Performance Analysis and Design Space Exploration of On-Chip Interconnection Networks

Abbas Eslami Kiasari

Doctoral Thesis in Electronic and Computer Systems
KTH Royal Institute of Technology
Stockholm, Sweden 2013

TRITA-ICT/ECS AVH 13:21
ISSN 1653-6363
ISRN KTH/ICT/ECS/AVH-13/21-SE
ISBN 978-91-7501-923-9

KTH School of Information and
Communication Technology,
Department of Electronic Systems
SE-164 40 Kista, SWEDEN

Academic dissertation for the Degree of Doctor of Philosophy in Electronic and Computer Systems at KTH Royal Institute of Technology to be publicly defended on Wednesday, 18 December 2013 at 13:00 in Sal D, Forum, Isafjordsgatan 39, Kista.

Tryck: Universitetservice US AB

Abstract

The advance of semiconductor technology, which has led to more than one billion transistors on a single chip, has enabled designers to integrate dozens of IP (intellectual property) blocks together with large amounts of embedded memory. These advances, along with the fact that traditional communication architectures do not scale well have led to significant changes in the architecture and design of integrated circuits. One solution to these problems is to implement such a complex system using an on-chip interconnection network or network-on-chip (NoC). The multiple concurrent connections of such networks mean that they have extremely high bandwidth. Regularity can lead to design modularity providing a standard interface for easier component reuse and improved interoperability.

The present thesis addresses the performance analysis and design space exploration of NoCs using analytical and simulation-based performance analysis approaches. At first, we developed a simulator aimed to performance analysis of interconnection networks. The simulator is then used to evaluate the performance of networks topologies and routing algorithms since their choice heavily affect the performance of NoCs. Then, we surveyed popular mathematical formalisms – queueing theory, network calculus, schedulability analysis, and dataflow analysis – and how they have been applied to the analysis of on-chip communication performance in NoCs. We also addressed research problems related to modelling and design space exploration of NoCs.

In the next step, analytical router models were developed that analyse NoC performance. In addition to providing aggregate performance metrics such as latency and throughput, our approach also provides feedback about the network characteristics at a fine-level of granularity. Our approach explicates the impact that various design parameters have on the performance, thereby providing invaluable insight into NoC design. This makes it possible to use the proposed models as a powerful design and optimisation tool.

We then used the proposed analytical models to address the design space exploration and optimisation problem. System-level frameworks to address the application mapping and to design routing algorithms for NoCs were presented. We first formulated an optimisation problem of minimizing average packet latency in the network, and then solved this problem using the simulated annealing heuristic. The proposed framework can also address other design space exploration problems such as topology selection and buffer dimensioning.

Acknowledgement

I am happy that the day has come when I can write these lines. This dissertation is about packets (message) traversing communication networks. I am blessed to be surrounded by a great network and this is my stage to send out some messages.

First and foremost, I would like to thank my advisor, Prof. Axel Jantsch, for being such a great mentor during all these years at KTH Royal Institute of Technology. It was really an honour for me to have an opportunity to be in his research team which has helped me greatly to learn new things both in research and life. Without his inspiration, patience, friendship and our stimulating discussions, this thesis would have never been possible. Besides my advisor, I would like to extend my sincere gratitude to Assoc. Prof. Zhonghai Lu for his support of my work in his role as secondary advisor. I would also like to thank Assoc. Prof. Ingo Sander for his kind and generous feedbacks for his role as the reviewer of my thesis.

I am particularly indebted to my Masters advisor, Prof. Hamid Sarbazi-Azad of the Sharif University of Technology for being such a great source of inspiration. I also want to take the opportunity to thank all my teachers throughout the years. There are many whom I owe thanks.

I wish to express my love and gratitude to all my family members, especially my father, for their encouragements and supports throughout all my studies from primary school to current level. My special thanks go to my beloved wife, Fahimeh Jafari, for the invaluable support she has given me during the last eleven years, and undoubtedly, without her support I could not continue my studies after the Bachelor's degree. Thank you very much Fahimeh! Last, but not least, I would like to thank my friends at ICT school, especially Dr. Farzad Kamrani, for making my life better and sharing memorable times with me in Sweden.

Dedicated to my first teacher,
Mr. Reza Mortazavi Kiasari,
who taught me to write.

Table of Contents

List of Figures

List of Tables

List of Acronyms

ASIC	Application-Specific Integrated Circuits
APL	Average Packet Latency
AxD	Average hop x standard Deviation
BDF	Boolean Dataflow
CAR	Congestion-Aware Routing
CDF	Cumulative Distribution Function
CDG	Channel Dependency Graph
CG	Communication Graph
CM	Cycle Mean
CMP	Chip Multiprocessor
CSDF	Cyclo-Static Dataflow
CV	Coefficient of Variation
DDF	Dynamic Dataflow
DOR	Dimension-Order Routing
FCFS	First-Come First-Serve
FIFO	First-In First-Out
HSDF	Homogeneous Synchronous Dataflow
IP	Intellectual Property
LAR	Latency-Aware Routing
LCFS	Last-Come First-Serve
MCM	Maximum Cycle Mean
MMPP	Markov-Modulated Poisson Process
MMS	Multimedia System
MPSoC	Multiprocessor System-on-Chip
MoC	Model of Computation
NoC	Network-on-Chip
pdf	Probability Density Function
PE	Processing Element
PR	Priority
PSDF	Parameterized Synchronous Dataflow
RR	Round Robin
RS	Random Service
RTL	Register Transfer Level
SADF	Scenario-Aware Dataflow
SDF	Synchronous Dataflow
SoC	System-on-Chip
TG	Topology Graph
VOPD	Video Object Plane Decoder
VRDF	Variable Rate Dataflow

List of Publications

• Conference Proceedings

1. Dara Rahmati, Abbas Eslami Kiasari, Shaahin Hessabi, and Hamid Sarbazi-Azad, "A Performance and Power Analysis of WK-Recursive and Mesh Networks for Network-on-Chips," *In the Proceedings of the 24th IEEE International Conference on Computer Design* (*ICCD*), pp. 142-147, San Jose, CA, USA, Oct. 2006.

2. Abbas Eslami Kiasari, Dara Rahmati, Hamid Sarbazi-Azad, and Shaahin Hessabi, "A Markovian Performance Model for Networks-on-Chip," *In the Proceedings of the 16th Euromicro International Conference on Parallel, Distributed and Network-Based Processing* (*PDP*), pp. 157-164, Toulouse, France, Feb. 2008.

3. Abbas Eslami Kiasari, Shaahin Hessabi, and Hamid Sarbazi-Azad, "PERMAP: A Performance-Aware Mapping for Application-Specific SoCs," *In the Proceedings of the 19th IEEE International Conference on Application-specific Systems, Architectures and Processors* (*ASAP*), pp. 73-78, Leuven, Belgium, Jul. 2008.

4. Abbas Eslami Kiasari, Hamid Sarbazi-Azad, and Shaahin Hessabi, "Caspian: A Tunable Performance Model for Multi-Core Systems," *In the Proceedings of the 14th European Conference on Parallel and Distributed Computing* (*Euro-Par*), Lecture Notes in Computer Science, vol. 5168, pp. 100-109, Canary Island, Spain, Aug. 2008.

5. Dara Rahmati, Abbas Eslami Kiasari, Hamid Sarbazi-Azad, and Shaahin Hessabi, "Power-Efficient Routing Algorithm for Torus NoCs," *In Proceedings of the International Conference on Contemporary Computing* (*IC3*), pp. 211-220, Uttar Pradesh, India, Aug. 2008.

6. Abbas Eslami Kiasari, Axel Jantsch, and Zhonghai Lu, "A Framework for Designing Congestion-Aware Deterministic Routing," *In the Proceedings of the 3rd International Workshop on Network-on-Chip Architectures* (*NoCArc*)*, Held in conjunction with the 43rd Annual IEEE/ACM International Symposium on Microarchitecture* (*MICRO-43*), pp. 45-50, Atlanta, Georgia, USA, Dec. 2010.

• Tutorial

7. Abbas Eslami Kiasari, Axel Jantsch, Marco Bekooij, Alan Burns, and Zhonghai Lu, "Analytical Approaches for Performance Evaluation of Networks-on-Chip," *In the Proceedings of the International Conference on Compilers, Architectures and Synthesis for Embedded Systems* (*CASES*), pp. 211-212, Tampere, Finland, Oct. 2012.

• Journal Papers

8. Dara Rahmati, Hamid Sarbazi-Azad, Shaahin Hessabi, and Abbas Eslami Kiasari, "Power-efficient Deterministic and Adaptive Routing in Torus Networks-on-Chip," *Microprocessors and Microsystems: Embedded Hardware Design*, vol. 36, no. 7, pp. 571-585, Oct. 2012.

9. Abbas Eslami Kiasari, Zhonghai Lu, and Axel Jantsch, "An Analytical Latency Model for Networks-on-Chip," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 21, no. 1, pp. 113-123, Jan. 2013.

10. Abbas Eslami Kiasari, Axel Jantsch, and Zhonghai Lu, "Mathematical Formalisms for Performance Evaluation of Networks-on-Chip," *ACM Computing Surveys*, vol. 45, no. 3, article no. 38, Jun. 2013.

• Book Chapter

11. Abbas Eslami Kiasari, Axel Jantsch, and Zhonghai Lu, A Heuristic Framework for Designing and Exploring Deterministic Routing Algorithm for NoCs, M. Palesi and M. Daneshtalab, editors, Routing Algorithms in Networks-on-Chip, Springer, 2013, ISBN 978-1-4614-8273-4.

Part I

Introduction

Chapter 1

Introduction

This chapter presents the frame of this thesis work, namely the area of Network-on-Chip (NoC). The design challenges of interconnection networks in many-core architectures motivate our focus on developing new analysis techniques for performance evaluation. At first, the evolution of digital systems is briefly reviewed and then design challenges and structure of NoCs are described. After that, the contribution of the thesis is highlighted. The chapter concludes by presenting the outline of the rest of the thesis.

The last few years have witnessed the emergence of many and varied advanced computing systems. There has also been a shift from personal computers towards portable computing devices that offer uninterrupted internet access. Embedded systems, which were traditionally designed as application-specific integrated circuits, have become more versatile, scaled-down computing systems.

1.1 Evolution of Digital Systems

For the last few decades, general-purpose processors and application-specific integrated circuits (ASICs) have existed as separate parts of digital system design. Usually, application-specific systems are designed in order to efficiently implement a specific application and minimize the implementation costs of that application. An example of such a system is the digital camera. General-purpose computer systems, on the other hand, are primarily intended for the purpose of generality and, often, high performance. In the last few years, however, application-specific systems have increasingly focused on parallel processing, which has led to a movement from single core architectures to many-core architectures. The evolution of the two types is shown in Figure 1.1.

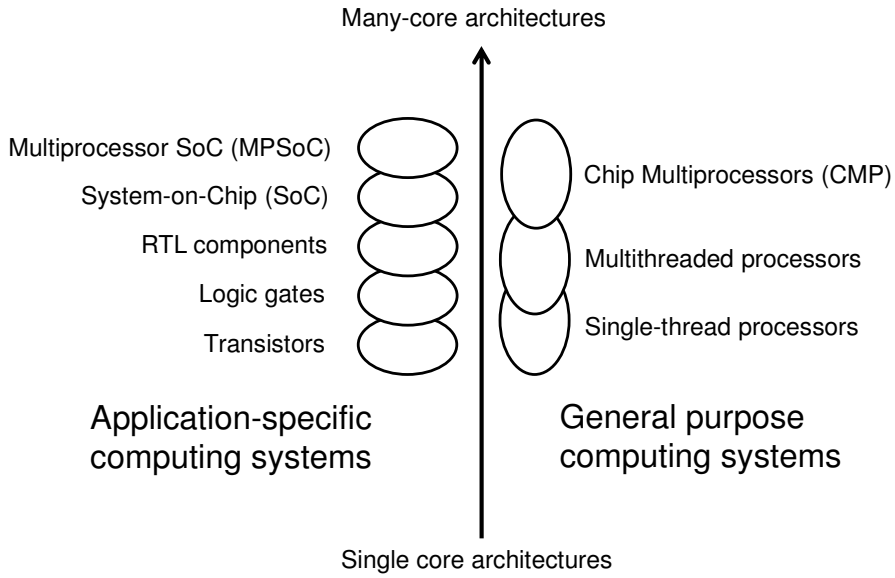


Figure 1.1: Evolution of application-specific and general purpose computing systems

Originally, application-specific computing systems were small gate-level optimised circuits; over the years, they have evolved into complex embedded hardware platforms. Each subsequent generation of technology has enabled systems to become more complex than the last, which has led to calls for new design methodologies to reduce the costs of system design. Consequently, the basic components of ASICs have increased, both in size and complexity, from single transistors to gates and register transfer level (RTL) components, and most recently to pre-designed cores that may even be entire processors. This evolution can be seen on the left-hand side of Figure 1.1. It is possible in the current era of System-on-Chip (SoC) design to create an entire system composed of pre-designed cores and then integrate the system on a single chip. A logical next step in this evolution, therefore, is to design multiprocessor SoCs (also known as MPSoCs) by adding more and more processing elements. This step makes it possible to meet the processing demands of future applications without a significant increase in the design effort and development costs. The emergence of MPSoCs has brought with it numerous new challenges related to parallel computing; at the same time, it has also led to application-specific system design converging with current trends in the design of general-purpose computer systems.

In the search for high performance, general-purpose computer systems have exploited parallelism. Multiprocessor systems have been created by using an interconnection network to connect processors. At the same time, the evolution of technology has led to increasing transistor densities that have made it possible to integrate increasingly complex processor designs into a single chip. Because of this evolution, along with reduced returns from processor pipeline improvements, processor designs are no longer the single-threaded single core processors they once were and, as the right-hand side of Figure 1.1 shows, have become multithreading processors. This evolution has continued in recent years into chip multiprocessor (CMP), with the integration of multiple general purpose processor cores and caches into a single chip. CMPs have become the standard for general purpose systems and future systems are expected to have an increased number of on-chip processor cores.

1.2 Design Challenges

The evolution of digital systems along with the fact that shared-medium busses do not scale well or completely utilize the potentially available bandwidth, have led to significant changes in the architecture and design of integrated circuits. Shrinking feature sizes, combined with relative increases in overall chip size, have caused interconnects to start behaving like lossy transmission lines. Line delays have become much longer than gate delays, which has caused synchronization problems between cores. In long interconnects and in clocking networks, a significant amount of power is dissipated. This trend only gets worse with increased clock frequencies and decreased feature sizes. One solution to these problems is to implement SoCs using an on-chip interconnection network or *network-on-chip* (NoC) which was proposed as "an architecture for billion transistor era" in the beginning of the millennium [Hemani et al. 2000]. The multiple concurrent connections of such networks mean that they have extremely high bandwidth. Regularity can lead to design modularity, which provides a standard interface for easier component reuse and improved interoperability. The fact that the networking resources are shared helps increase overall performance and scalability. However, design automation faces new challenges as a result of the trend towards many-core architecture designs. This section describes some of these challenges, with particular focus on the early analysis tools related to communication infrastructure.

Communication performance is an important design criterion, and the accurate prediction thereof is particularly important, as well as challenging, in the early design stages. Even for a design that has a register-transfer level (RTL) specification or a lower-level implementation, the sensitivity to variation in workloads means that it can still be difficult to estimate performance

accurately. In its early stages, a design may only consist of an interconnected set of components without an RTL embodiment. In such cases, estimates are often formed by scaling a detailed analysis of similar existing designs. The first stage of early analysis is a performance model, either analytic or simulation-based, that embodies a number of assumptions about the design architecture. The NoC must not only be modelled in a way that is sufficiently accurate to evaluate the communication performance, but it must also be fast enough to deal with the challenge of huge design space exploration; this can include finding an optimal network topology, a low congestion routing algorithm, efficient application mapping to processing cores, and sufficient configuration of buffers.

1.3 Problems and Contributions

In the present thesis, the author studied modelling, performance analysis and optimisation of NoC communication architectures, and presented novel design methodologies for NoCs design space exploration. The problems and contributions in the present thesis are divided into the following categories:

1. Design optimisation with simulation performance models
   - *Problem*: Study a new topology and routing algorithms for NoCs
   - *Contribution*: Propose frameworks to design deadlock-free and balanced routing algorithms
2. Analytical performance models
   - *Problem*: Evaluate the average end-to-end packet latency in NoCs
   - *Contribution*: Propose analytical models to estimate the latency
3. Design optimisation with analytical performance models
   - *Problem*: Application mapping and routing problems in NoCs
   - *Contribution*: Propose mapping and routing algorithms that minimises the network congestion and latency

More details and discussions about problems, contributions, and their limitations are provided in Chapter 3.

1.4 Thesis Organization

This Ph.D. thesis is a collection of papers presented with a general introduction to the topic in Part I. The next chapter surveys the related work and other background. The problem formulation and detailed contributions are introduced in Chapter 3. Chapter 4 draws conclusions and presents directions of future work. Part II of the thesis consists of published papers including 1 book chapter, 3 journal papers and 7 conference proceedings.

Chapter 2

Background and Related Work

This chapter is structured in three sections. The first section provides a background in NoC architecture details and the second section presents some of the concepts in the area of performance evaluation. Finally, the third section surveys related work in mathematical formalisms for performance evaluation of NoCs.

2.1 NoC Building Blocks

An on-chip network can be designed by breaking it down into its various building blocks; namely, its topology, flow control, routing, link architecture, and router microarchitecture.

Topology: An NoC is made up of router nodes and channels. The logic connections between the network's nodes and channels are determined by the network topology. Figure 2.1 shows an NoC with a 6x6 mesh topology.

Routing: The routing algorithm determines the path that a message takes through the network to reach its destination. The ability of a routing algorithm to balance load directly impacts the performance of the network.

Switching: Switching mechanism determines the way in which a network allocates resources to messages as they travel through the network. The switching mechanism allocates and de-allocates buffers and channel bandwidth to the packets that wait for them. While it is possible to allocate resources to packets in their entirety (this is done using store-and-forward and virtual cut-through switching), it is impractical to implement it in the NoCs because of the large buffer resources required. The most common method for on-chip networks is to handle flow control at the flit level. Allocating buffers and channel bandwidth on the smaller granularity of flits, as opposed to entire packets, makes it possible to design routers that have smaller buffers.

Router microarchitecture: The following components comprise a generic router microarchitecture: router state, input buffers, allocators, routing logic, and a crossbar (or switch). It is common to pipeline router functionality in order to improve throughput. The main contributor to communication latency is the delay in the on-chip network through each individual router. Because of this, researchers have made significant efforts to reduce router pipeline stages and improve throughput.

Link architecture: All on-chip network prototypes to date have utilized conventional pipelined wires and full-swing logic. Pipelined wires utilize repeaters in order to increase signal reach, but studies are underway to identify alternative link architectures, such as optical networks.

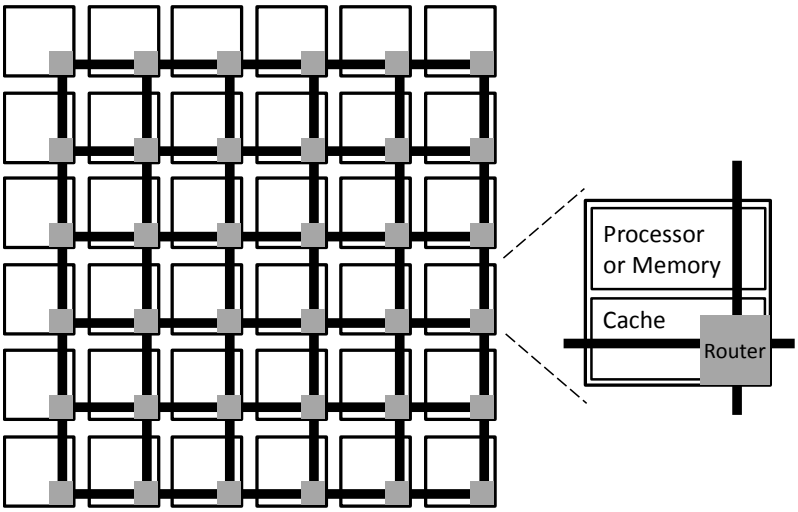


Figure 2.1: An NoC with 6x6 mesh topology

2.2 Performance Evaluation Methods

When designing SoCs, performance is a key factor to take into account. Performance evaluation methods are generally divided into two main areas: performance measurement and performance modelling. Measurement has an advantage over modelling in that it obtains the performance of the real system rather than that of a model of the system. A system may contain interactions that affect performance and are difficult to capture in a model. If these interactions can be captured, perhaps in a detailed model, this model may take an extremely long time to program and run. Performance measurement, on the other hand, can be achieved once a system has been built and instrumented and is functional. This means that performance measurement cannot be used in the SoC design process, and modelling is required in order to predict performance.

Performance modelling is divided into simulation modelling and analytic modelling. Consequently, performance models range widely, from simple analytically tractable models to extremely detailed trace-driven simulation models. Along with the quantitative predictions that are obtained, a principle benefit of performance modelling is the insight into the structure and behaviour of a system that developing a model can create. This insight can be especially valuable during system design and can result in the early discovery and correction of design flaws. Also, it is common to use performance measurement and both analytic and simulation performance models during a system's life cycle. As more information about the design of a system becomes available, more detailed models can be developed.

2.2.1 Simulation

Simulation is a versatile and useful tool to use when evaluating SoC performance. Whereas there are limitations on the range of features that can be modelled using analytic techniques, a simulation model can be constructed to a level of detail that is almost arbitrary, which makes it possible to model extremely complex situations that are analytically intractable. In fact, validation of analytic models is one of the main applications of simulation. Two distinct types of simulation have become widespread in the performance evaluation of interconnection networks [Dally and Towles 2004]. The first of these, *trace-driven workload simulation*, simulates a deterministic model that is driven by a sequence – or trace – obtained from measurements of an existing system. The primary use of trace-driven simulations has been to study the performance of storage hierarchies and processor pipelines. *Synthetic workload simulation* typically simulates a queueing model driven by sequences of random or pseudorandom numbers that have user-specified distributions.

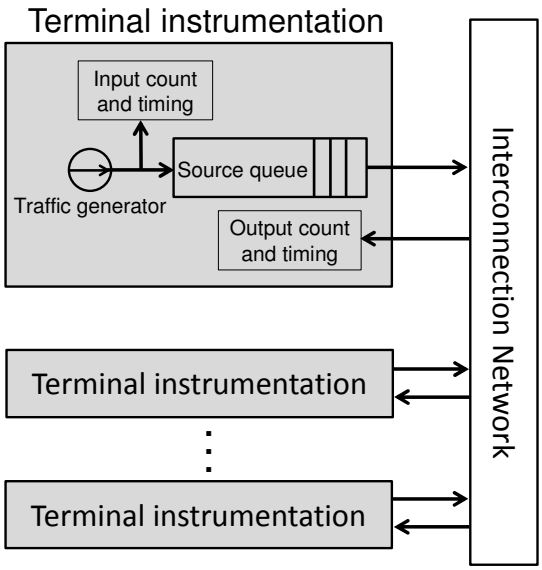
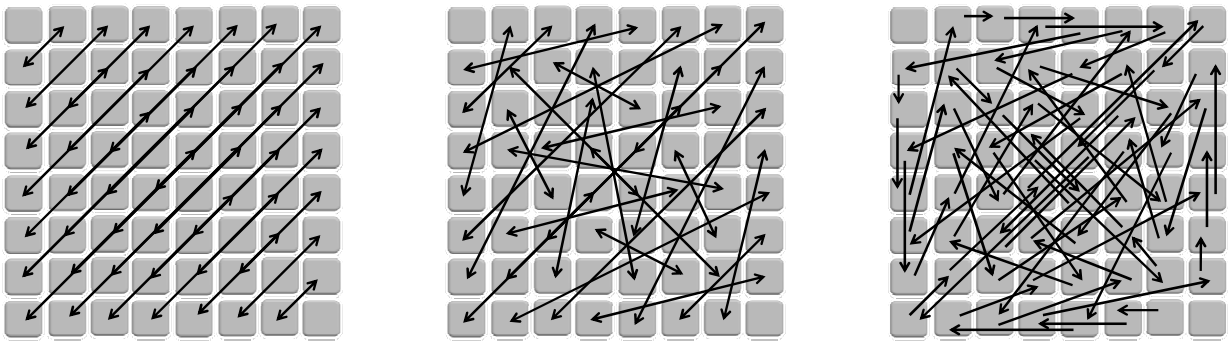


Figure 2.2: Structure of an interconnection network simulation engine

Figure 2.2 shows the structure of an interconnection network simulator. Traffic generator modules produce packet flows that form the synthetic workload in the network and input count and timing module records the number and timing properties of generated packets. Similarly, output count and timing module records the number and timing properties of received packets. By analysing these values, we can estimate the latency and throughput of flows in the system.

Due to complexity of developing and controlling of trace-driven workloads, synthetic workloads are used frequently in NoCs simulation. Not only are synthetic workloads easy to design and manipulate, but they can also capture the most noticeable features of the realistic workloads [Dally and Towles 2004]. Synthetic workload is comprised of three independent distributions: temporal distribution, spatial distribution, and packet length distribution. Temporal distribution refers to the distribution of interarrival time of packets while spatial distribution represents the distribution of the destination of packets in the network. Some examples of temporal distributions are periodic process, Poisson process, and bursty process. In a periodic process, the packets interarrival times are fixed and known while Poisson process incorporates fluctuations in the interarrival times based on the exponential distribution. An example of bursty traffic modelling is given in Section 3.1.1. Uniform, transpose, bit-reversal, and shuffle traffic patterns are well-known examples of spatial distributions used in NoC simulations [Gratz and Keckler 2010]. Figure 2.3 shows these distributions in the 8x8 mesh topology.



(a) (b) (c)

Figure 2.3: (a) Transpose, (b) bit-reversal, and (c) shuffle spatial traffic patterns in 8x8 mesh network

Usually, in a simulation process, resources are idle at the beginning, which results in error in the measurement of network performance metrics. For instance, early injected packets experience less contention and lower latency while later packets see more contention as buffers are starting to fill up gradually. During this *warm-up period* bias is inevitable in simulation results because initial observations are not completely representative of the *steady state*. The best way to eliminate initial observations is to discard them. Once the warm-up period has been completed, observations are considered to come from a steady-state process.

The fact that the input processes that drive a simulation (such as packet size, service times, and interarrival times) are random variables means that the output of such simulations are also random. Estimates of system performance measures are only yielded from runs of the simulation – these are random variables themselves, which makes them subject to sampling error. Consider the example of a simulation experiment that produces observations of a random process $\{X_i, i = 1,2, \dots, N\}$, in which steady-state mean $\mu = E[X_i]$ is expected. In such a case, an interval CI will exist such that $probability\{\mu \in CI\} = CL$, where CL is the *confidence level* and CI is the *confidence interval*.

A number of analysis and data collection methods have been proposed in order to calculate both the CI and CL during steady-state simulation [Pawlikowski 1990]. A common method of estimating the steady-state mean and the variance of collected data with a given CI and CL and is *batch means*. This method involves dividing a series of steady-state observations, with a length *N*, into *B* contiguous and non-overlapping batches with size *M* $(N = M.B)$. The first batch comprises transient information, so once the first batch has been discarded, the next step is to calculate the global mean $\left(\bar{X} = \frac{1}{B-1}\sum_{i=2}^{B}\bar{X}_i\right)$, the batch mean for each of *B*-1 batches $(\overline{X_2}, \overline{X_3}, \dots, \overline{X_B})$, and the variance of those batch means $\left(s_X^2 = \frac{1}{B-2}\sum_{i=2}^{B}(\bar{X}_i - \bar{X})^2\right)$. The next step is to construct a confidence interval with the form $\bar{X} \pm \beta$; where β is the confidence interval half-length provided by $\beta = \frac{s_X}{\sqrt{B-1}} t_{B-2,\frac{\alpha}{2}}$, where $t_{B-2,\frac{\alpha}{2}}$ is the upper $\left(1 - \frac{\alpha}{2}\right)$ critical point of the *student's t distribution* with *B*-2 degrees of freedom [Alexopoulos and Seila 1996]. Put another way, $Probability\{\bar{X} - \beta \leq \mu \leq \bar{X} + \beta\} = 1 - \alpha$. By determining an upper bound for β $(\beta < \varepsilon\bar{X})$ where ε is a pre-specified relative precision, the confidence interval will be $\left((1 - \varepsilon)\bar{X}, (1 + \varepsilon)\bar{X}\right)$. In the present thesis, it is supposed that 10 batches (*B* =10) of *M* observations are collected. If the simulation results are not sufficiently accurate, the simulation will be repeated with larger values of *M*. Since $\beta = \frac{1}{3} s_X t_{8,\alpha/2}$ and $\beta < \varepsilon\bar{X}$ we have $\frac{1}{3} s_X t_{8,\alpha/2} < \varepsilon\bar{X}$, and therefore $s_X/\bar{X} < 3\varepsilon/t_{8,\alpha/2}$. In other words, the *coefficient of variation* $(s_X/\bar{X})$ should be less than a pre-defined threshold. This threshold $\left(3\varepsilon/t_{8,\alpha/2}\right)$ is a function of CI and CL. Table 2.1 shows various coefficient of variation for some confidence intervals and confidence levels. In the experimental results of this thesis the confidence interval and the confidence level are assumed 0.02 and 0.99 respectively. In other words, the coefficient of variation is less than 0.0179.

Table 2.1: Calculated coefficient of variation for some confidence intervals and confidence levels

CL \ CI	0.01	0.02	0.05	0.10	0.20
0.999	0.0060	0.0119	0.0298	0.0595	0.1190
0.99	0.0089	0.0179	0.0447	0.0894	0.1788
0.98	0.0104	0.0207	0.0518	0.1036	0.2072
0.95	0.0130	0.0260	0.0650	0.1301	0.2602
0.90	0.0161	0.0323	0.0806	0.1613	0.3226
0.80	0.0215	0.0429	0.1074	0.2147	0.4295

2.2.2 Analytic modelling

Analytic performance modelling has become widely acknowledged as a cost-effective evaluation technique with which to estimate the performance of interconnection networks [Dally and Towles 2004]. The cost-effectiveness of analytic models stems from the fact that they are based on efficient solutions to mathematical equations. In order for such equations to have a tractable solution, however, it is necessary to make certain simplifying assumptions regarding the structure and behaviour of the queueing network model. Consequently, analytic models cannot capture all of the detail that can be built into simulation models. Nevertheless, the key resources and workload requirements for many types of systems can be modelled analytically with enough realism for them to provide insights into the bottlenecks and key parameters that affect system performance.

There is a widely held belief that carefully constructed analytic models can estimate average job throughputs and device utilizations with up to 90 per cent accuracy and average response time with up to 70 per cent accuracy [Lazowska et al. 1984]. For a preliminary new system design, these levels of accuracy are usually sufficient. An analytic model can make it possible to understand the key factors that affect the performance of a proposed system, as well as helping to determine how sensitive performance is to parameter changes. An analytic model can help provide guidelines regarding the overall design of a system and also help develop more detailed simulation models as the design matures. For example, an analytic model could determine the areas to focus on when building a simulation model. If there is no issue with performance in a certain subsystem, this suggests that the subsystem does not need to be modelled in great detail.

SoC designers often use performance models when making early architecture and design decisions. Engineers will typically construct a performance model and then compare future technology options based on performance model projections. With such a goal in mind, engineers will start by developing the application and architecture models separately, before mapping the application to the architecture and using a performance model to evaluate the selected application–architecture combination. This concept is discussed in detail in the following chapter.

Most current performance models of NoCs tend to rely on simulations. NoC designers have used detailed simulations to explore the design space in order to address performance analysis. Simulation tools are accurate and flexible, but the complexity of modern SoCs limits what can be reasonably simulated. Simulation-based design processes are also disadvantaged by the non-linear behaviour of system performance, which makes it hard to draw conclusions from the simulation results in terms of how to adapt the system hardware or its programming. It can also be difficult to determine the SoC's worst-case behaviour. Using simulation experiments increases the computational intensity of searching for efficient architectures, and does not scale well with the size of networks. As a result, the simulation simply cannot be used in optimisation loops.

An alternative to the aforementioned approach is an analytical model that can estimate the desired performance metrics in much less time. Analytical models can reduce the large design space in a much shorter time than simulation can. Therefore, deriving accurate analytical models for performance prediction of NoCs is justified in order to eliminate the need for time-consuming simulations. Engineers can use the information provided during the performance analysis step in any optimisation loop for NoCs, such as the topology selection, buffer allocation, and application mapping. While high-level models conceal many complex technological aspects, they also facilitate rapid exploration of the NoC design space. As well as providing the timing properties of the system, analytical models also provide useful feedback about the system's behaviour. Accurate simulations can be set up at later steps of the design process, when the design space has been reduced to a small number of practical choices. In short, analytical models have earned a place alongside simulation in the analysis of NoC performance analysis and are likely to grow in importance as NoCs become increasingly complex and irregular.

Latency is one of the most critical design challenges for on-chip interconnection network architectures [Owens et al. 2007]. Latency plays a significant role in the NoC-based system's

performance since it is introduced to every communication pair within the system such as processor units, local memory, shared memory, and cache blocks. Furthermore, it has a direct effect on throughput and power consumption in NoCs. Latency is also the main design challenge in systems with critical timing demands such as real-time SoCs. Therefore, the present thesis assesses the performance in terms of packet latency.

2.3 Mathematical Formalisms for Performance Evaluation

Performance evaluation is important to NoC designers who aim to provide either the highest level of performance at a given cost or a minimum level of performance at the lowest possible cost. In both cases, a reliable measure of performance is indispensable. However, the former case typically focuses on average performance, while the main metric for the latter case is worst-case performance. The worst-case execution time is of particular concern in real-time systems, such as automotive or avionic applications. In such systems, it is important to know the amount of time that might be required in a worst-case scenario in order to guarantee that the task will always complete its jobs before the predetermined deadline. The downside of the worst-case-based design is that it results in resource over-dimensioning. Because of this, average-case-based design methods are usually used for non-time critical applications to increase the efficiency of the system. Below, the author of the present thesis reviews the basic concepts and applications of four analytical performance evaluation methods that are popular for average-case and worst-case performance analysis of NoCs: *queueing theory, network calculus, schedulability analysis*, and *dataflow analysis*.

2.3.1 Queueing Theory

Queueing theory is a branch of probability theory. Figure 2.4 shows a queueing system in which a population of *customers* enters a *service facility* that includes one or multiple servers. If a new customer arrives and all servers are busy, that customer will enter a *queue* and wait until a server becomes available. In order to analyse such a system, it is necessary to identify the arrival process, but also the structure and discipline of the service facility.

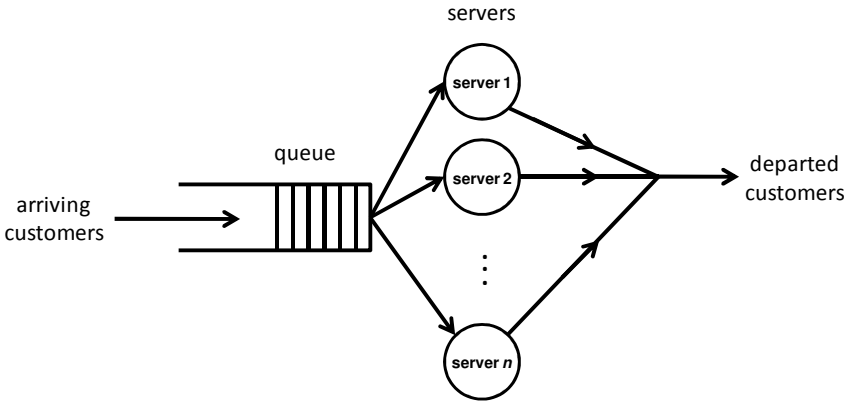


Figure 2.4: Model of a queueing system.

Queueing theory specifies the arrival process and service time probabilistically. In terms of the structure and discipline of the service facility, it is necessary to specify a range of additional quantities, including the amount of storage capacity available to handle waiting customers, the number of available service stations, and the queueing discipline. Along with distributions of interarrival and service times, queueing systems may vary in terms of the number of servers, the capacity of a queue (whether it is infinite or finite), and the service discipline.

The *Kendall notation* briefly characterizes the queueing systems [Bolch et al. 2006]. One description of a queueing system description looks like $A/B/m/K - S$, where A denotes the distribution of the customer interarrival time, B the distribution of the service time, m the number of servers, K the maximum capacity of the queue in a finite case (if $K = \infty$, then this letter is

omitted), and the *S*, which is optional, denotes the service discipline used. Omitting *S* always results in the service discipline being FCFS. For *A*, the following abbreviations are common:

- *M* (Markov property): The exponential distribution with an average arrival rate of λ customers/time unit. Put another way, the number of customers follows a Poisson distribution with an average of one customer per $1/\lambda$ time unit.
- *D* (Deterministic): The interarrival times are constant and have the same value.
- *G* (General): General distribution, not specified further. At least the mean and the variance will be known in most cases.

Similarly, these same notations (*M, D,* and *G*) are used to specify *B* to describe the distribution of service time.

Having specified a queueing system, it is appropriate to identify the measures of performance and effectiveness that the analysis generates. The main variables of interest are the average waiting time for a customer, the number of customers in the queue, the length of the continuous interval during which the server is busy or idle, and the backlog of unfinished work expressed in units of time. Because these quantities are all random variables, we look for their complete probabilistic description, such as their distribution functions. In most applications, however, it is sufficient to calculate the first few moments (mean, variance, etc.). The scope of queueing theory applies when several servers are arranged in a network and customers move through the network to visit a number of different servers.

Applications in NoCs

Techniques for evaluating the performance of NoCs have been inherited from distributed and parallel processing research. A significant number of the previous analytical latency models used in off-chip networks have been formulated for a specific topology and traffic pattern [Kim and Das 1994; Kiasari et al. 2008]. One of the uses of queueing theory is in the estimation of average performance metrics, including average packet latency, average resource utilization, average energy and power consumption, and average throughput. System designers utilize metrics such as these when making decisions related to solving problems such as module placement, link capacity, routing decisions, and buffer configuration.

The analytical model proposed by Guan et al. [1993] concerns a general topology with an exponential packet length distribution and features high complexity for high dimensional networks. Hu and Kleinrock [1997] used queueing theory in their presentation of a general analytical model for wormhole routing that estimates the average packet latency in interconnection networks. Kim et al. [2005] developed a queueing theory-based model in order to provide rapid performance estimates during the design cycle; this model quantifies the performance and energy behaviour of on-chip networks. Kim et al.'s model assumed that packet arrivals at all input channels have Markov property. Hu et al.'s [2006] study of $M/M/1/K$ queueing models solved a series of nonlinear equations in order to analyse the current buffer size configuration in a timely manner and detect performance bottlenecks in the router channels. They then used this model in buffer sizing problems in packet-switched NoCs. Specifically, given the traffic characteristics of the target application and the overall budget of the available buffering space, Hu et al.'s proposed model automatically assigns the buffer depth for each input channel, in different routers across the chip, in such a way that minimizes the average packet latency in the system. Based on $M/M/1$ queueing model, Guz et al. [2007] proposed an analytical delay model for virtual channelled wormhole networks for link capacity allocation in NoC-based systems. This assignment algorithm allocates network resources efficiently in order to meet quality of service (QoS) and performance requirements.

Hur et al.'s [2008] performance analysis of hard and soft on-chip networks for FPGAs applied Jackson's [1957] queueing model to analyse the performance of a multiprocessor SoC. They also used Jackson's model to analyse circuit-switched NoCs and show that hardwired networks perform significantly better than conventional soft NoCs. Foroutan et al. [2009] presented a case

study using an analytical method based on Markov chain stochastic processes for latency evaluation of an NoC that was arranged in a 2D mesh topology with a deterministic routing algorithm and a uniform traffic pattern. Foroutan et al. [2010] proposed a generic analytical model that estimates communication latencies and link-buffer utilizations that have a given application mapped on wormhole-switched NoCs. The present study correctly models the resulting interdependencies between the routers.

Ogras et al. [2010] proposed an analytical performance model for wormhole-switched NoCs and used the $M/G/1$ queueing model to compute the average number of packets at each buffer. This model provides three performance metrics; average buffer utilization, network throughput, and average packet latency. Cheng et al. [2011] presented another analytical model with which to estimate the communication performance of wormhole-switched NoCs; their model supports arbitrary network topology with virtual channels. Cheng et al. used a routing path decomposition approach to generate a series of ordered link categories in order to resolve the inherent dependency of successive links occupied by a packet. They then used $M/M/1$ and $M/M/1/K$ queueing models to derive the transmission latency of network components. Krimer et al.'s [2011] analytical model, which was inspired by industrial work-flow modelling techniques, introduced a packet-level static timing analysis for wormhole-switched NoCs with virtual channels. This model is reliant on a reduced Markov chain that represents the network state, including the occupancy of all buffers, and also handles any topology, link capacities, and buffer sizes and provides per-flow delay analysis. Wang et al.'s [2011] proposed performance analytical model uses a semi-Markov process to estimate the average packet latency in NoCs. The authors used the process to describe the behaviour of each link in the network and calculate the header flit delay.

Varatkar and Marculescu [2004] showed that the traffic of some multimedia applications display a long-range dependent behaviour that has a considerable impact on queueing performance. Because of the inability of the traditional Poisson arrival process to capture such a traffic pattern, the performance properties of interconnection networks must be re-examined in the context of more realistic traffic models before practical implementations reveal their potential faults. With this in mind, Min and Ould-Khaoua's [2004] proposed analytical queueing model was designed for wormhole-switched networks in the presence of self-similar traffic. However, their approach is limited to the k-ary n-cube networks and uniform traffic pattern. Their study showed that such networks suffer considerable performance degradation when they are subjected to self-similar traffic; this finding emphasizes the need to improve network performance in order to ensure efficient support for traffic of this type.

A queueing system can only be analysed if something is known about the laws governing the arrival pattern, the characteristics of the service facility, and the logic governing the behaviour of the queue. Queueing theory deals with the mathematical analysis of such systems subject to demands that have occurrences and lengths that, generally speaking, can only be specified probabilistically.

2.3.2 Network Calculus

Network calculus is a mathematical framework used to derive the worst-case bounds on maximum latency and backlog, both in a single node and a network of nodes. Accordingly, network calculus can be seen as a theory with which to analyse performance guarantees in computer networks. Network calculus was pioneered by Cruz [1991a; 1991b], based on which Chang [2000] and Le Boudec and Thiran [2001] further developed the network calculus theory and based it on min-plus algebra. The basic elements of this algebra are *arrival curves* and *service curves* as abstractions of application traffic and network elements, respectively. Like conventional system theory, a network calculus consists of an input function, a transfer function, and an output function. The difference between network calculus and a conventional system theory is

that the former uses min-plus algebra and replaces addition and multiplication with minimum and addition, respectively.

In network calculus theory, the input function and output function are described by the cumulative functions $R(t)$ and $R^*(t)$, respectively. These functions represent the number of bits (words, or packets) that can be seen on the input and output data flow in time interval $[0, t]$. Functions R and R^* are, obviously, always monotonically increasing functions. System S, which receives input data and delivers the output data after a variable delay, could take the form of something like a single buffer served at a constant rate, a complex communication node, or even a complete network.

The *backlog* is the number of bits held inside the system. In cases where the system is a single buffer, the backlog determines the queue length. In a more complex system, the backlog is the number of bits "in transit," assuming that input and output can be observed simultaneously. Therefore, for a lossless system, the *backlog* at time t is

$$b(t) = R(t) - R^*(t). \tag{2.1}$$

The virtual delay at time t would be experienced by a bit that arrived at time t as long as all bits received before that bit are served first. Therefore, the virtual delay at time t is

$$d(t) = \inf_{\tau \geq 0}\{ R(t) \leq R^*(t + \tau)\}. \tag{2.2}$$

The infimum of a set (*inf*) is similar to the minimum. The difference is that the minimum of a set is the smallest element of the set and is, naturally, in the set. The infimum of a set, on the other hand, is the *greatest lower bound* of the set, and does not need to be in the set. $d(t)$ is the smallest value that satisfies $R^*(t + d(t)) = R(t)$. Network calculus theory refers to the input and transfer functions as the arrival curve and the service curve, respectively.

By way of example, providing guarantees to traffic flows requires a specific form of support in the network to limit the traffic sent by sources. This support is provided using the concept of an arrival curve. Given an increasing function $a(t)$ defined for $t \geq 0$, we can say that an input flow $R(t)$ is constrained by $a(t)$ only if the following applies for all $s \leq t$:

$$R(t) - R(s) \leq a(t - s). \tag{2.3}$$

A common arrival curve is the *leaky bucket* arrival curve (or affine arrival curve), which is defined by

$$\alpha(t) = \gamma_{\rho,\sigma} = \rho t + \sigma, \qquad t \geq 0. \tag{2.4}$$

where ρ is the rate of the flow (measured in units of data per time unit) and σ limits the burstiness of the flow (in units of data). Such an arrival curve enables a source to send σ bits at once, but no more than ρ bits/second over the long run. Cruz [1991a] was the first to propose the (σ, ρ) traffic characterization; Figure 2.5 shows the corresponding arrival curve.

