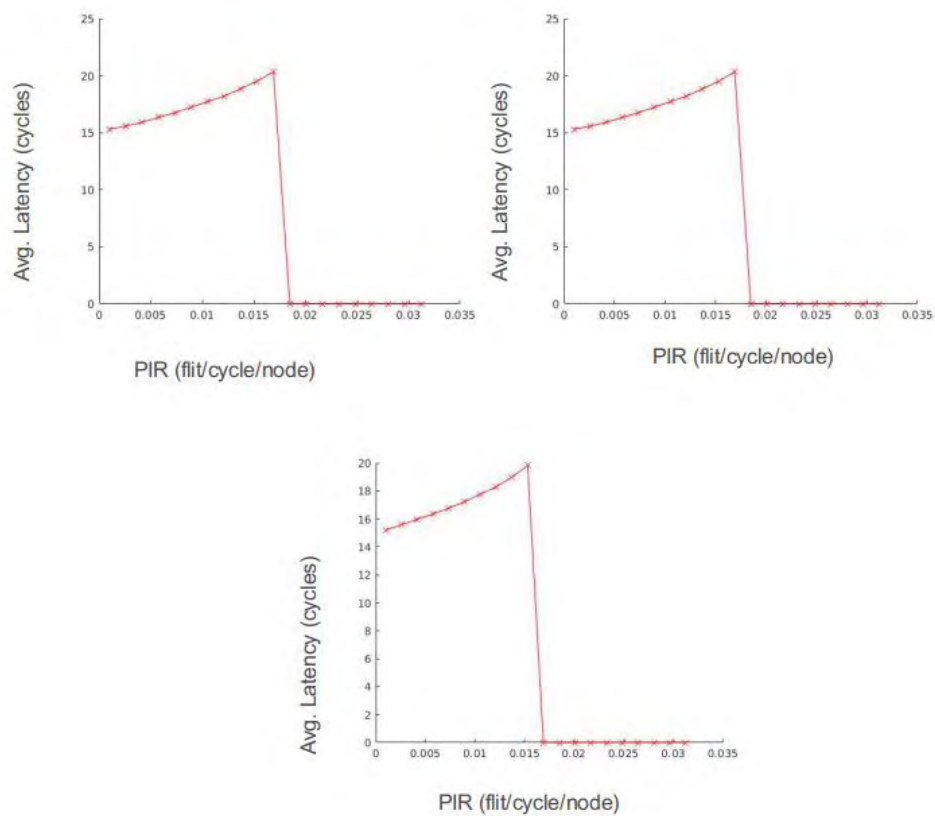


Figure 23 Latency for adaptive routing under bufferlevel selection (top-left) vs nop selection (top-right) vs random selection(bottom)