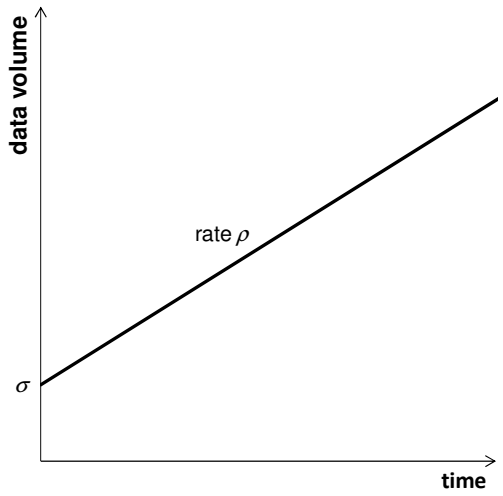


Figure 2.5: Leaky bucket (affine) arrival curve.

The service curve represents the minimum service levels of network elements (router, channel, etc.) and often abstracts a scheduling policy. Consider the example of a system S and a flow-through S with input and output functions R and R^*. S can be said to offer a service curve β to the flow only if some $s \leq t$ exists for all t such that

$$R^*(t) \geq R(s) + \beta(t-s). \tag{2.5}$$

A well-defined service curve is the latency-rate function $\beta_{R,T}$,

$$\beta_{R,T}(t) = R(t-T)^+ = \begin{cases} R(t-T), & t > T, \\ 0, & \text{otherwise.} \end{cases} \tag{2.6}$$

where R is the service rate and T is the maximum response delay of the node [Stiliadis and Varma 1998]. Figure 2.6 below illustrates such a service curve, which is widely used to model the routers in a network.

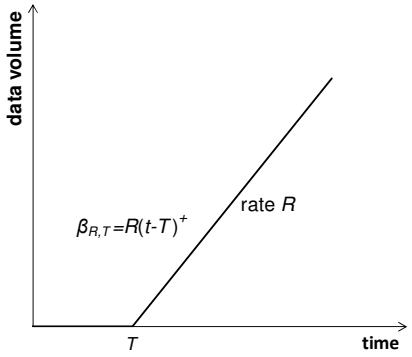


Figure 2.6: A latency-rate service curve.

One use of network calculus is in determining the backlog and delay of a flow that is constrained by an arrival curve and traverses a system offering a known service curve. For instance, the maximum delay, maximum backlog, and output traffic characterization in a system with leaky bucket arrival curve and latency-rate service curve are as follows [Le Boudec and Thiran 2001]:

$$b_{max} = \sigma + \rho T, \tag{2.7}$$

$$d_{max} = T + \sigma/R, \tag{2.8}$$

$$\alpha^*(t) = \gamma_{\rho,\sigma+\rho T} = \rho t + \sigma + \rho T. \tag{2.9}$$

Applications in NoCs

Zhang [1995] surveyed several of the service disciplines that have been proposed in the literature in an effort to provide per-connection end-to-end performance guarantees in packet-switching networks. Zhang discussed a number of issues and trade-offs regarding the design of service disciplines for guaranteed performance service and presented a general framework for studying and comparing these disciplines. Zhang's survey is an excellent overview of guaranteed service in networks that are potentially applicable to NoCs.

Network calculus can help estimate worst-case flow delays and backlogs in a given system. Qian et al. [2009c] performed an investigation of per-flow flit and packet worst-case delay bounds in on-chip wormhole networks. They proposed certain analysis models for flow control and for link and buffer sharing; then, based on these analysis models, they obtained an open-ended service analysis model that captured the combined effect of flow control and link and buffer sharing. Their service analysis model computed leftover service curves for individual flows, before deriving their flit and packet delay bounds.

Lu et al.'s [2009] study defined a regulation spectrum for lossless flow regulation, which was then used to reduce delays and backlog bounds in SoC architectures. Based on the regulation spectrum, Jafari et al. [2010] formulated optimisation problems with which to minimize total buffers and buffer variations under QoS constraints, and also performed a regulation analysis for best-effort networks. Bakhouya et al. [2011] presented a methodology with which to analyse and evaluate on-chip interconnects in terms of a number of performance and cost metrics, including energy consumption, latency, and area requirements. They also used a given traffic pattern to compare spidergon, 2D mesh, and WK-recursive topologies, and found that the WK-recursive topologies outperformed the others in all considered metrics. Lu [2011] used network calculus to analyse and determine the delay and buffer bounds for TDM virtual circuits that crossed synchronous clock domains.

Qian et al.'s [2009a] study applied network calculus to NoCs with the aim of analysing delay and backlog bounds for self-similar traffic. They showed that a deterministic arrival curve cannot constrain self-similar traffic, and then went on to prove that leaky bucket arrival curves can constrain self-similar traffic if an additional parameter – excess probability – is used to capture the burstiness of the traffic exceeding the arrival envelope. In Qian et al. [2010a], the worst-case delay bound was derived for an individual flow on packet-switched best-effort NoCs. The authors derived the leftover service curve for the flows by first constructing a *contention tree* [Lu et al. 2005] that captures the contention of each flow with other interfering flows along its routing path, and then proceeded to scan the tree. Tagged flows contend directly with interfering flows, and interfering flows may contend both with each other and then with the tagged flow again. In turn, this indirect contention may influence the performance of the tagged flow. In order to decompose a complex contention scenario, they identified three primitive contention patterns; they analysed these three scenarios and then derived their basic analytical models, with a focus on the derivation of the service curve provided by the tandem. The authors of the present study have assumed sufficiently large buffers in routers. [Qian et al. 2009b] and [Qian et al. 2010b] considered bounded buffers and virtual channels, respectively.

Network calculus has emerged as a new theory with which to analyse performance bounds in network-based systems. Unlike queueing theory, network calculus deals with worst-case analysis rather than average-case analysis and has therefore becoming a promising formalism for analysing quality of service. Network calculus makes it possible to derive the worst-case bounds on backlog, maximum latency, and minimum throughput.

2.3.3 Schedulability Analysis

Schedulability analysis is a mathematical formalism that investigates the timing properties of real-time systems. This technique was originally developed with the purpose of analysing computation systems [Liu and Layland 1973; Leung and Whitehead 1982; Lehoczky et al. 1989] and was then applied to such communication platforms as multicomputers [Li and Mutka 1994] and NoCs [Shi and Burns 2008]. Schedulability analysis usually involves modelling tasks with sporadic and periodic models. Periodic tasks are released at regular intervals, while sporadic tasks are released arbitrarily, albeit with specified minimum time intervals between releases. Schedulability analysis uses a scheduling policy, a given set of periodic and sporadic tasks, and their worst-case execution time to determine the possibility of scheduling these tasks in such a way that deadlines are never missed. The earliest results from schedulability analysis and real-time scheduling were obtained based on restrictive assumptions regarding the task in question and the underlying architecture. The task is comprised of a certain fixed number of independent tasks that are mapped on a single processor; these tasks each have a fixed period and are released periodically, while the deadlines equal the periods and the task execution times are fixed. The assumptions in later studies were more relaxed, including multiprocessor systems, deadlines that were less than or equal to the periods, data dependency relationships among the tasks, and sporadic tasks.

Real-time systems are usually equipped with a *schedulability test* [Wu et al. 2010], the purpose of which is to determine whether each admitted task can meet its deadline. New tasks are not admitted unless they pass the schedulability test. Schedulability tests can either be *direct* or *indirect*. A direct schedulability test calculates the worst-case response time of the tasks, and a set task can only be schedulable if the worst-case response time of each separate task is less than or equal to its deadline. Although such tests are accurate, calculating the response times involves a high computing cost. The most common type of indirect schedulability test is a *utilization-based* test, which tests system resource utilization in order to determine task schedulability. A new task can only be admitted if its utilization is lower than a pre-derived bound. A task set for a utilization-based schedulability test is schedulable when the task's utilization is lower than a pre-derived bound.

Application in NoCs

It is essential that the SoC communication platform provides different levels of service for the different application components on the same network. There are stringent requirements for real-time communication; the correctness relies on the communication result and also on the completion time bound. If a destination receives a data packet too late, that packet may be useless. The worst-case acceptable time metric is considered to be the deadline of the packet. If all the packets belonging to a set of real-time traffic flows over the network meet their deadlines regardless of the arrival order of the packet set, then these flows are considered *schedulable*. In systems like this, schedulability analysis investigates the schedulability of flows in the network and adopts an iterative approach that estimates, in the context of a network-based system, the maximum end-to-end latency of flows.

Shi and Burns [2008] suggested an offline schedulability analysis approach that discusses real-time on-chip communication with fixed-priority scheduling and wormhole switching. They proved the NP-hard nature of determining the precise schedulability of a real-time traffic flow over an on-chip network. However, by evaluating a diverse range of relationships among the traffic flows, they also provided a determinant upper bound on the schedulability of real-time traffic flows. They proposed a method that predicts a packet network's latency based on direct contention and indirect contention from traffic flows with higher priority. Wormhole switching with fixed-priority pre-emption is one possible solution for real-time on-chip communication; however, the hardware implementation cost of such a solution is high. The solution proposed by

Shi and Burns [2009] involves utilizing a priority share policy that reduces the resource overhead while, at the same time, achieving the hard real-time service guarantees. However, the analysis process is complicated by the blocking that the priority share policy introduces. To counter this, Shi and Burns [2010] suggested an analysis scheme with a per-priority basis that would compute the total time window at each priority level rather than at each traffic flow. Shi and Burns efficiently determined schedulability by checking each flow's release instance at the corresponding priority window. Furthermore, by building on this static analysis, for a given set of tasks and network topology, they also proposed a task-mapping and priority assignment algorithm that met the hard time bounds with reduced hardware overhead.

Schedulability analysis focuses on real-time systems and determines whether or not a real-time system can meet its deadline; it also aims to assign priorities to tasks in order for each task to meet its deadline.

2.3.4 Dataflow Analysis

A dataflow graph is a model-of-computation (MoC) in which a number of concurrent processes use unbounded FIFO channels to communicate with each other [Lee and Parks 1995]. Writing to these channels is a non-blocking method, but reading from these channels is a blocking method [Jantsch and Sander 2005]. A dataflow program is a directed graph that consists of nodes (actors) that represent communication, and arcs representing ordered sequences (streams) of data units (tokens). Studies have shown that, for concurrent implementation on parallel hardware, dataflow graphs are valuable in digital signal processing applications (such as audio and video applications) [Jantsch and Sander 2005]. There are a range of dataflow MoCs, depending on the specifications regarding the firing rules, consumption, and production. These MoCs differ in terms of their *analysability*, their *expressiveness and succinctness,* and their *implementation efficiency* [Stuijk et al. 2011]. Among the dataflow models for streaming applications, the synchronous dataflow (SDF) model is the most popular and widely studied nowadays [Bekooij et al. 2005]. SDF places further restrictions on the general dataflow model, in that each process produces and consumes a fixed number of tokens for each individual firing [Lee and Messerschmitt 1987]. In the cyclo-static dataflow (CSDF) model [Lauwereins et al. 1994; Bilsen et al. 1996], the number of tokens that an actor consumes and produces varies cyclically. Each cycle has a fixed number of phases and each actor produces or consumes a fixed number of token in each phase, although behaviour may vary from one phase to another. Non-synchronous and data-dependent behaviour means that SDF and CSDF are not able to express some streaming applications [Buck 1994], although it is possible to address this problem by extending the SDF model to permit some actors with data-dependent behaviour. The SDF model can be further generalized using Boolean dataflow (BDF) [Buck 1993], in which the number of tokens that is produced and consumed will depend on the value of a token read from a dedicated control input. The dynamic dataflow (DDF) model [Lee and Parks 1995] is a Boolean dataflow model that has an additional variation; namely, the control actors mentioned in the BDF model can read multiple token values and, based on what the control actors read, the data actors can be fired conditionally. The incomplete knowledge at compile time means that BDF and DDF MoCs require a run-time scheduling mechanism in order to determine the point at which an actor becomes executable.

Applications in NoCs

Classical dataflow models are not timed. The timing properties of a system can be addressed by associating a worst-case execution time with each actor [Sriram and Bhattacharyya 2009]. This association makes it possible to assess aspects of the NoC-based system's timing behaviour, such as its throughput and latency. Each actor has a worst-case execution time added to it, and the specified number of tokens is produced and consumed within that execution time. An actor's self-edge is used to model the fact that the previous execution must be completed before starting

the next execution. It is possible to indirectly model scheduling policies by changing the worst-case execution time into the worst-case response time [Bekooij et al. 2005].

In streaming applications, throughput is an important performance indicator that has been well studied in the literature on dataflow models [Dasdan and Gupta 1998; Dasdan 2004; Ghamarian et al. 2006]. All of those studies have focused on analysing HSDFs and can only be applied to SDFs by converting them to HSDF [Lee and Messerschmitt 1987; Sriram and Bhattacharyya 2009]. Next, the throughput is determined using maximum cycle mean (MCM) analysis. In order to determine the MCM, it is necessary to determine the maximum of the cycle means of all the simple cycles in the HSDF graph, where a cycle's cycle mean (CM) c is the sum of the actors' response times on c, divided by the number of initial tokens on cycle c. The maximum throughput of the graph that can be attained relates to 1/MCM. Another prominent performance metric is latency, although the research into this metric has been minimal. Sriram and Bhattacharyya [2009] studied the latency for the HSDFs. Although the latency of an SDF can be computed by converting it to a HSDF, this conversion can lead to an exponential increase in the number of nodes in the graph, which makes the prediction of performance metrics prohibitively expensive [Stuijk et al. 2006]. Moreira and Bekooij [2007] offered a closed-form expression of the latency of HSDF graphs, providing useful bounds on the maximum latency for jobs with periodic, bursty, and sporadic sources, in addition to a technique that checks latency requirements. Ghamarian et al.'s [2007] latency minimization technique, which works directly on SDFs, computes the minimal achievable latency for an SDF and provides an execution scheme that provides the minimal latency.

Within the context of a real-time embedded multiprocessor system, Bekooij et al. [2005] utilized SDF models in order to derive the end-to-end temporal behaviour of jobs. Hansson et al. [2009] and Hansson and Goossens [2010] explained the method for constructing a CSDF model that conservatively models an NoC connection. They then used the proposed dataflow model to dimension the buffer size in network interfaces in order to guarantee the system's performance, and they also showed that buffer sizes are determined using a run time that is comparable to that of analytical methods and results that are comparable to those of exhaustive simulation. Wiggers et al.'s [2007] proposed algorithm determines close-to-minimal buffer capacities for CSDF graphs in a way that satisfies the throughput requirement and constraints on maximum buffer capacities. They also demonstrated that a CSDF model can reduce resource requirements in comparison to an SDF model.

2.3.5 Comparison of Formalisms

These formalisms are compared based on the event model, the node model, and the analysis output of the formalisms.

- The event model refers to the data packet representation. The event model in queueing theory is the probability distribution of the packets' interarrival time. In network calculus, the events are modelled by an upper bound for the number of packets. Schedulability analysis models the events using periodic and sporadic models in which the minimum interarrival times represent a flow. As with queueing theory and network calculus, schedulability analysis does not involve any dependency between events. Dataflow analysis uses tokens that are produced and consumed by nodes to model events. In the output ports, the production of new tokens (events) is dependent on the availability of tokens in the input ports, which means that the only formalism that captures the dependency between the events is dataflow analysis.
- The modelling of nodes is based on the nodes' service time. Service time in queueing theory is specified probabilistically. Network calculus models nodes using a function that characterizes the minimum number of bits a node must transmit in a given time interval. Scheduling analysis models node based on their worst-case delay and on the scheduling policy. The worst-case node delay and scheduling policy represent a node in dataflow analysis.

- The analysis results differ among each of the four formalisms. The fact that queueing theory deals with probability models enables it to compute average-case performance metrics such as average throughput, average packet latency, average resource utilization, and average energy and power consumption. Network calculus computes worst-case packet latency and maximum backlog, while schedulability analysis estimates worst-case latency in order to determine whether the flow is schedulable. Finally, dataflow analysis determines worst-case latency and throughput.

This chapter presented some background on networks-on-chip and performance evaluation methods. Then, the basic concepts and applications of four analytical performance evaluation methods that are popular for average-case and worst-case performance analysis of NoCs were reviewed.

Chapter 3

Problem Description and Solution Overview

This chapter is started by describing a generic synthesis flow for NoCs, from the application specification through to tape-out. The chapter then formulates the problem and reviews the contributions in this thesis

3.1 NoC Synthesis Flow

The flow starts with application modelling, followed by NoC performance analysis and, finally, NoC verification. The present thesis focuses on the second of these steps.

3.1.1 Application Modelling

NoCs target single-chip multicore systems that implement multiple applications. Such systems are complex, and this fact, along with strict requirements on power, performance, cost, and time-to-market, places a high degree of pressure on the design team. Such situations are usually handled by developing the application and platform models separately [Lieverse et al. 2001]. The application models include all of the computation and communication tasks. Workload characterization is usually performed in order to obtain traffic models that can be used for analysis and optimisation. These application models must be sufficiently scalable and flexible for them to be analysed quickly. Also, in order to have confidence in the predicted results, it is essential that the key behaviour of the application in the model be captured. The workload has a significant influence on the performance of NoC-based systems. Performance evaluation follows the GIGO principle of "Garbage in Garbage out". Evaluation of a system using incorrect workloads is likely to lead to incorrect results that cannot be relied upon. An application that has software components may require a code partitioning step [Ryoo et al. 2007]. This step is a crucial part of extracting as much parallelism from the application as possible. Only after this step is it possible to truly exploit the concurrency provided by the NoC architectures. Lastly, in situations where the NoC platform is either general-purpose or likely to accommodate a large set of applications, it is possible to use random traffic models such as uniform traffic as the application model. Flexible tools that can be used to exercise characteristics of the target NoC platform include traffic models or realistic traces.

Many performance analyses of interconnection networks have used the Poisson model, and many papers in numerous application domains are based on this stochastic assumption [Hu et al. 2006]. However, analysis of the multimedia applications in NoCs reveals bursty patterns of traffic over a range of time scales [Varatkar and Marculescu 2004]. Because the Poisson process is not able to accurately model the bursty traffic, the Markov-modulated Poisson process (MMPP) model [Fischer and Meier-Hellstern 1993] has been used in the present study to model the temporal burstiness of traffic in NoCs. Several studies have employed MMPP in order to model traffic burstiness in a temporal domain [Fischer and Meier-Hellstern 1993]. Figure 3.1 shows a two-state MMPP in which arrival traffic follows a Poisson process that has rates λ_0 and λ_1. The transition rate from state 0 to 1 is r_0, and the rate from state 1 to state 0 is r_1.

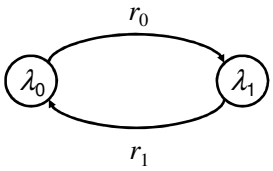


Figure 3.1: Two-state MMPP model

The present study uses the notation MMPP(k) for the two-state MMPP in which $\lambda_1 = k\lambda_0$. In Figure 3.2, the number of packet arrivals in a node against time is shown for varying values of traffic burstiness. Figure 3.2.a clearly shows that the Poisson process (k =1) is not able to model the traffic burstiness, while Figures 3.2.b through 3.2.f indicate that greater k will result in a greater intensity of packet burstiness.

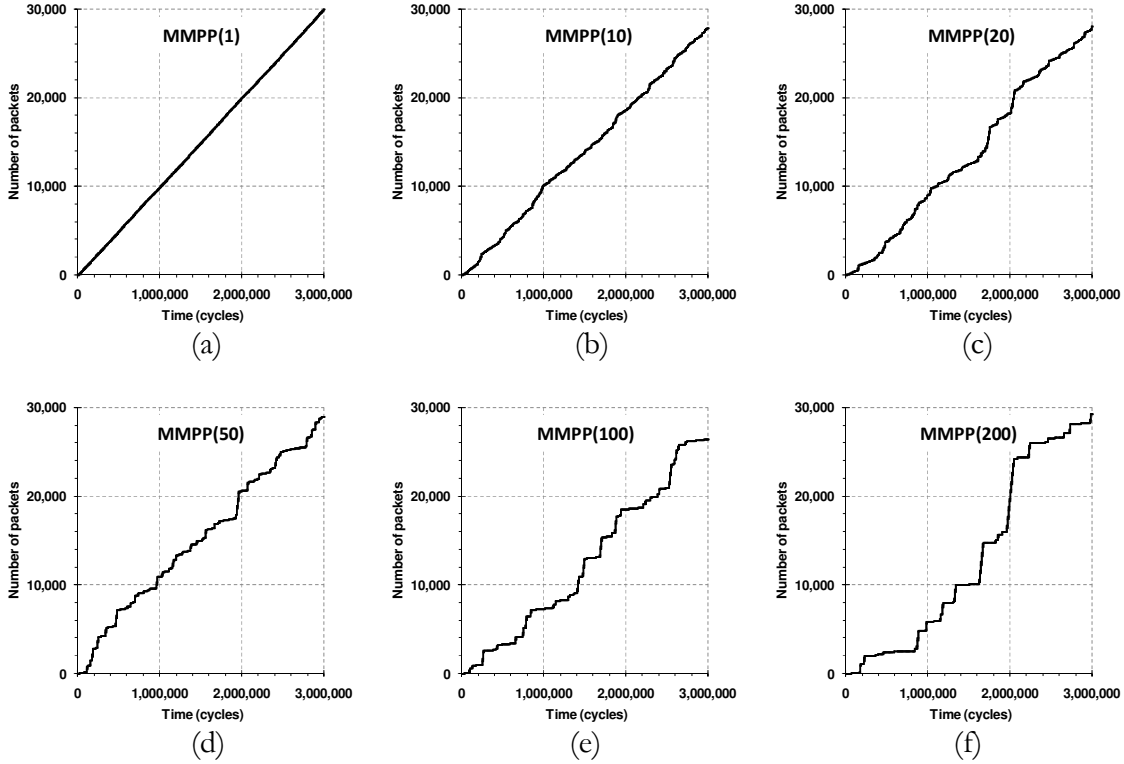


Figure 3.2: Number of packets against time in the MMPP model for (a) k =1 (Poisson model), (b) k =10, (c) k =20, (d) k =50, (e) k =100, (f) k =200.

3.1.2 NoC Performance Analysis

This phase aims to determine the NoC architecture and the mapping of the target application onto this architecture that meets the goals and constraints of the design. The application is first mapped onto the target architecture, and performance analysis is then conducted to determine whether the chosen application–architecture combination satisfies the imposed design constraints. The success of this methodology is heavily dependent on the availability of adequate performance analysis tools that can guide the overall design process. So far, performance evaluation of NoC designs has been based on simulation, as Figure 3.3 shows. However, simulation is a very time-consuming process and provides minimal insights into how various design parameters affect actual network performance. Accordingly, it is virtually impossible to use simulation for optimisation purposes.

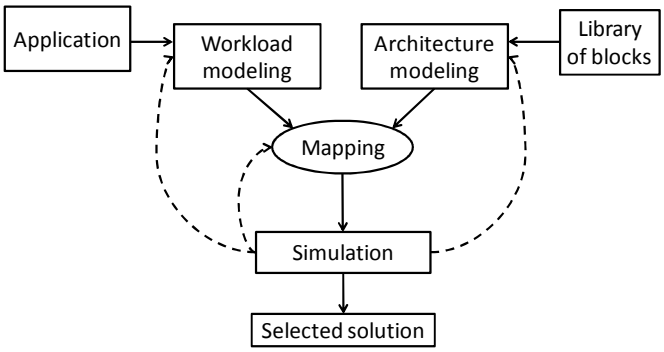


Figure 3.3: NoC design flow.

The main goal of the present thesis is to develop an approach to NoC performance analysis that not only provides accurate performance figures, but also reveals the relationship among application mapping, network topology, buffer configuration, and routing algorithms. The resulting model is used to achieve accurate and fast performance estimates and also to guide the NoC design process in an optimisation loop.

Figure 3.3 shows a simulation based design flow while Figure 3.4 presents an alternative approach. In order for the analysis to be used in an optimisation loop, it must be tractable and must also provide meaningful feedback to the designer. Time-consuming simulations can only be conducted later on, typically after the design space has already been reduced to a few practical choices. This makes it possible to conduct rapid design space exploration and to identify high-quality solutions within a short and predictable period of time.

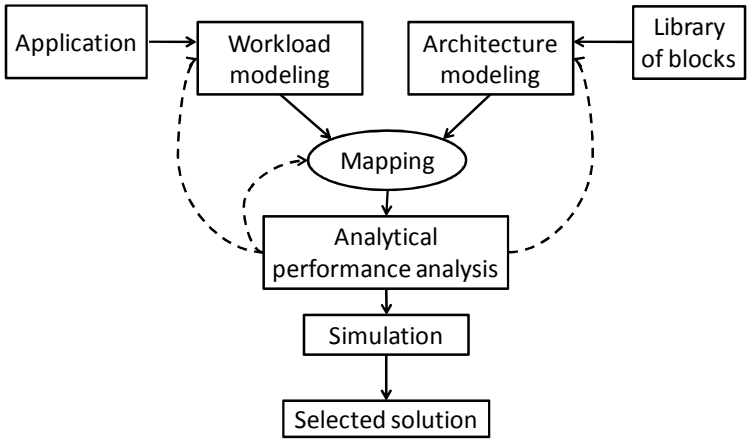


Figure 3.4: An alternative design flow approach for NoCs.

3.1.3 NoC Verification

Once a particular architecture–mapping pair has been selected, the subsequent and final phase involves implementing the NoC communication architecture by instantiating the components from a communication library and carrying out the synthesis process. This phase includes simulating and verifying the final design to meet user-defined design constraints and goals. Synthesis, layout generation, and floor-planning steps are performed at the end, just before tape-out.

3.2 Detailed Contribution

This section summarizes the contributions in this thesis, which are divided into the following three main categories:

- Design optimisation with simulation performance models (Papers 1, 5, and 8)
- Analytical performance models (Papers 2, 4, and 9)
- Design optimisation with analytical performance models (Papers 3, 6, and 11)

The author of the thesis is the second contributor to the simulation-based design optimisation and the main contributor to the analytical performance models and the main contributor to the design optimisation with analytical performance models. Here is a short list of publications.

1. A Performance and Power Analysis of WK-Recursive and Mesh Networks for Network-on-Chips, ICCD 2006.
2. A Markovian Performance Model for Networks-on-Chip, *PDP* 2008.
3. PERMAP: A Performance-Aware Mapping for Application-Specific SoCs, *ASAP* 2008.
4. Caspian: A Tunable Performance Model for Multi-Core Systems, *Euro-Par* 2008.
5. Power-Efficient Routing Algorithm for Torus NoCs, *IC3* 2008.
6. A Framework for Designing Congestion-Aware Deterministic Routing, *NoCArc* 2010.
7. Analytical Approaches for Performance Evaluation of Networks-on-Chip, *CASES* 2012.
8. Power-efficient Deterministic and Adaptive Routing in Torus Networks-on-Chip, *Microprocessors and Microsystems*, 2012.
9. An Analytical Latency Model for Networks-on-Chip, *IEEE Transactions on VLSI*, 2013.
10. Mathematical Formalisms for Performance Evaluation of Networks-on-Chip, *ACM Computing Surveys*, 2013.
11. A Heuristic Framework for Designing and Exploring Deterministic Routing Algorithm for NoCs, *Springer*, 2013.

The full list of papers has been given on pages xx-xxi and the papers are included in Part II of the thesis.

3.2.1 Design Optimisation with Simulation Performance Models (Papers 1, 5, and 8)

The network characteristics are affected by various NoC topologies, including the average inter-node distance, communication flow distributions, and the total wire length. Researchers have proposed a number of new parallel computer architectures in recent years for building massively parallel computer systems that aim to increase computation speed. One significant drawback of such networks is the absence of any predefined modules for when fabricating the models onto a monolithic chip. This limitation is due to the fact that such networks are not truly expansible, while the irregularity of node degrees also makes them costly in terms of VLSI implementation.

Vecchia and Sanges [1988] proposed recursively scalable network topologies for VLSI implementation, referred to as WK-recursive networks. They can be constructed recursively by grouping basic modules. Any d-node complete graph can serve as the basic module. WK(d,t) is used to denote a WK-recursive network of level t whose basic modules are some d-node complete graph, where $d > 1$ and $t \geq 1$. Each node of WK(d,t) is uniquely identified by a sequence of t numbers, and each of its edges is represented by a pair of nodes. The node set of WK(d,t) is denoted by $\{a_t a_{t-1} \ldots a_2 a_1 \mid a_i \in [0,d\text{-}1] \text{ for } 1 \leq i \leq t\}$. The adjacency is defined as follows: $a_t a_{t-1} \ldots a_2 a_1$ is adjacent to (1) $a_t a_{t-1} \ldots a_2 b$ where $0 \leq b \leq d - 1$ and $b \neq a_1$, and (2) $a_t a_{t-1} \cdots a_{i+1} a_{i-1} (a_i)^{i-1}$ if $a_i \neq a_{i-1}$ and $a_{i-1} = a_{i-2} = \ldots = a_2 = a_1$, where $(a_i)^{i-1}$ represents i -1 consecutive a_i's. In WK(d,t), each node is of degree d, and there are totally d^t nodes and $d + d(d^t - 1) / 2$ edges. Figure 3.5 shows the topology of WK(4,2) network.

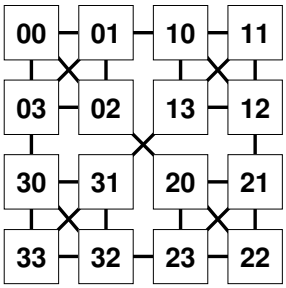


Figure 3.5: The topology of WK(4,2) with 16 nodes.

The WK-recursive networks offer a high degree of scalability, regularity, and symmetry, which helps them conform well to the implementation and modular design of distributed systems that involve many computing elements. Vecchia and Sanges [1988] described the VLSI implementation of the WK-recursive networks and developed a routing algorithm that defined which physical channels need to be traversed, but did not address the use of virtual channels. Usually, virtual channels are used to design deadlock-free routing algorithms and increase their performance. The author of the thesis proposed a deadlock-free routing algorithm to be applied to WK-recursive networks. The next step was to assess and compare performance factors of same-sized mesh with WK-recursive networks under similar working conditions. The results of the simulation showed that, for medium and low traffic loads, the WK-recursive network performed better the mesh topology. Paper 1 in Part II provides more detailed information regarding the results.

The routing algorithm, like the network topology, has an important role to play in the design of high-performance NoCs. XY routing is the simplest most commonly used routing algorithm for mesh NoCs [Dally and Towles 2004]. However, deadlock may occur when XY routing is applied to the torus topology because of each dimension's channel dependency between various packets that occurs as a consequence of added wrap-around links. Using multiple virtual channels provides flexibility in terms of designing new deadlock-free routing algorithms, although this comes at the cost of more complex hardware and greater area, which leads to increase power consumption [Banerjee et al. 2004]. A deadlock-free routing algorithm in the torus requires a minimum of two virtual channels for each physical channel in order to break the cycles within the channel dependency graph into spirals [Dally and Towles 2004]. Papers 5 and 8 present a new systematic approach to the design of routing algorithms for torus NoCs, based on which a deadlock-free deterministic routing algorithm is proposed. The new algorithm, which is named *TRANC* (Torus Routing Algorithm for NoCs), uses only one virtual channel for each physical channel. TRANC offers the low level of power consumption of a mesh NoC, while also providing performance comparable to that of a torus NoC by using XY routing that has two virtual channels for each physical channel. TRANC even produces better performance for light traffic thanks to a zero switching time between virtual channels compared to a torus NoC which uses two virtual channels and XY routing. As a secondary contributor of these papers, the author of the thesis proposed the minimality factor and an optimality factor to find routes that are deadlock-free and as optimal and minimal as possible. These parameters show the extent to which an algorithm can balance network traffic. In fact, a routing algorithm requires a measure that shows whether all the links are being utilized properly in terms of the distribution of the packet destination address.

3.2.2 Analytical Performance Models (Papers 2, 4, and 9)

As noted in Section 2.2, a suitable analytical model is able to predict the performance of an on-chip network many times faster than a simulation. Therefore, it is logical to pursue models that accurately analyse the performance of popular NoCs. As the main contributor, the author of the thesis proposed three analytical models that estimate the average packet latency in NoCs.

Markovian model: In paper 2, the author of the thesis proposed a Markovian performance model which only supports wormhole-switched torus network with uniform traffic and Poisson arrival process. Using a queuing-based approach, the average delay due to path contention, virtual channel and crossbar switch arbitration is computed. The performance results from the analytical models are then validated against those obtained from the simulator. Comparison with simulation results indicates that the proposed analytical model is quite accurate and can be used as an efficient design tool by SoC designers.

Caspian model: The author of the present thesis proposed the Caspian model in paper 4 which is the first analytical model to predict the average end-to-end packet latency in NoCs with bursty traffic. The model aimed to minimize prediction costs while providing prediction accuracy. This goal was accomplished using a G/G/1 priority queueing model that is used for wormhole-switched networks under arbitrary traffic pattern. Although Caspian has a good degree of accuracy it is limited to networks with dimension-order routing algorithm and single flit buffers.

PQ model: To overcome the limitations of Caspian model, the thesis author proposed PQ model (Performance Queueing model) in paper 9 as an extension to Caspian. Input for PQ model includes an application communication graph, a topology graph, buffer configuration, a mapping vector, and a routing matrix. PQ model then estimates the average packet latency and router blocking time. It works for arbitrary network topology under arbitrary traffic patterns. However, PQ model is limited to with deterministic routing algorithms. The author of the thesis also analysed the computational complexity of the algorithm and showed that the PQ model has time requirement $O(n^{5/2})$ for 2D mesh networks.

3.2.3 Design Optimisation with Analytical Performance Models (Papers 3, 6, and 11)

In the next step, the analytical models in Section 3.2.2 are used for design space exploration of NoCs. The author of the present thesis is the main contributor of papers 3, 6, and 11.

PERMAP: In paper 3, the author of the thesis used the Caspian model in an application mapping algorithm and presented PERMAP, a PERformance-aware MAPping algorithm that maps a task graph onto a generic NoC architecture in a way that minimizes the average communication delay. PERMAP is then used to map a video application onto a tile-based NoC, and experimental results show that the proposed mapping algorithm is fast and robust. The main limitation of the PERMAP is that the search space is explored randomly.

CAR: In paper 6, the thesis author presented a system-level Congestion-Aware Routing (CAR) framework for designing minimal deterministic routing algorithms. CAR exploits the peculiarities of the application workload in order to spread the load evenly across the network. To this end, an optimisation problem of minimizing the level of congestion in the network is formulated and then the problem is solved by using the simulated annealing heuristic. The proposed framework ensures deadlock-free routing, even in networks without virtual channels. However, CAR calculates the level of congestion based on average traffic rate and does not consider the traffic burstiness.

LAR: The thesis author presented Latency-Aware Routing (LAR) to support the traffic burstiness in NoCs in paper 11. To this end, the PQ analytical model is used to estimate the average packet latency. Experiments with both synthetic and realistic workloads show the effectiveness of the approach. The results show that maximum sustainable throughput of the network is improved for different applications and architectures. The main limitation of LAR is that it is a static approach and the traffic pattern must be known in the design time.

In paper 10, the author of the thesis reviewed four popular mathematical formalisms – queueing theory, network calculus, schedulability analysis, and dataflow analysis – and how they have been applied to NoC performance analysis. The paper also discusses the respective strengths and weaknesses of each formalism, their suitability for a specific purpose, and the attempts that have been made to bridge these analytical approaches. Furthermore, the author of the thesis and his advisor organized a tutorial on the same topic that is summarised in paper 7. Table 3.2 shows the thesis author contributions proposed in each paper along with the topic, role, problem and main limitations.

In addition to above mentioned papers, papers 7 and 10 are also included in the present thesis. There is no novel research contribution in these papers and they review popular mathematical formalisms and their application to the performance analysis of NoCs.

Table 3.2: The thesis author's contributions

Topic	My role	Paper	Problem	My contribution	Main limitation(s)
Simulation-based performance analysis	Secondary contributor	Paper 1	Study a new topology for NoCs	Propose a deadlock-free routing for WK-recursive topology	For large networks, the proposed routing algorithm requires a large amount of buffer.
		Paper 5	Low power routing algorithm in NoCs	Propose the minimality factor and the optimality factor to have a balanced load in the network	The approach is limited to torus networks and deterministic routing.
		Paper 8	Low power routing algorithm in NoCs	Propose the minimality factor and optimality factor to find a balanced load in the network	The approach is limited to torus networks.
Analytical performance analysis	main contributor	Paper 2	Evaluate the average latency and energy consumption in NoCs	Propose a Markovian model for latency estimation	It only supports torus network with uniform traffic and Poisson arrival process.
		Paper 4	Evaluate the average end-to-end packet latency in NoCs	Propose an analytical model to estimate the latency in case of bursty traffic	It is limited to networks with dimension-order routing algorithm and single flit buffers.
		Paper 9	Evaluate the average end-to-end packet latency in NoCs	Propose an analytical model to estimate the latency in case of arbitrary topology and bursty traffic	It only supports deterministic routing.
Design space exploration		Paper 3	Application mapping to NoC architecture	Use a queueing theory-based model to map the application on NoC	The search is done randomly and the model only considers the single flit buffers.
		Paper 6	Congestion-aware routing	Propose a routing algorithm that minimises the network congestion	The solution does not consider the traffic burstiness.
		Paper 11	Latency-aware routing	Use an analytical model to find the optimised routes for traffic flows	The traffic pattern must be known in the design time.
Survey		Paper 7 and Paper 10	There is no novel research contribution in these papers and they review popular mathematical formalisms and their application to the performance analysis of NoCs.		

Chapter 4

Summary and Outlook

This chapter concludes the thesis by giving a summary of the previous chapters and an overview about future directions for research.

4.1 Summary

In the near future, the scaling down of semiconductor technology will enable implementations that include thousands of communicating IP blocks on a single chip. Successful integration of these blocks relies on the design of communication architectures that are truly scalable. To date, the most promising solution has been given by structured communication with an NoC. The present thesis studied the modelling, analysis, and optimisation of on-chip interconnection networks and presents novel design methodologies for NoC design. A summary of our work is provided below.

- It is vital to have formal models for network architectures and on-chip routers in order to identify and solve key research problems. The author of the present thesis reviewed four popular mathematical formalisms – queueing theory, network calculus, schedulability analysis, and dataflow analysis – and how they have been applied to the analysis of on-chip interconnection networks.
- The thesis author developed novel analytical models that analyse NoC communication performance. In addition to providing aggregate performance metrics such as latency and throughput, our approach also provides feedback about the network characteristics at a fine-level of granularity. Our approach explicates the impact that various design parameters have on the performance, thereby providing invaluable insight into NoC design. This makes it possible to use the proposed approach as a powerful design and optimisation tool.
- Using the proposed analytical models, system-level frameworks are presented that address application mapping and routing algorithms for NoCs. To this end, the author of the thesis first formulated an optimisation problem of minimizing average packet latency in the network, and then solved the problem. The proposed framework can also address other design space exploration problems such as topology selection and buffer dimensioning.

4.2 Outlook

As part of future direction, a number of worthy research challenges were identified. In what follows, we summarise these directions.

Bridging the formalisms: Each of the reviewed formalisms has its own advantages and difficulties also differs somewhat in purpose; therefore, none of them can easily replace all of the others. Although each formalism has issues that require further study, the most urgent need is for research into integrated approaches to the problems of system performance analysis. Each formalism can be extended in certain directions; however, these extensions usually face problems of complex mathematics or are considered cumbersome and unnatural. Accordingly, the author of the thesis argues that the best solution would be comprehensive frameworks that combine two or more formalisms. For instance, queueing theory and network calculus could be combined to offer both worst-case and average-case analysis. The result of this combination could be merged with dataflow analysis in order to naturally model event dependencies and build a bridge to

simulation. However, there is a need to explore and understand the relations between these models, as well as the possible and useful transformations between them.

Energy analysis: One of the major concerns regarding the design of multi-core chips is power consumption. Because of this, it is important to analyse the power–performance trade-off of an NoC design. Therefore, the proposed performance analysis tool can be extended for use with available power models [Kahng et al. 2012] and tools for NoC components [Soteriou et al. 2007] in order to achieve fast and accurate power/performance analysis.

Reliability analysis: Fault-tolerance and reliability of NoCs is becoming a critical issue due to several artifacts of deep sub-micron technologies. Therefore, it is important for a designer to have access to fast methods for evaluating the performance and reliability of an on-chip network.

Dynamic workload: The work in the present thesis addresses the performance analysis of NoCs for a given application. One direction for future work would be focus on exploring more dynamic workload variations and additional benchmarks.

Bibliography

ALEXOPOULOS, C., SEILA, A. F. 1996. Implementing the Batch Means Method in Simulation Experiments. *Winter Simulation Conference*. 214-221.

BAKHOUYA, M., SUBOH, S., GABER, J., EL-GHAZAWI, T. A., AND NIAR, S. 2011. Performance evaluation and design tradeoffs of on-chip interconnect architectures. *Simulation Modeling Practice and Theory*. 19, 6, 1496-1505.

BANERJEE, N., VELLANKI, P., AND CHATHA, K. S. 2004. A Power and Performance Model for Network-on-Chip Architectures. In *Proceedings of the Design, Automation and Test in Europe, (DATE-04)*, 1250-1255.

BEKOOIJ, M. J. G., HOES, R., MOREIRA, O., POPLAVKO, P., PASTRNAK, M., MESMAN, B., MOL, J. D., STUIJK, S., GHEORGHITA V., AND VAN MEERBERGEN J. 2005. Dataflow analysis for real-time embedded multiprocessor system design, chapter 15, Dynamic and robust streaming between connected consumer-elecronic devices, Kluwer Academic Publishers.

BILSEN, G., ENGELS, M., LAUWEREINS, R., AND PEPERSTRAETE, J. 1996. Cyclo-static dataflow. *IEEE Trans. Signal Process.* 44, 2, 397-408.

BOLCH, G., GREINER, S., DE MEER, H., AND TRIVEDI, K. S. 2006. *Queueing Networks and Markov Chains: Modeling and Performance Evaluation with Computer Science Applications*, 2nd Edition, John Wiley & Sons.

BUCK, J. T. AND LEE, E. A. 1993. Scheduling dynamic dataflow graphs with bounded memory using the token flow model. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing: Plenary, Special, Audio, Underwater Acoustics, VLSI, Neural Networks - (ICASSP'93)*, vol. I. IEEE Computer Society, 429-432.

BUCK, J. T. 1994. A dynamic dataflow model suitable for efficient mixed hardware and software implementations of DSP applications. In *Proceedings of the 3rd international workshop on hardware/software co-design (CODES'94)*. IEEE Computer Society, 165-172.

CHANG, C.-S. 2000. *Performance Guarantees in Communication Networks*. Springer-Verlag, London, UK.

CHENG, A.-L., PAN, Y., YAN, X.-L., HUAN, R.-H. 2011. A general communication performance evaluation model based on routing path decomposition. *J. Zhejiang Univ. - Sci. C (Comput. & Electron.)* 12, 7, 561-573.

CRUZ, R. L. 1991a. A calculus for network delay, part I: Network elements in isolation. *IEEE Trans. Inf. Theory*. 37, 1, 114-131.

CRUZ, R. L. 1991b. A calculus for network delay, part II: Network analysis. *IEEE Trans. Inf. Theory*. 37, 1, 132-141.

DALLY, W. J. AND TOWLES, B. 2004. Principles and Practices of Interconnection Networks, Morgan Kaufmann Publishers Inc., First edition.

DASDAN, A. 2004. Experimental analysis of the fastest optimum cycle ratio and mean algorithms. *ACM Trans. Des. Autom. Electron. Syst.* 9, 4, 385-418.

DASDAN, A. AND GUPTA, R. 1998. Faster maximum and minimum mean cycle algorithms for system performance analysis. *IEEE Trans. Computer-Aided Design Integr. Circuits Syst.*17, 10, 889-899.

Fischer W. and Meier-Hellstern K. 1993. The Markov-Modulated Poisson Process (MMPP) Cookbook. *Performance Evaluation*, 18, 2, 149-171.

FOROUTAN, S., THONNART, Y., HERSEMEULE, R., AND JERRAYA, A. 2009. Analytical computation of packet latency in a 2D-mesh NoC. In *Proceedings of the Joint IEEE North-East Workshop on Circuits and Systems and TAISA Conference*, 1-4.

FOROUTAN, S., THONNART, Y., HERSEMEULE, R., AND JERRAYA, A. 2010. An analytical method for evaluating network-on-chip performance. In *Proceedings of the 13th Conference on Design, Automation and Test in Europe (DATE'10)*.1629-1632.

GHAMARIAN, A. H., GEILEN, M. C. W., STUIJK, S., BASTEN, T. THEELEN, B. D., MOUSAVI, M. R., MOONEN, A. J. M., AND BEKOOIJ, M. J. G. 2006. Throughput analysis of synchronous data flow graphs. In *Proceedings of the 6th International Conference on Application of Concurrency to System Design (ACSD'06)*. IEEE Computer Society, 25-36.

GHAMARIAN, A. H., STUIJK, S., BASTEN, T., GEILEN, M. C. W., AND THEELEN, B. D. 2007. Latency minimization for synchronous data flow graphs. In *Proceedings of the 10th Euromicro Conference on Digital System Design Architectures, Methods and Tools (DSD'07)*. IEEE Computer Society, 189-196.