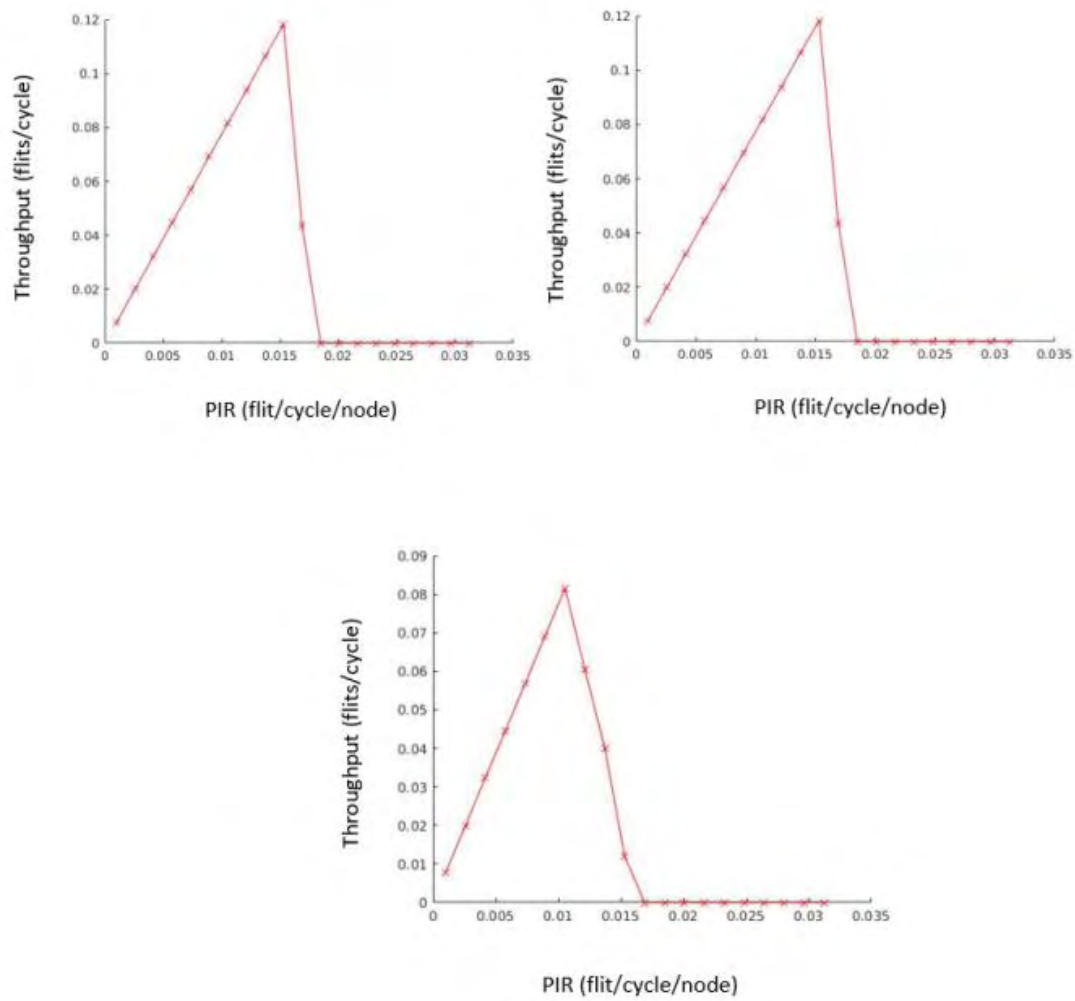


Figure 24 Throughput for adaptive routing under bufferlevel selection (top-left) vs nop selection (top-right) vs random selection (bottom)

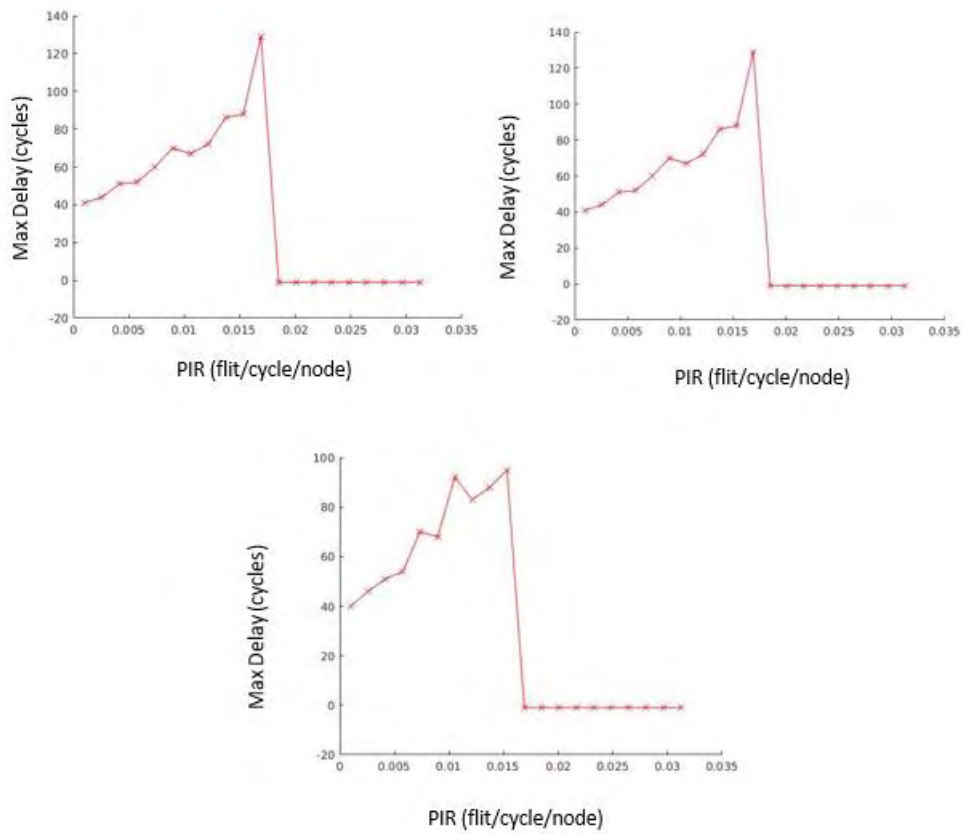


Figure 25 Max Delay for adaptive routing under bufferlevel selection (top-left) vs nop selection (top-right) vs random selection (bottom)

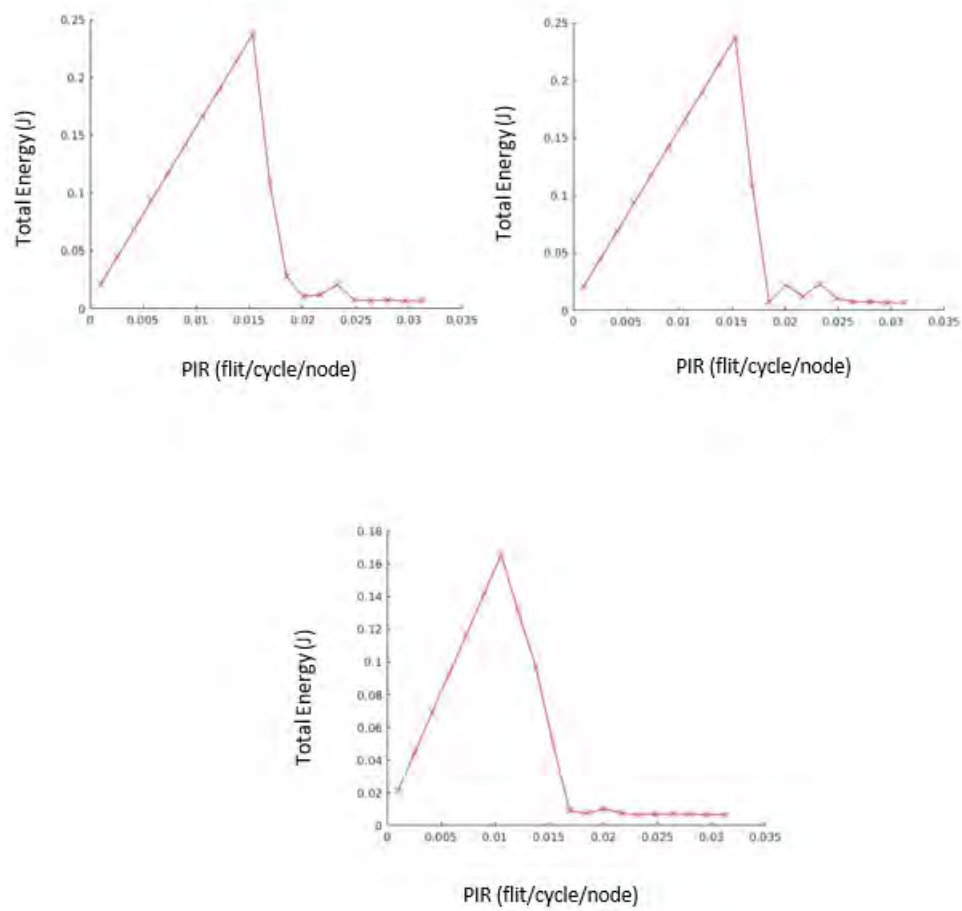


Figure 26 Total Energy for adaptive routing under bufferlevel selection (top-left) vs nop selection (top-right) vs random selection (bottom)

Conclusion and Future Work

This thesis is a survey regarding the NoC technology, NoC simulations and NoC performance study. After analyzing the need of NoCs, the NoC architecture and design space are presented. We focus on methods and parameters to assess the performance of a NoC and introduce and compare different Academic NoCs simulators that support different topologies. We focus on Access Noxim simulator and explain its capabilities, limitations and usage. Finally we run a set of experiments to evaluate the performance of a 3D mesh NoC with Access Noxim.

It is observed that while the PIR increases, the Latency increases as well. On the other hand, while the PIR increases until 0.045, the Throughput increases as well but there is a decrease at 0.05. Moreover, while the Simulation Cycle increases the Latency remains at the same levels and the Throughput decreases.

Under the same traffic pattern, different routing algorithms have different performance. XYZ and oddeven routing algorithms have been compared under random and transpose1 traffic. The results indicate that XYZ routing algorithm under random traffic has the best performance with maximum Average Latency 75 cycles, maximum Throughput 0.25 flits/cycle and maximum Max Delay 1500 cycles. Oddeven routing algorithm under random traffic has better performance in terms of Total Energy consumption since it decreases after 0.02 PIR.

A set of experiments has been designed in order to observe the property of Random, Buffer-level and NoP selection under adaptive routing algorithm and decide which way to route to get a better performance. It turned out that adaptive routing has better performance under Random Selection Function in terms of Average Latency with maximum value 20 cycles, Max Delay with maximum value 100 cycle, and Total Energy with maximum value 0.17J. NoP and Bufferlevel Selection Functions have better Throughput with maximum value 0.12 flits/cycle.