GRATZ, P. V. AND KECKLER, S. W. 2010. Realistic workload characterization and analysis for networks-on-chip design , In *Proceedings of the 4th Workshop on Chip Multiprocessor Memory Systems and Interconnects, Held in conjunction with the 16th IEEE International Symposium on High-Performance Computer Architecture (HPCA-16)*.

GUAN, W., TSAI, W., AND BLOUGH, D. 1993. An analytical model for wormhole routing in multicomputer interconnection networks. In *Proceedings of the International Parallel Processing Symposium*, 650-654.

GUZ, Z., WALTER, I., BOLOTIN, E., CIDON, I., GINOSAR, R., AND KOLODNY, A. 2007. Network delays and link capacities in application-specific wormhole NoCs. *J. VLSI Design*, 2007, article 90941, 15 pages.

HAMANN, A., JERSAK, M., RICHTER, K., AND ERNST, R. 2004. Design space exploration and system optimization with SymTA/S - symbolic timing analysis for systems. In *Proceedings of the 25th IEEE International Real-Time Systems Symposium (RTSS'04)*. IEEE Computer Society, 469-478.

HANSSON, A., WIGGERS, M., MOONEN, A., GOOSSENS, K., AND BEKOOIJ, M. J. G. 2009. Enabling application-level performance guarantees in network-based systems on chip by applying dataflow analysis. *IET* Comput. Digital Tech. 3, 5, 398 - 412.

HANSSON, A. AND GOOSSENS, K. 2010. *On-chip Interconnect with Aelite: Composable and Predictable Systems*, Springer.

Hemani, A., Jantsch, A., Kumar, S., Postula, A., Öberg, J., Millberg, M. And Lindqvist, D. 2000. Network on a Chip: An architecture for billion transistor era. *Proceedings of the Norchip Conference.*

HU, P.-C. AND KLEINROCK, L. 1997. An analytical model for wormhole routing with finite size input buffers. In *Proceedings of the 15th International Teletraffic Congress*. 549-560.

HU, J., OGRAS, U. Y. AND MARCULESCU, R. 2006. System-level buffer allocation for application-specific networks-on-chip router design. *IEEE Trans. Computer-Aided Design Integr. Circuits Syst.* 25, 12, 2919-2933.

HUR, J. Y., GOOSSENS, K., AND MHAMDI, L. 2008. Performance analysis of soft and hard single-hop and multi-hop circuit-switched interconnects for FPGAs. In *Proceedings of the IFIP International Conference on Very Large Scale Integration*, 224-229.

JACKSON, J. R. 1957. Networks of waiting lines. *Operations Research*, 5, 518-521.

JAFARI, F., LU, Z., JANTSCH, A., AND YAGHMAEE, M. H. 2010. Buffer optimization in network-on-chip through flow regulation. *IEEE Trans. Computer-Aided Design Integr. Circuits Syst.* 29, 12, 1973-1986.

JANTSCH, A., AND SANDER, I. 2005. Models of computation and languages for embedded system design. *IEE Proceedings of Computers and Digital Techniques*, 152, 2, 114-129.

KAHNG, A. B., LI, B., PEH, L. S. SAMADI, K. 2012. ORION 2.0: A Power-Area Simulator for Interconnection Networks. *IEEE Trans. VLSI Syst.* 20, 1, 191-196.

KIASARI, A. E., SARBAZI-AZAD, H., AND OULD-KHAOUA, M. 2008. An accurate mathematical performance model of adaptive routing in the star graph. *Future Generation Computer Systems,* 24, 6, 461-474.

KIM, J. AND DAS, C. R. 1994. Hypercube communication delay with wormhole routing. *IEEE Trans. Comput.* 43, 7, 806-814.

KIM, J., PARK, D., NICOPOULOS, C., VIJAYKRISHNAN, N., AND DAS, C. R. 2005. Design and analysis of an NoC architecture from performance, reliability and energy perspective. In *Proceedings of the ACM Symposium on Architecture for Networking and Communications Systems (ANCS'05)*. ACM Press, 173-182.

KRIMER, E., KESLASSY, I., KOLODNY, A., WALTER, I., AND EREZ, M. 2011. Static timing analysis for modeling QoS in networks-on-chip. *J. Parallel Distrib. Comput.* 71, 5, 687-699.

LAUWEREINS, R., WAUTERS, P., ADE, M., PEPERSTRAETE, J. A. 1994. Geometric parallelism and cyclo-static data flow in GRAPE-II. In *Proceedings of the International Workshop on Rapid System Prototyping*, 90-107.

LAZOWSKA, E. D., ZAHORJAN, J., GRAHAM, G. S. AND SEVCIK, K. C. 1984. Quantitative System Performance - Computer System Analysis Using Queueing Network Models. Englewood Cliffs, NJ: Prentice-Hall.

LE BOUDEC J.-Y. AND THIRAN, P. 2001. *Network Calculus: A Theory of Deterministic Queuing Systems for the Internet*. Springer-Verlag.

LEE, E. A. AND MESSERSCHMITT, D. G. 1987. Synchronous data flow. *Proceedings of the IEEE*, 75, 9, 1235-1245.

LEE, E. A. AND PARKS, T. M. 1995. Dataflow process networks. *Proceedings of the IEEE*, 83, 5, 773-799.

LEHOCZKY, J. P., SHA, L., AND DING, Y. 1989. The rate monotonic scheduling algorithm: exact characterization and average case behavior. In *Proceedings of the IEEE Real-Time Systems Symposium*, 166-171.

LEUNG, J. Y. T. AND WHITEHEAD, J. 1982. On the complexity of fixed priority scheduling of periodic, real-time tasks. *Performance Evaluation*, 2, 4, 237-250.

LI, J.-P. AND MUTKA, M. W. 1994. Priority based real-time communication for large scale wormhole networks. In *Proceedings of the 8th International Symposium on Parallel Processing,* IEEE Computer Society, 433-438.

LIEVERSE, P., VAN DER WOLF, P. VISSERS K. AND DEPRETTERE, E. 2001. A methodology for architecture exploration of heterogeneous signal processing systems. *Journal of VLSI Signal Processing Systems for Signal, Image and Video Technology*, 29. 3, 197-207.

LIU C. L. AND LAYLAND, J. W. 1973. Scheduling algorithms for multiprogramming in a hard-real-time environment. *J. ACM* 20, 1, 46-61.

LIU, J. W. S. 2000. *Real-Time Systems (1st ed.)*. Prentice Hall.

LU, Z., JANTSCH, A., AND SANDER, I. 2005. Feasibility analysis of messages for on-chip networks using wormhole routing. In *Proceedings of the Asia and South Pacific Design Automation Conference (ASP-DAC'05)*. ACM Press, 960-964.

LU, Z., MILLBERG, M., JANTSCH, A., BRUCE, A., VAN DER WOLF, P., AND HENRIKSSON, T. 2009. Flow regulation for on-chip communication. In *Proceedings of the Design, Automation and Test in Europe Conference (DATE'09)*. 578-581.

LU. Z. 2011. Cross clock-domain TDM virtual circuits for networks on chips. In *Proceedings of the 5th ACM/IEEE International Symposium on Networks-on-Chip (NoCS'11)*. ACM Press, 209-216.

MIN, G. AND OULD-KHAOUA, M. 2004. A performance model for wormhole-switched interconnection networks under self-similar traffic. *IEEE Trans. Comput.* 53, 5, 601-613.

MOREIRA, O. M. AND BEKOOIJ, M. J. G. 2007. Self-timed scheduling analysis for real-time applications. *EURASIP J. Advances in Signal Processing*, 2007, id: 083710.

OGRAS, U. Y., BOGDAN, P., AND MARCULESCU, R. 2010. An analytical approach for network-on-chip performance analysis. *IEEE Trans. Comp.-Aided Des. Integ. Cir. Sys.* 29, 12, 2001-2013.

OWENS, J. D., DALLY, W. J., HO, R., JAYASIMHA, D. N., KECKLER, S. W., PEH L. S. 2007. Research Challenges for On-Chip Interconnection Networks. *IEEE Micro.* 27, 5, 96-108.

PAWLIKOWSKI, K. 1990. Steady-State Simulation of Queueing Processes: A Survey of Problems and Solutions. *ACM Computing Surveys*, 22, 2, 123–170.

QIAN, Y., LU, Z., AND DOU, W. 2009a. Applying network calculus for performance analysis of self-similar traffic in on-chip networks. In *Proceedings of the 7th IEEE/ACM International Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS'09)*. ACM Press, 453-460.

QIAN, Y., LU, Z., AND DOU, W. 2009b. Analysis of worst-case delay bounds for best-effort communication in wormhole networks on chip. In *Proceedings of the 3rd ACM/IEEE International Symposium on Networks-on-Chip (NOCS'09)*. IEEE Computer Society, 44-53.

QIAN, Y., LU, Z., AND DOU, W. 2009c. Worst case flit and packet delay bounds in wormhole networks on chip. *IEICE Trans. Fundamentals of Electronics, Communications and Computer Sciences, Special Section on VLSI Design and CAD Algorithms*, E92-A, 12, 3211-3220.

QIAN, Y., LU, Z., AND DOU, W. 2010a. Analysis of worst-case delay bounds for on-chip packet-switching networks. *IEEE Trans. Comp.-Aided Des. Integ. Cir. Sys.* 29, 5, 802-815.

QIAN, Y., LU, Z., AND DOU, W. 2010b. QoS scheduling for NoCs: strict priority queueing versus weighted round robin. In *Proceedings of the 28th International Conference on Computer Design (ICCD'10)*, 52-59.

RYOO, S., UENG, S., CHRISTOPHER, I. R., KIDD, R. E., FRANK, M. I., AND HWU, W. W. 2007. Automatic discovery of coarse-grained parallelism in media applications. *Transactions on High-Performance Embedded Architectures and Compilers*, 1, 1, 187-206.

SHI, Z. AND BURNS, A. 2008. Real-time communication analysis for on-chip networks with wormhole switching. In *Proceedings of the 2nd ACM/IEEE International Symposium on Networks-on-Chip (NOCS'08)*. IEEE Computer Society, 161-170.

SHI, Z. AND BURNS, A. 2009. Real-time communication analysis with a priority share policy in on-chip networks. In *Proceedings of the 21st Euromicro Conference on Real-Time Systems (ECRTS)*. IEEE Computer Society, 3-12.

SHI, Z. AND BURNS, A. 2010. Schedulability analysis and task mapping for real-time on-chip communication. *Real-Time Systems*. 46, 3, 360-385.

SOTERIOU, V., EISLEY, N., WANG, H., LI, B. PEH, L.-S. 2007. Polaris: A System-Level Roadmapping Toolchain for On-Chip Interconnection Networks. *IEEE Trans. VLSI Syst.* 15, 8, 855-868.

SRIRAM, S., AND BHATTACHARYYA, S. S. 2009. *Embedded Multiprocessors: Scheduling and Synchronization* (2nd ed.). CRC Press.

STILIADIS, D. AND VARMA, A. 1998. Latency-rate servers: a general model for analysis of traffic scheduling algorithms. *IEEE/ACM Trans. Netw.* 6, 5, 611-624.

STUIJK, S., GEILEN, M. C. W., AND BASTEN, T. 2006. Exploring trade-offs in buffer requirements and throughput constraints for synchronous dataflow graphs. In *Proceedings of the 43rd Annual Design Automation Conference (DAC'06)*. ACM Press, 899-904.

STUIJK, S., GEILEN, M. C. W., THEELEN, B. D., AND BASTEN, T. 2011. Scenario-aware dataflow: modeling, analysis and implementation of dynamic applications. In *Proceedings of the International Conference on Embedded Computer Systems,* 404-411.

VARATKAR, G. V. AND MARCULESCU, R. 2004. On-chip traffic modeling and synthesis for MPEG-2 video applications. *IEEE Trans. Very Large Scale Integr. Syst.* 12, 1, 108-119.

VECCHIA G. D. AND SANGES C. 1988. A recursively scalable network VLSI implementation. *Future Generation Computer Systems*, 4(3) 235-243.

WANG, J., LI, Y., AND PENG, Q. 2011. A novel analytical model for network-on-chip using semi-Markov process. *Advances in Electrical and Computer Engineering*, 11, 1, 111-118.

WIGGERS, M. H., BEKOOIJ, M. J. G., AND SMIT, G. J. M. 2007. Efficient computation of buffer capacities for cyclo-static dataflow graphs. In *Proceedings of the 44th Annual Design Automation Conference (DAC'07)*. ACM Press, 658-663.

WU, J., LIU, J.-C., AND ZHAO, W. 2010. A general framework for parameterized schedulability bound analysis of real-time systems. *IEEE Trans. Comput.* 59, 6, 776-783.

ZHANG, H. 1995. Service disciplines for guaranteed performance service in packet-switching networks. *Proceedings of the IEEE*, 83, 10, 1374-1396.

Part II

Included Papers

Paper 1

A Performance and Power Analysis of WK-Recursive and Mesh Networks for Network-on-Chips

Dara Rahmati
Abbas Eslami Kiasari
Shaahin Hessabi
Hamid Sarbazi-Azad

In the Proceedings of the 24th IEEE International Conference on Computer Design (ICCD), pp. 142-147, San Jose, CA, USA, Oct. 2006.

A Performance and Power Analysis of WK-Recursive and Mesh Networks for Network-on-Chips

D. Rahmati[1], A. E. Kiasari[1,2], S. Hessabi[1], H. Sarbazi-Azad[1,2]
[1] Department of Computer Engineering,
Sharif University of Technology,
Tehran, Iran
[2] School of Computer Science,
Institute for Studies in theoretical Physics and Mathematics (IPM),
Tehran, Iran

Abstract
Network-on-Chip (NoC) has been proposed as an attractive alternative to traditional dedicated wires to achieve high performance and modularity. Power efficiency is one of the most important concerns in NoC architecture design. The choice of network topology is important in designing a low-power and high-performance NoC. In this paper, we propose the use of the WK-recursive networks to be used as the underlying topology in NoC. We have implemented VHDL hardware model of mesh and WK-recursive topologies and measured the latency results using simulation with these implementation. We also propose a novel approach in high level power modeling based on latency for these topologies and show that the power consumption of WK-recursive topology is less than that of the equivalent mesh on a chip.

Index Terms—System-on-chips, Network-on-chips, Mesh, WK-Recursive mesh, Routing, Power, Performance.

1. Introduction

With the advance of the semiconductor technology, the enormous number of transistors available on a single chip allows designers to integrate dozens of IP (Intellectual Property) blocks together with large amounts of embedded memory. Such IPs can be CPU or DSP cores, video stream processors, high-bandwidth I/O, *etc.* Shared-medium busses do not scale well, and do not fully utilize potentially available bandwidth. As the feature sizes shrink, and the overall chip size relatively increases, the interconnects start behaving as lossy transmission lines. Line delays have become very long as compared to gate delays causing synchronization problems between cores. A significant amount of power is dissipated on long interconnects and in clocking network. This trend only worsens as the clock frequencies increase and the features sizes decrease. Lowering the power supply voltage and designing low swing circuits decrease the overall power consumption at the cost of higher data errors.

One solution to these problems is to treat systems on a chip implemented using *micro-networks*, or *Networks on Chips* (NoCs). Networks have a much higher bandwidth due to multiple concurrent connections. Regularity enables design modularity, which in turn provides a standard interface for easier component reuse and better interoperability. Overall performance and scalability increase since the networking resources are shared. Scheduling of traffic on shared resources prevents latency increases on critical signals.

Power efficiency is one of the most important concerns in NoC architecture design. Consider a 10×10 tile-based NoC, assuming a regular mesh topology and 32 bit link width in 0.18*um* technology and minimal spacing, under 100Mbit/s pair-wise communication demands,

interconnects will dissipate 290W of power [4]. Thus, reducing the power consumption on global interconnects is a key factor to the success of NoC designs.

The choice of network topology is important in designing a low-power NoC. Different NoC topologies can dramatically affect the network characteristics, such as average inter-IP distance, total wire length, and communication flow distributions. These characteristics in turn determine the power efficiency of NoC architectures. In recent years, several new parallel computer architectures have been proposed in the literature for building massively parallel computer systems to increase computation speed. A major drawback for these networks is that there are not predefined modules existent for them when they are fabricated onto a monolithic chip. The reason is that they are not truly expansible. In addition, the irregularity of node degrees also makes them costly for VLSI implementation.

The WK-recursive networks [7] are a class of recursively scalable networks with many desirable properties. They offer a high degree of regularity, scalability, and symmetry, which very well conform to a modular design and implementation of distributed systems involving a large number of computing elements. In [7], a VLSI implementation of the WK-recursive networks is described, and a routing algorithm is developed. This algorithm defines the physical channels which must be traversed, but does not address the use of virtual channels. Virtual channels are usually used to increase performance and to design deadlock free routing algorithms. In this paper we propose a new virtual channel selection policy for WK-recursive network which results in a deadlock-free routing algorithm.

In this research, we compare the two most important performance factors (latency and power) of the same size mesh and WK-recursive networks for NoC implementation. To this end, we have implemented a hardware model using VHDL for accurate simulation of the networks in question. A routing algorithm for the WK-recursive network has been proposed and the performance of the two networks under similar working conditions has been assessed and compared. We also develop a high level power consumption model to compare the candidate networks with their energy requirements.

2. WK-recursive network structure

The WK-recursive networks can be constructed recursively by grouping basic modules. Any d-node complete graph can serve as the basic module. Throughout this paper, we use WK(d,t) to denote a WK-recursive network of level t whose basic modules are some d-node complete graph, where $d > 1$ and $t \geq 1$. Each node of WK(d,t) is uniquely identified by a sequence of t numbers, and each of its edges is represented by a pair of nodes. We define WK(d,t) formally as follows:

Definition 2.1. The node set of WK(d,t) is denoted by $\{a_t a_{t-1} \ldots a_2 a_1 \mid a_i \in [0,d\text{-}1] \text{ for } 1 \leq i \leq t\}$. The adjacency is defined as follows: $a_t a_{t-1} \ldots a_2 a_1$ is adjacent to (1) $a_t a_{t-1} \ldots a_2 b$ where $0 \leq b \leq d - 1$ and $b \neq a_1$, and (2) $a_t a_{t-1} \cdots a_{i+1} a_{i-1} (a_i)^{i-1}$ if $a_i \neq a_{i-1}$ and $a_{i-1} = a_{i-2} = \ldots = a_2 = a_1$, where $(a_i)^{i-1}$ represents i -1 consecutive a_i's. Besides, an *open edge* is incident to $a_t a_{t-1} \ldots a_2 a_1$ if $a_t = \ldots = a_2 = a_1$. Each edge of WK(d,t) is assigned a label as follows: 0 if it is of type(1), i - 1 if it is of type (2), and t if it is an open edge. The edges of type (1) are referred to as *substituting edges,* and the edges of type (2) are referred to as *flipping edges*.

In Fig. 1, the topologies of 16-node WK(4,2) and mesh(4x4) are shown. In WK(d,t), each node is of degree d, and there are totally d^t nodes and $d + d(d^t - 1) / 2$ edges. In [2], it has been shown that the diameter of WK(d,t) depends on d and equal to 2^t -1.

Definition 2.2. For any two nodes U and V in WK(d,t), we define $U =_i V$ if they belong to the same subnetwork of level i in WK(d,t), and $U \neq_i V$ if they belong to two distinct subnetworks of level i in *WK(d,t)*.

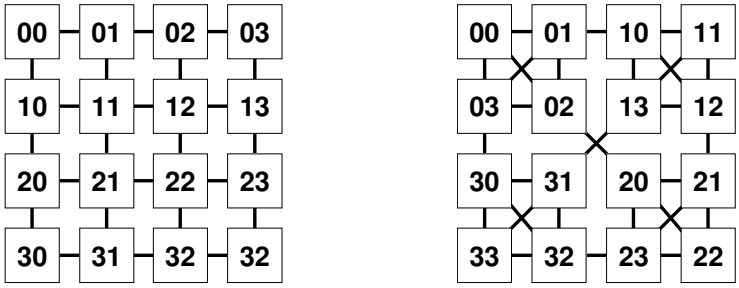


Fig. 1. The topologies of (a) Mesh(4x4) and (b) WK(4,2) with 16 nodes.

Definition 2.3. For each i, $1 \leq i \leq t$, a node $S = a_t a_{t-1} \ldots a_2 a_1$ in WK(d,t) is called an *i-frontier,* if $S = a_t a_{t-1} \ldots a_{i+1} (a_i)^i$.

Remarks. Two nodes connected by a flipping edge of label i are i-frontiers. Also, by definition, if a node is an i-frontier, it is also a j-frontier for $1 \leq j < i$.

2.1. Routing algorithms

Each wormhole routing algorithm includes two important parts: (1) physical channel selection rule and (2) virtual channel selection rule. Physical channel selection rule chooses the next physical channel to route the message and virtual channel selection rule chooses the proper virtual channel in selected physical channel by considering of deadlock avoidance conditions.

2.2. Physical channel selection rule

Suppose S and T are the source node and destination node in WK(d,t), respectively. A routing path between them can be constructed as follows [1].

$S \neq_{i-1} T$. This can be easily done by examining the identifiers of S and T from the left, and finding the first position where they differ.

Step 2. Determine the flipping edge, say (W,X), such that $S \neq_{i-1} W$ and $X \neq_{i-1} T$. The flipping edge is the bridge between the two sub-networks of level i-1 where S and T reside. The nodes W and X are (i-1)-frontiers, and they can uniquely be determined by examining the identifiers of S and T.

Step 3. Determine the routing path from S to W, and the routing path from X to T, recursively. The routing path from S to T is the concatenation of the routing path from S to W, the flipping edge (W,X), and the routing path from X to T.

2.3. Virtual channel selection rule

After selection of next physical channel according to the algorithm in previous section, suitable virtual channel must be selected in this physical channel by considering deadlock avoidance conditions.

2.3.1. The Positive-hop (PHop) policy

In the PHop policy [1], a message is placed in a virtual channel of class 0 in the source node upon injection into the network. During routing, a message is placed in the virtual channel of class i in an intermediate node if it has already completed i hops. Since the maximum number of hops a message can take equals the diameter of the network, the maximum number of virtual channel classes required in each node equals the diameter of the network; for WK(d,t), this number equals 2^t -1.

2.3.2. The Flipping-hop (FHop) policy

We now propose a novel virtual channel selection rule for WK-recursive networks which has less requirements than PHop policy. In PHop policy, the virtual channel class number is incremented in each hop but in FHop policy a message is placed in a virtual channel of class 0 in the source node and in an intermediate node, and the virtual channel class number is incremented if the selected physical channel is corresponding to a flipping edge. Since the maximum number of flipping edges in a path equals the half of diameter of the network [2], the maximum number of virtual channel classes required in each node equals the half of the diameter; for WK(*d,t*), this number equals $2^{\lfloor t/2 \rfloor}$.

2.4. Performance of PHop and FHop

To compare the performance of these routing algorithms, we have developed a VHDL based cycle accurate model for evaluating the latency and power of NoC based interconnection architectures. The design is parameterized on (i) size of packets, (ii) length and width of physical links, (iii) number and depth of virtual channels. This simulation model can be used for the WK-recursive networks of any size with wormhole switching. We compare the performances of PHop and FHop wormhole routing algorithms. We have simulated these routing algorithms for WK(4,2). Also we have considered fixed length messages of 32 flits. Nodes generate traffic independently of each other, and which follows a Poisson process. For the destination address of each message, we have considered the uniform traffic pattern.

In Fig. 2, the average message latency is plotted against message generation rate for WK(4,2) network with FHop routing algorithm and with 32-flit messages. In WK(4,2), we need 3 and 2 virtual channel classes for PHop and FHop routing algorithms and we have also used 2 and 3 virtual channels in each class, respectively. Therefore, there are 6 virtual channels per physical channel in both networks that can ensure a fair comparison under almost equal hardware cost. Also in our simulation experiments all virtual channels are of 1-flit depth. As you see in Fig. 2, for low and medium traffic loads, these routing algorithms have the same latency, but they begin to behave differently for high traffic loads and around the saturation region. The FHop routing algorithm has better performance than PHop routing algorithm. Therefore, from now on, we use the FHop routing algorithm for message routing in WK-recursive networks.

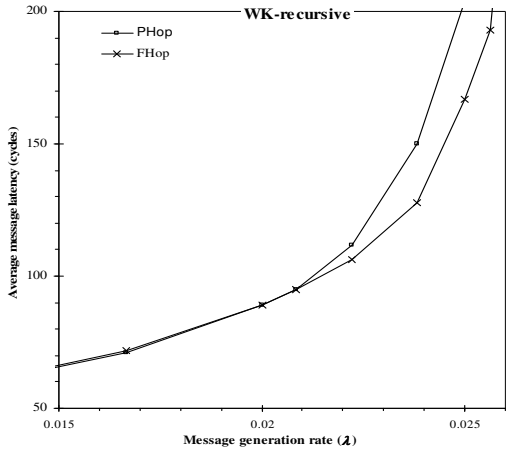


Fig. 2. Message latency in WK(4,2) with PHop and FHop routing algorithms.

3. Mesh and WK-recursive hardware model and latency comparison

The top most shared component in this hardware model is the NoC node, in which *PE* (Processing Element) and *router* are the main components. The *PE* is a module that injects/ejects

the generated/receiving packets based on a traffic model like uniform, hotspot, etc. Routers receive packets on their input channels and after routinga packet based on the routing algorithm and destination address, the packet is sent to the selected output channel. Fig. 3 shows internal structure of a *node*. A *router* consists of several different parts such as *Address Extractor* which determines and manipulates the packet headers and buffers some flits of the packet, *Multiplexer* and *De-Multiplexer* which handle the virtual channel operations, *Selector* unit which applies the virtual channel selection rule, *Crossbar switch* which directly connects each input channel to each unoccupied output channel, *Reservator* unit which controls the crossbar switch and other related sub-modules. When a specific topology like mesh or WK-recursive is supposed to be modeled by such components, a top-level wrapper module is implemented that connects several nodes of this type to each other based on the structure of the specified topology. We have simulated the hardware models of the mesh and WK-recursive networks to extract accurate quantities, e.g. latency values.

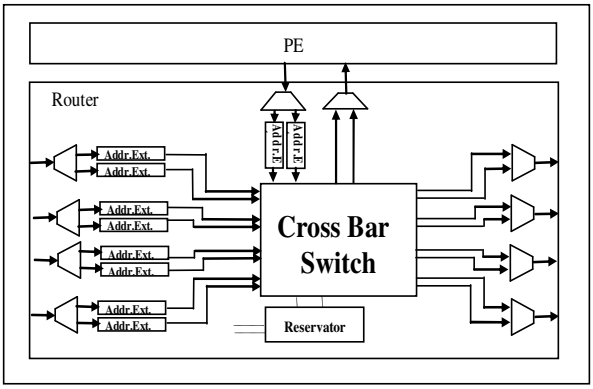


Fig. 3. Hardware implementation of a node with a PE and a Router.

In Fig. 4, the average message latency is plotted as a function of average message generation rate at each node for a 4x4 mesh interconnection network with XY routing algorithm [3] and a WK(4,2) network with FHop routing algorithm. In WK(4,2), we need 2 virtual channel classes for FHop routing algorithm. For low traffic loads, the WK-recursive provides a better performance compared to the mesh network, but they begin to behave differently near high traffic regions. It is notable that a usual advice on using any networked system is "not take the network working near saturation region". Having considered this and also the fact that most of networks rarely enter such traffic regions, we can conclude that the WK-recursive network can outperform its equivalent mesh network when average message latency is considered.

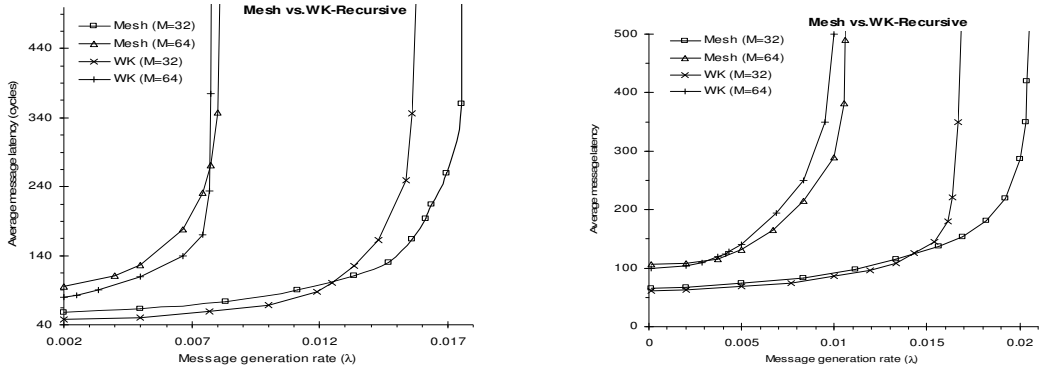


Fig. 4. Message latency in WK(4,2) and Mesh(4x4) with (a) 2 and (b) 4 virtual channels.

4. Mesh and WK-recursive power modeling and comparison

Reducing power consumption is required in today's semiconductor designs. Silicon technology advances have made it possible to pack millions of transistors switching at high clock speeds on a single chip. While these advances bring unprecedented performance to electronic products, they pose difficult power dissipation and distribution problems [5]. These problems must be addressed, because consumers demand longer battery life in addition to lower cost in computers, battery-operated systems, medical devices, telecommunications equipment and many high-volume consumer products. In this section, we propose a novel high level approach for modeling the power consumption of mesh and WK-recursive NoCs. The model computes power dissipation for a packet crossing the network, thus static and dynamic powers are both included in our analysis.

We first introduce some assumptions and definitions used in our analysis.

- The average distance of the mesh and WK-recursive networks are defined as D_{mesh} and D_{wk} .
- The uniform traffic pattern is used for message destination address.
- The length of wire between two switches is fixed.
- The power consumption is calculated for two different operating regions of the NoC, namely the low and high traffic regions. The low traffic region in a NoC is the region that there is no packet contention or the contention is rare. Also the high traffic is defined as a region that packet blocking is frequently occurred but there is no packet deadlock and network does not enter the saturation region. Let λ be the packet generation rate at a node and λ_s be the value of λ at the saturation point (the point from where the saturation region starts). Let the low traffic load be defined as $\lambda \leq 0.6\lambda_s$ and the high traffic region be defined as $0.6\lambda_s < \lambda < \lambda_s$.

As defined in [6][8], the average energy consumed for transferring a packet between two nodes is as follows:

E_P: Total energy dissipated for packet transfer
E_B: Energy dissipated for packet buffering
E_S: Switching energy dissipated for packet transfer
E_W: Energy dissipated in wires for packet transfer
$E_P = E_B + E_S + E_W$

We have

$$E_{BS} = E_B + E_S \,,\; E_P = E_{BS} + E_W \tag{1}$$

We consider both the buffering and switching (router) energy in a single parameter E_{BS}. Also we define E^C_W as average wiring energy which is dissipated between two switches for a packet transfer, $E^C_{BS,\,Low}$ is average buffering and switching energy which is dissipated in a switch for a packet transfer in low traffic and $E^C_{BS,\,High}$ for the high traffic. $E_{blocking}$ is the average blocking energy which is dissipated for a packet transfer in high traffic. This parameter represents the extra buffering energy which is dissipated in switch buffers during packet blocking. We estimate this parameter in the next section where we calculate the energy dissipation for high traffic mode of operation. At last we define

$$\alpha = E^C_W \,/\, E^C_{BS,\,Low} \tag{2}$$

in order to simplify the calculations. It shows the relation of wiring and router energy dissipation for a packet transfer.

4.1. Low traffic modeling

In low traffic region, packets are transferred across the network with no contention or negligible contention. Using equation (1) and (2), the average total energy dissipated for a packet transfer in the mesh and WK-recursive network topologies can be computed as:

$$E_{P,\,WK,\,Low} = (D_{WK}+1).\,E^{C}_{BS,\,Low} + D_{WK}.\,E^{C}_{W}$$

and the same formula for mesh network:

$$E_{P,\,Mesh,\,Low} = (D_{Mesh}+1).\,E^{C}_{BS,\,Low} + D_{Mesh}.\,E^{C}_{W}$$

Therefore, we define K as follows:

$$K = \frac{E_{P,WK,Low}}{E_{P,Mesh,Low}} = \frac{(\alpha+1)D_{WK}+1}{(\alpha+1)D_{Mesh}+1} \tag{3}$$

where a is the parameter in equation (2) and K shows the ration of the energy dissipated for Mesh and WK-recursive in low traffic region for a packet transfer. For Mesh (4x4) and WK(4,2), we have D_{WK}=2.21, D_{Mesh}=2.67 and thus

$$K = \frac{2.21\alpha + 3.21}{2.67\alpha + 3.67}$$

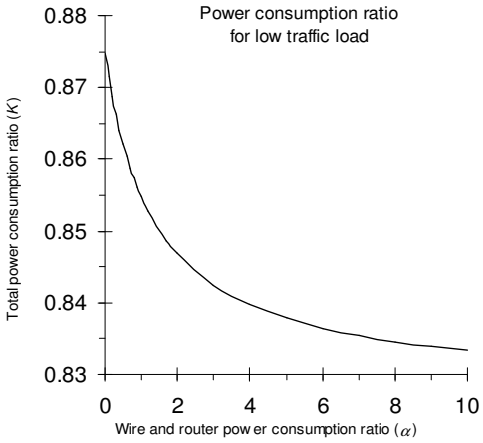


Fig. 5. The ratio of a packet transfer power dissipation for a WK(4,2) and mesh(4x4) as function of a for low traffic load.

Depending on all values of a, the power consumption ratio may vary from 0.83 to 0.88 as is shown in Fig. 5. This means that the WK-recursive consumes lower amount of energy than its mesh counterpart in low traffic.

4.2. High traffic modeling

Near and in high traffic regions, the energy consumption is heavily affected by the packet contention. In this case a large amount of the dissipated energy comes from packet blocking (energy consumed for buffering). The other forms of energy dissipation in wires, switching hardware, and general buffering is almost the same for low traffic load.

Let us define two parameters T_{Mesh} and T_{WK} representing the average blocking time of a packet when crossing the network. The energy dissipated for a packet transfer can be written as

$$E_{P,\,WK,\,High} = E_{P,\,WK,\,Low} + E_{blocking,\,WK}$$
$$E_{P,\,Mesh,\,High} = E_{P,\,Mesh,\,Low} + E_{blocking,\,Mesh} \quad (4)$$

also, the average energy consumed due to the message blocking is proportional to its average blocking time and also the average number of switches it traverses, i.e. $E_{blocking,\,WK} \propto T_{WK}.(D_{WK}+1)$ and $E_{blocking,\,Mesh} \propto T_{Mesh}.(D_{Mesh}+1)$. By using a constant we can write
$E_{blocking,\,WK} = C.T_{WK}.(D_{WK}+1)$,
$E_{blocking,\,Mesh} = C.T_{Mesh}.(D_{Mesh}+1)$.

Note that since the hardware components used in both topologies are almost equal, we have used one constant for both equations. Considering $C = \varepsilon.E^{C}_{BS,\,Low}$ we have

$$E_{P,\,WK,\,High} = D_{WK}(1+\alpha+\varepsilon T_{WK}) + \varepsilon.\,T_{WK}+1$$
$$E_{P,\,Mesh,\,High} = D_{Mesh}(1+\alpha+\varepsilon T_{Mesh}) + \varepsilon.\,T_{Mesh}+1 \quad (5)$$

Thus the energy consumption ratio in this case equals

$$K = \frac{E_{p,wk,high}}{E_{p,mesh,high}} \text{ or}$$

$$K = \frac{D_{wk}(1+\alpha+\varepsilon\times T_{wk})+\varepsilon\times T_{wk}+1}{D_{mesh}(1+\alpha+\varepsilon\times T_{mesh})+\varepsilon\times T_{mesh}+1} \quad (6)$$

Again, for the sake of present discussion, let us consider specific cases of 16-node mesh and WK-recursive networks.
To calculate the value of T_{wk} and T_{mesh}, we use the average message latencies shown in Fig. 4. It is clear that T_{WK} and T_{Mesh} can be calculated by reducing the average message latency in low traffic load from that in high traffic load. Thus, from Fig. 4(b), we have T_{WK}=63 and T_{mesh}=57, we can write

$$K = \frac{202\varepsilon+2.21\alpha+3.21}{209\varepsilon+2.67\alpha+3.67}.$$

Fig. 6 shows the *K* as a function of *a* and *ε*. As can be seen in the figure, the values for the case of $\varepsilon = 0$ is the same as Fig. 5 in low traffic region and as the value of ε increases (blocking power increases), power dissipation in the two networks tend to be equal. For smaller values of *a*, this trend happens quicker.

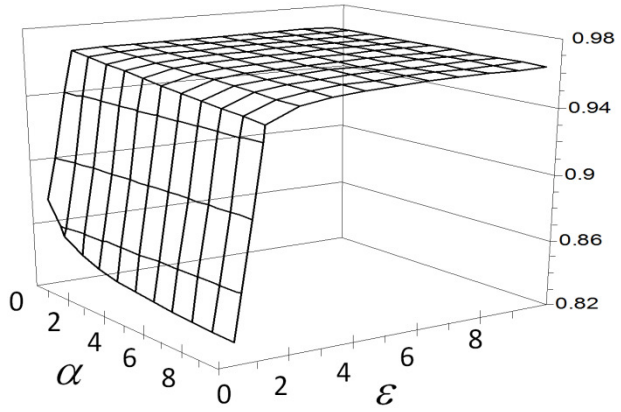


Fig. 6. The ratio of packet transfer power dissipation in the mesh(4x4) and WK(4,2) for different values of α and ε in high traffic region.

5. Conclusion and future works

Mesh topology has been used in a variety of interconnection network applications especially for NoC design. However, the WK-recursive network has not been studied yet as the underlying topology for NoCs. In this paper, we proposed a latency and power consumption comparative analysis for these two topologies (mesh and WK-recursive) and showed that the latency of the WK-recursive network for low traffic loads is superior to the mesh topology. The power consumption in the WK-recursive is also less than that of the mesh network for low traffic loads while the power consumption in the two networks is almost equal for high traffic loads. We also proposed a high level approach for modeling the power consumption of the two topologies based on the latency parameters. This approach can be applied to other topologies for NoC designs.

Our next objective is to develop a combined accurate analytical model of power consumption and performance of NoCs and validating it for different network topologies under different working conditions.

References

[1] R. V. Boppana and S. Chalasani, "A framework for designing deadlock-free wormhole routing algorithms", *IEEE Transactions on Parallel and Distributed Systems (TPDS)*, 7(2): 169-183, 1996.

[2] D. R. Duh and G. H. Chen, "Topological properties of WK-recursive networks," *Journal of Parallel and Distributed Computing (JPDC),* 23(3): 468-474, 1994.

[3] C. J. Glass and L. M. Ni, "The turn model for adaptive routing," *Proceedings of the International Symposium on Computer Architecture (ISCA)*, pp. 278-287, 1992.

[4] Y. Hu, H. Chen, Y. Zhu, A. A. Chien and C. Cheng, "Physical synthesis of energy-efficient networks-on-chip through topology exploration and wire style optimizations," *Proceedings of the International Conference on Computer Design (ICCD)*, pp. 111-118, 2005.

[5] D. L. Liu and C. Svensson, "Power consumption estimation in CMOS VLSI chips", *IEEE Journal of Solid-State Circuits (JSSC)*, 29(6): 663-670, 1994.

[6] M. Naderi, B. Javadi, H. Pedram, A. Afzali-Kusha, and M. K. Akbari, "An asynchronous viterbi-decoder for low-power applications", Pr*oceedings of the Power and Timing Modeling, Optimization and Simulation* (*PATMOS*), pp. 471-480, 2003.

[7] G. D. Vecchia and C. Sanges, "A recursively scalable network VLSI implementation," *Future Generation Computer Systems,* 4(3) 235-243, 1988.

[8] T. T. Ye,"On-chip multiprocessor communication network design and analysis," *Ph.D. dissertation, Stanford University*, 2003.

Paper 2

A Markovian Performance Model for Networks-on-Chip

Abbas Eslami Kiasari
Dara Rahmati
Hamid Sarbazi-Azad
Shaahin Hessabi

In the Proceedings of the 16th Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP), pp. 157-164, Toulouse, France, Feb. 2008.

A Markovian Performance Model for Networks-on-Chip

A. E. Kiasari [†,‡], D. Rahmati [‡], H. Sarbazi-Azad [‡,†], and S. Hessabi [‡]
[†] IPM School of Computer Science, Tehran, Iran
[‡] Sharif University of Technology, Tehran, Iran
kiasari@ipm.ir, d_rahmati@ce.sharif.edu, azad@ipm.ir, hessabi@sharif.edu

Abstract
Network-on-Chip (NoC) has been proposed as a solution for addressing the design challenges of future high-performance nanoscale architectures. Thus, it is of crucial importance for a designer to have access to fast methods for evaluating the performance of on-chip networks. To this end, we present a Markovian model for evaluating the latency and energy consumption of on-chip networks. We compute the average delay due to path contention, virtual channel and crossbar switch arbitration using a queuing-based approach, which can capture the blocking phenomena of wormhole switching quite accurately. The model is then used to estimate the power consumption of all routers in NoCs. The performance results from the analytical models are validated with those obtained from a synthesizable VHDL-based cycle accurate simulator. Comparison with simulation results indicate that the proposed analytical model is quite accurate and can be used as an efficient design tool by SoC designers.

1. Introduction

The International Technology Roadmap for Semiconductors (ITRS) predicts that chips with over 4 billion transistors operating at 10 GHz speeds will be commercialized before the end of the decade [1]. Assuming these predictions, within a few years they will bring about a significant change to the architecture and design of integrated circuits. The current design methodologies are not expected to cope with such multi-billion transistor design challenges. To meet the design productivity and signal integrity challenges of next-generation designs, packet switched NoC architectures [2][3][6] have been proposed as a solution to the global interconnect problems.

It is not clear which NoC-based architecture is best suited for a specific application. The trade-off in average message latency versus power consumption of an interconnection network is also an open question. The trade-off analysis can be performed by varying the following design parameters: topology, physical links bandwidth, buffer allocation scheme, switching technique and routing algorithm. Many studies, e.g. Dally and Towles's [6], have investigated the 2D torus NoC architecture (as shown in Figure 1.b). The Torus architecture is basically the same as a regular mesh (Figure 1.a); the only difference is that the switches at the edges are connected to the switches at the opposite edge through wraparound channels. Every switch has five ports, one connected to the local IP and the others connected to the closest neighboring switches.

Owning to its low buffering requirement *wormhole switching* has been widely employed in multicomputers. Another advantage of wormhole switching is that, in the absence of blocking, message latency is almost independent of the distance between source and destination nodes. In this switching technique, messages are broken into *flits*, each of a few bytes, for transmission and flow control. The *header* flit, containing routing information, is used to govern routing and the remaining data flits follow in a pipelined fashion. If the header is blocked, the other flits are blocked in situ. The advantage of this technique is that it reduces the impact of message distance on the latency under light traffic. Yet, as network traffic increases, messages may experience large delays to cross the network due to the chain of blocked channels. To overcome this, the flit buffers associated with a given physical channel are organised into several virtual channels, each representing a "logical" channel with its own buffer and flow control logic. Virtual channels are allocated independently to different messages and compete with each other for the physical

bandwidth. This decoupling allows messages to bypass each other in the event of blocking, using network bandwidth that would otherwise be wasted.

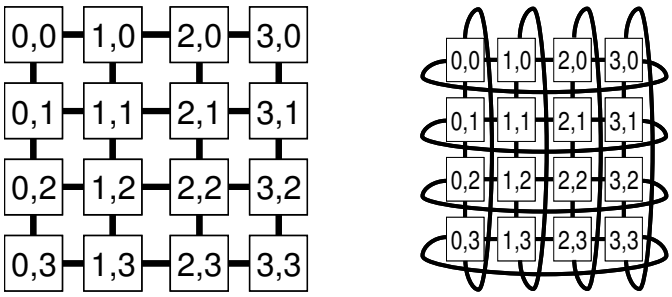


Figure 1: The topology of (a) 4X4 mesh and (b) 4X4 torus networks.

Routing algorithms establish the path between the source and destination nodes. Routing can be deterministic or adaptive. With adaptive routing, the path taken by a message is affected by the traffic on network channels. In deterministic routing, messages with the same source and destination always traverse the same path. This form of routing results in a simpler router implementation [9] and has been used in many practical systems. In this research for deadlock free routing, a restricted virtual channel allocation scheme, in the context of deterministic routing, is enforced. In this scheme, the V virtual channels of a given physical channel are split into two sets: $VC_1 = \{v_3, v_4, \ldots, v_V\}$ and $VC_2 = \{v_1, v_2\}$. A message at node address $C = x_c y_c$ and destined to node $D = x_d y_d$, can choose any of the V-2 virtual channels in VC_1 of dimension X (and then Y). If all these virtual channels are busy, the message uses v_1 when $x_c < x_d$ (or $y_c < y_d$), otherwise it crosses v_2 [9]. In some of physical channels either v_1 or v_2 is never used and we can use V-1 virtual channels in VC_1 for more adaptivity [9].

Simulation is an approach to evaluate the performance of an interconnection network for a specific configuration. But, depending on the complexity of the interconnection network and resources available, this technique may be too time-consuming to perform. Another approach is utilization of an analytical model of the system. An appropriate analytical model can predict the performance of a specific on-chip network in a fraction of the time that simulation would take. Thus, it is justified to be in pursuit of accurate analytical models for the performance of popular NoCs such as the torus.