Another research direction would be to compare flow control algorithms with different packet and buffer sizes. Access Noxim source code could be modified to change the arbitration algorithm and compare for example the performance among original arbiter, fixed priority, round-robin and matrix arbiter under the same traffic. Finally, the thermal distribution of each simulation could be plotted.

Bibliography

- [1] M. Coppola, M. D. Grammatikakis, R. Locatelli, G. Maruccia, and L. Pieralisi, Design of cost-efficient interconnect processing units: Spidergon STNoC. CRC press, 2008.
- [2] Kamal, Rajeev & Yadav, Neeraj. (2012). NOC AND BUS ARCHITECTURE: A COMPARISON. International Journal of Engineering Science and Technology. 4.
- [3] Kundu, Santanu; Chattopadhyay, Santanu (2014). Network-on-chip: the Next Generation of System-on-Chip Integration (1st ed.). Boca Raton, FL: CRC Press.
- [4] Bjerregaard T, Mahadevan S (2006). A survey of research and practices of network-on-chip. ACM Comput Surv 38
- [5] Cota É, Morais A, Lubaszewski M.S (2012). Reliability, Availability and Serviceability of Networks-on-Chip. Retrieved from 10.1007/978-1-4614-0791-1
- [6] L.-S. Peh, S. Keckler, and S. Vangal, "On-chip networks for multicore systems," in Multicore Processors and Systems, ser. Integrated Circuits and Systems, S. W.Keckler, K. Olukotun, and H. P. Hofstee, Eds. Springer US, 2009, pp. 35–71.[Online]. Available: http://dx.doi.org/10.1007/978-1-4419-0263-4_2
- [7] Dally WJ, Towles BP (2004) Principles and practices of interconnection networks. The Morgan Kaufmann series in computer architecture and design. Morgan Kaufmann, Burlington
- [8] Chelcea T, Nowick SM (2001) Robust interfaces for mixed-timing systems with application to latency-insensitive protocols. In: Proceedings of the 38th design automation conference (DAC), Las Vegas, pp 21–26
- [9] Glass C, Ni L (1994) The turn model for adaptive routing. J Assoc Comput Mach 41(5):874–902
- [10] Zeferino CA, Susin AA (2003) SoCIN: a parametric and scalable network-on-chip. In: Proceedings of the 16th symposium on integrated circuits and systems design (SBCCI), Sao Paulo, pp 169–174
- [11] Millberg M Nilsson E, Thid R, Jantsch A (2004). Guaranteed bandwidth using looped containers in temporally disjoint networks within the nostrum network-on-chip. In: Proceedings of design, automation and testing in Europe conference (DATE), Paris, pp 890–895
- [12] Goossens K, Dielissen J, Radulescu A (2005) Æthereal network on chip: concepts, architectures and implementations. IEEE Design Test Comput 22(5):414–421
- [13] Chiu G-M (2000) The odd-even turn model for adaptive routing. IEEE Trans Parallel Distrib Syst 11(7):729–738
- [14] OCPIP (2011) <http://www.ocpip.org/>. Accessed 25 May 2011
- [15] VSI Alliance (2011) Virtual component interface standard version 2. VSI alliance. www.vsi.org. Accessed 26 May 2011
- [16] ARM (2011) AMBA advanced extensible interface (AXI) protocol specification, version 2.0. <http://www.arm.com>. Accessed 26 May 2011