The rest of the paper is organized as follows. Section 2 reviews related work and highlights our contributions. Section 3 proposes a mathematical performance model for wormhole-switched NoCs. In Section 4, a simple model for power consumption of routers in torus-based NoCs is presented. Validation of the proposed performance and power models are realized in Section 5 using the results obtained from simulation experiments. Finally, Section 6 concludes the paper.

2. Related Work and Novel Contribution

The design of NoCs is commonly formulated as a constrained optimization problem [13][17][19]. Therefore, performance analysis techniques that can be used in optimization loops are extremely important. Traditional work on performance evaluation in parallel computing uses either time-consuming simulation (*e.g.*, [5] and [10]) or provides analytical models.

The authors in [14] consider the buffer sizing problem and present a performance model based on an M/M/1/K queuing model. However, the approach is not applicable to wormhole switched networks. Related work about analysis techniques for wormhole routing comes mainly from parallel computing and macro-network research communities. Many studies target specific network topologies such as *k*-ary *n*-cubes [7][22] and hypercubes [23]. The study presented in [11] is not restricted to a particular topology, but it assumes an exponential message length distribution and it has a very high complexity for high dimensional networks. A more general

analytical model for wormhole routing is presented in [15]. The model provides average packet latency estimates using a sophisticated analysis.

The main contribution of the work herein is a novel performance model for on-chip routers which can be used to develop a thorough performance analysis for torus network topology with deterministic routing and wormhole switching under Poisson arrival process and uniform traffic pattern. Then, we propose a high level approach for modeling the power consumption of routers for different values of basic NoC parameters. Hence, it can be invoked in any fast and accurate power and performance estimations.

3. The Latency Model

3.1. Model Assumptions

The following assumptions are made when developing the proposed performance model. These assumptions have been widely used in the literature [4][12][16][18][20]:

- Messages are broken into some packet of fixed length of M flits which are the unit of switching.
- Message destinations are uniformly distributed across the network nodes.
- Nodes generate traffic independently of each other, and follow a Poisson process, with a mean rate of λ_g messages/cycle/node.
- V virtual channels per physical channel are used. These virtual channels are used according to XY routing algorithm.
- The local queue in the source node has infinite capacity. Moreover, messages at the destination node are transferred to the local processing element (PE) as soon as they arrive to their destinations.

3.2. Model Description

The model computes the mean message latency as follows. First, the mean network latency, $\overline{S}$, that is the time to cross the network is determined. Then, the mean waiting time seen by a message in the source node to be injected into the network, $\overline{W_s}$, is evaluated. Finally, to model the effect of virtual channels multiplexing, the mean message latency is scaled by a factor, $\overline{V}$, representing the average degree of virtual channels multiplexing that takes place at a given physical channel. Therefore, the mean message latency can be written as

$$Latency = (\overline{S} + \overline{W_s})\overline{V} . \quad (1)$$

In the 2-D torus network with a uniform traffic pattern, the average number of hops that a message traverses before reaching its destination is equal to $\overline{d} = k/2$ [8], where k is the number of nodes in each dimension. Consequently, the average rate at which messages enter nodes of the network is equal to $\overline{d}$ times the message generation rate. On the other hand, since the four output channels of each node are equally utilized, the arrival rate of messages to any network channel, denoted λ_c, is equal to $\overline{d}\lambda_g / 4$.

Since the torus interconnection network is symmetric, averaging the network latencies seen by the messages generated by only one node for all other nodes gives the mean message latency in the network. Let S_{ij} be the network latency of a message which should traverse i hops in dimension X and then j hop in dimension Y. The network latency, S_{ij}, seen by the message consists of two parts: one is the delay due to the actual message transmission time, and the other is due to the blocking time in the network. Therefore, S_{ij} can be written as

$$S_{ij} = (i+j)t_w + (i+j+1)(t_r + t_s) + Mt_w + iP_{B_x}\overline{W_x} + jP_{B_y}\overline{W_y} , \quad (2)$$

where M is the message length, P_{B_x} and P_{B_y} are the probabilities of a message being blocked at a hop of dimension X and Y, and $\overline{W_x}$ and $\overline{W_y}$ are the average waiting time for acquiring a virtual channel in dimension X and Y, respectively. Also t_r, t_s and t_w are the routing time of a message, crossing time of a flit over the crossbar switch and transfer time of a flit across a physical channel, respectively.

Averaging message latencies for all the possible destination nodes for a typical message yields the mean network latency as

$$\overline{S} = \sum_i \sum_j P_{ij} S_{ij} , \tag{3}$$

where P_{ij} is the probability of a message traverses i hops in dimension X and j hops in dimension Y. It is easy to see that $P_{ij} = N_{ij}/(k^2\text{-}1)$ where N_{ij} is the number of nodes that can be the destination of a $(i+j)$-hop message. Obviously when k is odd, N_{ij} maybe equal to 2 or 4 and when k is even N_{ij} maybe equal to 1, 2, or 4 for different values of i and j.

The XY routing algorithm results in a different service times on the channels of X and Y dimension. Therefore determination of the network latency starts at the ejection channel and works backward to the source channel of the message. In the steady state, the rate of messages that exit the network through ejection channels is equal to the injection rate of messages, which is equal to the generation rate λ_g. Utilization of the ejection channel (in each node) is therefore equal to $Mt_w\lambda_g$. Given that messages are of fixed length, there is no variance in service time. By using an M/G/1 queueing model [18], we can calculate the waiting time at an ejection channel as

$$\overline{W_{ej}} = \left(Mt_w\right)^2 \lambda_g / 2(1 - Mt_w\lambda_g) . \tag{4}$$

When a message needs to traverse one of the hops of dimension Y, it is delayed an average amount of time $\overline{W_y}$ before acquiring a virtual channel. The probability of v virtual channels of a hop in dimension Y being busy is denoted by $P_{y,v}$.

Considering the scheme used for virtual channel allocation which is described before, the probability of a message being blocked at a hop of dimension Y is given by $P_{B_y} = P_{y,V} + P_{y,V-1}/V$ in which two cases are considered. The first expression, $P_{y,V}$, corresponds to the case where all the virtual channels are busy and the second, to where all but v_1 or v_2 (the one not corresponding to the direction of the message) are busy. If a message is blocked at a hop, the message is delayed by as much time as it takes all the flits of a blocking message to finish traversing that hop. If none of the messages occupying the virtual channels terminate after traversing that hop, the blocked message will additionally be delayed by the average waiting time encountered by a blocking message in the rest of its path to its destination.

When a message passes a channel in dimension Y and reaches to a router, it may choose ejection channel with probability $P_{y\to ej} = 2/(\lfloor k/2 \rfloor + 1)$ for the next hop. Therefore, the probability of a message being blocked at a hop of dimension Y and none of the blocking messages terminating after traversing that hop, can be denoted as

$$P_{d_y} = P_{y\to ej}(1 - P_{y\to ej})^{V-1}\frac{P_{y,V}}{V} + (1 - P_{y\to ej})^{V} P_{y,V} + (1 - P_{y\to ej})^{V-1}\frac{P_{y,V-1}}{V} , \tag{5}$$

in which three cases are considered, each corresponding to one of the products. Enumerated from left to right, the first product corresponds to the case when V virtual channels are busy, but the message allocating one of them (v_1 or v_2, such that it can not be traversed by the blocked message) does terminate in the following node. The second and third cases correspond, respectively to when V and V-1 virtual channels are busy and none of the messages allocating these virtual channels terminate in the following node.

$\overline{W}_y$, the average waiting time of a blocked message to acquire a virtual channel at a hop of dimension Y, when it is considered that no other message is blocked at that hop, can be obtained as the product of the aggregate of the average waiting time of blocking messages in each of the

hops of the remainder of their paths, and the conditional probability that none of the blocking messages terminate after traversing the channel, given that the channel is already known to be blocked (resulting in P_{d_y} / P_{B_y}), plus the average length of a message times the channel cycle time, t_w (to account for the time it takes for the flits of a message to be transmitted over a single channel). This is expressed in the following equation:

$$\hat{W}_y = \frac{P_{d_y}}{P_{B_y}} P_{y \to y} \overline{W_y} + \overline{W_{ej}} + (M/2)t_w, \tag{6}$$

in which $\overline{W_y}$ is the average blocking time at a hop of dimension Y when it is considered that other messages may be blocked at the same hop and $P_{y \to y}$ is the probability of a message traversing dimension Y given that it has already traversed a hop in dimension Y. By considering the topology of torus network and deterministic routing algorithm, the value of $P_{y \to y}$ compute as $(\lfloor k/2 \rfloor - 1)/(\lfloor k/2 \rfloor + 1)$. If freed virtual channels are granted to blocked messages on a first-come-first-serve basis (which is usually the case), $\overline{W_y}$ can be calculated as

$$\overline{W_y} = \hat{W}_y \overline{N_y}, \tag{7}$$

in which $\overline{N_y}$ is the average number of waiting messages at a hop of dimension Y. Therefore, the meaning of this is that, the actual average blocking time at a channel of dimension Y is equal to the average waiting time of a blocked message to acquire a virtual channel at that hop when considering that no other message is blocked at that hop, times the average number of blocked messages at the channel. By considering Eq. (6) and (7) we can write

$$\overline{W_y} = \frac{\overline{W_{ej}} + (M/2)t_w}{1/\overline{N_y} - P_{d_y} P_{y \to y} / P_{B_y}}. \tag{8}$$

The average network latency of messages that traverse their first hop in dimension Y, excluding the blocking delay of the first hop, $\overline{S_y}$, is defined as the sum of the average blocking delay that messages face at the other hops of dimension Y, the transfer time of all the flits of a message over a channel (Mt_w) and the waiting time at the ejection channel $\overline{W_{ej}}$. Thus,

$$\overline{S_y} = P_{yf \to y} \overline{W_y} + \overline{W_{ej}} + Mt_w, \tag{9}$$

where $P_{yf \to y}$ is the probability of a message traversing dimension Y given that it has started its journey from dimension Y and equal to $1 - 1/\lfloor k/2 \rfloor$.

Similarly, the average service time of a message that traverses dimension X as its first route, $\overline{S_x}$, includes the blocking delay that the message faces at subsequent hops in dimension X and the ejection channel, and the actual transmission time. When a message passes a channel in dimension X and reaches to a router, it can choose one of three following paths for its next hop:

- ejection channel with probability $P_{x \to ej} = 2/\left(k\left(\lfloor k/2 \rfloor + 1\right)\right)$, (10)
- dimension Y with probability $P_{x \to y} = 2(k-1)/\left(k\left(\lfloor k/2 \rfloor + 1\right)\right)$, (11)
- dimension X with probability $P_{x \to x} = \left(\lfloor k/2 \rfloor - 1\right)/\left(\lfloor k/2 \rfloor + 1\right)$. (12)

By using the same approach used for dimension Y we have:

$$\hat{W}_x = \frac{P_{d_x}}{P_{B_x}} (P_{x \to x} \overline{W_x} + P_{x \to y} \overline{W_y}) + \overline{W_{ej}} + (M/2)t_w \tag{13}$$

for dimension X. Therefore,

$$\overline{W_x} = \frac{P_{d_x} P_{x\to y} \overline{W_y} / P_{B_x} + \overline{W_{ej}} + (M/2)t_w}{1/\overline{N_x} - P_{d_x} P_{x\to x} / P_{B_x}} \tag{14}$$

and

$$\overline{S_x} = P_{xf\to x} \overline{W_x} + P_{xf\to y} \overline{W_y} + \overline{W_{ej}} + Mt_w . \tag{15}$$

where $P_{xf\to x}$ is the probability of a message traverses dimension X if it has passed its first hop in dimension X and equal to $1-1/\lfloor k/2 \rfloor$ and $P_{xf\to y}$ is the probability of a message traverses dimension Y given that it has started its journey from dimension X and equal to $(k-1)/(k\lfloor k/2 \rfloor)$.

The probability $P_{y,v}$, that v $(0 \le v \le V)$ virtual channels are busy at a physical channel in dimension Y, can be determined using a Markovian model shown in Figure 2. State π_v, corresponds to v virtual channels being requested. The transition rate out of state π_v to state π_{v+1} is λ_c while the rate out of state π_v to state π_{v-1} is $1/\overline{S_y}$ ($\overline{S_y}$ is given by Eq. (9))

The probability that v virtual channels are busy, $0 \le v < V$, is the probability of being in state v, i.e. $P_{y,v}$=Pr(π_v). However, the probability that V virtual channels are busy is the summation of the probabilities of being in states v $(V \le v < \infty)$ i.e. $P_{y,V} = \sum_{k=V}^{\infty} \Pr(\pi_k)$. The steady-state solution of the Markovian model yields the probability $P_{y,v}$ to be

$$P_{y,v} = \begin{cases} (1-\lambda_c \overline{S_y})(\lambda_c \overline{S_y})^v, & 0 \le v < V, \\ (\lambda_c \overline{S_y})^v, & v = V. \end{cases} \tag{16}$$

In a similar manner $P_{x,v}$ can be determined by using a similar Markov chain while the rate out of state π_v to state π_{v-1} is $1/\overline{S_x}$.

The average number of waiting messages at a hop of dimension X and Y can be computed as

$$\overline{N_x} = \sum_{k=1}^{\infty} \Pr(\pi_{V+k}) = \frac{(\lambda_c \overline{S_x})^{V+1}}{1-\lambda_c \overline{S_x}}, \tag{17}$$

$$\overline{N_y} = \frac{(\lambda_c \overline{S_y})^{V+1}}{1-\lambda_c \overline{S_y}} . \tag{18}$$

A message originating from a given source node sees a network latency of $\overline{S}$ (given by Eq. (3)). Modeling the local queue in the source node as an M/G/1 queue, with the mean arrival rate λ_g / V (recalling that a message in the source node can enter the network through any of the V virtual channels) and service time $\overline{S}$ with an approximated variance $\left(\overline{S} - M\right)^2$ yields the mean waiting time seen by a message at source node as [8]

$$\overline{W_s} = \frac{(\lambda_g / V)\left(\overline{S}^2 + \left(\overline{S} - M\right)^2\right)}{2\left(1-(\lambda_g / V)\overline{S}\right)} . \tag{19}$$

When multiple virtual channels are used per physical channel they share the bandwidth in a time multiplexed manner. The average degree of multiplexing of virtual channels that take place at a physical channel in dimension X, can be estimated by [18] $\overline{V_x} = \left(\sum_{v=1}^{V} v^2 P_{x,v}\right) / \left(\sum_{v=1}^{V} v P_{x,v}\right)$ and similarly for dimension Y is $\overline{V_y} = \left(\sum_{v=1}^{V} v^2 P_{y,v}\right) / \left(\sum_{v=1}^{V} v P_{y,v}\right)$. Therefore, the average degree of virtual channel multiplexing for a physical channel becomes $\overline{V} = (\overline{V_x} + \overline{V_y})/2$.

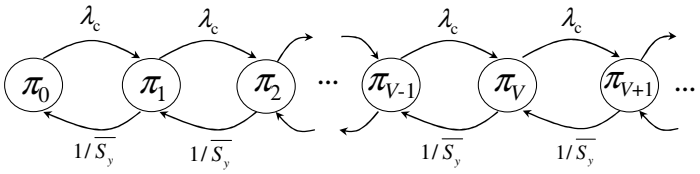


Figure 2: Markov process for occupying and releasing virtual channels associated with a physical channel at dimension *Y*.

4. Router Power Model

In this section, we propose a simple high level approach for modeling the power consumption of torus network for different values of basic NoC parameters. Our proposed model considers both dynamic and static power consumption. Static power is related to the leakage current of transistors in steady state, while dynamic power is related to *switching activity* (the changing of internal logic levels) and the associated charging of internal load capacitances.

There are two different types for the case of energy consumption in an NoC router. First, the router dissipates energy during message movement and second energy dissipation of a quiet router. Quiet router is a router with no messages or blocked messages. In other words, when there is not any transition in the router, we have a quiet router. Therefore the dissipated energy in the quiet network is only static energy while message movement dissipates both static and dynamic energy. Let P_s and P_d be the static and dynamic power dissipated in a buffer for one flit data. In torus network there are k^2 routers and each router has 5 input channels (4 channels from neighbor routers and one injection channel) and each channel has V buffers (virtual channels). Therefore, static power consumption of all routers is $5k^2VP_s$. Since a message has M flits and on average, a message traverses $\overline{d}+1$ routers to reach its destination, the dynamic power consumption of one message is $M(\overline{d}+1)P_d$. Also according to the assumptions given in Section 3.1., on average, λ_g messages reaches to a router in one cycle. Therefore dynamic power consumption of all messages is $\lambda_g M(\overline{d}+1)P_d$. Note that due to the limitation of routing resources (switches and interconnect wires), accepted traffic will saturate at a certain value of the injection load λ_s where λ_s is the peak data rate sustainable by the network which has found by the Markovian model described in previous section. Now we can model total power consumption as

$$P_{total} = \begin{cases} 5k^2VP_s + M(\overline{d}+1)P_d\lambda_g, & \lambda_g < \lambda_s, \\ 5k^2VP_s + M(\overline{d}+1)P_d\lambda_s, & \lambda_g \geq \lambda_s. \end{cases} \quad (20)$$

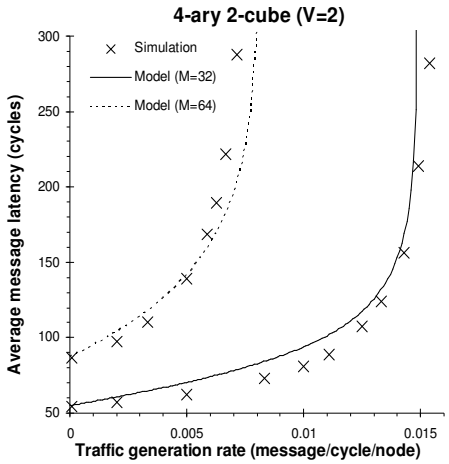
This simple analytical model shows that power consumption has a linear relation to the packet injection rate. P_s and P_d are constant and depend on system voltage, frequency and fabrication technology. These power numbers are obtained from the synthesized VHDL designs.

5. Validation of the Model

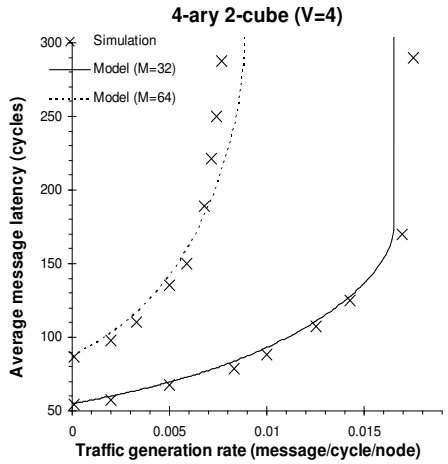
The proposed analytical model has been validated through a VHDL-based cycle accurate simulator. To achieve a high accuracy in the simulation results, we use the batch means method [21] for simulation output analysis. There are 10 batches and each batch includes up to 70000 messages depending on the traffic injection rate and network size. Statistics gathering was inhibited for the first batch to avoid distortions due to the startup transient. The standard deviation of latency measurements is less than 2% of the mean value. The simulator uses the same assumptions as the analysis. Numerous validation experiments have been performed for

several combinations of network sizes, message lengths, and number of virtual channels to validate the model.

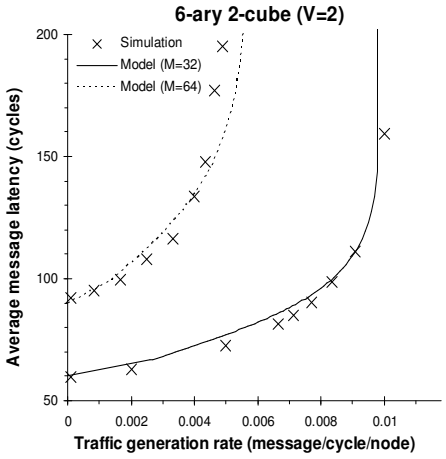
Figure 3 depicts latency results predicted by the model explained in the previous section, plotted against those provided by the simulator for the 4X4 and 6X6 torus NoCs with V=2 and 4 virtual channels per physical channel, and two different message lengths M=32 and 64 flits. The horizontal axis in the figure shows the traffic generation rate at each node while the vertical axis shows the mean message latency. The figures reveal that in all cases the analytical model predicts the mean message latency with high degree of accuracy in the steady state regions. Moreover, the model predictions are still good even when the network operates in the heavy traffic region, and when it starts to approach the saturation region. However, some discrepancies around the saturation point are apparent. These can be accounted for by the approximations made to ease the derivation of different variables, e.g. the approximation made to estimate the variance of the service time distribution at an injection channel (Eq. (19)). Such an approximation greatly simplifies the model as it allows us to avoid computing the exact distribution of the message service time at a given channel, which is not a straightforward task due to interdependencies between service times at successive channels as wormhole routing relies on a blocking mechanism for flow control.



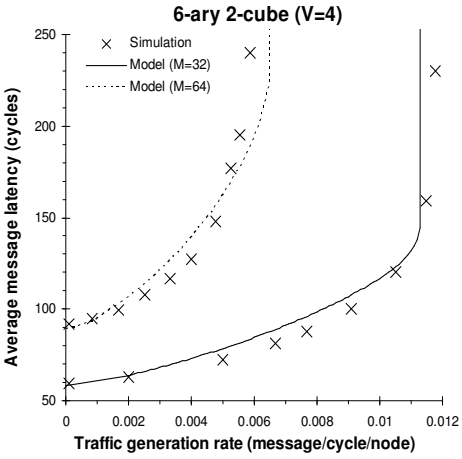
(a) 4X4 torus with 2 virtual channels



(b) 4X4 torus with 4 virtual channels



(c) 6X6 torus with 2 virtual channels



(d) 6X6 torus with 4 virtual channels

Figure 3: The average message latency predicted by the model against simulation results for a 4X4 and a 6X6 torus NoC with V=2 and 4 virtual channels and messages length M=32 and 64 flits.

Power consumption of each router is determined by modeling the design in VHDL and using Synopsys Power Compiler [2]. This tool does not include the long wires between the logic blocks. Figure 4 compares power consumption results from the analytical model and simulation experiments. The simulation environment is the same as in Section 3.1. As can be seen in the figure, the results from the proposed analytical model closely match the experimental ones.

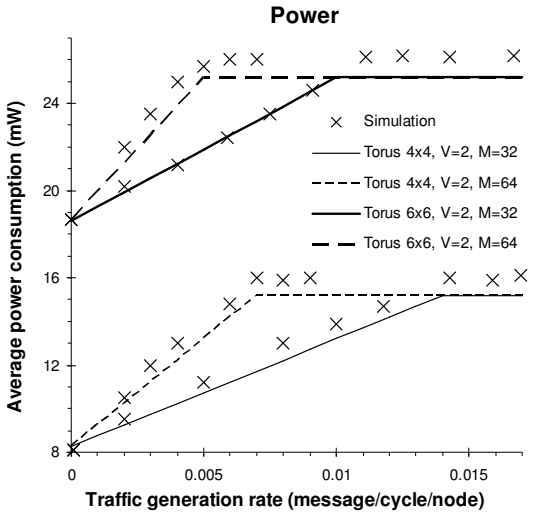


Figure 4: Total power consumption of routers in analytical model and simulation

6. Conclusion and Future Work

In this paper, we first proposed an analytical model to predict the average message latency of a wormhole-switched 2D torus NoC with deterministic routing. Also a simple high level model for predicting the power consumption of routers in NoCs was presented. Both models capture the effect of virtual channel multiplexing on the average message latency and power consumption. Simulation experiments have revealed that the proposed models predict message latency and power consumption of on-chip torus networks quite accurate.

We plan to advance this research in several directions. One possible direction is to extend this approach to the other network-on-chip topologies. Another important extension is to accommodate this analytical model with various channel buffer depth. We are also working on a G/G/1 queueing model (instead of M/G/1 model) which is applicable to NoCs with arbitrary arrival process and arbitrary traffic pattern. The main challenge comes from the difficulty involved in the calculation of the packets interarrival time for each channel.

Refrences

[1] International Technology Roadmap for Semiconductors (ITRS), 2005 edition, http://www.itrs.net/.

[2] Synopsys Power Compiler User Guide, Synopsys Inc., 2005, http://www.synopsys.com/.

[3] L. Benini and G. De Micheli, "Networks on Chips: A New SoC Paradigm," *IEEE Computer*, pp. 70–78, 2002.

[4] X. Chen and L. Peh, "Leakage Power Modeling and Optimization in Interconnection Networks," *In Proceedings of the International Symposium on Low Power Electronics and Design*, pp. 90–95, 2003.

[5] G. Chiu, "The Odd-Even Turn Model for Adaptive Routing," *IEEE Transaction on Parallel and Distributed Systems*, Vol. 11, No. 7, pp. 729–738, 2000.

[6] W.J. Dally and B. Towles, "Route Packets, Not Wires: On-Chip Interconnection Networks," *In Proceedings of the Design Automation Conference*, pp. 683–689, 2001.

[7] W.J. Dally, "Performance Analysis of *k*-ary *n*-cube Interconnection Networks," *IEEE Transactions on Computers*, Vol. 39, No. 6, pp. 775–785, 1990.

[8] J.T. Draper and J. Ghosh, "A Comprehensive Analytical Model for Wormhole Routing in Multicomputer systems," *Journal of Parallel and Distributed Computing,* Vol. 23, No. 2, pp. 202–214, 1994.

[9] Duato J., Yalamanchili S., and Ni L., *Interconnection Networks: An Engineering Approach*, Morgan Kaufmann, 2002.

[10] C.J. Glass and L.M. Ni, "The Turn Model for Adaptive Routing," In Proceedings of the International Symposium on Computer Architecture, pp. 441–450, 1998.

[11] W. Guan, W. Tsai, and D. Blough, "An Analytical Model for Wormhole Routing in Multicomputer Interconnection Networks," *In Proceedings of International Parallel Processing Symposium*, pp. 650–654, 1993.

[12] Z. Guz, I. Walter, E. Bolotin, I. Cidon, R. Ginosar, and A. Kolodny, "Efficient Link Capacity and QoS Design for Network-on-Chip," *Design, Automation, and Test in Europe,* pp. 9–14, 2006.

[13] J. Hu and R. Marculescu, "Energy- and performance-aware mapping for regular NoC architectures," *IEEE Transaction on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 24, No. 4, pp. 551–562, 2005.

[14] J. Hu, U.Y. Ogras, and R. Marculescu, "System-Level Buffer Allocation for Application-Specific Networks-on-Chip Router Design," *IEEE Transaction on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 25, No. 12, pp. 2919-2933, 2006.

[15] P. Hu and L. Kleinrock, "An Analytical Model for Wormhole Routing with Finite Size Input Buffers," *In Proceedings of the International Teletraffic Congress*, 1997.

[16] J. Kim, D. Park, C. Nicopoulos, N. Vijaykrishnan, and C.R. Das, "Design and Analysis of an NoC Architecture from Performance, Reliability and Energy Perspective," *In Proceedings of the Symposium on Architecture for Networking and Communications Systems*, pp. 173–182, 2005.

[17] S. Murali and G. De Micheli, "Bandwidth-constrained Mapping of Cores onto NoC Architectures," *In Proceedings of the conference on Design, Automation and Test in Europe,* Vol. 2, pp. 896–901, 2004.

[18] H.H. Najaf-abadi and H. Sarbazi-Azad, "An Accurate Combinatorial Model for Performance Prediction of Deterministic Wormhole Routing in Torus Multicomputer Systems," *In Proceedings of the International Conference on Computer Design*, pp. 548–553, 2004.

[19] U.Y. Ogras and R. Marculescu, "'It's a small world after all': NoC performance optimization via long-range link insertion," *IEEE Transaction on VLSI*, Vol. 14, No. 7, pp. 693 – 706, 2006.

[20] P.P. Pande, C. Grecu, M. Jones, A. Ivanov, and R. Saleh, "Performance Evaluation and Design trade-offs for Network-on-Chip Interconnect Architectures," *IEEE Transactions on Computers*, Vol. 54, No. 8, pp. 1025 – 1040, 2005.

[21] K. Pawlikowski, "Steady-State Simulation of Queueing Processes: A Survey of Problems and Solutions," *ACM Computing Surveys*, Vol. 22, No. 2, pp. 123–170, 1990.

[22] H. Sarbazi-Azad, M. Ould-Khaoua, and L.M. Mackenzie, "Analytical Modeling of Wormhole-Routed *k*-Ary *n*-Cubes in the Presence of Hot-Spot Traffic," *IEEE Transaction on Computers*, Vol. 50, No. 7, pp 623-634, 2001.

[23] H. Sarbazi-Azad, M. Ould-Khaoua, and L. M. Mackenzie, "Communication Delay in Hypercubes in the Presence of Bit-Reversal Traffic," Parallel Computing, Vol. 27, No. 13, pp. 1801-1816, 2001.

PERMAP: A Performance-Aware Mapping for Application-Specific SoCs

Paper 3

Abbas Eslami Kiasari
Shaahin Hessabi
Hamid Sarbazi-Azad

In the Proceedings of the 19th IEEE International Conference on Application-specific Systems, Architectures and Processors (ASAP), pp. 73-78, Leuven, Belgium, Jul. 2008.

PERMAP: A Performance-Aware Mapping for Application-Specific SoCs

A. E. Kiasari[†,‡], S. Hessabi[†], and H. Sarbazi-Azad[†,‡]
[†]Sharif University of Technology
Tehran, Iran
[‡]IPM School of Computer Science
Tehran, Iran
kiasari@ipm.ir, hessabi@sharif.edu, azad@ipm.ir

Abstract
Future System-on-Chip (SoC) designs will need efficient on-chip communication architectures that can provide efficient and scalable data transport among the Intellectual Properties (IPs). Designing and optimizing SoCs is an increasingly difficult task due to the size and complexity of the SoC design space, high cost of detailed simulation, and several constraints that the design must satisfy. For efficient design of SoCs, an efficient mapping of IPs onto Networks-on-Chip (NoCs) is highly desirable. Towards this end, we have presented PERMAP, a PERformance-aware MAPping algorithm which maps the IPs onto a generic NoC architecture such that the average communication delay is minimized. This is accomplished by a performance analytical model which can be used for any arbitrary network topology with wormhole routing. The algorithm is used for mapping a video application onto a tile-based NoC and experimental results show that PERMAP is fast and robust.

1. Introduction

The International Technology Roadmap for Semiconductors (ITRS) predicts that chips with over 4 billion transistors operating at 10 GHz speeds will be commercialized before the end of the decade [1]. With the advance of the semiconductor technology, the enormous number of transistors available on a single chip allows designers to integrate dozens of IP (Intellectual Property) blocks together with large amounts of embedded memory. Such IPs can be CPU or DSP cores, video stream processors, high-bandwidth I/O controllers, etc.

Shared-medium busses do not scale well, and do not fully utilize potentially available bandwidth. As the feature sizes shrink, and the overall chip size relatively increases, the interconnects start behaving as lossy transmission lines. Line delays have become very long as compared to gate delays, causing synchronization problems between cores. A significant amount of power is dissipated on long interconnects and in clocking network. This trend only worsens as the clock frequencies increase and the feature sizes decrease. Lowering the power supply voltage and designing low swing circuits decrease the overall power consumption at the cost of higher data errors.

One solution to these problems is to implement System-on-Chip (SoC) using micro-networks, or Networks-on-Chip (NoCs) [2][4][8]. Networks have a much higher bandwidth due to multiple concurrent connections. Regularity enables design modularity, which in turn provides a standard interface for easier component reuse and better interoperability. Overall performance and scalability increase since the networking resources are shared. Scheduling of traffic on shared resources prevents latency increases on critical signals.

To exploit this regular architecture, the design flow needs the following steps: First, the application needs to be divided into a graph of concurrent tasks. The task may be, for example, a software task to be executed in an embedded processor, a hardware task to be executed in an embedded FPGA or a co-processor, or an input/output operation. This task graph is a directed graph, where each vertex represents one selected IP, and each directed arc represents the communication volume from source IP to destination IP. Then, the designer needs to decide to

which network node each selected IP should be mapped such that the metrics of interest are optimized. More precisely, given the assigned task graph, this last phase determines the topological placement of these IPs onto different network nodes. The feasible placements are evaluated according to a specific cost function, as area fragmentation, power consumption, packet latency or link usage, in order to get the best performance.

The mapping phase (that is, the topological placement of the IPs onto the on-chip tiles) represents a problem, especially in the context of the regular architecture, as it significantly impacts the performance metrics of the system. The overall objective of this research is mapping IPs on to a network architecture. In particular, we consider mesh topology, and the mapping of IPs to their cross-points. We describe a mapping scheme called PERMAP (PERformance-aware MAPping) that minimizes the average communication delay.

To this end, we first show the impact of IPs mapping on the communication performance of a given system. An efficient analytical model is then proposed to predict the communication performance of an SoC. Finally, the proposed model is used as a powerful tool with the goal of finding a legal mapping which has an acceptable performance.

2. Related Work

Hu and Marculescu [9] presented a static mapping heuristic. The main goal of the approach is to reduce the overall power consumption by decreasing the consumed energy on communication. The authors proposed a mapping approach named Communication Weighted Model (CWM), modeling applications as graphs, where the vertices are the tasks and the edges are the communications between tasks. The weight of each edge corresponds to the number of bits exchanged between tasks.

Another mapping algorithm is presented in [11] for satisfying the bandwidth constraints of a mesh NoC and minimizing the average delay. The average hop count is used to approximate the average packet latency. This metric, however, ignores the queuing delays and network contention.

Authors in [10] present an approach that uses a genetic algorithm to map an application, described as a parameterized task graph, on a mesh-based NoC architecture so as to minimize the execution time.

3. Motivation

In 2006, the National Science Foundation initiated a workshop to identify the on-chip communication challenges. Workshop members did agree that latency and power are the two most critical crosscutting design challenges for on-chip interconnection network architectures [12].

To show that the IP mapping heavily affects the communication latency, we consider two different IP mapping of a video application to the tiles of a 4X4 mesh on-chip network. As an example of a video processing application, the task graph of a Video Object Plane (VOP) decoder [15] is shown in Figure 1. Each block in the figure corresponds to an IP and the numbers near the edges represent the bandwidth (in MBytes/sec) of the data transfer, for a 30 frames/sec MPEG-4 movie with 1920×1088 resolution [15]. Then, the system is simulated for these mapping configurations and the corresponding average packet latency (APL) values are plotted against packet generation

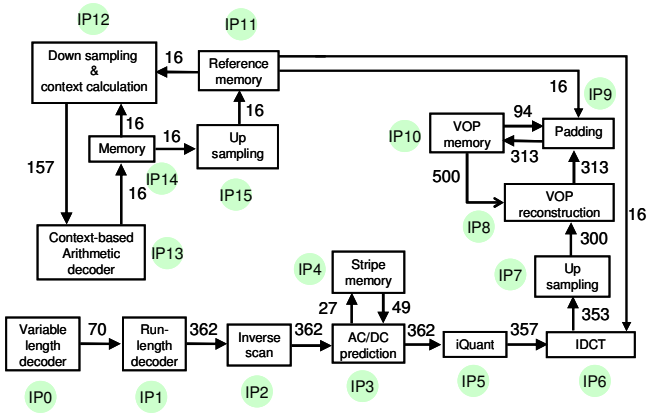


Figure 1: Task graph of a Video Object Plane (VOP) decoder [15].

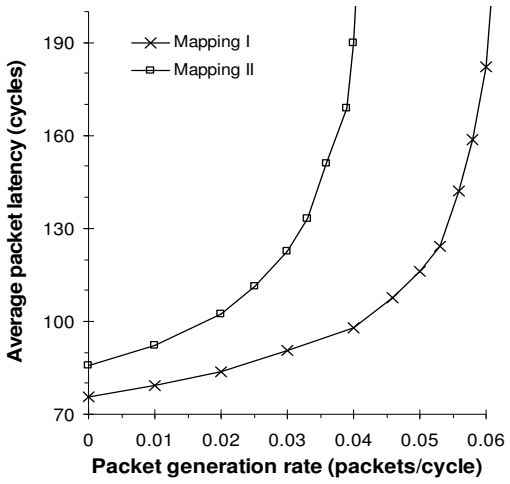


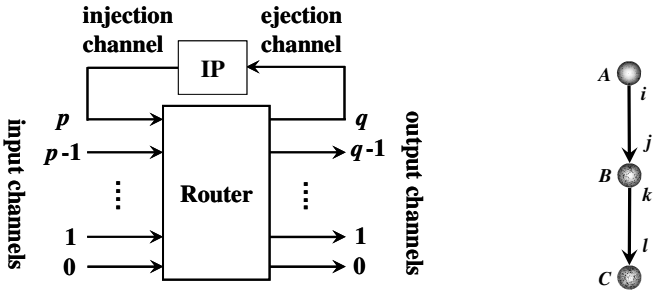
Figure 2: Average packet latency (APL) of a video application for two different mapping configurations vs. packet generation rate.

rate in Figure 2. As can be seen in the figure, the communication performance of a system is dependent on how the IPs of the task graph are assigned to the tiles of the network.

Unfortunately, the mapping problem is *NP-hard* [6]. The search space of the problem increases factorially with the system size. Even for a system with 4X4 tiles, there can be 16! ($\approx 2x10^{13}$) mappings which are almost impossible to enumerate. In the following section, we propose an efficient analytical model which can be used to find nearly optimal solutions in reasonable time.

4. Performance Analysis

If the performance is measured in terms of average packet latency, then maximizing the performance means, in fact, minimizing the end-to-end packet latency. In this section, we derive an analytical performance model for on-chip networks using a G/G/1 [3] priority queueing model. It can be used for any arbitrary network topology with wormhole routing under any arbitrary traffic pattern.



(a) (b)

Figure 3: (a) A general structure for a node in an SoC (b) A two hop path from node A to node C.

4.1. Assumptions and Notations

We consider input buffered routers with $p+1$ input channels, $q+1$ output channels, and target wormhole flow control under deterministic routing algorithms. The structure of a single node is depicted in Figure 3.a. Each node contains a router and an IP capable of generating and/or receiving packets.

Packets are injected into the network on input port p (injection channel) and leave the network from output port q (ejection channel). Generally, each channel connects output port j of node N to input port i of node M. So, we denote this channel OC_j^N (jth output channel of router N) or IC_i^M (ith input channel of router M). Messages are broken into some packets of fixed length of M flits. The routing decision delay for a packet, crossing time of a flit over the crossbar switch, and transfer time of a flit across a wire between two neighboring routers are t_r, t_s, and t_w, respectively. Also the transfer time of a flit across the injection and ejection channels are considered to be t_w.

Let $P^{S\to D}$ be the probability of the packet transmitted from the source node at router S (R^S) to the destination node at router D (R^D). Likewise, the traffic arrival rate of the header flits from IC_i^N to OC_j^N is given by $\lambda_{i\to j}^N$ packets/cycle. Also we assume that the packet injection process to the router R^N has a general distribution with mean value of α^N packets/cycle. The average packet latency (L) is used as the performance metric. We assume that the packet latency spans the instant when the packet is created, to the time when the packet is delivered to the destination node, including the queuing time spent at the source. We also assume that the packets are consumed immediately once they reach their destination nodes.

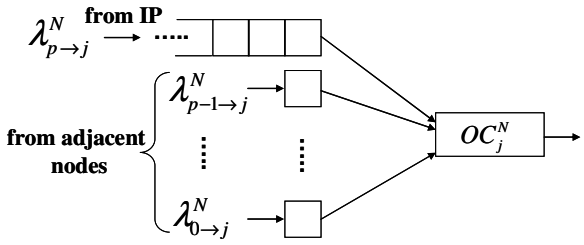


Figure 4: Queueing model of a channel of an arbitrary topology.

4.2. Analytical Model

In Figure 3.b consider a packet which is generated in IP^A, and reaches its destination (IP^C) after traversing R^A, R^B, and R^C. The latency of this packet ($L^{A\to C}$) consists of two parts: the latency of header flit ($L_h^{A\to C}$) and the latency of body flits (L_b). In other words

$$L^{A\to C} = L_h^{A\to C} + L_b \tag{1}$$

$L_h^{A\to C}$ is the time from when the packet is created in IP^A, until when the header flit is reached to the IP^C, including the queueing time spent at the source node and intermediate nodes. In Figure 3.b, $L_h^{A\to C}$ can be computed as:

$$L_h^{A\to C} = 3\left(t_r + t_s\right) + 4t_w + W_{p\to i}^{A} + W_{j\to k}^{B} + W_{l\to q}^{C}$$

where $W_{i\to j}^{N}$ is the mean waiting time for a packet from IC_i^N to OC_j^N. Since the body flits follow the header flit in a pipelined fashion, L_b is given by $L_b = (M-1)(t_s + t_w)$. The only unknown parameter for computing the latency is $W_{i\to j}^{N}$. The value of $W_{i\to j}^{N}$ can be calculated in a straightforward manner using a queuing model. The basic element in the model is a G/G/1 priority queue (the customer interarrival time and server's service time follow general distributions and queues have one server to provide the service). A router is primarily modeled based on nonpreemptive priority queuing system 11.

Now, let us consider, for instance, the *j*th output channel of R^N (OC_j^N). As can be seen in Figure 4, this channel is modeled as a server in a priority queueing system with $p+1$ classes (IC_0^N to IC_p^N), the arrival rate $\lambda_{i\to j}^{N}$ $(0 \le i \le p)$, served by one server (OC_j^N) of service rate μ_j^N. Note that since all incoming packets are similar, the service times of all packets are equal. Both interarrival and service times are independent and identically distributed with arbitrary distributions.

Since the input channels (except injection channel) have one flit rooms, we should compute the average waiting time for the head of class *i*. Using a technique similar to that employed in literature for priority queues [3]11, we can write

$$W_{i\to j}^{N} = \begin{cases} \overline{R_j^N} \,/\left(1-\rho_{p\to j}^{N}\right), & i = p, \\ \dfrac{1+\rho_{i+1\to j}^{N} - \sigma_{i+2\to j}^{N}}{1-\sigma_{i+1\to j}^{N}} W_{i+1\to j}^{N}, & 0 \le i < p. \end{cases} \tag{2}$$

where $\overline{R_j^N}$ is the residual service time of OC_j^N seen by an incoming header flit and $\sigma_{i\to j}^{N} = \sum_{k=i}^{p+1} \rho_{i\to j}^{N}$. In a G/G/1 queueing system, $\overline{R_j^N}$ is approximated by [3]

$$\overline{R_j^N} \approx \sum_{i=0}^{p} \rho_{i\to j}^{N} \frac{C_{A_{i\to j}^N}^2 + C_{B_j^N}^2}{2\mu_j^N} \tag{3}$$

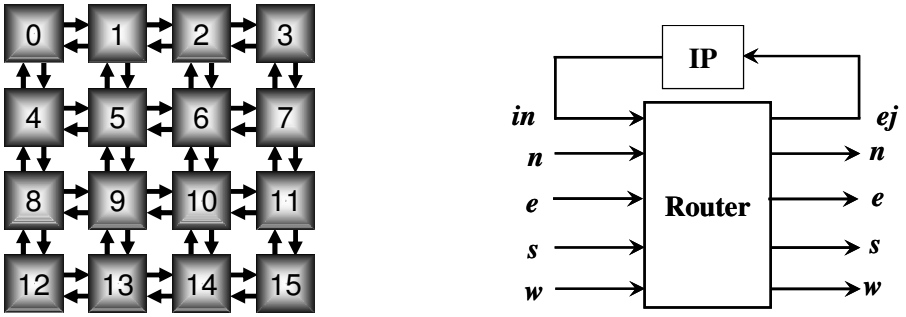
Since we do not have enough insight about the first and second moments of interarrival time, we suppose that $C_{A_{i\to j}^N}$ is constant for all input channels in the network and equal to the coefficient of variation (CV) of the arrival process to network ($C_{A_{i\to j}^N} = C_A$). So, we can rewrite Eq. (3) as:

$$\overline{R_j^N} \approx \frac{1}{2}\lambda_j^N \left(b_j^N\right)^2 \left(C_A^2 + C_{B_j^N}^2\right) \tag{4}$$

Therefore, to compute $W_{i\to j}^{N}$ we must calculate the average arrival rate over OC_j^N (λ_j^N), and also first and second moments of the service time of OC_j^N. Assuming the network is not overloaded, the arrival rate over OC_j^N can be calculated using the following general equation:

$$\lambda_j^N = \sum_{\forall S}\sum_{\forall D} \alpha^S \times P^{S\to D} \times \Re\left(S \to D, OC_j^N\right) \quad (5)$$

In Eq. (5), the routing function $\Re\left(S \to D, OC_j^N\right)$ equals 1 if the packet from IP^S to IP^D passes through OC_j^N; it equals 0 otherwise. Note that we assume a deterministic routing algorithm, thus the function of $\Re\left(S \to D, OC_j^N\right)$ can be predetermined.



(a) (b)

Figure 5: An SoC with (a) 4X4 mesh network and (b) its router structure.