- [17] Philips Semiconductors (2002) Device Transaction Level (DTL) protocol specification, version 2.2
- [18] Pasricha S, Dutt N (2008) On-chip communication architectures: system on chip interconnect (systems on silicon). Morgan Kaufmann, Burlington
- [19] Duato J, Yalamanchili S, Ni LM (2003) Interconnection networks: an engineering approach. Morgan Kaufmann, Burlington
- [20] Dv, Nam & Ho, Mau-Viet & Nguyen, Van-Cuong & Son, Ngo & Effiong, Charles. (2017). The Analyzes of Network-on-Chip Architectures Based on NOXIM Simulator. 603-611. 10.1007/978-3-319-49073-1_64.
- [21] Singh, J.K., et al: Performance evaluation of different routing algorithms in network on chip. In: 2013 IEEE Asian Pacific Conference on Postgraduate Research in Microelectronics and Electronics (2013)
- [22] N. Jiang, G. Michelogiannakis, D. Becker, B. Towles, and W. Dally, "Booksim interconnection network simulator," Online, <https://nocs.stanford.edu/cgi-bin/trac.cgi/wiki/Resources/BookSim>.
- [23] F. Fazzino, M. Palesi, and D. Patti, "Noxim: Network-on-chip simulator," URL: <http://sourceforge.net/projects/noxim>, 2008.
- [24] A. T. Tran and B. M. Baas, "Noctweak: A highly parameterizable simulator for early exploration of performance and energy of networks on-chip," Tech. rep., VLSI Computation Lab, ECE Department, University of California, Davis, Tech. Rep., 2012.
- [25] S. K. Mandal, N. Gupta, A. Mandal, J. Malave, J. D. Lee, and R. Mahapatra, "Nocbench: a benchmarking platform for network on chip," in Workshop on Unique Chips and Systems (UCAS), 2009.
- [26] H. Wang, et al., "Orion: A Power-Performance Simulator for Interconnection Networks," in 35th Annual IEEE/ACM International Symposium on Microarchitecture, 2002, pp. 294 - 305.
- [27] A. Kahng, et al., "ORION 2.0: A Fast and Accurate NoC Power and Area Model for Early-Stage Design Space Exploration " in Proceedings of Design Automation and Test in Europe, 2009.
- [28] Orion tool website. Available: www.princeton.edu/~peh/orion.html
- [29] M. Lis, et al., "DARSIM: a parallel cycle-level NoC simulator," in 6th Annual Workshop on Modeling, Benchmarking and Simulation, 2010.
- [30] M. Ali, et al., "Using the NS-2 Network Simulator for Evaluating Network on Chips (NoC)," in International Conference on Emerging Technologies, 2006, pp. 506 - 512.
- [31] NS-2 website. Available: <https://www.isi.edu/nsnam/ns/>
- [32] R. Kourdy, et al., "Using the NS-2 Network Simulator for Evaluating Multi Protocol Label Switching in Network-on-Chip," in 2nd International Conference on Computer Research and Development, 2010, pp. 795 - 799.
- [33] R. Lemaire, et al., "Performance evaluation of a NoC-based design for MC-CDMA telecommunications using NS-2," in The 16th IEEE International Workshop on Rapid System Prototyping, 2005, pp. 24 - 30.
- [34] Kai-Yuan Jheng, Chih-Hao Chao, Hao-Yu Wang, and An-Yeu Wu, "Trafficthermal mutual-coupling co-simulation platform for three-dimensional Network-on-Chip," in Proc. int. Sym. VLSI Design Automation and

Test (VLSI-DAT), pp.135-138, April 2010.

- [35] Wikipedia, "SystemC", <http://en.wikipedia.org/wiki/SystemC>
- [36] "Hotspot", <http://lava.cs.virginia.edu/HotSpot/>
- [37] "Noxim, the NoC Simulator", <http://noxim.sourceforge.net>
- [38] A. Ganguly, K. Chang, P. P. Pande, B. Belzer and A. Nojeh, "Performance evaluation of wireless networks on chip architectures," in Proceedings of the 10th International Symposium on Quality of Electronic Design, pp. 350-355, 2009.
- [39] C. H. Chao, K. Y. Jheng, H. Y. Wang, J. C. Wu, and An-Yeu Wu, "Traffic- and thermal-aware run-time thermal management scheme for 3D NoC systems," in Proc. ACM/IEEE Int. Symp. Networks-on-Chip (NoCS), Grenoble, France, pp. 223-230, May 2010.
- [40] Wang Zhang, Ligang Hou, Jinhui Wang, Shuqin Geng, Wuchen Wu, "Comparison Research between XY and Odd-Even Routing Algorithm of a 2-Dimension 3X3 Mesh Topology Network-on-Chip," *gcis*, vol. 3, pages 329-333, 2009 WRI Global Congress on Intelligent Systems, 2009
- [41] Holsmark, R.D, Kumar, S, Catania, V, Palesi, M.: 'Application specific routing algorithms for networks on chip', *IEEE Trans. Parallel Distrib. Syst.*, 2009, 20, (3), pp. 316–330
- [42] Catania, V, Palesi, M., Patti, D, Acsia, G.: 'Implementation and analysis of a new selection strategy for adaptive routing in networks-on-chip', *IEEE Trans. Comput.*, 2008, 57, (6), pp. 809–820
- [43] Dalirsani A., Holst S., Elm M., Wunderlich H., "Structural Test for Graceful Degradation of NoC Switches", *European Test Symposium*, pp. 183-188, 2011.