Although λ_j^N can be computed exactly for all topologies by Eq. (5), service time moments of the output channels cannot be computed in a direct manner by a general formula for any topology and any routing algorithm. To compute the moments of the service time of the output channels we first divide the channels into some groups based on their routing order and then an index is assigned to the groups opposite of the routing order, from ejection channel to injection channel. Then, we estimate the first two moments of the service time for the output channels. Determination of the channel service time moments starts at the ejection channel and works backward to the source of the packets. Therefore, the contention delay from lower numbered groups can then be thought of as adding to the service time of packets on higher numbered groups. In other words, to determine the waiting time of channels in group *k,* we must calculate the waiting time of all channels in group $k-1$. This approach is dependent to the topology and routing algorithm. Here we derive an analytical performance model for XY routing [7] in a mesh network. Due to the popularity of the mesh network, our analysis focuses on this topology but the modeling approach used here can be equally applied for other topologies after few changes in the model.

We consider a system which is composed of 4X4 tiles interconnected by a 2D mesh network as shown in Fig 5.a. Also a typical on-chip router for a 2D mesh is shown in Figure 5.b. The five input channels of each router are represented with *in* (injection), *n* (north), *e* (east), *s* (south), and *w* (west). Also these channels are assigned to priority classes from index 5 (the highest priority) to 1 (the lowest priority), respectively. *ej* is representing the *ejection* channel.

In XY routing, the injected packets into the network traverse in the row of the source up to the column of the destination, and then go straight through to the destination. Finally, it is fed to the destination node via ejection channel. We divide the output channels of the mesh network into 4 groups based on their routing order: (1) ejection channels, (2) north and south channels, (3) east and west channels, and (4) injection channels. In the ejection channel of R^N the header flit and body flits are accepted in t_w and L_b cycles, respectively. So, we can write $\overline{b_{ej}^N} = t_w + L_b$ and since all

packets have the same service time, there is not any variation in the service times. In other words $C_{B_{ej}^N} = 0$. Now, we can determine the value of $W_{i\to ej}^N$ where $i \in \{n,e,s,w\}$.

After determination of the service time moments in group 1, in the next step, we should determine the first two moments of service time of channels in group 2, north and south channels. The moments of the service time for a north output channel is obtained by tracing each of paths from this channel to the network outputs (ejection channels). The service time along each path is a random variable; however, each is only a fraction of the packet length for all packet rates that can be sustained on the mesh network. Hence, the service time through a path is assumed to be a constant equal to the sum of the mean waiting times along the path plus L_b. Since each of these paths is not equally probable, the service time moments are weighted mean of all paths service times.

For example, consider the north output channel of R^9 (OC_n^9) in Figure 5.a. One possible path from OC_n^9 to a network output is directly to ejection channel of the adjacent node R^5, for an average service time of $L_5 = t_r + t_s + 2t_w + W_{s\to ej}^5 + L_b$ where $W_{s\to ej}^5$ was already computed. The second path from OC_n^9 to a network output is through north channel of R^5 and ejection channel of R^1, for an average service time of $L_1 = 2(t_r + t_s) + 3t_w + W_{s\to n}^5 + W_{s\to ej}^1 + L_b$ where $W_{s\to ej}^1$ was already computed and $W_{s\to n}^5$ can be computed with the same approach. Packets are routed under XY routing algorithm; therefore, all passing packet through OC_n^9 are generated in IP^8, IP^9, … IP^{15} and destined to IP^1 and IP^5 with the probability of $\alpha = \sum_{i=8}^{15} P^{i\to 1} / \left(\sum_{i=8}^{15} P^{i\to 1} + \sum_{i=8}^{15} P^{i\to 5}\right)$ and $1-\alpha$, respectively. So, the first and second moments of the service time can be estimated as $\overline{b_n^9} = \alpha L_5 + (1-\alpha) L_1$, and $\overline{(b_n^9)^2} = \alpha L_5^2 + (1-\alpha) L_1^2$, respectively. Now, we are able to calculate the coefficient of variation of the service time in OC_n^9, $C_{B_n^9}^2 = \overline{(b_n^9)^2} / \left(\overline{b_n^9}\right)^2 - 1$, and then the mean waiting time for the north channel of R^9 seen by other channels ($W_{i\to n}^9$) can be computed by Eq. (2). After computing the mean waiting time of all channels in group 2, the mean waiting time for other output channels (east, west and ejection channels) can be calculated using the same approach. Now, we can calculate the packet latency between any two nodes in the network by Eq. (1). The average packet latency is the weighted mean of these latencies as:

$$L = \sum_{s=0}^{15} \sum_{d=0}^{15} P^{s\to d} \times L^{s\to d} \qquad (6)$$

5. Experimental Results

This section reports on the accuracy and run time of the proposed approach. To evaluate the capability of our method for real applications, we applied it to the VOP decoder application [15] which is mapped to a 4×4 2D mesh network. We used two-state Markov Models [5] as stochastic traffic generators to model the bursty nature of the application traffic, with average communication bandwidth matching the applications' average communication bandwidth (shown in Figure 1). The average packet latencies obtained using the proposed method are compared against those obtained with a cycle-accurate flit-level simulator. Both the simulator and the analytical model are implemented in C++. Throughout the experiments, we consider an SoC with 128 byte packets, 32 bit flits, 4 cycle router delay (routing decision and switching delay are 3 cycles and 1 cycle, respectively), and 1 cycle wire delay. To achieve a high accuracy in the simulation results, we use the batch means method [13] for simulation output analysis. There are 10 batches and each batch includes up to 20,000 packets, depending on the traffic injection rate and network size. Statistics gathering was inhibited for the first batch to avoid distortions due to the startup transient. The standard deviation of latency measurements is less than 2% of the mean value.

Next, we assess the accuracy of our approach for different application mappings. We performed experiments for 1000 random mappings. For each mapping, the average packet latency is computed by using the proposed approach and by simulation, at 0.03 packets/cycle injection rate (see Figure 2). According to the simulation results, the best among all 1000 mappings is the mapping with ID 534, with an average latency of 88.81 cycles. The analytical model however reports the best mapping to be ID 264 with average latency of 97.24 cycles. The latency for mapping ID 264 found by simulation is 90.73 cycles. As such, the analysis approach selects a mapping whose latency is within 3% of the best one found by simulation. But, the analysis finds the best mapping more than 100,000 times faster. Thus, much more mappings can be explored within the same time budget using the proposed analytical technique.

Table 1: Minimum APL of some random mapping found by analytical model and corresponding APL obtained using simulation.

Number of random vectors	APL (Model)	APL (Sim.)	Error (%)	Model run time (sec)
10	115.08	125.96	8.64	≈ 0
100	98.52	97.50	1.05	0.015
1000	97.24	90.73	7.18	0.14
10,000	90.99	90.60	0.43	1.25
100,000	89.86	88.68	1.33	14.17
1,000,000	88.20	88.24	0.05	242.20

We used our analytical method to find the APL of 10 to one million different mappings and selected the best mapping (minimum APL among all mapping configurations). Then the APL of this mapping is calculated by simulation. Table 1 shows the APL of the best mapping which is found by our proposed approach and corresponding APL obtained using simulation. Also the relative error and analytical model run time of each set of random vectors are reported in Table 1. The best mapping configuration among 1 million different mapping vectors is shown in Figure 6.

Therefore, the proposed method can be used to prune the large design space in a very short time compared to simulation. Experiments performed on larger networks show several orders of magnitude achievable speed-up compared to a single simulation run. Considering that many simulations are needed to obtain high confidence intervals, the overall speed-up of the analytical approach is impressive. Moreover, the simulation runtime grows faster for heavier traffic, while the run-time of the analytical approach remains pretty much the same.

6. Conclusions

In this paper, we addressed the mapping problem for application specific SoC architectures. An efficient queueing-based model is used for performance prediction of an SoC. PERMAP uses this analytical model to map the IPs onto a generic NoC architecture such that the average communication delay is minimized. The model is used for mapping a video application onto tile-based NoCs. Experimental results show that we can generate high quality solutions with significantly less computational time. Although in this paper we focused on the tile-based architecture interconnected by a 2D mesh network with XY routing, our method can be adapted to other network topologies and routing schemes.

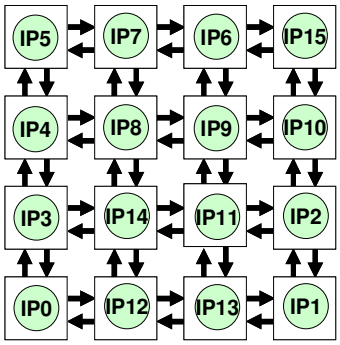


Figure 6: An efficient mapping of the VOP decoder application which is found by the analytical model.

References

[1] International Technology Roadmap for Semiconductors (ITRS), 2007 edition, http://www.itrs.net/.

[2] L. Benini and G. De Micheli, "Networks on Chips: A New SoC Paradigm", *IEEE Computer*, 2002, pp. 70–78.

[3] G. Bolch, S. Greiner, H. De Meer, and K.S. Trivedi, *Queueing Networks and Markov Chains: Modeling and Performance Evaluation with Computer Science Applications*, 2nd Edition, John Wiley and Sons, 2006.

[4] W.J. Dally and B. Towles, "Route Packets, Not Wires: On-Chip Interconnection Networks", *In Proceedings of the DAC*, 2001, pp. 683–689.

[5] W. Fischer and K. Meier-Hellstern, "The Markov-Modulated Poisson Process (MMPP) Cookbook", *Performance Evaluation*, Vol. 18, No. 2, 1993, pp. 149-171.

[6] M.R. Garey and D.S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, Freeman, 1979.

[7] C.J. Glass and L.M. Ni, "The Turn Model for Adaptive Routing", *In proceedings of the ISCA*, 1992, pp. 278-287.

[8] P. Guerrier and A. Greiner, "A Generic Architecture for on-chip Packet-Switched Interconnections", *In proceedings of the DATE*, 2000, pp. 250-256.

[9] J. Hu and R. Marculescu, "Energy- and Performance-Aware Mapping for Regular NoC Architectures", *IEEE Transaction on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 24, No. 4, 2005, pp. 551-562.

[10] T. Lei and S. Kumar, "A Two-step Genetic Algorithm for Mapping Task Graphs to a Network on Chip Architecture", *In Proceedings of the Euromicro Symposium on Digital Systems Design*, 2003, page 180, 2003.

[11] S. Murali and G. De Micheli, "Bandwidth-Constrained Mapping of Cores onto NoC Architectures", In Proceedings of the DATE, 2004, pp. 896- 901.

[12] J.D. Owens et al., "Research Challenges for On-Chip Interconnection Networks", *IEEE Micro*, Vol. 27, No. 5, 2007, pp. 96-108.

[13] K. Pawlikowski, "Steady-State Simulation of Queueing Processes: A Survey of Problems and Solutions", *ACM Computing Surveys*, Vol. 22, No. 2, 1990, pp. 123-170.

[14] H. Takagi, *Queueing analysis, Vol. 1: Vacation and priority systems*, Amsterdam, North-Holland, 1991.

[15] E.B. van der Tol and E.G. Jaspers, "Mapping of MPEG-4 Decoding on a Flexible Architecture Platform", SPIE, Vol. 4674, 2002, pp. 1-13.

Caspian: A Tunable Performance Model for Multi-Core Systems

Paper 4

Abbas Eslami Kiasari
Hamid Sarbazi-Azad
Shaahin Hessabi

In the Proceedings of the 14th European Conference on Parallel and Distributed Computing (Euro-Par), Lecture Notes in Computer Science, vol. 5168, pp. 100-109, Canary Island, Spain, Aug. 2008.

Caspian: A Tunable Performance Model for Multi-Core Systems

Abbas Eslami Kiasari[1,2], Hamid Sarbazi-Azad[2,1], and Shaahin Hessabi[2]
[1]IPM School of Computer Science, Tehran, Iran
[2]Sharif University of Technology, Tehran, Iran
kiasari@ipm.ir, {hessabi, azad}@sharif.edu

Abstract
Performance evaluation is an important engineering tool that provides valuable feedback on design choices in the implementation of multi-core systems such as parallel systems, multicomputers, and Systems-on-Chip (SoCs). The significant advantage of analytical models over simulation is that they can be used to obtain performance results for large systems under different configurations and working conditions which may not be feasible to study using simulation on conventional computers due to the excessive computation demands. We present Caspian[1], a novel analytic performance model, aimed to minimize prediction cost, while providing prediction accuracy. This is accomplished by using a G/G/1 priority queueing model which is used for arbitrary network topology with wormhole routing under arbitrary traffic pattern. The accuracy of this model is examined through extensive simulation results.

Keywords: Performance evaluation, Analytical model, Multi-core systems, G/G/1 queueing model.

1 Introduction

Multi-core system designers are constantly confronted with the challenge of designing high performance system while simultaneously meeting constraints such as communication latency, network throughput and design costs [4]. The problem of identifying multi-core system configurations is further exacerbated for the following reasons. First, multi-core systems are evolving into increasingly complex systems with a large number and type of components such as processors, memories, routers, and queues. As a consequence, designers must deal with a large architectural design space consisting of several interacting parameters. Furthermore, new workloads are composed of a large spectrum of programs with widely differing characteristics. System designers have addressed these problems in the past by exploring the design space using detailed simulations. However, this approach has high simulation costs due to the low speed of cycle-accurate simulators.

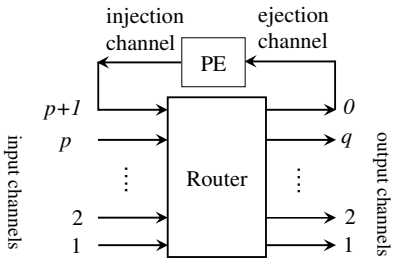


Figure 1: A general structure for a node in a generic multi-core system

[1] The Caspian Sea is the largest enclosed body of water on Earth by area, variously classed as the world's largest lake or a full-fledged sea. It lies between the southern areas of the Russian Federation and northern Iran [Wikipedia].

Performance models are frequently employed by multi-core system vendors in their design of future systems. Typically, engineers construct a performance model for one or two key applications, and then compare future technology options based on performance model projections. An analytical model that accurately characterizes the relationship between multi-core system performance and various implementation parameters would, in theory, obviate the need for detailed, expensive, and time consuming simulations. As an alternative to analytical models, in this research a tunable analytical modeling technique for multi-core system performance has been proposed and evaluated. The proposed approach, which is developed for wormhole flow control, provides buffer utilization, channels throughput, achievable throughput of the network, average waiting time for each channel, and average packet latency. These metrics can be conveniently used for design and optimization purposes, as well as obtaining quick performance estimates.

The main contribution of the work is a novel performance model, Caspian, for multi-core systems which can generalize the traditional delay models. Finally, the proposed model provides not only aggregate performance metrics, such as average latency and throughput, but also useful feedback about the network behavior. Hence, it can be invoked in any optimization loop for multi-core systems for fast and accurate performance estimations.

2 Performance Analysis

If the performance is measured in terms of average packet latency, then maximizing the performance is equivalent to minimizing the end-to-end packet latency. In this section, we derive an analytical performance model for multi-core systems using a G/G/1 [2] priority queueing model. It can be used for any arbitrary network topology with wormhole routing under any arbitrary traffic pattern.

2.1 Assumptions and Notations

We consider input buffered routers with $p+1$ input channels, $q+1$ output channels, and target wormhole flow control under deterministic routing algorithm. This form of routing results in a simpler router implementation and has been used in many practical systems [3]. So in this research we use the deterministic routing for deadlock free routing. The structure of a single node is depicted in Figure 1. Each node contains a router and a Processing Element (PE) capable of generating and/or receiving packets.

Packets are injected into the network on input port $p+1$ (injection channel) and leave the network from output port 0 (ejection channel). Generally, each channel connects output port j of node N to input port i of node M. So, we denote this channel OC_j^N (jth output channel of router N) or IC_i^M (ith input channel of router M). Messages are broken into some packets of fixed length of M flits, as listed in Table 1 along with other parameters. The routing decision delay for a packet, crossing time of a flit over the crossbar switch, and transfer time of a flit across a wire between two neighboring routers are t_r, t_s, and t_w, respectively. Also the transfer time of a flit across the injection and ejection channels are considered to be t_w.

Let $P^{S\to D}$ be the probability of packet transmission from the source node at router S (R^S) to the destination node at router D (R^D). Likewise, the traffic arrival rate of the header flits from IC_i^N to OC_j^N is given by $\lambda_{i\to j}^N$ packets/cycle. Also we assume that the packet injection process to the router R^N has a general distribution with mean value of α^N packets/cycle. The average packet latency (L) is used as the performance metric. Similar to previous works [1], [7], [8], we assume that the packet latency spans the instant when the packet is created, to the time when the packet is delivered to the destination node, including the queuing time spent at the source. We also assume that the packets are consumed immediately once they reach their destination nodes.

Table 1: Parameter notation.

t_r	Spent time for packet routing decision (*cycles*)	*Platform specific parameters*
t_s	Delay of crossbar switch (*cycles*)	
t_w	Spent time for transmitting a flit between two adjacent router (*cycles*)	
M	The size of a packet (*flits*)	
L	The average packet latency (*cycles*)	
$\Re$	Routing function	
R^N	The router located at address N	
PE^N	The processing element located at address N	
IC_i^N	The ith input channel of router R^N	
OC_j^N	The jth output channel of router R^N	
S_j^N	set of all source nodes for packets which pass through OC_j^N	
D_j^N	set of all destination nodes for packets which pass through OC_j^N	
$P^{S\to D}$	The probability of a packet generated by PE^S to be delivered to PE^D	*Application specific parameters*
α^N	The average packet injection rate of PE^N (*packets/cycle*)	
$\lambda_{i\to j}^N$	The average packet rate from IC_i^N to OC_j^N (*packets/cycle*)	
λ_j^N	The average packet rate to OC_j^N (*packets/cycle*) $\left(\lambda_j^N = \sum_i \lambda_{i\to j}^N\right)$	
μ_j^N	The average service rate of OC_j^N (*packets/cycle*)	
$\overline{b_j^N}$	The average service time of OC_j^N (*cycles*) $\left(\overline{b_j^N} = 1/\mu_j^N\right)$	
$\overline{\left(b_j^N\right)^2}$	The second moment of the service time of OC_j^N	
$C_{B_j^N}$	The CV (coefficient of variation) for service time of the OC_j^N	
$C_{A_{i\to j}^N}$	The CV for interarrival time of packets from IC_i^N to OC_j^N	
$\rho_{i\to j}^N$	The fraction of time that the OC_j^N is occupied by packets from IC_i^N	
$W_{i\to j}^N$	The average waiting time for a packet from IC_i^N to OC_j^N (*cycles*)	

2.2 Analytical Model

In Figure 2 consider a packet which is generated in IP^A, and reaches its destination (IP^C) after traversing R^A, R^B, and R^C. The latency of this packet ($L^{A\to C}$) consists of two parts: the latency of header flit ($L_h^{A\to C}$) and the latency of body flits (L_b). In other words

$$L^{A\to C} = L_h^{A\to C} + L_b \tag{1}$$

$L_h^{A\to C}$ is the time from when the packet is created in IP^A, until when the header flit is reached to the IP^C, including the queueing time spent at the source node and intermediate nodes. In Figure 2, $L_h^{A\to C}$ can be computed as

$$L_h^{A\to C} = W_{p+1\to i}^{A} + t_w + t_r + t_s + t_w + t_r + W_{j\to k}^{B} + t_s + t_w + t_r + W_{l\to 0}^{C} + t_s + t_w \quad (2)$$

where $W_{i\to j}^{N}$ is the mean waiting time for a packet from IC_i^N to OC_j^N. Note that in Figure 2 the channel between *B* and *C* can be addressed with OC_k^B or IC_l^C. Since the body flits follow the header flit in a pipelined fashion, L_b is given by

$$L_b = (M-1)(t_s + t_w) \quad (3)$$

The only unknown parameter for computing the latency is $W_{i\to j}^{N}$. This value can be calculated in a straightforward manner using a queuing model. The basic element in the model is a G/G/1 priority queue (the customer interarrival time and server's service time follow general distributions and queues have one server to provide the service). A router is primarily modeled based on nonpreemptive priority queuing system [11]

Now, let us consider, for instance, the *j*th output channel of R^N (OC_j^N). As can be seen in Figure 3, this channel is modeled as a server in a priority queueing system with $p+1$ classes (IC_1^N to IC_{p+1}^N), with arrival rates $\lambda_{i\to j}^N$ $(1 \le i \le p+1)$, served by one server (OC_j^N) of service rate μ_j^N. Note that since all incoming packets are similar, the service times of all packets are equal. Both interarrival and service times are independent and identically distributed with arbitrary distributions.

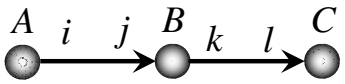


Figure 2: A two hop packet from node A to node C

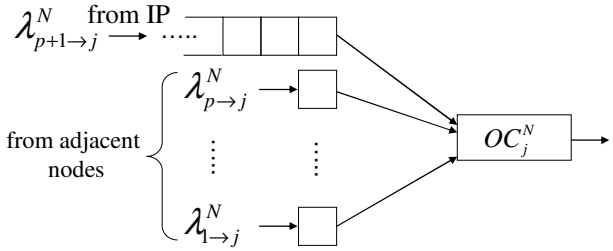


Figure 3: Queueing model of a channel of an arbitrary topology

Since the input channels (except injection channel) have one flit rooms, we should compute the average waiting time for the head of class *i*. Using a technique similar to that employed in literature for priority queues [2], [11] we can write

$$W_{i\to j}^{N} = \begin{cases} \overline{R_j^N} / \left(1-\rho_{i\to j}^N\right), & i = p+1, \\ \dfrac{1+\rho_{i+1\to j}^N - \sigma_{i+2\to j}^N}{1-\sigma_{i+1\to j}^N} W_{i+1\to j}^N, & 1 \le i \le p. \end{cases} \quad (4)$$

where $\overline{R_j^N}$ is the residual service time of OC_j^N seen by an incoming header flit and $\sigma_{i\to j}^N = \sum_{k=i}^{p+1} \rho_{i\to j}^N$. In a G/G/1 queueing system, $\overline{R_j^N}$ is approximated by [2]:

$$\overline{R_j^N} \approx \sum_{i=0}^{p} \rho_{i\to j}^N \frac{C_{A_{i\to j}^N}^2 + C_{B_j^N}^2}{2\mu_j^N} \quad (5)$$

Since we do not have enough insight about the first and second moments of interarrival time, we suppose that $C_{A_{i\to j}^N}$ is constant for all input channels in the network and equal to the coefficient of variation (CV) of the arrival process to network ($C_{A_{i\to j}^N} = C_A$). So, we can rewrite Eq. (5) as

$$\overline{R_j^N} \approx \frac{1}{2}\lambda_j^N \left(\overline{b_j^N}\right)^2 \left(C_A^2 + C_{B_j^N}^2\right) \tag{6}$$

Therefore, to compute $W_{i\to j}^N$ we must calculate the average arrival rate over OC_j^N (λ_j^N), and also first and second moments of the service time of OC_j^N. Assuming the network is not overloaded, the arrival rate over OC_j^N can be calculated using the following general equation

$$\lambda_j^N = \sum_{\forall S}\sum_{\forall D} \alpha^S \times P^{S\to D} \times \Re\left(S \to D, OC_j^N\right) \tag{7}$$

In Eq. (7), the routing function $\Re\left(S \to D, OC_j^N\right)$ equals 1 if the packet from PE^S to PE^D passes through OC_j^N; it equals 0 otherwise. Note that we assume a deterministic routing algorithm, thus the function of $\Re\left(S \to D, OC_j^N\right)$ can be predetermined.

Although λ_j^N can be computed exactly for all topologies by Eq. (7), service time moments of the output channels cannot be computed in a direct manner by a general formula for any topology and any routing algorithm. To compute the moments of the service time of the output channels we first divide the channels into some groups based on their routing order and then an index is assigned to the groups opposite of the routing order, from ejection channel to injection channel. Then, we estimate the first two moments of the service time for the output channels. Determination of the channel service time moments starts at the ejection channel and works backward to the source of the packets. Therefore, the contention delay from lower numbered groups can then be thought of as adding to the service time of packets on higher numbered groups. In other words, to determine the waiting time of channels in group *k,* we must calculate the waiting time of all channels in group $k-1$. This approach is dependent to the topology and routing algorithm. Here we derive an analytical performance model for for e-cube routing [4] in a hypercube network. Due to the popularity of the hypercube network for multicomputer vendors [4], our analysis focuses on this topology but the modeling approach used here can be equally applied for other topologies after few changes in the model.

We consider a system which is composed of 2^n processing cores interconnected by an n dimensional hypercube (H_n). Packets are injected into the network on crossbar input port $n+1$ and leave on output port 0. A dimension-i channel connects output port i of a node to input port i of another node that differs only in the ith bit of its address. (In this paper, the least significant bit is bit 1). The $n+1$ input channels of each router are represented with $n+1$ (injection channel) and 1 to n (dimension 1 to n). Also these channels are assigned to priority classes from index $n+1$ (the highest priority) to 1 (the lowest priority), respectively. It is assumed that a static total ordering of input channel priorities exists. The packet arriving on the higher priority input channel will receive use of the crossbar output first. E-cube routing specifies that a packet sent between two nodes be first routed in the most significant dimension in which the addresses differ, then in the next most significant dimension in which they differ, etc. Finally, it is fed to the destination node via ejection channel. By restricting the order in which the dimensions may be traversed, the possibility of cycles is removed, eliminating deadlock.

We divide the output channels of the hypercube network into $n+2$ groups based on their dimension numbers. Injection and ejection channels are located in group $n+1$ and 0, respectively, and physical channels are located in groups 1 to n. In the ejection channel (group 0) of R^N the header flit and body flits are accepted in t_w and L_b cycles, respectively. So, we can write $\overline{b_0^N} = t_w + L_b$

and since all packets have the same service time, there is not any variation in the service times. In other words $C_{B_0^N} = 0$. Now, we can determine the value of $W_{i\to 0}^N$ where $1 \le i \le n$.

The moments of service time for OC_j^N, $j > 0$, are obtained by tracing each of the 2^{j-1} paths from output j to the network outputs (ejection channels). Since each of these paths is not equally probable, the service time moments are weighted mean of each path service time. If S_j^N and D_j^N be the sets of all possible source and destination nodes for a packet which passes through OC_j^N, respectively, a passing packet through OC_j^N are destined to $M \in D_j^N$ with the probability of $\sum_{\forall S \in S_j^N} P^{S\to M} / \sum_{\forall S \in S_j^N} \sum_{\forall D \in D_j^N} P^{S\to D}$. The contention delays along each path are random variables; however each is only a fraction of the packet length for all packet rates that can be sustained on the hypercube.

For example, consider the output channel in dimension 2 of R^0 (OC_2^0) in an n-dimensional hypercube. One possible path from OC_2^0 to a network output is directly to ejection channel of the adjacent node R^2, for an average service time of $L_1 = t_w + \left(t_r + W_{2\to 0}^2 + t_s\right) + t_w + L_b$ where $W_{2\to 0}^2$ was already computed. The second path from OC_2^0 to a network output is through dimension 1 of R^2 and ejection channel of R^3, for an average service time of $L_2 = t_w + \left(t_r + W_{2\to 1}^2 + t_s\right) + t_w + \left(t_r + W_{1\to 0}^3 + t_s\right) + t_w + L_b$ where $W_{1\to 0}^3$ was already computed and $W_{2\to 1}^2$ can be computed with the same approach. So, the first and second moments of the service time can be estimated as

$$\overline{b_2^0} = \frac{\sum_{\forall S \in S_2^0} P^{S\to 2}}{\sum_{\forall S \in S_2^0} \sum_{\forall D \in D_2^0} P^{S\to D}} L_1 + \frac{\sum_{\forall S \in S_2^0} P^{S\to 3}}{\sum_{\forall S \in S_2^0} \sum_{\forall D \in D_2^0} P^{S\to D}} L_2$$

and

$$\overline{\left(b_2^0\right)^2} = \frac{\sum_{\forall S \in S_2^0} P^{S\to 2}}{\sum_{\forall S \in S_2^0} \sum_{\forall D \in D_2^0} P^{S\to D}} L_1^2 + \frac{\sum_{\forall S \in S_2^0} P^{S\to 3}}{\sum_{\forall S \in S_2^0} \sum_{\forall D \in D_2^0} P^{S\to D}} L_2^2$$

Now, we are able to calculate the coefficient of variation of the service time in OC_2^0, $C_{B_2^0}^2 = \overline{\left(b_2^0\right)^2} / \left(\overline{b_2^0}\right)^2 - 1$, and then the mean waiting time for OC_2^0 seen by other channels ($W_{i\to 2}^0$, $i > 2$) can be computed with Eq. (4). After computing the mean waiting time of all channels in group 2, the mean waiting time for other output channels (channels in groups 3, 4, …, n+1) can be calculated using the same approach. Now, we can calculate the packet latency between any two nodes in the network by Eq. (1). The average packet latency is the weighted mean of these latencies as

$$L = \sum_{S=0}^{2^n-1} \sum_{D=0}^{2^n-1} P^{S\to D} \times L^{S\to D} \tag{8}$$

3. Model Validation

The proposed analytical model has been validated through a discrete-event simulator that mimics the behavior of the network at the flit level. To achieve a high accuracy in the simulation results, we use the batch means method [9] for simulation output analysis. There are 10 batches and each batch includes up to 500,000 packets depending on the traffic injection rate and network size. Statistics gathering was inhibited for the first batch to avoid distortions due to the startup transient. The standard deviation of latency measurements is less than 2% of the mean value.

For the destination address of each packet, we have considered the uniform and hotspot traffic patterns [10]. Packets are transferred to the local PE through the ejection channel as soon as they arrive at their destinations. Nodes generate packets independently of each other, and which follows a Poisson process. It means that the time between two successive packet generations in a PE is distributed exponentially, so for the first time we run the model program with $C_A = 1$. The Poisson model simplicity made it widely used in many performance analysis studies, and there are a large number of papers in very diverse application domains that are based on this stochastic assumption [5].

Using the proposed performance model, the CV of the interarrival time (C_A) can be adjusted to account for conformity between model and simulation results. Figure 4 shows the flow chart used for tuning of the performance model. The model is run with various C_A until the predicted latency by the model is matched to its corresponding simulated value. The tuning procedure is run for the H_8 with $t_r = t_s = t_w = 1$ and $C_A = 1.0498$ is obtained for average packet generation rate of $\lambda = 0.045$ packets/cycle and packet length of $M = 32$ flits. Result of this tuned model has been presented in Figure 5. The horizontal axis in the figure shows the traffic generation rate (packets/cycle) at each node while the vertical axis shows the mean packet latency (cycles). Tuned C_A is shown to give good quantitative agreement of model to simulation for a wide range of packet generation rate and packet length.

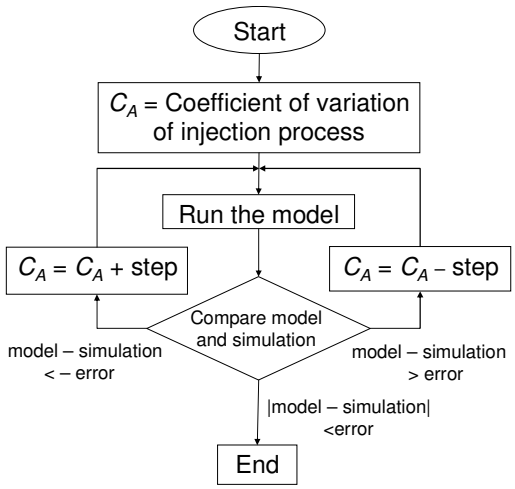


Figure 4: Flow chart showing the strategy of the performance model tuning to simulation.

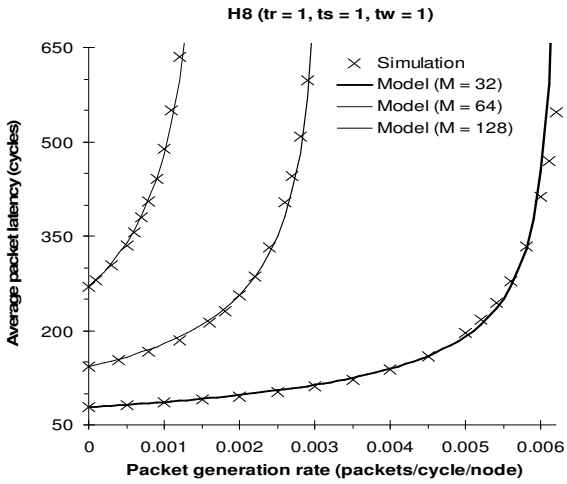
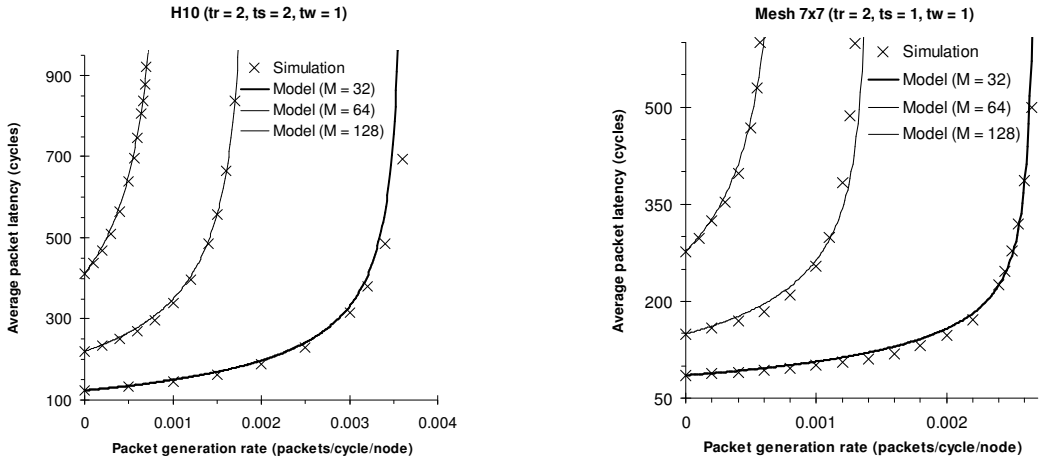


Figure 5. The average packet latency predicted by the tuned model against simulation results for an H_8.

Figure 6(a) illustrates average packet latency predicted by the tuned model, plotted against simulation results for H_{10} network. ($C_A = 1.0035$). Furthermore to verify the model accuracy for other topologies and non-uniform traffic pattern, we have modeled a 7X7 mesh network under hotspot traffic [10]. According to hotspot traffic pattern, there is a hot node in the network to receive the packets. Each node sends packets to the hot node with probability h, and sends packets to other nodes with probability $1 - h$. In our experiments, we consider the node 24 in the center of the network as a hot node with hotspot rate $h = 0.1$. The comparison results is shown in Figure 6(b) ($C_A = 0.8653$).

In [6], we have used Caspian and presented a performance-aware mapping algorithm which maps the IPs onto a generic System-on-Chip architecture such that the average communication delay is minimized.



(a) (b)

Figure 6: The average packet latency predicted by the tuned model against simulation results for (a) H_{10}, and (b) 7x7 mesh network with hotspot traffic.

4. Conclusions and Future Work

A novel methodology for predicting the communication performance of multi-core systems was proposed. The choice of the hypercube and mesh networks as the underlying interconnection architecture serves mostly as an example. In fact, our methodology can be modified to arbitrary topologies by adapting the analytical models accordingly to the target topology. Moreover, although we have evaluated our algorithm only for multi-core systems with dimension order routing, the approach is general enough to be applied to other deterministic and oblivious routing schemes.

We plan to advance this research in several directions. One possible direction is to extend this approach to multi-core systems with realistic workloads. Another important extension is to accommodate interconnection networks that support adaptive routing. The main challenge comes from the difficulty involved in the calculation of the arrival rate for each channel as multiple routing paths are possible in adaptive routing. Finally, we are extending this work for routers with finite size buffers.

References

1. Aljundi, A.C., Dekeyser, J., Kechadi, M.T., Scherson, I.D.: A Universal Performance Factor for Multi-criteria Evaluation of Multistage Interconnection Networks, Future Generation Computer Systems Vol. 22, No. 7, pp. 794-804, (2006)
2. Bolch, G., Greiner, S., De Meer, H., Trivedi, K.S.: Queueing Networks and Markov Chains: Modeling and Performance Evaluation with Computer Science Applications, 2nd Edition. John Wiley and Sons, (2006)
3. Duato, J.: Why Commercial Multicomputers Do Not Use Adaptive Routing. IEEE Technical Committee on Computer Architecture Newsletter, pp. 20-22, (1994)
4. Duato, J., Yalamanchili, C., Ni, L.: Interconnection Networks: An Engineering Approach. IEEE Computer Society Press, (2003)
5. Hu, J., Ogras, U.Y., Marculescu, R.: System-level Buffer Allocation for Application-Specific Networks-on-chip Router Design. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, Vol.25, No.12, pp. 2919-2933, (2006)
6. Kiasari, A.E., Hessabi, S., Sarbazi-Azad, H.: PERMAP: A Performance-Aware Mapping for Application-Specific SoCs. Proceedings of the Application-specific Systems, Architectures and Processors, (2008)

7. Kiasari, A.E., Rahmati, D., Sarbazi-Azad, H., Hessabi, S.: A Markovian Performance Model for Networks-on-Chip. In Proceedings of the Euromicro International Conference on Parallel, Distributed and Network-Based Processing, pp. 157-164, (2008)
8. Najafabadi, H.H., Sarbazi-Azad, H., Rajabzadeh, P.: Performance Modelling of Fully Adaptive Wormhole Routing in 2D Mesh-connected Multiprocessors, In Proceedings of the International Symposium on Modelling, Analysis, and Simulation of Computer and Telecommunication Systems, pp. 528-534, (2004)
9. Pawlikowski, K.: Steady-State Simulation of Queueing Processes: A Survey of Problems and Solutions. ACM Computing Surveys, Vol. 22, No. 2, pp. 123-170, (1990)
10. Sarbazi-Azad, H., Ould-Khaoua, M., Mackenzie, L.M.: Analytical Modeling of Wormhole-Routed k-Ary n-Cubes in the Presence of Hot-Spot Traffic. IEEE Transaction on Computers, Vol. 50, No. 7, pp 623-634, (2001)
11. Takagi, H.: Queueing analysis, Vol. 1: Vacation and Priority Systems. Amsterdam, North-Holland, (1991)

Power-Efficient Routing Algorithm for Torus NoCs

Paper 5

Dara Rahmati
Abbas Eslami Kiasari
Hamid Sarbazi-Azad
Shaahin Hessabi

In Proceedings of the International Conference on Contemporary Computing (IC3), pp. 211-220, Uttar Pradesh, India, Aug. 2008.

Power-Efficient Routing Algorithm for Torus NoCs

D. Rahmati[1], A. E. Kiasari[1,2], H. Sarbazi-Azad[1,2], S. Hessabi[1]
[1] Computer Engineering Dept., Sharif University of Technology
[2] IPM School of Computer Science
Tehran, Iran
{d_rahmati, kiasari}@ce.sharif.edu, {azad, hessabi}@sharif.edu

Abstract
Modern System-on-Chip (SoC) architectures use Network-on-Chip (NoC) for high-speed inter-node communication. NoC with torus interconnection topology is now popular due to its low dimension and simple structure. Torus NoC is very similar to the mesh NoC from a structural point of view, but has rather smaller diameter that makes it a suitable choice for NoCs. For a routing algorithm to be deadlock-free in a torus NoC at least two virtual channels should be used to avoid channel dependency, while mesh NoC can handle deadlock freedom using only one virtual channel. In this paper, we propose a novel approach on designing routing algorithms for mesh and torus NoCs. Also a deadlock free routing algorithm is proposed for Torus NoC that uses only one virtual channel per physical channel resulting in lower power consumption because of reduced hardware complexity and with no significant performance degradation. The algorithm works within a dimension and is applied to all dimensions individually for XY routing and various turn based deterministic routing algorithms like west first, north last and negative first. We have proved efficiency of the algorithm using simulation results obtained from synthesis of our implemented VHDL Register Transfer Level (RTL) model of NoC.

Keywords: SoC, NoC, Torus, Mesh, Performance, Power Consumption, Routing, Virtual Channel, Deadlock, VHDL RTL model.

1. Introduction

The simplest and hence widely used routing algorithm for the mesh NoCs is XY routing [1,2,3,8]. In this algorithm the packet is routed across the X axis and then across the Y axis until it reaches the destination node as shown in Fig.1. Since there are no wraparound links to connect the first and last nodes in each dimension, XY routing algorithm is deadlock free using only one virtual channel.

However, applying XY routing for the torus NoC may cause deadlock as a result of the channel dependency in each dimension between different messages [8]. By using more than one virtual channel there will be the flexibility of designing different deadlock free routing algorithms in the cost of hardware complexity, more area, and thus higher power consumption. Power consumption is the most important factor in the design and implementation of NoC architectures, while performance (network latency and throughput) is the key factor in multicomputer networks.In order to have a deadlock-free routing algorithm in the torus NoC, there should be at least two virtual channels to break the cyclic channel dependency, caused by wraparound links, into a spiral [4, 8, 10]. This is not the case when mesh NoCs are used without wrap-around links and thus requiring only one virtual channel. It is also shown that the number of virtual channels has a crucial effect on power consumed by the NoC [5, 6, 12].

In this paper, we first introduce IRN (Interconnection Routing Notation), a map-based systematic approach on designing routing algorithms for mesh and torus NoCs. This notation is also extendable for other interconnection topologies. We then use IRN and propose a deadlock free routing algorithm called TRANC (Torus Routing Algorithm for NoC) for the torus NoC that uses only one virtual channel.

The proposed routing algorithm enjoys the low power consumption of a mesh NoC while possessing a good performance (near a torus NoC). It even exhibits better performance for light traffic because of a zero switching time between virtual channels compared to a torus NoC using two virtual channels to implement XY routing. There is a slight decrease in the performance of TRANC for heavy traffics and near the saturation point of the NoC when compared to XY routing in the trous NoC. However, as mentioned before, power consumption is a dominant factor when comparing routing algorithms in NoCs, since the network rarely works near its saturation point of operation.

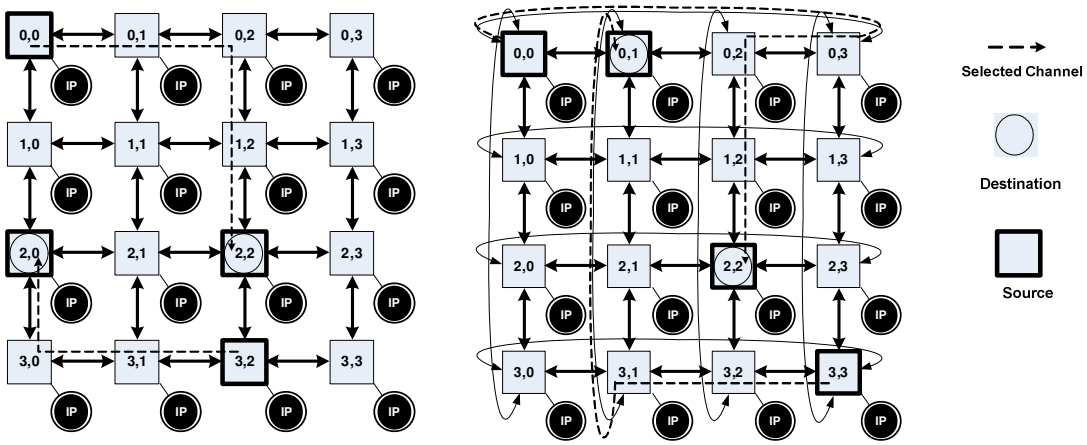


Fig. 1. A 4x4 mesh NoC (left) and a 4x4 torus NoC (right).

2. Routing in the Mesh and Torus NoCs

An $n \times n$ mesh or torus NoC consists of n^2 nodes arranged in a two-dimensional grid structure. Each node is addressed using an (x,y) tuple and has a neighboring node in the increasing and decreasing directions (positive and negative directions) in each dimension. The first node and the last node of each dimension are linked using a wraparound link in the torus NoC, while such a wraparound link does not exist in the mesh NoC. Fig. 1 shows a 4x4 mesh NoC and a 4x4 torus NoC. Each node in the network consists of two parts: IP and Router. Usually, the whole system (called SoC) except for the IPs is called the NoC.

2.1. Node structure in mesh and torus NoCs

A cycle accurate and synthesizable VHDL hardware model for NoCs has been implemented and several different topologies like mesh and torus have been tested based on it. The top most shared component in this hardware model is the NoC node in which *IP* and *router* are its main components. Fig. 2 shows the node structure in the implemented model.

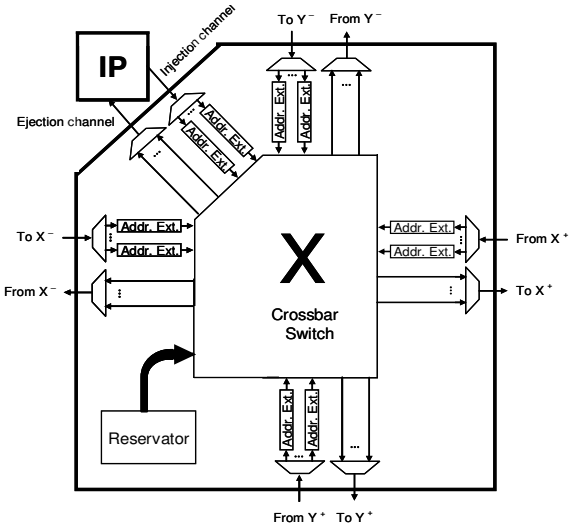


Fig. 2. Node structure in NoC

The IP can be a processor with some local memory, or any other module that can send/receive packets over the network. In our implementation it generates packets based on a traffic model like uniform distribution for packet destinations. Also each IP generates packets on intervals based on a Poisson distribution.

The router has five input and five output channels. A node uses four inputs and four output channels to connect to its neighboring nodes; two per dimension, one in each direction. The remaining channels are used by the IP to inject/eject messages to/from the network, respectively. Messages generated by the IP are injected into the network through the injection channel. Messages that arrive at the destination node are transferred to the IP through the ejection channel. The bandwidth of each channel is shared among a number, say V, of virtual channels. The hardware implementation of the router consists of several different units such as *Address Extractor* which determines and manipulates the packet headers and contains some buffer (of few flits) for each incoming virtual channel. It should be noted that the more the number of virtual channels is, the structure of the node is more complex. There are *Multiplexer* and *De-Multiplexer* units which handle the virtual channel operations, *Selector* unit which applies the virtual channel selection rule, *Crossbar switch* that can simultaneously connect multiple input channels to multiple output channels given that there is no contention over the output channels. *Reservator* unit which controls the crossbar switch and other related sub-modules. When a specific topology like mesh or torus is supposed to be modeled by such components, a top-level wrapper module is implemented that connects several nodes of this type to each other based on the structure of the specified topology. Based on this hardware model, different cases of mesh and torus topologies have been simulated and synthesized to extract accurate quantities, e.g. average message latency and power consumption values.

2.2. Routing in mesh and torus NoCs

Examples of XY-routing are also shown in Fig. 1 for torus and mesh networks (the route is indicated as dashed lines). The XY routing for mesh NoCs is straight forward: a message (or packet) first traverses its route towards its destination across X axis and then across Y axis. It is easy to see that such a routing algorithm prevent cyclic dependency in reserving and using network channels by the messages. The case is however different for the torus NoCs as where wraparound links can clearly make cyclic channel dependency resulting in deadlock situation. The straight forward deadlock free XY routing algorithm for this case needs at least two virtual channels per physical channel. In XY routing algorithm in the torus, the packet traverses first X dimension and then Y dimension (as in mesh NoCs); it uses the first virtual channel before reaching a wraparound link, thereafter it uses the second virtual channel until it reaches the

destination node. Thus, XY routing in the mesh NoCs requires one virtual channel per physical channel while it requires 2 virtual channels per physical channel in torus NoCs.

2.3. Performance and power consumption results

Fig. 3 shows the performance and power consumption of XY routing for a 4x4 mesh NoC with one and two virtual channels and a 4x4 torus NoC with two virtual channels. In the figure, horizontal axis shows the message generation rate at each node and the vertical axis shows either the average message latency (an important measure of NoC performance) or the energy consumed. The simulated topologies are of 4x4 nodes, message length of 32 flits (4 flits for the header and 28 data flits), and buffer size of 4 flits for each virtual channel. The figure shows that the torus exhibits a better performance compared to the mesh topology with one and 2 virtual channels per physical channel. This is because of the lower diameter and average inter-node distance in the torus NoC compared to its equivalent mesh network. The figure also shows that the number of virtual channels mainly determines the power consumption of the network and the torus NoC with two virtual channels has a larger amount of dissipated power due to the complexity of the switch and higher buffering requirement and extra wraparound links.

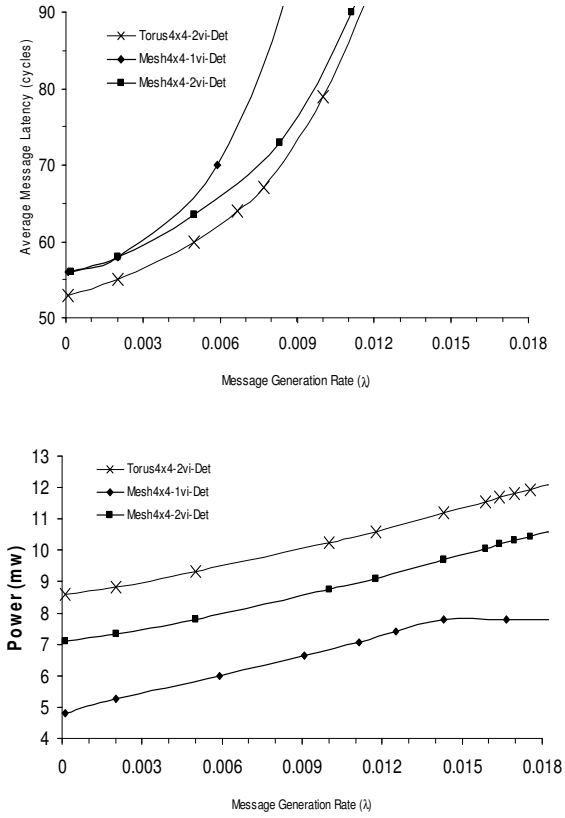


Fig. 3. Performance and power consumption of XY routing in a 4x4 mesh with one and two virtual channels and 4x4 torus with two virtual channels.

3. Interconnection Routing Notation (IRN)

We propose a new notation to extract new routing algorithms for torus and mesh NoCs. This notation can also be extended to other interconnection topologies. By using this notation, it is possible to have better understanding and formulation of routing algorithms for interconnection networks. Consider the XY routing algorithm for the 4x4 mesh network with one virtual channel per physical channel (as shown in Fig. 1). The *IRN Map* and *IRN Graph* for this algorithm are shown in Fig. 4. In XY routing algorithm, a packet first traverses the X axis and then continues its journey towards its destination along Y axis. The IRN notation only explores

the rule of movement through one axis (current axis or dimension). First row of the IRN Graph shows that if the source and destination nodes for a packet are adjacent across a dimension, then the packet moves towards the destination node directly with one step. The other rows show the direction of movements for distances more than one hop, individually for all node locations at a dimension. Corresponding to the IRN Graph, the IRN Map in each row shows the direction that the packet should traverse when it is in a specified location to cross the network to get closer to the destination. As shown in the figure, all the movements over the diameter of the map (or matrix), which means the packet should go from a smaller index to a bigger index, have a '+' sign. This sign indicates movement in the positive direction of that dimension; similarly the '-' sign is used in the lower part of the map.

Fig. 5 shows the notation for the proposed routing algorithm in the torus network but with only one virtual channel. At the first row of the IRN Graph the wraparound link is shown. Because of using wrap around links, a packet may reach its destination using positive or negative directions; but for a routing algorithm to be deadlock free only one of the directions should be selected. The plus and minus symbols with the circles refer to movements on the first row of IRN map in which it is not reasonable to select the opposite direction to reach the destination. Therefore, we suppose these moves are always unchangeable, and only the signs without the circle are selectable. It should be noted that there are 4 selectable moves that can make up 16 different routing algorithms some of which are deadlock free and some others are not. The goal is to find the best selection (being deadlock-free and as minimal and optimal as possible, which will be explored using minimality and optimality factors in next section). Here, the only change to the mesh XY routing is that the first and last nodes in a dimension can use the wraparound link for a one step movement. For example, in the case of 4x4 torus, nodes 0 and 3 can communicate with each other using the wraparound links.

3.1. The Rules

In order to complete the IRN Map, the following rules should be applied: In each column there should not be sign changing for more than once. This is because it may cause a livelock in the network. For the sake of minimality and optimality it is better to have equal number of '+' and '-' signs for the selectable area. The same case applies to the number of '+' and '-' in a row. Also there should not be more than one sign changes in a row. Fig. 6 shows a case for the 4x4 torus in which deadlock may occur. Deadlock is caused because of a loop between movements in positive or negative directions. There should be one row with all selectable '-' movements and also one row with all selectable '+' movements for the algorithm to be deadlock free. After filling these two rows, the other rows should be filled using the previous rules. Example for applying these rules is shown in Fig. 7 for the 6x6 torus NoC.

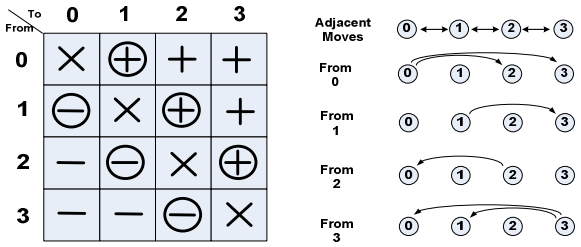


Fig. 4. The IRN Map and Graph for a 4x4 mesh using XY Routing.

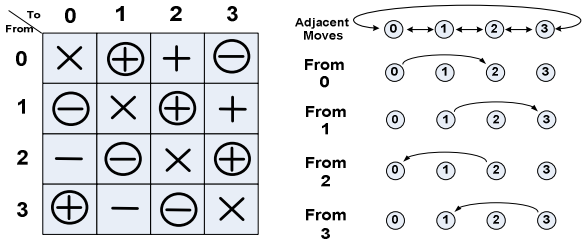


Fig. 5. The IRN Map and Graph for a sample routing in a 4x4 torus.

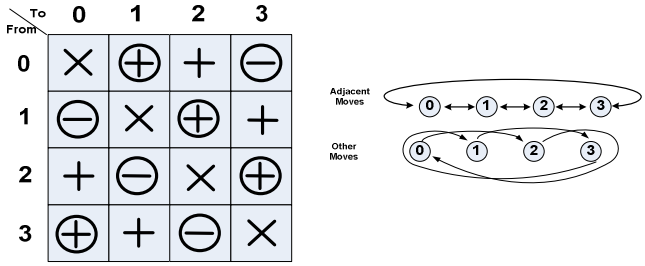


Fig. 6. A routing case in a 4x4 torus causing deadlock.

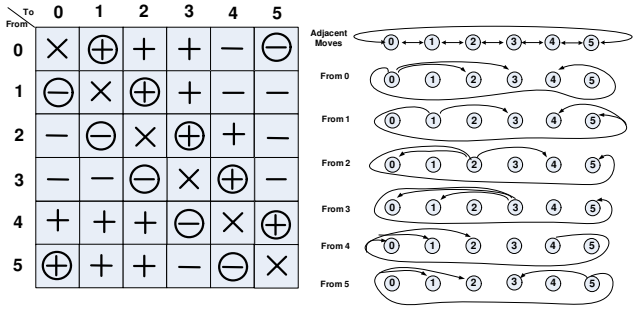


Fig.7. IRN Map and IRN Graph for routing in a 6x6 torus.

3.2. Optimality and Minimality

Minimality Factor (*M*): For a packet which traverses the network from the source to destination node, there is always a minimum number that determines the shortest path for the packet. Because of the limitations that the routing algorithm poses on the packet, the routing algorithm might not be always minimal.

As can be seen in Fig. 8, the selectable area of the IRN Map determines whether the proposed routing algorithm for a dimension is minimal or not. The fact is that for the dimensions with radix $n>4$ in the torus, there is not a minimal algorithm in which all the paths are the shortest possible ones. The shaded boxes in the figure show a subset of the selectable areas for odd and even values of n in which the minimality parameter is applicable.

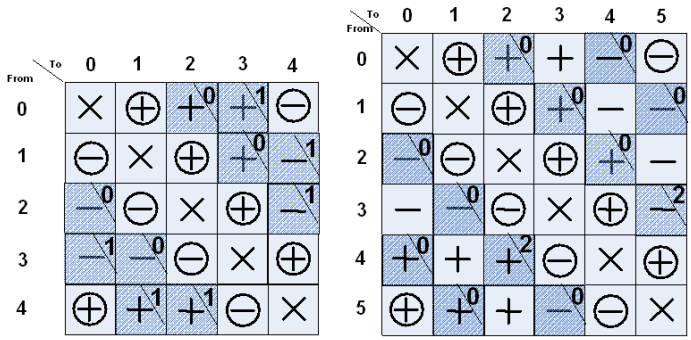


Fig.8. Minimality factor for some routing methods in a 5x5 torus (left) and a 6x6 torus (right).

A number is displayed in the top corner of the shaded boxes: it is *0* if the packet takes a direction with the shortest distance and other numbers show the extra steps that should be taken. When *n* is odd, all movements of the selectable area are shaded and when *n* is even this parameter is not applicable to the movements with distance $n/2$ from the source nodes, since both directions result in equal distance. At last for a specific dimension if we calculate the sum of all minimal path lengths when comparing different routing algorithms, the algorithm with the minimum total sum will be the best one (or here called the minimal one).

Optimality Factor (*Opt*): Although in some references in the area of interconnection networks, an optimal algorithm is known to be a *balanced* algorithm, here we propose a quantitative approach as a parameter based on IRN notation to measure the quality of traffic balance or *optimality factor*. This parameter describes how good an algorithm can balance the network traffic. In fact for a routing algorithm we need a measure that indicates if all the links are utilized properly with respect to the packet destination address distribution. For the case of uniform traffic, the links should be utilized equally. With uniform traffic pattern, it is supposed that all nodes send a packet to any other network nodes with equal chance, and therefore all the links should be utilized evenly. As shown in Fig. 9, some numbers have been presented on adjacent movements with the shaded boxes. Note that these movements are the representatives of their corresponding links and if we consider the sum of '+' ('-') signs for positive (negative) movements in their corresponding rows and columns (considering wraparounds), the result shows the number of times this link has been utilized. The numbers have been shown in lower corner of the shaded boxes and as discussed they should have the same value in order to have optimal routing; therefore the variance (*Opt*) of all the numbers is a good candidate to represent the optimality of routing algorithms: the smaller the *Opt* is, the more optimal the algorithm is. As shown in the figure, two different algorithms have been proposed for the 4x4 torus in which one of the algorithms is optimal since all the optimality numbers are equal to 2, therefore *Opt* is *0*. The other algorithm is not optimal, although both algorithms are minimal.

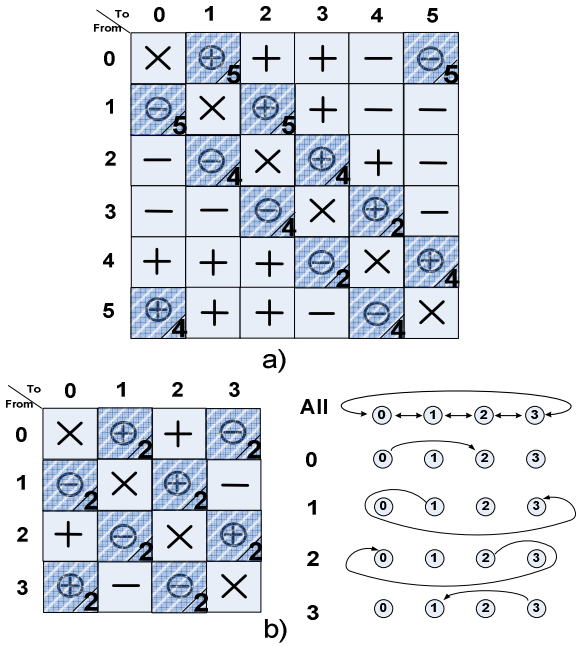


Fig. 9. IRN with optimality factors for different routing methods;
(a) Non-optimal in 6x6 torus, *Opt*=12 and
(b) Optimal in a 4x4 torus, *Opt*=0.

4. The TRANC Routing Algorithm

In this section, we introduce a new routing algorithm that is deadlock free and requires only one virtual channel per physical channel (i.e. no extra virtual channel to the existing physical channel is required). The algorithm uses an incremental approach based on the IRN notation in an $n \times n$ torus NoC.

4.1. Proposed Algorithm for any radix *n*

As shown in Fig. 10, the IRN Map for radix n is generated by adding a row and a column to the IRN Map of radix n-1, starting from n=4. The algorithm is straight forward for the 4x4 torus; for higher radices it is enough to add a row and a column as shown in Fig. 10. In order to complete the new row it is enough to fill the right most two boxes with '-' signs and others with '+' signs. Again for completing the new column it is enough to fill the two lower boxes with '+' signs and others with '-' signs. When two or more of the moves described in IRN Graph happen simultaneously, they may form a situation where some of the packets are waiting for other ones to free the path. In this situation, there is a packet contention. When the contention starts from a packet and lasts with the same packet, such that no activity is possible for the packets, it is said that a deadlock situation has occurred. A routing algorithm that never causes a deadlock situation is called a deadlock free routing algorithm.

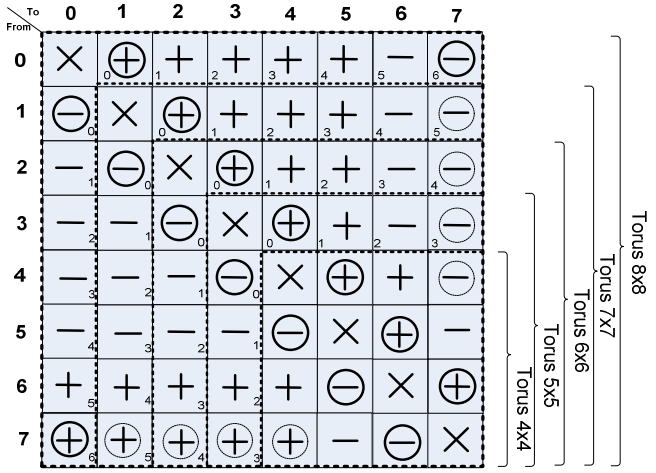


Fig. 10. The proposed IRN Map representing the TRANC routing algorithm in an *nxn* torus.

The TRANC algorithm is a deadlock free routing algorithm. The intuitive justification that it is deadlock free is extracted from the IRN Map and Graph in Fig. 7. As discussed before, there is not a positive movement of more than one step from node 3, and therefore the positive loop is broken in the network. The same is correct for negative movements, as there are not any negative movements of more than one step from node 4 to other nodes and therefore the negative loops also are broken. There is mathematical justification that TRANC is deadlock free, that we do not present it here. Furthermore the justification is also supported by the extensive simulation experiments we have realized for different scenarios. As discussed before for the dimensions with radix *n>4*, TRANC is not fully optimal and fully minimal but with good optimality and minimality factors for different radices. The reason is that each packet traverses the shortest possible path to reach the destination which ignores deadlock, not the shortest physical path which potentially causes a deadlock. Also the usage of wraparound links is not balanced compared to other links because of the rules that have been applied to the algorithm. It is possible to use a different approach for different radices based on IRN that may result in better optimality and minimality factors but justification of deadlock freedom for each radix should be done separately. We ignore this approach for the sake of present discussion.

In Fig. 12, a pseudo code for TRANC algorithm is given. As can be seen in the code, the complexity of the hardware that utilizes this algorithm compared to the classic XY routing algorithm includes the extra comparisons that should be done with *n*-1, *n*-2 and *n*-3 and since *n* is a constant number (when a fixed radix is implemented in hardware), only some comparison operations with some constant numbers are added to the code. Such simple comparisons do not require considerable power and do not impose noticeable delay in routing as will be shown later in the simulation results.

5. Simulation Results

We have implemented a VHDL cycle accurate and synthesizable hardware model for mesh and torus NoCs using both XY routing and TRANC with the possibility of using different number of virtual channels. To evaluate the performance and power dissipation for the proposed routing algorithm in comparison to XY routing two different network sizes (4x4 and 6x6 nodes) are considered. The message size is considered to be fixed and equal to 32 flits (or phits) and the destination of the messages is chosen uniformly over the network nodes. Messages are generated and entered into the network following a Poisson distribution. The VHDL implementation has been used for both performance evaluation using simulation tools and also power estimation using *Power Compiler* CAD tool [7, 12, 13].

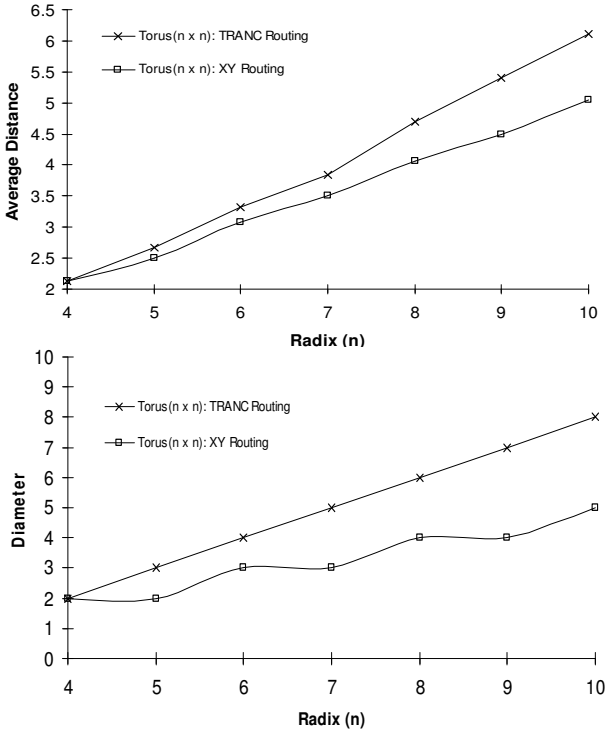


Fig. 11. The average inter-node distance and diameter using TRANC and XY routing algorithms.

A primarily evaluation of the TRANC using a simple C++ program shows that TRANC slightly increases the maximum and the average inter-node distance in the network (or message path length) compared to XY routing in the torus NoC. This is shown in Fig. 11 for different network radices. Note that for popular and current network sizes used in practice today (i.e. NoC with up to 6x6 nodes) the difference between the average and maximum inter-node distances for the two routing algorithms in torus NoCs is small. Therefore, the lower complexity of the router in TRANC (due to the fewer virtual channels used) can improve the performance and reduce the power dissipation in the network.

```
Algorithm TRANC for 2-Dimensional Torus NoCs.
    Inputs: Coordinates of current node (Xcurrent, Ycurrent),
            destination node (Xdest, Ydest), and radix n;
    Output: Selected output Channel
Begin
Xoffset := Xdest − Xcurrent;   Yoffset :=Ydest − Ycurrent;
if (Xoffset=0) and (Yoffset !=0) then return Ejection Channel;
else
  { if (Xoffset =1) or (Xoffset = -n+1) or
      (((Xdest = n-2) and (Xcurrent=n-4)) or (Xcurrent=n-2)) or
      (Xdest -2>= Xcurrent)
    then return X+;
    if (Xoffset = -1) or (Xoffset = n-1) or
      (((Xdest = n-3) and (Xcurrent=n-1)) or (Xcurrent=n-3)) or
      ((Xcurrent-2>= Xdest) or (Xdest =n-2))
    then return X-;
    if (Yoffset =1) or (Yoffset = -n+1) or
      (((Ydest = n-2) and (Ycurrent=n-4)) or (Ycurrent=n-2)) or
      (Ydest -2>= Ycurrent)
    then return Y+;
    if (Yoffset = -1) or (Yoffset = n-1) or
      (((Ydest = n-3) and (Ycurrent=n-1)) or (Ycurrent=n-3)) or
      ((Ycurrent-2>= Ydest) or (Ydest =n-2))
    then return Y-;
  }
End.
```

Fig. 12. Pseudo code for TRANC routing algorithm

Fig. 13 shows the simulation results for XY routing in mesh and torus NoCs and for TRANC routing algorithm in torus NoCs (for 4x4 and 6x6 wormhole-switched networks). The horizontal axis shows the traffic generation rate at each node while the vertical axis shows the average message latency (or dissipated power) in the network. As can be seen in the figure, the performance of TRANC routing (with one virtual channel) is slightly better than XY routing (using 2 virtual channels) while the power consumption is near that of a mesh NoC and much less than that of XY-routed torus (using 2 virtual channels). To have a unique measure to assess the suitability of the proposed algorithm for torus NoCs we have also used the product of average message latency and power consumption. Simulation results show that the proposed routing algorithm for the torus NoC using one virtual channel is superior to the equivalent mesh using XY routing (using one virtual channel) and equivalent torus NoC using XY routing with 2 virtual channels.

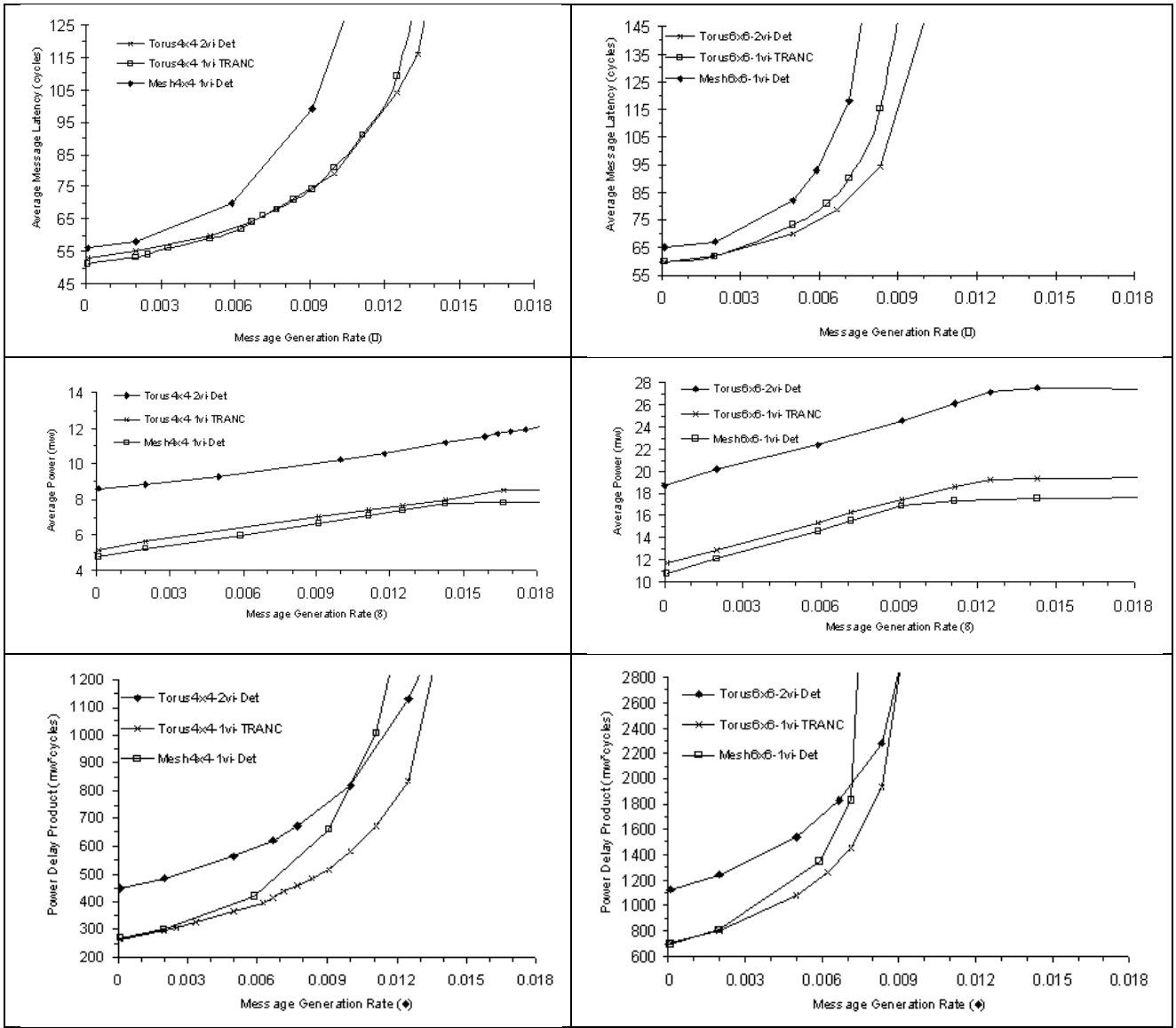


Fig. 13. Performance, power consumption, and power-delay product of XY routing in the mesh and torus NoCs and TRANC routing algorithm in the torus NoC with radices 4 and 6.

6. Conclusions

Current SoC designs have popularly employed point-to-point NoCs for inter-IP communication. The most popular NoCs are the mesh and torus networks. The mesh topology enjoys its simple structure and possibility of using XY routing with only one virtual channel. However, when wraparound links are used, in the torus NoCs, Two virtual channels should be used to ensure deadlock freedom for XY routing. On the other hand, adding virtual channels increases power dissipation, although performance is increased (compared to the mesh NoC) as a result of lower inter-IP distance caused by wraparound links in the torus.In this paper, a new network routing notation, IRN, and based on it a new routing algorithm for the torus NoCs (called TRANC) were presented. The TRANC routing algorithm for the torus NoC uses only one virtual channel and thus consumes lower energy compared to XY routing that requires 2 virtual channels. Note that simplicity of the router (less buffering and switching hardware complexity) can well compensate for the slightly increased inter-IP distance (compared to the XY routed torus NoC with 2 virtual channels) and result in a slightly higher performance. Our next objective is to design partially and fully-adaptive routing algorithms for torus NoCs with a minimum virtual channel requirement.

References

1. W. J. Dally, B. Towles, "Route packets, not wires: on-chip interconnection networks", Proc. DAC, pp. 684–689, June 2001.
2. R. V. Boppana, S. Chalasani, "A framework for designing deadlock-free wormhole routing algorithms", IEEE Transactions on Parallel and Distributed Systems (TPDS), 7(2): 169-183, 1996.
3. C. J. Glass, L. M. Ni, "The turn model for adaptive routing", Proceedings of the International Symposium on Computer Architecture (ISCA), pp. 278-287, 1992.
4. A. Singh, W. J. Dally, A. K. Gupta, B. Towles, "GOAL: A Load-balanced Adaptive Routing Algorithm for Torus Networks", in Proc. of the International Symp. on Comp. Arch., pp. 194-205, June, 2003.
5. Terry T. Ye, Luca Benini, Giovanni De Micheli, "Analysis of Power Consumption on Switch Fabrics in Network Routers", In Proceedings of DAC, 2002.
6. H-S Wang, L-S Peh, S. Malik, "Orion: A Power-Performance Simulator for Interconnection Network", In International Symposium on Microarchitecture, Istanbul, Turkey, November 2002.
7. D. L. Liu, C. Svensson, "Power consumption estimation in CMOS VLSI chips", IEEE Journal of Solid-State Circuits, 29(6): 663-670, 1994.
8. W. J. Dally, H. Aoki, "Deadlock-Free Adaptive Routing in Multicomputer Networks Using Virtual Channels", IEEE ransactions on Parallel and Distributed Systems, 4(4):466–475, April 1993.
9. Jae H. Kim, Ziqiang Liu, Andrew A. Chien, "Compressionless Routing: A Framework for Adaptive and Fault-tolerant Routing", IEEE Transactions on Parallel and Distributed Systems 1996.
10. W.J. Dally, C.L. Seitz, "Deadlock-free message routing in multiprocessor interconnection networks", IEEE Transactions on Computers, vol. C-36, no. 5, pp. 547-553, May 1987.
11. W. J. Dally, C. Seitz, "The torus routing chip", In Distributed Computing, pages 187 196, 1986.
12. N. Banerjee, P. Vellanki, K. S. Chatha, "A Power and Performance Model for Network-on-Chip Architectures", In Proceedings of DATE, Paris, France, February 2004.
13. K. Srinivasan, K. S. Chatha, "ISIS : A Genetic Algorithm based Technique for Custom On-Chip Interconnection Network Synthesis", In Proceedings of the 18th International Conference on VLSI Design (VLSID'05)

A Framework for Designing Congestion-Aware Deterministic Routing

Abbas Eslami Kiasari
Axel Jantsch
Zhonghai Lu

Paper 6

In the Proceedings of the 3rd International Workshop on Network-on-Chip Architectures (NoCArc), Held in conjunction with the 43rd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO-43), pp. 45-50, Atlanta, Georgia, USA, Dec. 2010.

A Framework for Designing Congestion-Aware Deterministic Routing

Abbas Eslami Kiasari, Axel Jantsch, and Zhonghai Lu
Royal Institute of Technology (KTH), Sweden
{kiasari, axel, zhonghai}@kth.se

ABSTRACT

In this paper, we present a system-level Congestion-Aware Routing (CAR) framework for designing minimal deterministic routing algorithms. CAR exploits the peculiarities of the application workload to spread the load evenly across the network. To this end, we first formulate an optimization problem of minimizing the level of congestion in the network and then use the simulated annealing heuristic to solve this problem. The proposed framework assures deadlock-free routing, even in the networks without virtual channels. Experiments with both synthetic and realistic workloads show the effectiveness of the CAR framework. Results show that maximum sustainable throughput of the network is improved by up to 205% for different applications and architectures.

Categories and Subject Descriptors

C.2.1 [Network Architecture and Design]: Network communications

General Terms

Algorithms, Design, Performance

1. INTRODUCTION

Thanks to high performance and low power budget of ASICs (application specific integrated circuits), they have been common components in the design of embedded systems-on-chip. Advances of semiconductor technology facilitate the integration of reconfigurable logic with ASIC modules in embedded systems-on-chip. Reconfigurable architectures are used as new alternatives for implementing a wide range of computationally intensive applications, such as DSP, multimedia and computer vision applications [1]. In the beginning of the current millennium, network-on-chip (NoC) emerged as a standard solution in the on-chip architectures [7][8]. In network-based systems, the performance of the communication infrastructure is critical, as it can represent the overall system performance bottleneck. The performance of networks depends heavily on the routing algorithm effectiveness, since it impacts all network metrics such as latency, throughput, and power dissipation.

Routing algorithms are generally categorized into deterministic and adaptive. A deterministic routing algorithm is oblivious of the dynamic network conditions and always provides the same path between a given source and destination pair. In contrast, in adaptive routing algorithms, besides source and destination addresses, network traffic variation plays an important role for selecting channels to forward packets. However, adaptive routing may cause packets to arrive out-of-order since they may be routed along different paths. The re-order buffers needed at the destination for ordering the packets impose large area and power on system [13]. Deterministic routers not only are more compact and faster than adaptive routers [4], but also guarantee in-order packet delivery. Therefore, it is not surprising that designers would like to use deterministic routing algorithms in the NoCs which suffer from limited silicon resources. However, in deterministic routing a packet cannot use alternative paths to avoid congested channels along its

route; this leads to degraded performance of the communication architecture at high levels of network throughput.

A well-designed routing algorithm utilizes the network resources uniformly as much as possible and avoids the congested channels, even in the presence of non-uniform traffic patterns, which are usual in the embedded systems. In this paper, we propose a system-level Congestion-Aware Routing (CAR) framework for designing minimal deterministic routing algorithms for network-based platforms. Especially, CAR is appropriate for reconfigurable embedded systems-on-chip which host several applications with high computational requirements and static workloads. Before the execution of a new application, the routing tables are configured with pre-computed routes, as well as other components in the system. After selecting the route and adding it to the packet, no further time is needed on routing at the intermediate nodes along the path. Due to advantages of table-based routing, it is one of the most widely used routing methods for implementing deterministic routing algorithm, e.g., IBM SP1 and SP2 [4].

To calculate the expected load on various channels in the network, CAR uses off-line analysis based on the global knowledge of application traffic. The results obtained from simulation experiments confirm that the proposed routing framework can find efficient routes for various networks and workloads.

The rest of the paper is organized as follows. We start by reviewing previous studies in Section 2. The CAR framework is proposed in Section 3. Experimental results in Section 4 show that our proposed approach can improve the system performance. Finally, concluding remarks are given in Section 5.

2. RELATED WORK

Turn model for designing partially adaptive routing algorithms for mesh and hypercube networks was proposed in [7]. Prohibiting minimum number of turns breaks all of the cycles and produces a deadlock-free routing algorithm. Turn model was used to develop the Odd-Even adaptive routing algorithm for meshes [3]. This model restricts the locations where some turns can be taken so that deadlock is avoided. In comparison with turn model, the degree of routing adaptivity provided by the Odd-Even routing is more even for different source-destination pairs.

DyAD routing scheme, which combines deterministic and adaptive routing, is proposed in [9] for NoCs, where the router works in deterministic mode when the network is not congested, and switches to adaptive mode when the network becomes congested. In [17] the authors extend routers of a network to measure their load and to send appropriate load information to their direct neighbours. The load information is used to decide in which direction a packet should be routed to avoid hot-spots. Recently, the authors in [14] present APSRA, a methodology to develop adaptive routing algorithms for NoCs that are specialized for an application or a set of concurrent applications. APSRA exploits the application-specific information regarding pairs of cores that communicate and other pairs that never communicate in the NoC platform to maximize communication adaptivity and performance.

Since all of these approaches are based on adaptive routing, they suffer from out-of-order packet delivery. Our proposed routing framework overcomes this problem while it spreads the load more evenly across the network.

Also, an application-aware oblivious routing is proposed in [11] that statically determines deadlock-free routes. The authors presented a mixed integer-linear programming approach and a heuristic approach for producing routes that minimize maximum channel load. However, in case of realistic workload, they did not study the effect of task mapping on their approach.

3. CAR FRAMEWORK

The CAR framework consists of 5 steps as its flowchart is shown in Figure 1. At first, we represent the architecture and application using *topology graph* (TG) and *communication graph* (CG), respectively. Then we construct the *channel dependency graph* (CDG) based on TG and CG. In the

third step, an acyclic CDG is extracted by deleting some edges from CDG to guarantee the deadlock freedom. After that, we find all possible shortest paths for each flow to create the routing space. Finally, we formulate an optimization problem over the routing space and solve it. In the following subsections, each step is described in detail.

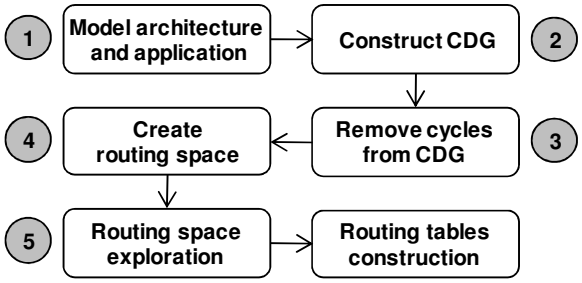


Figure 1: The flowchart of CAR framework

3.1 Model Architecture and Application

In order to characterize the network performance, a network model is essential. As shown in Figure 2, a directed graph, which is called topology graph (TG), can represent the topology of network architecture. Vertices and edges of TG show nodes and links of the network, respectively. Every node in TG contains a core and a router. Such a core is a local computing or a storage region.

An application can be modelled by a graph called communication graph (CG). CG is a directed graph, where each vertex represents one selected task, and each directed arc represents the communication volume from source task to destination task.

3.2 Construct Channel Dependency Graph

Dally and Seitz simplified designing deadlock-free routing algorithms with a proof that an acyclic channel dependency graph

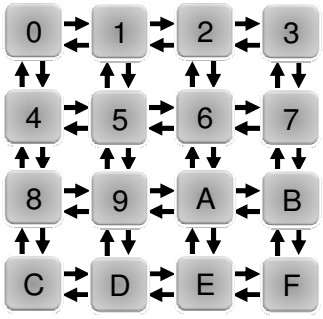
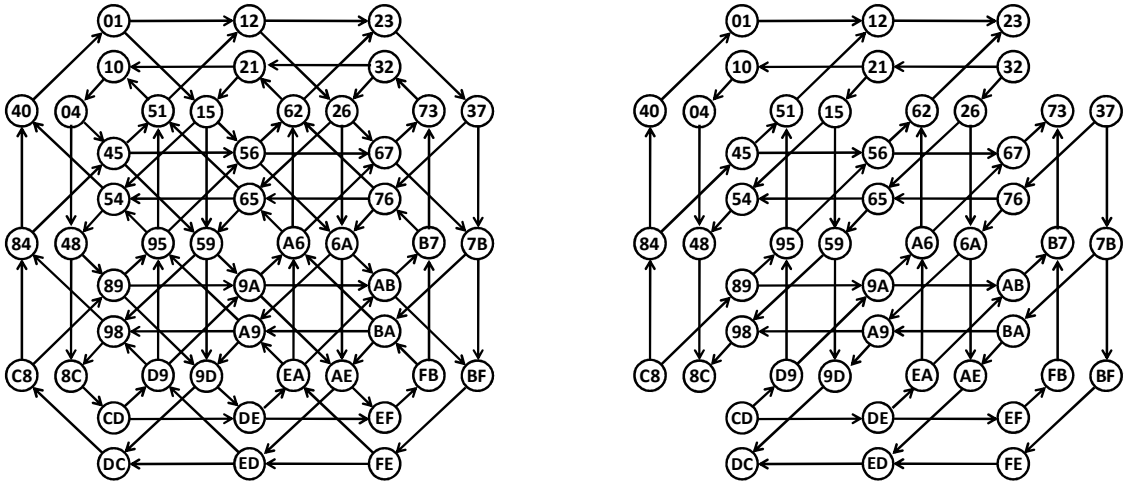


Figure 2: TG of a 4X4 mesh network

(CDG) guarantees deadlock freedom [5]. Each vertex of the CDG is a channel in TG. For instance, vertex 01 in Figure 3 corresponds to the channel from node 0 to node 1 in Figure 2. There is a directed edge from one vertex in CDG to another if a packet is permitted to use the second channel in TG immediately after the first one. To find the edges of a CDG, we use the *Dijkstra's algorithm* to find all shortest paths between source and destination of any flows in corresponding TG. CDG of a 4X4 mesh network (Figure 2) under minimal fully adaptive routing is shown in Figure 3.a, when any two nodes have the need to communicate such as in the uniform traffic pattern.



(a) (b)

Figure 3: CDG of 4X4 mesh network for minimal fully adaptive routing under (a) uniform and (b) transpose traffic patterns

3.3 Remove Cycles from CDG

Traditional routing algorithms, such as *dimension-order routing* (DOR) and turn model, extract an acyclic CDG by systematically removing some edges from CDG regardless of the traffic pattern. This may result in poor performance of routing algorithm due to prohibition of unnecessary turns. For instance, as shown in Figure 3.b, there is no cycle in CDG of 4X4 mesh network under transpose traffic pattern, which the node in row *i* and column *j* sends packets to the node in row *j* and column *i*. However, traditional routing algorithms conservatively remove some edges from CDG.

We modify the *depth-first-search* (*dfs*) algorithm to find cycles in a given CDG. Since we want to remove minimum number of edges, we delete an edge from CDG which is shared among more cycles. Note that, this edge is removed if the reachability of all flows is guaranteed. For example, in a CDG of 4X4 mesh network, shown in Figure 3.a, there are 6,982,870 cycles and the edge from vertex 40 to vertex 01 is shared among 5,041,173 cycles. Thus by removing this edge from CDG, the number of cycles is considerably reduced to 1,941,697. These steps are repeated again while there is a cycle in CDG. Table 1 shows the numbers of cycles found by CAR in CDG of different mesh networks. As it can be vividly seen, number of cycles is exponentially grown with the size of TG and it takes a long time to find all cycles in the CDG. Hence, we find cycles in CDG till certain number of cycles, and then remove an edge from CDG which is shared among more cycles.

Table1: Number of cycles in CDG of mesh networks.

TG	Number of cycles in corresponding CDG
Mesh (2X2)	2
Mesh (2X3)	8
Mesh (3X3)	292
Mesh (3X4)	14,232
Mesh (4X4)	6,982,870
Mesh (4X5)	3,656,892,444

3.4 Create Routing Space (RS)

In this step, we apply Dijkstra's algorithm to the acyclic CDG to find all shortest paths between source and destination of flows in corresponding TG and create a set of *f* flows $RS = \{F_1, F_2, \dots, F_f\}$ where *f* is the number of all flows in the system. $F_i = (\lambda_i, n_i, P_i)$, where λ_i and n_i are the packet generation rate and the number of available shortest paths for flow *i*, respectively. Also, P_i is itself a set and includes all n_i routes for flow *i*.

Usually more than one shortest path is available between two nodes $(n_i > 1)$ in the routing space *RS*, so it is reasonable to choose a path such that the load is evenly spread across the network. In the next subsection, we formulate an optimization problem over *RS* to find a suitable route for each flow and then use the simulated annealing heuristic to solve this problem.

3.5 Routing Space Exploration

In this subsection, we define an optimization problem to explore the routing space of *RS*. It is essential to define *decision variables* and *objective functions* in formulating the optimization problem. As previously mentioned, our goal is to select a path for flow *i* $(1 \le i \le f)$ among n_i available paths. Therefore, we define $X = \{x_1, x_2, \dots, x_f\}$ as decision variables in the space of *RS* where x_i refers to a path number for flow *i* $(1 \le x_i \le n_i)$. A routing algorithm prevents congestion in the network by balancing the load over network channels, so the standard deviation of channels throughput can be used as a criterion for load balance in the network. The more balanced the channel load, the smaller the standard deviation of channels throughput. Hence, we consider standard deviation of channels throughput as an objective function.

Assuming the network is not overloaded, the throughput of channel *c* in TG, θ_c, can be calculated using the general equation

$$\theta_c = \sum_{i=1}^{f} \lambda_i \times R(i, c) \quad (1)$$

where the $R(i, c)$ is a Boolean function and equals 1 if the flow *i* passes through channel *c* and equals 0 otherwise. Note that we assume a deterministic routing algorithm, thus the function of $R(i, c)$ can be predetermined, regardless of topology and routing algorithm. After computing θ_c for all channels in the TG, the standard deviation of channels throughput can be calculated using the following equation

$$\sigma = \sqrt{\frac{1}{L}\sum_{i=1}^{L}(\theta_i - \bar{\theta})^2} \quad (2)$$

where *L* is the number of channels in the network and $\bar{\theta}$ is the average throughput of all channels. If the load is completely balanced over the channels, then $\sigma = 0$. CAR framework uses the simulated annealing heuristic to minimize the objective function (σ) as described briefly in the following.

The name and inspiration of simulated annealing algorithm come from physical annealing technique in metallurgy. To simulate the physical annealing process, simulated annealing algorithm will randomly choose a neighbour solution to replace the current solution. As we mentioned before, $X = \{x_1, x_2, \dots, x_f\}$ is the set of decision variables where x_i is the path number for flow *i* $(1 \le x_i \le n_i)$. To choose a neighbour of *X*, we generate a random number *r* where $1 \le r \le f$ to choose a flow, and then generate another random number x_r^{new} where $1 \le x_r^{new} \le n_r$ and $x_r^{new} \ne x_r$ to choose another path for flow *r*. The new solution is accepted based on an equation that depends on the difference of the objective function values between the two states. We follow the Metropolis algorithm [2] as the acceptance criterion which accepts all downhill moves (from higher value to lower value) and probabilistically accepts uphill moves (from lower value to higher value). The acceptance of uphill moves allows saving the method from becoming stuck at a local minimum. Detailed information about simulated annealing approach can be found in [12].

4. Experimental Results

To evaluate the capability of CAR framework, we developed a discrete-event simulator that mimics the behaviour of routing algorithm in the networks at the flit level. Due to the popularity of the mesh network in NoC domain, our analysis focuses on this topology but CAR framework can be equally applied for other topologies without any change. We compare the performance of CAR with DOR which becomes XY routing algorithm in 2D mesh networks.

To achieve a high accuracy in the simulation results, we use the batch means method [13] for simulation output analysis. There are 10 batches and each batch includes 1000 up to 1,000,000 packets depending on the workload type, packet injection rate, and network size. Statistics gathering was inhibited for the first batch to avoid distortions due to the startup transient. The standard deviation of latency measurements is less than 1.8% of the mean value. As a result, the confidence level and confidence interval of simulation results are 0.99 and 0.02, respectively.

For the sake of comprehensive study, numerous validation experiments have been performed for several combinations of workload types and network size. In what follows, the capability of CAR will be assessed for both synthetic and realistic traffic patterns. Since their applications differ starkly in purpose, these classes of NoC have substantially different traffic patterns.

4.1 Synthetic Traffic

Synthetic traffic patterns used in this research include *uniform, transpose, shuffle, bit-complement,* and *bit-reversal* [4]. After developing models describing spatial traffic distributions, we should use an appropriate model to model the temporal traffic distribution. In the case of synthetic traffics, we use the Poisson process for modelling the temporal variation of traffic. It means that the time between two successive packet generations in a core is distributed exponentially. The Poisson model widely used in many performance analysis studies, and there are a large number of papers in many application domains that are based on this stochastic assumption.

The average packet latencies in the 4X4 and 8X8 mesh networks are plotted against offered load in the network in Figure 4 and Figure 5, respectively. We observe that under uniform and bit-complement traffic patterns CAR converges to DOR, because in such traffic patterns the standard deviation of channels throughput is minimum for DOR. This result is consistent with other results reported in [3][7][9][14]. The main reason is that the DOR distributes packets evenly in the long term [7]. Previous works, Odd-Even [3], turn model [7], DyAD [9], and APSRA [14] indicate that in the case of uniform traffic, their proposed approaches underperform DOR. However, as can be seen in Figure 4.a and 5.a, our proposed framework has the same performance as DOR for different traffic loads.

Figure 4.b and 4.c compare the latency of DOR and CAR in 4X4 mesh network under transpose and bit-reversal workloads, respectively. It can be vividly seen that CAR considerably outperforms DOR. In these cases, CAR can find routes for flows such that the standard deviation of channels throughput equals zero. This means that the load is completely balanced across the network channels. Also, in the case of 8x8 mesh network, CAR has better performance than DOR as shown in Figure 5.b and 5.c. Figure 4.d and 5.d reveal that under shuffle traffic pattern CAR slightly outperforms DOR.

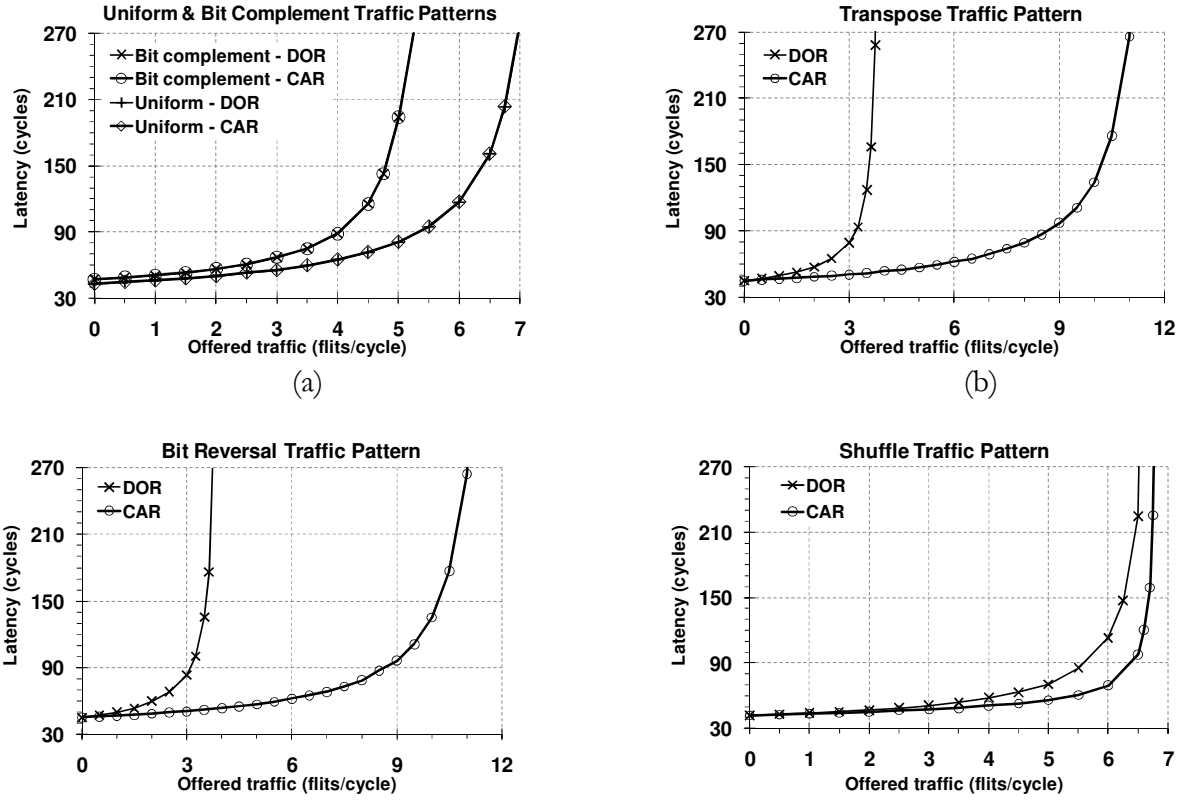


Figure 4: Average packet latency under (a) uniform and bit-complement, (b) transpose, (c) bit-reversal, and (d) shuffle traffic patterns in 4X4 mesh network

Table 2 shows the maximum sustainable throughput of the network for each workload and for each routing algorithm in 4X4 and 8X8 mesh networks. It also shows the percentage improvement of CAR over DOR and reveals that on average CAR outperforms DOR. The maximum load that the network is capable of handling using CAR is improved by up to 205%.
Also, the performance of CAR framework is compared against DyAD routing scheme [9] which combines deterministic and adaptive routing algorithms. We simulate the uniform and transpose workloads on the similar architecture (6X6 mesh network) and compare their improvement over DOR. Table 3 shows the percentage improvement of DyAD and CAR over DOR. In case of uniform workload, DyAD underperforms DOR while CAR has the same performance as DOR. In case of transpose traffic pattern, DyAD and CAR give about 62% and 56% improvement over DOR, respectively. This means that our deterministic routing policy can compete with adaptive routing policies (DyAD switches to adaptive mode under high traffic load) and meanwhile guarantees in-order packet delivery.

Table 2: Improvement in maximum sustainable throughput of CAR as compared to DOR for different synthetic workloads.

Workload	4X4 mesh network			8X8 mesh network		
	DOR	CAR	Impr.	DOR	CAR	Impr.
Uniform	7.4	7.4	0	15.9	15.9	0
Transpose	3.8	11.6	205%	7.7	10.3	34%
Bit-comp.	5.6	5.6	0	8.8	8.8	0
Bit-rev.	3.8	11.6	205%	7.6	9.0	18%
Shuffle	6.6	6.9	5%	12.2	13.1	7%

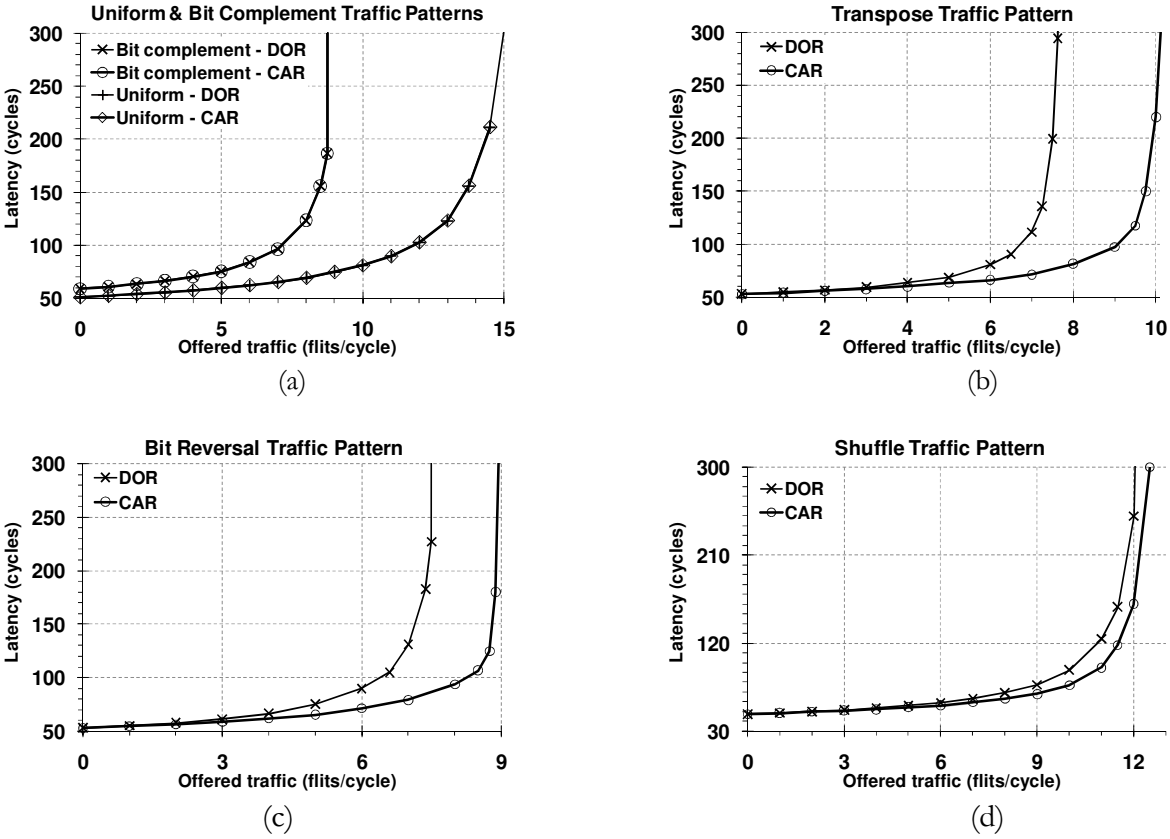


Figure 5: Average packet latency under (a) uniform and bit-complement, (b) transpose, (c) bit-reversal, and (d) shuffle traffic patterns in 8x8 mesh network

Table 3: Improvement in maximum sustainable throughput of DyAD and CAR over DOR.

Workload	Improvement over DOR	
	DyAD	CAR
Uniform	-21%	0
Transpose	62%	56%

4.2 Realistic Traffic

In case of realistic traffic, we consider two virtual channels for links to show the consistency of proposed framework with multiple virtual channel routing. As realistic communication scenarios, we consider a generic multimedia system (MMS) and the video object plane decoder (VOPD) application. MMS includes an H.263 video encoder, an H.263 video decoder, an mp3 audio encoder, and an mp3 audio decoder [10]. The communication volume requirements of this application are summarized in [10]. VOPD is an application used for MPEG-4 video decoding and its communication graph is available in [18]. Several studies reported the existence of bursty packet injection in the on-chip interconnection networks for multimedia traffic [16][19]. Poisson process is not the appropriate model in case of bursty traffic; consequently, we used two-state Markov modulated process as stochastic traffic generators to model the bursty nature of the application traffic [4]. The two states represent an "on" and "off" mode for injection process with average communication bandwidth matching the applications' average communication bandwidth.

Since in such systems, there are various types of cores with different bandwidth requirements, placement of tasks on a chip has strong effect on the system performance. To find a suitable mapping of these applications, we formulate another optimization problem to prune the large design space in a short time and then again use the simulated annealing heuristic to find a suitable mapping vector.

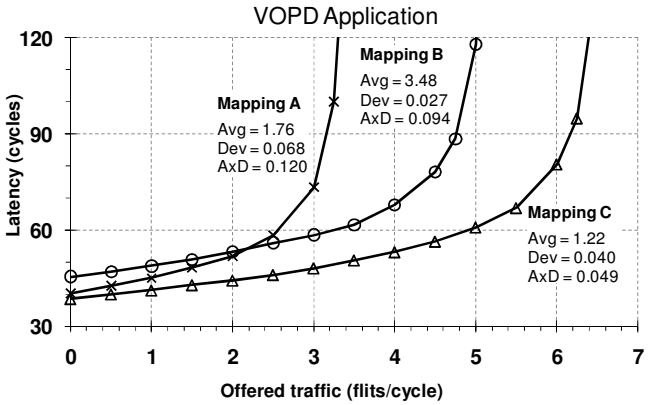


Figure 6: Average packet latency of VOPD application for three different mapping configurations vs. offered load

An efficient mapping tries to balance load over channels (minimizes the standard deviation of channels throughput) and also keeps the average hop count as small as possible. However, minimizing the standard deviation and minimizing the average hop are not always in the same direction. Therefore, to improve load balance, we have to increase the average path length [4]. To show how these parameters, average hop and standard deviation of channels throughput, affect the communication latency, we consider three different task mappings of VOPD application to the tiles of a 4X4 mesh on-chip network. Then, the system is simulated for these mapping configurations and the corresponding average packet latency values are plotted against offered load in Figure 6. Although the average hop in mapping A (1.76) is about half of the average hop in mapping B (3.48), mapping A underperforms mapping B. This is due to more balanced load in mapping B (the standard deviation of channels throughput in mapping B (0.027) is less than the standard deviation in mapping A (0.068)). On the other hand, the load in mapping B is more balanced in comparison to mapping C. However, due to smaller average hop in mapping C, it outperforms mapping B for all levels of network throughput. Thus, we define a new criterion named AxD (Average hop X standard Deviation) and use it as the objective function in the optimization problem of congestion-aware mapping (similar to congestion-aware routing).

Initially, we map task *i* to node *i* and then try to minimize the AxD through the simulated annealing approach. Figure 7.a shows that in the case of MMS application and DOR, for the initial mapping M1, AxD equals 0.57 and after a certain number of tries, the mapping vector converges to the mapping M4 with AxD = 0.04. Furthermore, AxD values for mappings M2 and M3, which are two local minimum points in simulated annealing process, are shown in the figure.

After the mapping phase, we apply the CAR framework to these four mapping vectors. Figure 7.a reveals that in case of mapping M1, CAR can significantly reduce the AxD from 0.57 to 0.24. This great difference is due to the unbalanced load of DOR. However, for more efficient mapping vectors (M2, M3, and M4), we achieve less improvement. Specially, in the case of best mapping (M4), AxD is reduced insignificantly from 0.0397 to 0.0395. It is reasonable that DOR is congestion-aware for the best mapping, because during the mapping problem solving process, we fix the routing policy to DOR and strive to minimize AxD for this routing policy. Figure 7.b shows that the simulation results confirm this conclusion. In the case of mapping M1, CAR significantly outperforms DOR, but in the case of M4, the latency is the same for both DOR and

CAR. Likewise, as shown in Figure 7.c and 7.d, for the VOPD application, the analysis result is the same as MMS application.
Figure 7.b and 7.d reveals that in case of application-specific traffic patterns, the improvement in the performance of the routing schemes highly depends on how the application tasks are mapped to the topology. This fact was not considered in the related works such as [11]. Also, Table 4 reports the maximum acceptable traffic for different mapping vectors under MMS and VOPD workloads. The better mapping vector results in smaller improvement in the saturation point.

Nowadays, in embedded systems-on-chip there are several different types of cores including DSPs, embedded DRAMs, ASICs, and generic processors which their places are fixed on the chip. On the other hand, such a system hosts several applications with completely different workload. Furthermore, modern embedded devices allow users to install applications at run-time, so a complete analysis of such systems is not feasible during design phase. As a result, it is not feasible to map all applications such that the load is balanced for all of them with specific routing algorithm and we should balance the load in routing phase.

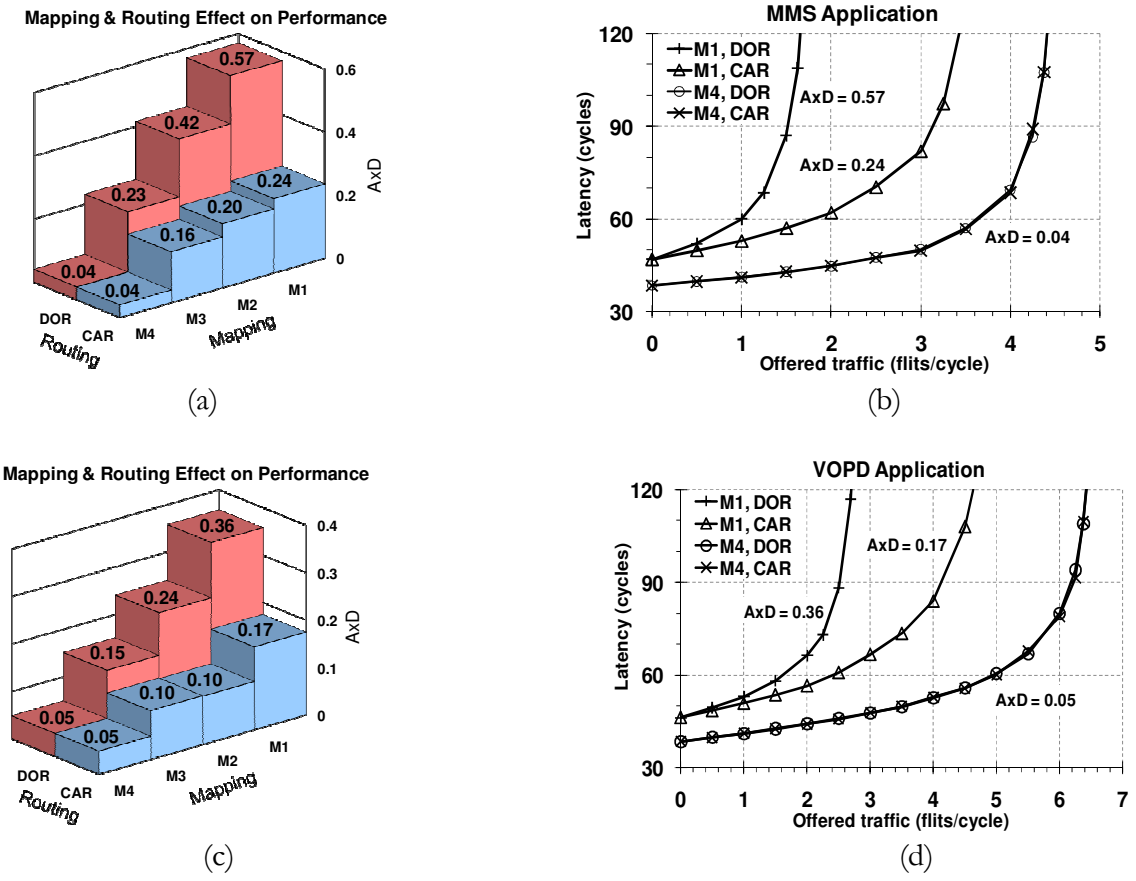


Figure 7: (a) The effect of mapping and routing on the performance of MMS application, (b) average packet latency for different mapping and routing schemes in the case of MMS workload, (c) the effect of mapping and routing on the performance of VOPD application, (d) average packet latency for different mapping and routing schemes in the case of VOPD workload

Table 4: Improvement in maximum sustainable throughput of CAR as compared to DOR for realistic applications.

Mapping	MMS application			VOPD application		
	DOR	CAR	Impr.	DOR	CAR	Impr.
M1	1.8	3.6	100%	2.9	4.9	69%
M2	2.1	3.9	86%	3.5	5.2	49%
M3	3.2	4.3	34%	5.1	6.4	25%
M4	4.7	4.7	0	6.7	6.7	0

In this section we used the CAR framework to find low congestion routes in the mesh network. Due to simplicity, regularity, and low cost merits of 2D mesh topology, it is the most popular one in the field of NoC. However, for large and 3D NoCs, which will be popular in the future, the communication in mesh architecture takes a long time. In the next subsection we use CAR to find deadlock-free paths in an arbitrary topology.

4.3 Find Routes in an Arbitrary Topology

To show the capability of CAR framework to find deadlock-free routes in an arbitrary topology, we consider the topology shown in Figure 8.a. CAR reports that under uniform traffic pattern there are 2 cycles in the corresponding CDG and by prohibiting turns 52 to 21 and 87 to 73 (shown in Figure 8.b) the deadlock-freedom is guaranteed.

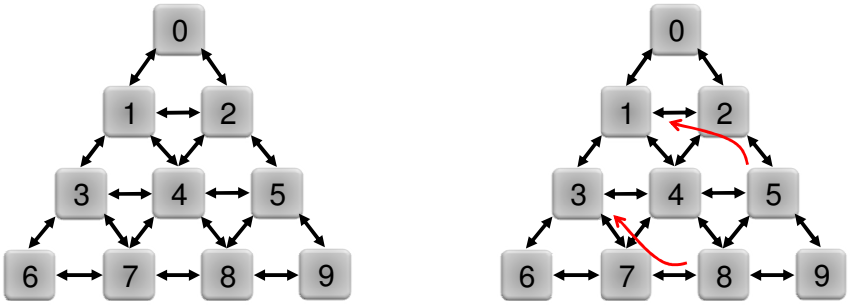


Figure 8: (a) A custom topology and (b) prohibited turns

Table 5 shows the routing table for node 0 of the topology in Figure 8.a. Each route in the table specifies a path from node 0 to a given destination as channels name. SE, SW, and EJ specify South East, South West, and ejection channels, respectively. To route a packet, the routing table is indexed by destination address to look up the pre-computed route by CAR. This route is then added to the packet. Since there are 7 channels in this network (E, S, NE, NW, SE, SW, and EJ), they can be encoded as 3-bit binary numbers. Also, there are techniques to reduce the size of routing tables [4][14].

Table 5: Routing table for node 0 of topology in Figure 8.a.

dst.	route	dst.	Route
0	No packet	5	SE, SE, EJ
1	SW, EJ	6	SW, SW, SW, EJ
2	SE, EJ	7	SE, SW, SW, EJ
3	SW, SW, EJ	8	SW, SE, SE, EJ
4	SW, SE, EJ	9	SE, SE, SE, EJ

5. Conclusion

On-chip packet routing is extremely crucial because it heavily affects performance and power. This calls for a great need of routing optimization. However, due to the diverse connectivity enabled by a network and the interferences in sharing network buffers and links, determining good routing paths, which are minimal and deadlock free for traffic flows, is nontrivial. In this paper, we have addressed the congestion-aware routing problem. With the analysis technique, we first estimate the congestion level in the network, and then embed this analysis technique into the loop of optimizing routing paths so as to quickly find deterministic routing paths for all traffic flows while minimizing the congestion level. Our experiments with both synthetic and realistic workloads show that we can extract high quality solutions with small computational time.

The proposed framework is appropriate for reconfigurable embedded systems-on-chip which run several applications with regular and repetitive computations on large set of data, e.g., multimedia and computer vision applications. CAR can not only design minimal and deterministic routing, but also can implement non-minimal and deadlock-free fully adaptive routing without virtual channels in arbitrary topology.

Reference

[1] K. Bondalapati and V.K. Prasanna, "Reconfigurable Computing Systems," *Proceedings of the IEEE*, 90(7):1201-1217, 2002.

[2] O. Catoni, "Metropolis, Simulated Annealing, and Iterated Energy Transformation Algorithms, Theory and Experiments," *Journal of Complexity* 12(4):595-623, 1996.

[3] G.-M. Chiu, "The Odd-Even Turn Model for Adaptive Routing," *IEEE Transactions on Parallel and Distributed Systems*, 11(7):729-738, 2000.

[4] W. J. Dally and B. Towles, Principles and Practices of Interconnection Networks, Morgan Kaufmann Publishers Inc., First edition, 2004.

[5] W.J. Dally and C.L. Seitz, "Deadlock-Free Message Routing in Multiprocessor Interconnection Networks," *IEEE Transactions on Computers*, 36(5):547-553, 1987.

[6] C.J. Glass and L.M. Ni, "The Turn Model for Adaptive Routing," *Journal of the Association for Computing Machinery*, 41(5):874-902, 1994.

[7] P. Guerrier and A. Greiner, "A Generic Architecture for on-chip Packet-Switched Interconnections," *Proceedings of the Design, Automation, and Test in Europe*, pp. 250-256, 2000.

[8] A. Hemani, *et. al.*, "Network on a Chip: An Architecture for Billion Transistor Era," *Proceedings of the IEEE NorChip*, pp. 166-173, 2000.

[9] J. Hu and R. Marculescu, "DyAD - Smart Routing for Networks-on-Chip," *Proceedings of the Design Automation Conference,* pp. 260-263, 2004.

[10] J. Hu and R. Marculescu, "Energy- and Performance-Aware Mapping for Regular NoC Architectures," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 24(4):551-562, 2005.

[11] M. A. Kinsy, *et. al.*, "Application-Aware Deadlock-free Obli- vious Routing," *Proceedings of the ISCA*, pp. 208-219, 2009.

[12] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. "Optimization by Simulated Annealing," *Science*, 220(4598):671–680, 1983.

[13] S. Murali, *et. al.,* "Analysis of Error Recovery Schemes for Networks on Chips", *IEEE Design and Test of Computers*, 22(5): 434-442, 2005.

[14] M. Palesi, *et. al.,* "Application Specific Routing Algorithms for Networks on Chip," *IEEE Transactions on Parallel and Distributed Systems*, 20(3):316-330, 2009.

[15] K. Pawlikowski, "Steady-State Simulation of Queueing Processes: A Survey of Problems and Solutions," *ACM Computing Surveys*, 22(2):123-170, 1990.

[16] V. Soteriou, H. Wang, L.-S. Peh, "A Statistical Traffic Model for On-Chip Interconnection Networks," *Proceedings of the MASCOTS*, pp. 104-116, 2006.

[17] W. Trumler, *et. al.*, "Self-optimized Routing in a Network-on-a-Chip," *IFIP World Computer Congress*, pp. 199-212, 2008.
[18] E.B. van der Tol and E.G. Jaspers, "Mapping of MPEG-4 Decoding on a Flexible Architecture Platform," *SPIE*, vol. 4674, pp. 1-13, 2002.
[19] G. Varatkar and R. Marculescu, "Traffic Analysis for On-chip Networks Design of Multimedia Applications," *Procee- dings of the Design Automation Conference,* pp. 795-800, 2002.

Analytical Approaches for Performance Evaluation of Networks-on-Chip

Abbas Eslami Kiasari
Axel Jantsch
Marco Bekooij
Alan Burns
Zhonghai Lu

In the Proceedings of the International Conference on Compilers, Architectures and Synthesis for Embedded Systems (CASES), pp. 211-212, Tampere, Finland, Oct. 2012.

Paper 7

Analytical Approaches for Performance Evaluation of Networks-on-Chip

Abbas Eslami Kiasari
KTH Royal Institute of Technology, Sweden
kiasari@kth.se

Axel Jantsch
KTH Royal Institute of Technology, Sweden
axel@kth.se

Marco Bekooij
University of Twente, The Netherlands
marco.bekooij@nxp.com

Alan Burns
University of York, United Kingdom
alan.burns@york.ac.uk

Zhonghai Lu
KTH Royal Institute of Technology, Sweden
zhonghai@kth.se

ABSTRACT

This tutorial reviews four popular mathematical formalisms – *dataflow analysis, schedulability analysis, network calculus,* and *queueing theory* – and how they have been applied to the analysis of Network-on-Chip (NoC) performance. We review the basic concepts and results of each formalism and provide examples of how they have been used in on-chip communication performance analysis. The tutorial also discusses the respective strengths and weaknesses of each formalism, their suitability for a specific purpose, and the attempts that have been made to bridge these analytical approaches. Finally, we conclude the tutorial by discussing open research issues.

Categories and Subject Descriptors

C.4 [Performance of Systems]: Design studies, Modeling techniques, Performance attributes

General Terms

Design, Performance

Keywords

System-on-Chip, Network-on-Chip, Design methodology, Performance evaluation, Analytical modeling

1. INTRODUCTION

In modern system-on-chip (SoC), the on-chip communication infrastructure or network-on-chip (NoC) is a dominant factor for design, validation and performance analysis. SoC designers are interested in NoC performance evaluation since their goal is either to provide a minimum level of performance at lowest possible cost, or to provide the highest performance at a given cost. In both cases a reliable measure of performance is indispensible. However, in the first case the focus is typically on *worst-case performance*, while in the latter case the *average-case performance* is the main metric [8]. In real-time systems such as automotive or avionic applications, the worst-case execution time is of particular concern since it is important to know how much time might be needed in the worst-case to guarantee that the task will always finish its jobs before the predetermined deadline. However, the worst-case-based design results in resource over-dimensioning. Therefore, the average-case-based design methods are usually used for non-time critical applications to have a more efficient system.

Performance estimation tools can be classified in *simulation models* and *analytical models*. SoC designers have tackled performance analysis by exploring the design space using detailed simulations. Simulation tools are flexible and accurate, but often have to be complemented by an analytical performance modeling approach. In particular, analytical models can analyze the worst-case. An appropriate analytical model can estimate very early in the design phase the desired performance metrics in a fraction of time that simulation would take. Although the use of high-level models conceals a lot of complex technological aspects, it facilitates fast exploration of the NoCs

design space. Also, the analytical models provide not only the timing properties of the system, but also useful feedback about the system behavior. Hence, it can be invoked in any optimization loop for NoCs for fast and accurate performance estimations.

2. OVERVIEW OF THE TUTORIAL

This tutorial reviews the applicability and the application of dataflow models, schedulability analysis, network calculus, and queueing theory to NoC performance analysis. The key message of each presentation is described in the following subsections.

2.1 Dataflow Models (Marco Bekooij)

Timed dataflow models have been successfully applied to the derivation of the minimum throughput and maximum latency of network-on-chips [5]. Furthermore, these models are used to compute trade-offs between allocated bandwidth of the network connections and the required capacity of the buffers in the network [11]. Flow-control on the network connections results in cyclic dependencies in these dataflow models. However, such cyclic dependencies do not complicate dataflow analysis significantly. Also, the effects of starvation free arbitration are included in the dataflow models [13].

Dataflow models of networks can be created at different levels of abstraction depending on the required conciseness and accuracy of the model. Conservativeness of these levels of abstraction can be shown by making use of the earlier-the-better refinement relation and its transitivity property [4]. This refinement relation also implies that for proving the temporal requirements of a network, it suffices to show that an admissible schedule exists that adheres to these requirements. Approximation algorithms have been developed that compute these admissible schedules in polynomial time [12]. These algorithms are based on convex programming.

2.2 Schedulability Analysis (Alan Burns)

Scheduling analysis (SA) is a mathematical formalism used to confirm that all deadlines will be met in a real-time system. SA is usually applied to application tasks running on one or more CPUs/cores; but it is a general framework that allows the worst-case behavior of systems to be evaluated. Usually within SA, tasks are repetitive and are either released periodically or sporadically. Tasks can also suffer release jitter.

In this section we introduce a form of SA known as Response-Time Analysis (RTA) for analyzing resources that are scheduled by the common fixed priority dispatching policy. We then show how this analysis can be applied to determine the worst-case latencies for messages on a SoC. The analysis is then used to minimize the number of priority levels (and hence virtual channels) needed to deliver a system in which all messages are delivered by their deadlines. Background on the techniques to be introduced in this section can be found in standard textbooks [1]. The application of RTA to NoC message scheduling is described in [10].

2.3 Network Calculus (Zhonghai Lu)

Network calculus dealing with queuing systems is a formalism for design, analysis and implementation of performance guarantees in communication networks. The research was pioneered by Cruz in his seminal paper [3]. Chang systematically studied this subject [2]. In [6], stochastic network calculus generalizes the deterministic network calculus. Network calculus has been very successful when applied to achieve per-node and end-to-end QoS guarantees in Asynchronous Transfer Mode (ATM) networks, and Internet for both differentiated and integrated services. Recently it is applied to embedded real-time systems, off-chip networks such as wireless sensor networks, and on-chip networks [9].

This tutorial introduces the basics of network calculus within the context of on-chip networks. We begin by introducing the basic concepts such as arrival and service curves of network calculus to uncover the foundation for its elegance. Afterwards, we explain how closed-form formulas for calculating packet delay and backlog bounds can be obtained. With a clear-box approach on an

example, we then orient our attention to its application to on-chip networks, analyzing service curves of an on-chip router and a concatenation of routers and further deriving per-flow end-to-end delay bound formula. Finally we give a short summary, reviewing its strength and pointing out future perspectives.

2.4 Queueing Theory (Abbas Eslami Kiasari)

Queueing theory is a branch of probability theory which is concerned with the mathematical modeling and analysis of systems that provide service to stochastic demands. Typically, a queueing model represents a system by probability models of customers' arrival time and service time. Since queueing theory deals with probability models, it is used to compute average-case performance metrics such as average packet latency, average throughput, average energy and power consumption, and average resource utilization.

This section starts with an introduction to queueing theory which demonstrates how to model events and resources in packet-switched networks. Then, we continue the tutorial by briefly reviewing related research where queueing theory is used for performance evaluation and optimization of NoCs. Using a state-of-the-art queueing model [7], we give a numerical example to estimate the average latency of packets in NoCs.

2.5 Bridging the Formalisms (Axel Jantsch)

The general trade-off between abstraction and accuracy can be observed in the comparison between these four formalisms. Since each of the reviewed formalisms has different advantages and difficulties, and since they also partially differ in purpose, none of them can easily replace all others. There are definitely point problems for each formalism that are worthy for further studies, but research on integrated approaches to the problems of system performance analysis is most urgent. Although each formalism can be extended in various directions, these extensions typically run into problems of complex mathematics or they are perceived to be unnatural and cumbersome. Therefore, we believe that comprehensive frameworks that combine two or more formalisms would be most desirable.

3. REFERENCE

[1] A. Burns and A. Wellings. *Real-Time Systems and Programming Languages*. Addison-Wesley, 4th Ed., 2009.

[2] C. S. Chang. *Performance Guarantees in Communication Networks*. Springer-Verlag, 2000.

[3] R. L. Cruz. A calculus for network delay, part I: network elements in isolation; part II: network analysis. *IEEE Transactions on Information Theory*, 37(1):114-141, 1991.

[4] M. Geilen, S. Tripakis, and M. Wiggers. The earlier the better: a theory of timed actor interfaces. In *Proceedings of the ACM International Conference on Hybrid Systems: Computation and Control*, pages 23-32, 2011.

[5] A. Hansson, M. Wiggers, A. Moonen, K. Goossens, and M. Bekooij. Enabling application-level performance guarantees in network-based systems on chip by applying dataflow analysis. *IET Computers & Digital Techniques*, 3(5):398- 412, 2009.

[6] Y. Jiang and Y. Liu. *Stochastic Network Calculus*. Springer, 2008.

[7] A. E. Kiasari, Z. Lu, and A. Jantsch. An analytical latency model for networks-on-chip. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, Jan. 2012. doi: 10.1109/TVLSI.2011.2178620

[8] A. E. Kiasari, A. Jantsch, and Z. Lu. Mathematical formalisms for performance evaluation of networks-on-chip. *Accepted for publication in the ACM Computing Surveys.*

[9] Y. Qian, Z. Lu, and W. Dou. Analysis of worst-case delay bounds for on-chip packet-switching networks. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems,* 29(5):802-815, 2010.

[10] Z. Shi and A. Burns. Real-time communication analysis for on-chip networks with wormhole switching. In *Proceeding of the IEEE International Symposium on Networks-on-Chip (NoCS),* pages 161-170, 2008.

[11] M. Wiggers, M. Bekooij, M. Geilen, and T. Basten. Simultaneous budget and buffer size computation for throughput constraint task graphs. In *Proceedings of the Design, Automation and Test in Europe Conference and Exhibition (DATE)*, pages 1669-1672, 2010.
[12] M. Wiggers, M. Bekooij, and G. Smit. Efficient computation of buffer capacities for cyclo-static dataflow graphs. In *Proceedings of the Design Automation Conference (DAC)*, pages 658-663, 2007.
[13] M. Wiggers, M. Bekooij, and G. Smit. Monotonicity and run-time scheduling. In *Proceedings of the International Conference on Embedded Software*, pages 177-186, 2009.

Power-Efficient Deterministic and Adaptive Routing in Torus Networks-on-Chip

Dara Rahmati
Hamid Sarbazi-Azad
Shaahin Hessabi
Abbas Eslami Kiasari

Microprocessors and Microsystems: Embedded Hardware Design, vol. 36, no. 7, pp. 571-585, Oct. 2012.

Paper 8

Power-Efficient Deterministic and Adaptive Routing in Torus Networks-on-Chip

Dara Rahmati [a], Hamid Sarbazi-Azad [a], Shaahin Hessabi [a], Abbas Eslami Kiasari [b]
[a] Department of Computer Engineering, Sharif University of Technology, Iran
[b] School of Information and Communication Technology, KTH, Sweden

Abstract
Modern SoC architectures use NoCs for high-speed inter-IP communication. For NoC architectures, high-performance efficient routing algorithms with low power consumption are essential for real-time applications. NoCs with mesh and torus interconnection topologies are now popular due to their simple structures. A torus NoC is very similar to the mesh NoC, but has rather smaller diameter. For a routing algorithm to be deadlock-free in a torus, at least two virtual channels per physical channel must be used to avoid cyclic channel dependencies due to the warp-around links; however, in a mesh network deadlock freedom can be insured using only one virtual channel. The employed number of virtual channels is important since it has a direct effect on the power consumption of NoCs. In this paper, we propose a novel systematic approach for designing deadlock-free routing algorithms for torus NoCs. Using this method a new deterministic routing algorithm (called TRANC) is proposed that uses only one virtual channel per physical channel in torus NoCs. We also propose an algorithmic mapping that enables extracting TRANC-based routing algorithms from existing routing algorithms, which can be both deterministic and adaptive. The simulation results show power consumption and performance improvements when using the proposed algorithms.

Keywords: SoC, NoC, Torus, Mesh, Performance evaluation, Power consumption, Routing, Adaptive, Deterministic, Virtual channel, Deadlock, VHDL.

1. Introduction

There is a large body of work on performance evaluation and design tradeoffs of multicomputer interconnection networks and NoC architectures [1-18]. Although performance (e.g. network latency and throughput) is the key factor in designing multicomputer interconnection networks, in the design and implementation of NoC architectures, power consumption is the most important factor while performance is the second important measure. There have been several works reported to design performance-power efficient NoCs [19-24]. Chiu, in [15], introduces a new turn model based routing algorithm, named Odd-Even, for designing adaptive wormhole routing algorithms for meshes without virtual channels. The model restricts the locations, where some turns can be taken due to the state of the packet, to an even or an odd numbered column. This way, the proposed routing algorithm avoids deadlock. In [22], the effect of traffic localization on energy dissipation in NoC-based interconnects is investigated through system level simulation. Authors show that energy reduction of up to 50% can be achieved by exploiting communication locality. In [24], authors compare mesh and torus topologies under different routing algorithms and traffic models for their performance and power consumption.

The simplest and hence widely used routing algorithm for the mesh NoCs is XY routing [11,16,17,25]. In XY routing algorithm, the packet is first routed across X axis and then across Y axis until it reaches the destination node (as shown in Fig.1). However, applying XY routing for the torus topology may cause deadlock due to the channel dependency in each dimension between different messages [11] as a result of added wrap-around links (with respect to the mesh topology). By using more than one virtual channel, there will be the flexibility of designing

different deadlock-free routing algorithms for the cost of extra hardware complexity, more area, and thus higher power consumption.

In order to have a deadlock-free routing algorithm in the torus, there should be at least two virtual channels per physical channel to break cycles in the channels dependency graph into spirals [11, 13, 18]. This is not the case when a mesh is used that requires only one virtual channel to prevent deadlocks. However, it is shown that the number of virtual channels plays a crucial role in power consumption [26, 27, 30].

In this paper, we introduce IRN (Interconnection Routing Notation), a novel map-based systematic approach to design routing algorithms for mesh and torus NoCs. This notation can also be extended for other interconnection topologies. We then use IRN and propose a deadlock-free deterministic routing algorithm, called *TRANC* (Torus Routing Algorithm for NoCs), for a torus NoC that uses only one virtual channel per physical channel. The TRANC routing algorithm enjoys low power consumption level of a mesh NoC while providing a comparable performance to a torus NoC using XY routing with 2 virtual channels per physical channel. It even exhibits better performance for light traffic because of a zero switching time between virtual channels compared to a torus NoC using 2 virtual channels implementing XY routing. There is a slight decrease in the performance of TRANC for heavy traffics (and near the saturation point) when compared to XY routing in the torus.

We have significantly extended our previous work [33] by proposing a new mapping method that extracts a new TRANC-based routing algorithm. In this method, existing routing algorithms for mesh networks can be easily converted to their counterpart TRANC algorithms for torus networks. As an example of using this method, we have added partial and full adaptivity to TRANC in a torus (called *Adaptive-TRANC*). The simulation results show the efficiency of the new algorithms compared to their original counterparts. Our analysis considers real application traffic patterns, in addition to synthetic loads, and thus verifying the efficiency of the proposed algorithms in different working scenarios.

2. Routing in the Mesh and Torus NoCs

An $n \times n$ mesh or torus NoC consists of n^2 nodes arranged in a two-dimensional grid structure. Each node is addressed as (x,y) where x indicates the position of the node along X dimension and y indicates its position along Y dimension. A node can have a neighboring node in the increasing and decreasing directions (positive and negative directions) in each dimension. The first node and the last node (border nodes) in each dimension are linked using a wrap-around link in the torus NoC, while such wrap-around links do not exist in the mesh NoC. Fig. 1 shows a 4x4 mesh NoC and a 4x4 torus NoC. Each node in the network consists of two parts: IP (intellectual property) and Router.

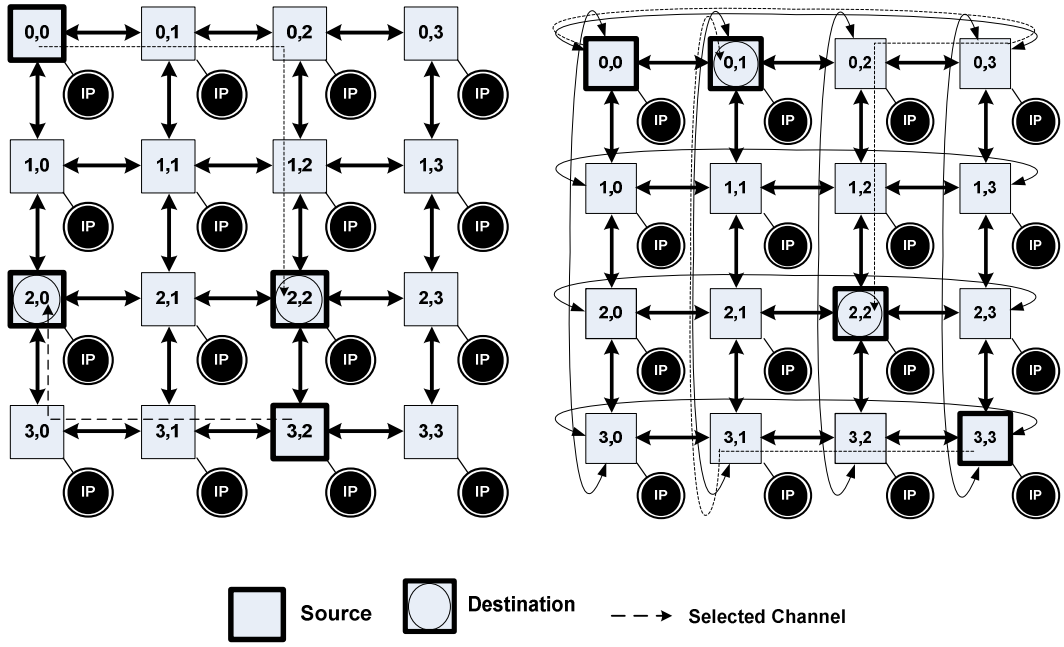


Figure 1. A 4x4 mesh NoC (left) and a 4x4 torus NoC (right).

2.1. Node structure in mesh and torus NoCs

A cycle accurate and synthesizable VHDL hardware model for mesh and torus NoCs has been implemented and several different topologies have been designed and tested based on it. The top most shared component in this hardware model is the NoC node where *IP* and *router* are the main components. Fig. 2 shows the node structure in the implemented model.

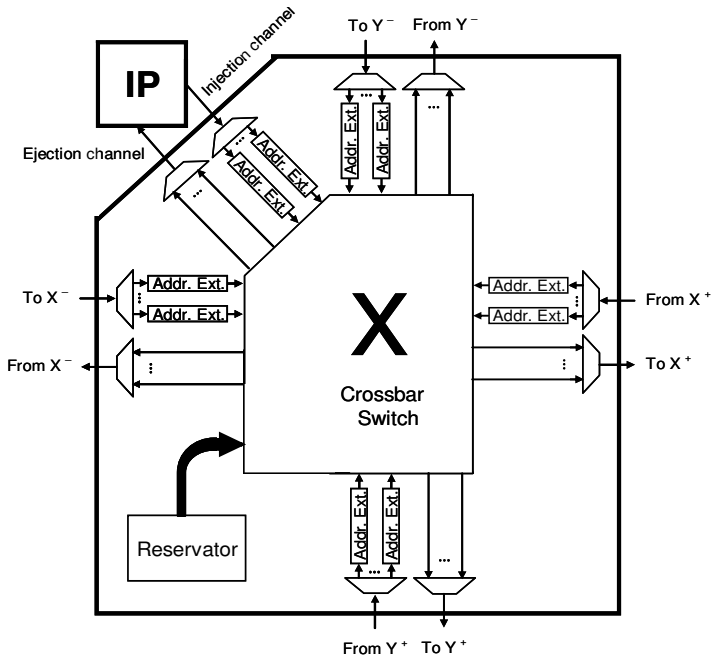


Figure 2. Node structure in a mesh or trous NoC

The IP can be a processor with some local memory, a memory module in a shared memory architecture, or any other module that can send/receive packets over the network. In our implementation, an IP generates packets based on a traffic model, such as uniform distribution, for determining packet destinations. Also, each IP generates packets on intervals based on a

Poisson distribution when working with synthetic workloads or the exact generation time indicated by real application traffic loads.

The router has five input and five output ports or channels. A node uses four input and four output channels to connect to its neighboring nodes, two per dimension, one in each direction. The remaining channels are used by the IP to inject/eject messages to/from the network, respectively. Messages generated by the IP are injected into the network through the injection channel. Messages that arrive at a destination node are transferred to the local IP through ejection channel. The bandwidth of each channel is shared among a number of virtual channels. The hardware implementation of the router consists of several different units such as *Address Extractor* which determines and manipulates the packet headers and contains some buffer (of few flits) for each incoming virtual channel. It should be noted that the more the number of virtual channels is, the more complex the node structure is. There are *Multiplexer* and *De-Multiplexer* units which handle the virtual channel operations, *Selector* unit which applies the virtual channel selection rule, *Crossbar switch* that can simultaneously connect multiple input channels to multiple output channels given that there is no contention over the output channels. *Reservator* unit which controls the crossbar switch and other related sub-modules. When a specific topology like mesh or torus is supposed to be modeled by such components, a top-level wrapper module is implemented that connects several nodes of this type to each other based on the structure of the specified topology. Based on this hardware model, different cases of mesh and torus topologies have been synthesized and simulated to extract accurate quantities, e.g. average message latency and power consumption values.

2.2. Routing Algorithms

Fig. 1 shows the topological structure of a 4x4 mesh and a 4x4 torus. Examples of XY routing are also shown in each network (the route is indicated as dashed lines). The XY routing for mesh NoCs is straightforward: a message (or packet) first traverses its route towards its destination across X axis and then across Y axis. It is easy to see that such a routing algorithm prevents cyclic dependency in reserving and using network channels by messages [17]. The case is however different for the torus NoCs, where wraparound links can clearly make cyclic channel dependency resulting in deadlock situation. The straightforward deadlock-free XY routing algorithm for this case needs at least two virtual channels per physical channel. In XY routing algorithm for the torus, the packet first traverses X dimension and then Y dimension (as in mesh NoCs) using two virtual channels; it uses the first virtual channel before reaching a wraparound link, thereafter it uses the second virtual channel until it reaches the destination node. Thus, XY routing in the mesh NoCs requires one virtual channel per physical channel while it requires 2 virtual channels per physical channel in torus NoCs.

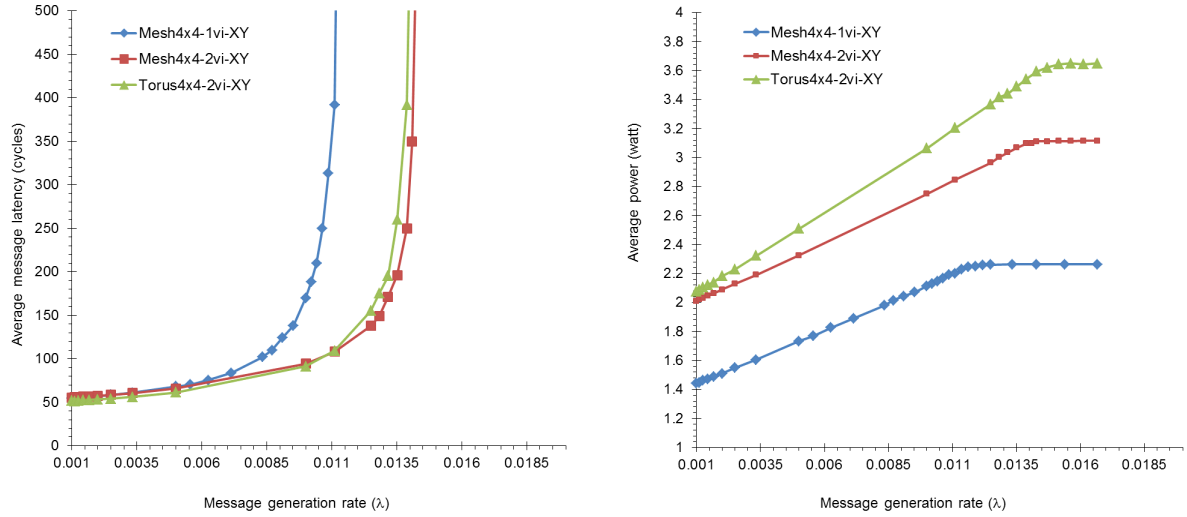


Figure 3. Performance and power consumption of XY routing in a 4x4 mesh with 1 and 2 virtual channels and a 4x4 torus with 2 virtual channels.

2.3. Performance and power consumption results

Fig. 3 shows the average message latency (as the main performance measure) and power consumption of XY routing in a 4x4 mesh NoC with one and two virtual channels, and in a 4x4 torus NoC with two virtual channels. In this figure, horizontal axis shows the message generation rate at each node and the vertical axis shows either the average message latency or the power consumption. The message length is of 32 flits length (4 flits for the header and 28 data flits), and buffer size of 4 flits for each virtual channel. As physical implementation suggests [34-36], the wire length for the torus is considered twice the length of the link in its mesh counterpart. Also a 2 GHz clock frequency with NVT type transistor and 65 nm VLSI technology is considered to extract power consumption values. The figure shows that the torus exhibits a better performance compared to the mesh topology with one and 2 virtual channels per physical channel for low traffic regions and better performance than the mesh with one virtual channel and marginally worse than the mesh with 2 virtual channels in high traffic region. This is because of the lower diameter and average inter-node distance in the torus NoC for low traffic loads and the dominant factor of inefficient bandwidth allocation compared to its equivalent mesh network in high traffics. The figure also shows that the number of virtual channels is the main factor to determine power consumption of the NoC; a mesh with two virtual channels, in average, has a dissipated power of near twice the power consumption of the mesh with 1 virtual channel. This is because of the complexity of switches and higher buffering requirements in the network with 2 virtual channels. A torus NoC has even more power consumption (compared to the mesh with 2 virtual channels) due to its extra wrap-around links.

3. Interconnection Routing Notation

We propose a new notation, called Interconnection Routing Notation (IRN), to extract new routing algorithms for torus and mesh NoCs. This notation can also be extended to other interconnection topologies. By using this notation, it is possible to have better understanding and formulation of routing algorithms for NoCs. Consider the XY routing algorithm for the 4x4 mesh network with one virtual channel per physical channel (as illustrated in Fig. 1). The *IRN Map* and *IRN Graph* for this algorithm are shown in Fig. 4. In XY routing algorithm, a packet first traverses the X axis and then continues its journey towards its destination along Y axis. The

IRN notation only explores the rule of movement through one axis (current axis or dimension). First row of the IRN Graph shows that if the source and destination nodes for a packet are adjacent across a dimension, then the packet moves towards the destination node directly with one step. The other rows show the direction of movements for distances more than one hop, individually for all node locations at a dimension. Corresponding to the IRN Graph, the IRN Map in each row shows the direction that the packet should traverse when it is in a specified location to cross the network to get closer to the destination. As shown in the figure, all the movements over the diameter of the map (or matrix), which means the packet should go from a smaller index to a larger index, have a '+' sign. This sign indicates movement in the positive direction of that dimension; similarly the '-' sign is used in the lower part of the map.

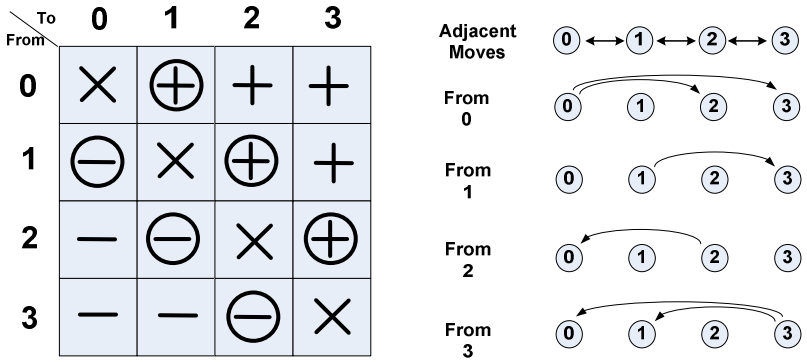


Figure 4. The IRN Map and Graph for a 4x4 mesh using XY Routing.

Fig. 5 shows the notation for the proposed routing algorithm in the torus network with only one virtual channel. At the first row of the IRN Graph the wrap-around link is shown. Because of using wrap-around links, a packet may reach its destination using positive or negative directions, but for a routing algorithm to be deadlock-free only one of the directions should be selected. The '+' and '-' symbols in circles refer to movements on the first row of IRN map where it is not reasonable to select the opposite direction to reach the destination. Therefore, we suppose these moves are always unchangeable, and only the signs without a circle are selectable. It should be noted that there are 4 selectable moves that can make up 16 different routing algorithms some of which deadlock-free and some with deadlock. The goal is to find the best selection (being deadlock-free and as minimal and optimal as possible, which will be explored using minimality and optimality factors discussed in the next section). Here, the only change to the mesh XY routing is that the first and last nodes in a dimension can use the wrap-around link for a one-step movement. For example, in case of 4x4 torus, nodes 0 and 3 can communicate with each other using the wrap-around links.

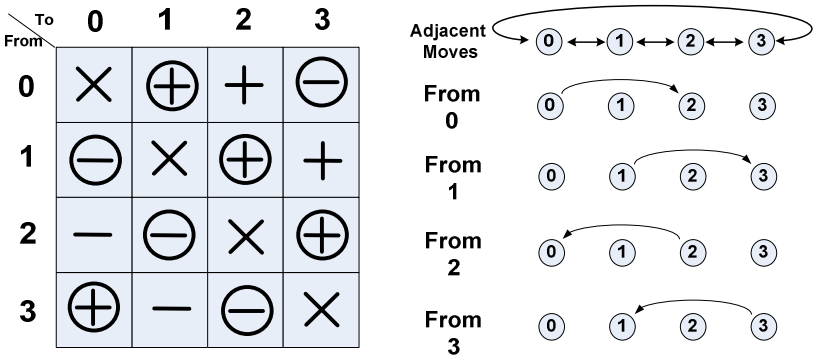


Figure 5. IRN Map and Graph for routing in a 4x4 torus NoC.

3.1. The Rules

In order to extract the IRN Map, the following rules should be applied: in each column, there should exist more than one sign change. Otherwise, it may cause livelock. For the sake of minimality and optimality it is much better to have equal number of '+'s and '-'s for the selectable area. The same case applies to the number of '+' and '-' symbols in a row. Also, there should not be more than one sign change in a row. Fig. 6 shows a case for the 4x4 torus in which deadlock may occur. Deadlock is caused because of a loop between movements in positive or negative directions. There should be one row with all selectable '-' movements and also one row with all selectable '+' movements for an algorithm to be deadlock-free. After filling these two rows, the other rows should be filled using the previous rules. Examples of applying these rules are shown in Fig. 7 for a 5x5 torus and a 6x6 torus.

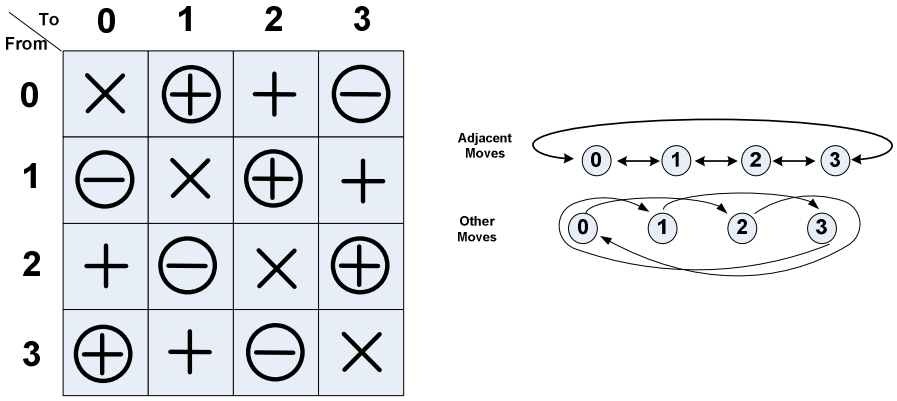


Figure 6. A routing case in a 4x4 torus causing deadlock.

3.2. Optimality and Minimality

3.2.1. Minimality Factor (*M*)

For a packet which traverses the network from the source node to destination node, there is always a minimum number that determines the shortest path taken by the packet. Because of the limitations that the routing algorithm makes for the packet, the routing algorithm might not be always minimal. As can be seen in Fig. 8, the selectable area of the IRN Map determines whether the proposed routing algorithm for a dimension is minimal or not. In fact for the dimensions of radix n>4 in the torus, there is no a minimal algorithm where all the paths are the shortest possible ones. The shaded boxes in the figure show a subset of the selectable areas for odd and even values of n in which the minimality parameter is applicable. A number is displayed in the top corner of the shaded boxes: it is 0 if the packet takes a direction with the shortest distance and other numbers show the extra steps that should be taken. When n is odd, all movements of the selectable area are shaded and when n is even this parameter is not applicable to the movements with distance $n/2$ from the source nodes, since both directions result in equal distance. At last, for a specific dimension, if we calculate the sum of all minimal path lengths when comparing different routing algorithms, the algorithm with the minimum total sum will be the best one (here called the minimal one).

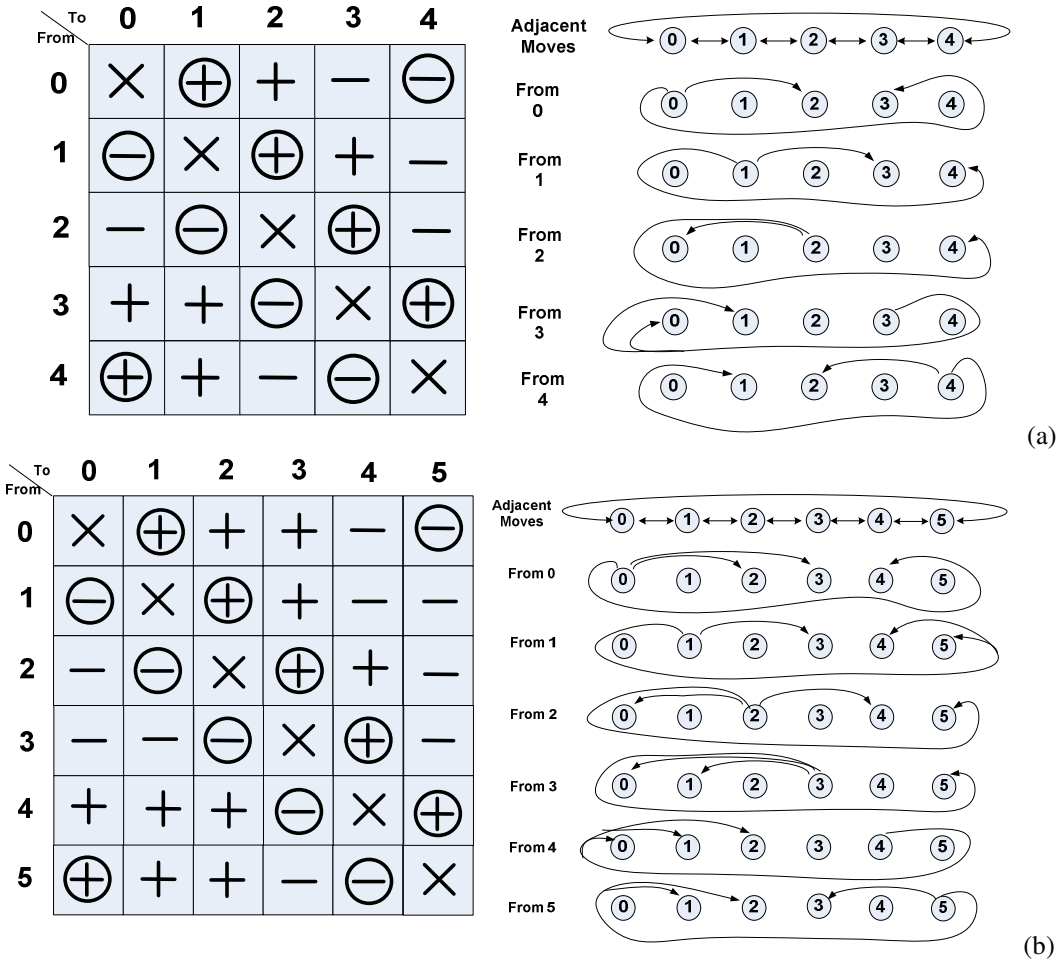


Figure 7. IRN Map and IRN Graph for routing in (a) a 5x5 torus and (b) a 6x6 torus.

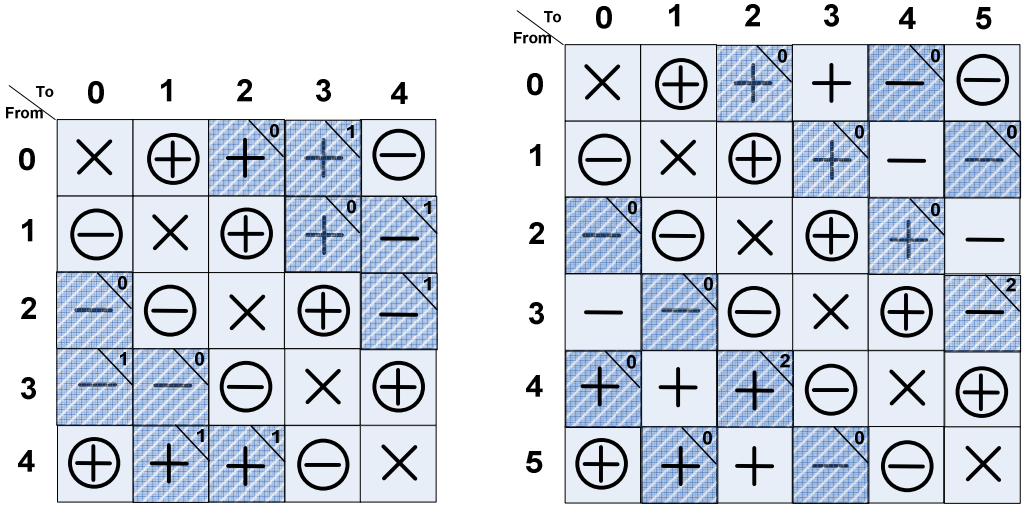


Figure 8. Minimality factor for some routing methods in a 5x5 torus (left) and a 6x6 torus (right).

3.2.2. Optimality Factor (*Opt*)

Although in some references in the area of interconnection networks, an optimal algorithm is known to be a *balanced* algorithm, here we propose a quantitative approach as a parameter based on IRN notation to measure the quality of traffic balance or *optimality factor*. This parameter describes how good an algorithm can balance the network traffic. In fact for a routing algorithm, we need a measure that indicates if all the links are utilized properly with respect to the packet destination address distribution. For the case of uniform traffic, the links should be utilized equally. With uniform traffic pattern, it is supposed that all nodes send a packet to all other network nodes with equal chance, and therefore all the links should be utilized evenly. As shown in Fig. 9, some numbers have been presented on adjacent movements with shaded boxes. Note that these movements are the representatives of their corresponding links and if we consider the number of '+' ('-') signs for positive (negative) movements in their corresponding rows and columns (considering wrap-arounds), the result shows the number of times this link is used. The numbers are shown in lower corner of the shaded boxes and as discussed they should have the same value in order to have optimal routing; therefore, the variance (*Opt*) of all numbers is a good candidate to represent the optimality of routing algorithms. That is, the smaller the *Opt*, the more optimal the algorithm is. As shown in the figure, two different algorithms are proposed for a 4x4 torus. One of the algorithms is optimal since all optimality numbers are equal to 2; therefore, *Opt* is 0. The other algorithm is not optimal, although both algorithms are minimal.

4. Deterministic TRANC Routing Algorithm

In this section, we introduce a new deterministic routing algorithm for the torus NoC (*TRANC*) that is deadlock-free and requires only one virtual channel per physical channel. Also, in the following section, we describe how to extend the algorithm to support adaptive routing algorithms based on TRANC. The algorithm uses an incremental approach based on the IRN notation in an $n \times n$ torus NoC.

4.1. The proposed algorithm for an arbitrary radix *n*

As shown in Fig. 10, the IRN map for a radix n ring is generated by adding a row and a column to the IRN map of radix n-1 ring, starting from n=4. The algorithm is straight forward for the 4x4 torus; for higher radices it is enough to add a row and a column as shown in Fig. 10.

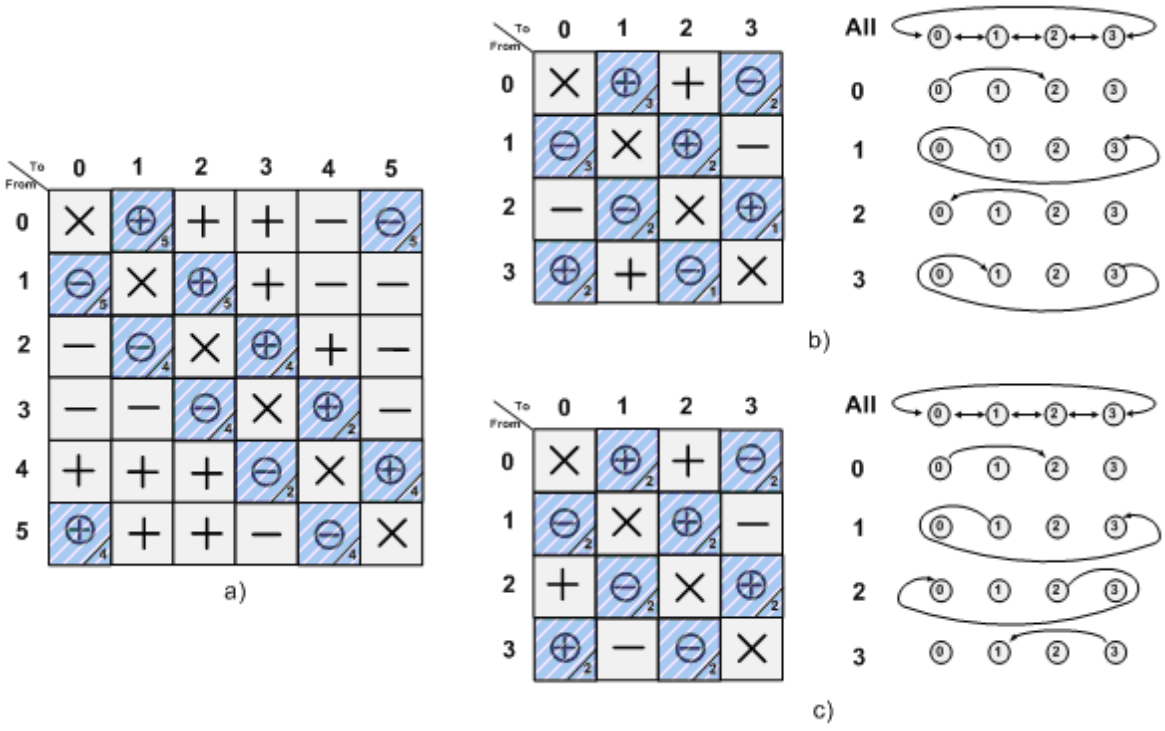


Figure 9. IRN with optimality factors for different routing methods; (a) Non-optimal in 6x6 torus, Opt=12, (b) Non-optimal in a 4x4 torus, Opt=4, and c) Optimal in a 4x4 torus, Opt=0.

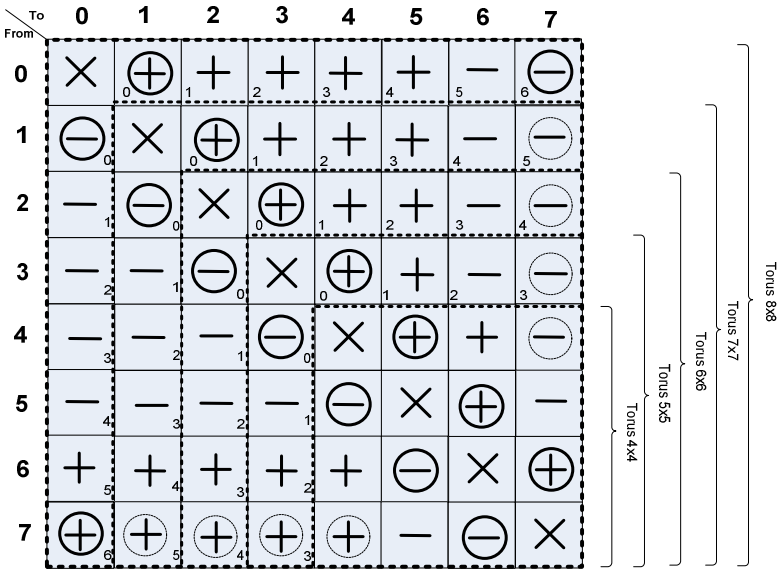


Figure 10. The proposed IRN Map representing the TRANC routing algorithm in an *n*x*n* torus.

In order to complete the new row, it is enough to fill the right most two boxes with '-' signs and others with '+' signs. Again for completing the new column, it is enough to fill the two lower boxes with '+' signs and others with '-' signs. When two or more of the moves described in the IRN graph happen simultaneously, they may form a situation where some of the packets are

waiting for other ones to free the path. In this situation, there is a packet contention. When the contention starts from a packet and lasts with the same packet, such that no activity is possible for the packets, it is said that a deadlock situation has occurred. A routing algorithm that never causes a deadlock situation is called a deadlock-free routing algorithm.

The TRANC routing algorithm is a deadlock-free. The intuitive justification that the algorithm is deadlock-free is extracted from the IRN map and graph in Fig. 7.b. As discussed before, there is not a positive move of more than one step from node 3, and therefore positive cyclic dependencies are broken in the network. The same is correct for negative movements, as there is not any negative movement of more than one step from node 4 to other nodes, and therefore negative cyclic dependencies are also broken.

For a more analytical justification let us propose the following rules:

Rule 1: A one-step movement in the IRN in which the source and destination nodes are adjacent never causes a deadlock situation.

Rule 2: Suppose there are two movements m_1 and m_2, where movement m_1 is a subset of movement m_2. If movement m_1 causes a deadlock then movement m_2 may cause the same deadlock.

Rule 3: For the case of the IRN, the deadlocks from positive movements are not corresponded to the deadlocks from the negative movements. Thus, they can be processed separately.

As discussed earlier the algorithm for higher radices can be extracted from smaller radices. For the case of a 4x4 torus, it can be easily seen that the algorithm is deadlock-free. Now, we show that if algorithm is deadlock-free for radix n then it is also deadlock-free for radix n+1. To do so, we suppose the claim is not true and thus at least there is a case where the algorithm is deadlock-free for radix n but is not deadlock-free for radix n+1. We show that it is impossible.

Fig. 11 shows the different cases of movements for radix n with node index from 0 to n-1 and also the added node 0' for radix n+1. Considering the routing algorithm in Fig. 10, if we suppose there is a deadlock situation for radix n+1, it should be one of these 3 different following cases (here, without loss of generality, we only consider the positive movements; negative movements can be processed in the same manner using rule 3):

Case 1: Node 0' is not the start or end point of none of the movements that have caused deadlock (Fig. 11.a). If we eliminate this node, we should find the routing algorithm for radix n. In this case, it can be seen that the algorithm for radix n has deadlock.

Case 2: We suppose 0' is the end of some movements that may cause a deadlock in radix n+1(Fig. 11.b). From rule 1, rule 2 and also Fig. 10, it can be seen that all the '+' signs in the newly added row for radix n+1 either represent a one-step movement or has a superset movement in radix n; therefore, by eliminating these positive movements, we will have the case of radix n, but this is a deadlock situation for radix n.

Case 3: We suppose 0' is the start of some movements that have caused deadlock in radix n+1 (Fig. 11.c). From rule 2 and also Fig. 10, it can be seen that the two '+' signs in the newly added column for radix n+1 have a superset '+' sign in radix n; therefore, by eliminating this positive movements, we will end up with the case of radix n in a deadlock situation.

Considering all above cases, we can conclude that TRANC is deadlock-free. This is also approved through the extensive simulation experiments we have realized for different scenarios. As discussed before for the dimensions of radix n>4, TRANC is not fully optimal and fully minimal but has good optimality and minimality factors for different radices. The reason is that each packet traverses the shortest possible path to reach the destination that prevent deadlocks, not the physically shortest paths that may cause deadlocks. Also, the use of wrap-around links is not balanced compared to other links because of the rules that have been applied to the algorithm. It is possible to use a different approach for different radices based on the IRN that can result in better optimality and minimality factors, but justification of deadlock freedom for each radix should be done separately.

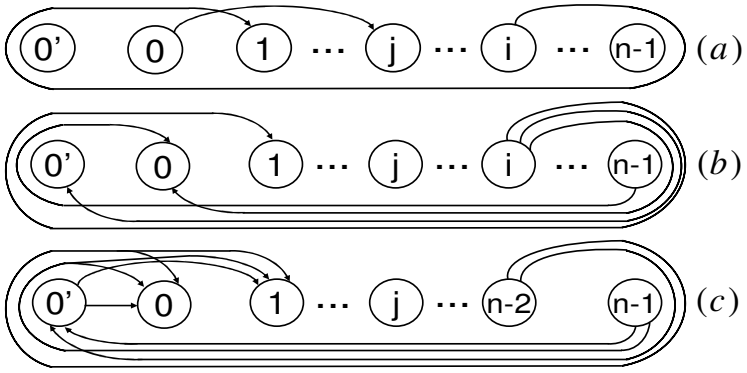


Figure 11. Different cases of movements when upgrading from radix n to n+1, a) 0' is not the start or end of any movement that causes deadlock, b) 0' is the end for some of the movements that causes deadlock c) 0' is the start of some of the movements that cause deadlock.

In Fig. 12, a pseudo code for the TRANC routing algorithm is shown. As can be seen in the figure, the complexity of the hardware that runs this algorithm compared to the classic XY routing algorithm includes some extra comparisons that should be done with n-1, n-2 and n-3 and since n is a constant number (when a fixed radix is implemented in hardware), only some comparison operations with some constant numbers are added to the code. Such simple comparisons do not require considerable power and do not impose noticeable delay in routing as will be shown later in simulation results.

5. Adaptive-TRANC Routing Algorithm

As described in previous sections, (deterministic) TRANC routing algorithm utilizes the extra wrap-around links independently in each dimension of a torus network, compared to its counterpart mesh network with XY-routing and outperforms the performance without the need of adding extra virtual channels. In order to extend TRANC to support adaptive routing, we consider the basic XY routing algorithm for the mesh network in Fig. 13 and compare it to (deterministic) TRANC routing algorithm in Fig. 12. It is obvious that we can consider a mapping for dimension k (x, y or others in multi-dimensional networks) to extract TRANC from XY routing. That is:

($K\_offset > 0$) ↔ *TRANC_K_Plus*

where

TRANC_K_Plus := (*K_offset* =1) or (*K_offset* = -*n*+1) or (((*K_dest* = *n*-2) and (*K_current*=*n*-4)) or (*K_current*=*n*-2)) or (*K_dest* -2>= *K_current*)

and

($K\_offset < 0$) ↔ *TRANC_K_Minus*

where

TRANC_K_Minus := (*K_offset* = -1) or (*K_offset* = *n*-1) or (((*K_dest* = *n*-3) and (*K_current*=*n*-1)) or (*K_current*=*n*-3)) or ((*K_current*-2>= *K_dest*) or (*K_dest* =*n*-2)).

```
Algorithm TRANC for 2-Dimensional Torus NoCs.
    Inputs: Coordinates of current node (Xcurrent, Ycurrent),
            destination node (Xdest, Ydest), and radix n;
    Output: Selected output Channel
Begin
 Xoffset := Xdest – Xcurrent;   Yoffset :=Ydest –Ycurrent;

 if (Xoffset=0) and (Yoffset =0) then  return Ejection Channel;
 else
   {
     if (Xoffset =1) or (Xoffset = -n+1) or
        (((Xdest = n-2) and (Xcurrent=n-4)) or (Xcurrent=n-2)) or
        (Xdest -2>= Xcurrent)
      then  return X+;

     if (Xoffset = -1) or (Xoffset = n-1) or
        (((Xdest = n-3) and (Xcurrent=n-1)) or (Xcurrent=n-3)) or
        ((Xcurrent-2>= Xdest) or (Xdest =n-2))
      then  return X-;

     if (Yoffset =1) or (Yoffset = -n+1) or
        (((Ydest = n-2) and (Ycurrent=n-4)) or (Ycurrent=n-2)) or
        (Ydest -2>= Ycurrent)
      then  return Y+;

     if (Yoffset = -1) or (Yoffset = n-1) or
        (((Ydest = n-3) and (Ycurrent=n-1)) or (Ycurrent=n-3)) or
        ((Ycurrent-2>= Ydest) or (Ydest =n-2))
      then  return Y-;
   }
End
```

Figure 12. Pseudo code for TRANC routing algorithm

Fig. 14 shows the WEST-FIRST adaptive routing algorithm for a mesh network [32]. Using the condition mapping approach described above, it is possible to modify the algorithm to be used for the torus. Fig. 15 shows the resulted algorithm. In the new algorithm, "*X*-" movement correctly describes west direction packet forwarding in a 2-D torus except for the wrap-around link at the left most nodes of the network. In these nodes, "*X*-" translates to go to the right most node in the current Y dimension. The same approach is applied for "*X*+","*Y*-" and "*Y*+" packet forwarding. This is also correct for fully adaptive routing algorithms. As an example, Fig.16 shows Duato's fully adaptive routing for the 2-D mesh network and Fig. 17 shows its corresponding algorithm for the 2D torus, based on TRANC algorithm. In these algorithms X_a and Y_a are the set of adaptive virtual channels and X_b and Y_b are the set of deterministic virtual channels. *Select*() function returns the first input argument in order of appearance, which is the index of a virtual channel and is not currently occupied [32].

```
Algorithm XY for 2-Dimensional Mesh NoCs.
    Inputs: Coordinates of current node (Xcurrent, Ycurrent),
            destination node (Xdest, Ydest), and radix n;
    Output: Selected output Channel
Begin
 Xoffset := Xdest − Xcurrent;   Yoffset :=Ydest −Ycurrent;

 if (Xoffset=0) and (Yoffset =0) then  return Ejection Channel;
 else
   {
    if (Xoffset > 0)   then  return X+;
    if (Xoffset < 0)   then  return X-;
    if (Yoffset > 0)   then  return Y+;
    if (Yoffset < 0)   then  return Y-;
   }
End.
```

Figure 13. Pseudo code for XY routing algorithm in Mesh

```
Algorithm West-First for 2-Dimensional Mesh NoCs.
    Inputs: Coordinates of current node (Xcurrent, Ycurrent),
            destination node (Xdest, Ydest), and radix n;
    Output: Selected output Channel
Begin
 Xoffset := Xdest − Xcurrent;   Yoffset :=Ydest −Ycurrent;

 if (Xoffset=0) and (Yoffset =0) then  return Ejection Channel;
 else
   {
    if (Xoffset < 0)   then
        return X-;
    if (Xoffset > 0 and Yoffset < 0)   then
        return (Select (X+,Y-));
    if (Xoffset > 0 and Yoffset > 0)   then
        return (Select (X+,Y+));
    if (Xoffset > 0 and Yoffset = 0)   then
        return X+;
    if (Xoffset = 0 and Yoffset < 0) then
        return Y-;
    if (Xoffset = 0 and Yoffset > 0) then
        return Y+;
   }
End.
```

Figure 14. Pseudo code for West First routing algorithm in Mesh

```
Algorithm TRANC West-First for 2-Dimensional Torus NoCs.
     Inputs: Coordinates of current node (Xcurrent, Ycurrent),
             destination node (Xdest, Ydest), and radix n;
     Output: Selected output Channel
Begin
 Xoffset := Xdest − Xcurrent;   Yoffset :=Ydest −Ycurrent;

 if (Xoffset=0) and (Yoffset =0) then  return Ejection Channel;
 else
   {
     if (TRANC_X_Minus)   then
         return X-;
     if (TRANC_X_Plus and TRANC_Y_Minus)   then
         return (Select (X+,Y-));
     if (TRANC_X_Plus and TRANC_Y_Plus)   then
        return (Select (X+,Y+));
     if (TRANC_X_Plus and Yoffset = 0)   then
         return X+;
     if (Xoffset = 0 and TRANC_Y_Minus) then
        return Y-;
     if (Xoffset = 0 and TRANC_Y_Plus) then
        return Y+;
   }
End.
```

Figure 15. Pseudo code for TRANC West First Routing algorithm in Torus

Algorithm Duato's Fully Adaptive Algorithm for 2-Dimensional Mesh NoCs.

Inputs: Coordinates of current node (*Xcurrent*, *Ycurrent*),
destination node (*Xdest*, *Ydest*), and radix *n*;
Output: Selected output Channel

```
Begin
 Xoffset := Xdest − Xcurrent;   Yoffset :=Ydest −Ycurrent;

 if (Xoffset=0) and (Yoffset =0) then  return Ejection Channel;
 else
   {
        if (Xoffset < 0 and Yoffset< 0) then
            return (Select(Xa−, Ya−, Xb−));
        endif
        if (Xoffset < 0 and Yoffset > 0) then
            return ( Select(Xa−, Ya+, Xb−));
        endif
        if  (Xoffset < 0 and Yoffset = 0) then
             return ( Select(Xa−, Xb−));
        endif
        if  (Xoffset > 0 and Yoffset < 0) then
             return ( Select(Xa+, Ya−, Xb+));
        endif
        if  (Xoffset > 0 and Yoffset > 0) then
             return ( Select(Xa+, Ya+, Xb+));
        endif
        if  (Xoffset > 0 and Yoffset = 0) then
             return ( Select(Xa+, Xb+));
        endif
        if  (Xoffset = 0 and Yoffset < 0) then
             return ( Select(Y a−, Yb−));
        endif
        if  (Xoffset = 0 and Yoffset > 0) then
              return ( Select(Y a+, Yb+));
        endif
    }
End.
```

Figure 16. Pseudo code for Duato's fully adaptive routing algorithm in Mesh

```
Algorithm TRANC  Fully Adaptive Algorithm for 2-Dimensional Torus NoCs.
     Inputs: Coordinates of current node (Xcurrent, Ycurrent),
             destination node (Xdest, Ydest), and radix n;
     Output: Selected output Channel
Begin
 Xoffset := Xdest − Xcurrent;   Yoffset :=Ydest −Ycurrent;

 if (Xoffset=0) and (Yoffset =0) then  return Ejection Channel;
 else
   {
        if (TRANC_X_Minus and TRANC_Y_Minus) then
            return (Select(Xa−, Ya−, Xb−));
        endif
        if (TRANC_X_Minus and TRANC_Y_Plus) then
            return ( Select(Xa−, Ya+, Xb−));
        endif
        if  (TRANC_X_Minus and Yoffset = 0) then
            return ( Select(Xa−, Xb−));
        endif
        if  (TRANC_X_Plus 0 and TRANC_Y_Minus) then
            return ( Select(Xa+, Ya−, Xb+));
        endif
        if  (TRANC_X_Plus and TRANC_Y_Plus) then
            return ( Select(Xa+, Ya+, Xb+));
        endif
        if  (TRANC_X_Plus and Yoffset = 0) then
            return ( Select(Xa+, Xb+));
        endif
        if  (Xoffset = 0 and TRANC_Y_Minus) then
            return ( Select(Y a−, Yb−));
        endif
        if  (Xoffset = 0 and TRANC_Y_Plus) then
             return ( Select(Y a+, Yb+));
        endif
    }
End.
```

Figure 17. Pseudo code for TRANC Fully Adaptive Routing algorithm for 2-D Torus based on Duato's fully adaptive algorithm.

The same mapping approach can be used also for other adaptive routing algorithms like Linder-Harden partially adaptive, *opt-y* fully adaptive, Disha [32].

6. Experimental Results

A primarily evaluation of the TRANC using a simple C++ program shows that TRANC slightly increases the maximum and the average inter-node distance in the network when compared to XY routing in the torus. This is shown in Fig. 18 for different network radices. Note that for popular and current network sizes used in practice (i.e. NoC with up to 6x6 nodes) the difference between the average and maximum inter-node distances for the two routing algorithms in torus NoCs is small. Therefore, the lower complexity of the router in TRANC (due to the less virtual channels used) can improve the performance and reduce the power dissipation in the network.

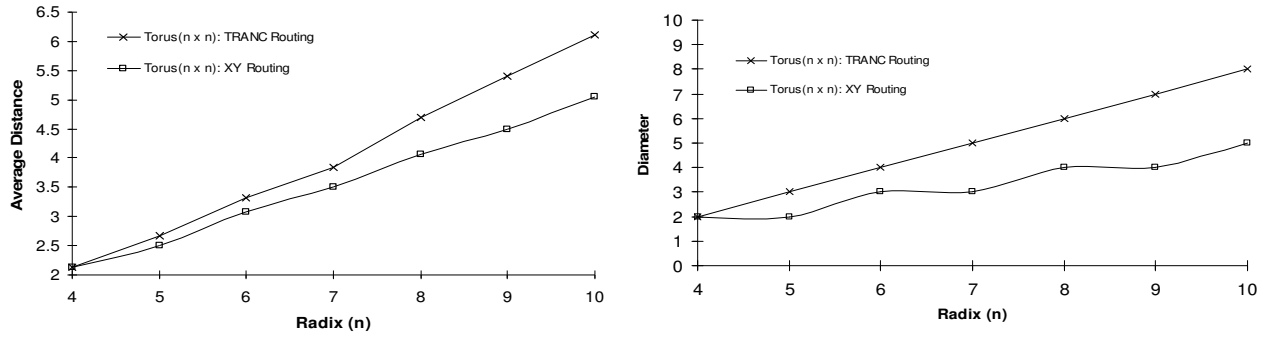


Figure 18. The average inter-node distance and diameter using TRANC and XY routing algorithms.

A cycle accurate and synthesizable VHDL model for the mesh and torus NoCs has been implemented. XY, TRANC and Adaptive-TRANC routing algorithms with the possibility of using different number of virtual channels per physical channel are implemented. The VHDL implementation is used for both performance evaluation and power estimation using *Power Compiler* CAD tool [28, 30, 31].

To evaluate the performance and power dissipation for the proposed routing algorithm compared to XY routing in the first scenario, two different network sizes (4x4 and 6x6 NoCs) are considered. The destination of the messages is chosen uniformly over the network nodes. Messages are generated and entered into the network following a Poisson distribution. Fig. 19 shows the simulation results for XY routing in mesh and torus NoCs and for (deterministic) TRANC routing algorithm in torus NoCs (for 4x4 and 6x6 wormhole-switched networks). The horizontal axis shows the traffic generation rate at each node while the vertical axis shows the average message latency (or dissipated power) in the network. As can be seen in the figure, the performance of TRANC routing (with one virtual channel) is better than XY routing in the mesh (using one virtual channel) and almost equal to the torus XY routing (using 2 virtual channels) for radix 4 and slightly worse for radix 6, while the power consumption is near that of a mesh NoC and much less than that of the XY-routed torus (using 2 virtual channels). To have a unique measure to assess the suitability of the proposed algorithm for torus NoCs, we have also used the product of average message latency and power consumption. Simulation results show that the proposed routing algorithm for the torus NoC using one virtual channel is superior to its equivalent mesh using XY routing (with one virtual channel) and equivalent torus NoC using XY routing with 2 virtual channels for low and medium traffic loads (as can be seen in Fig. 19).

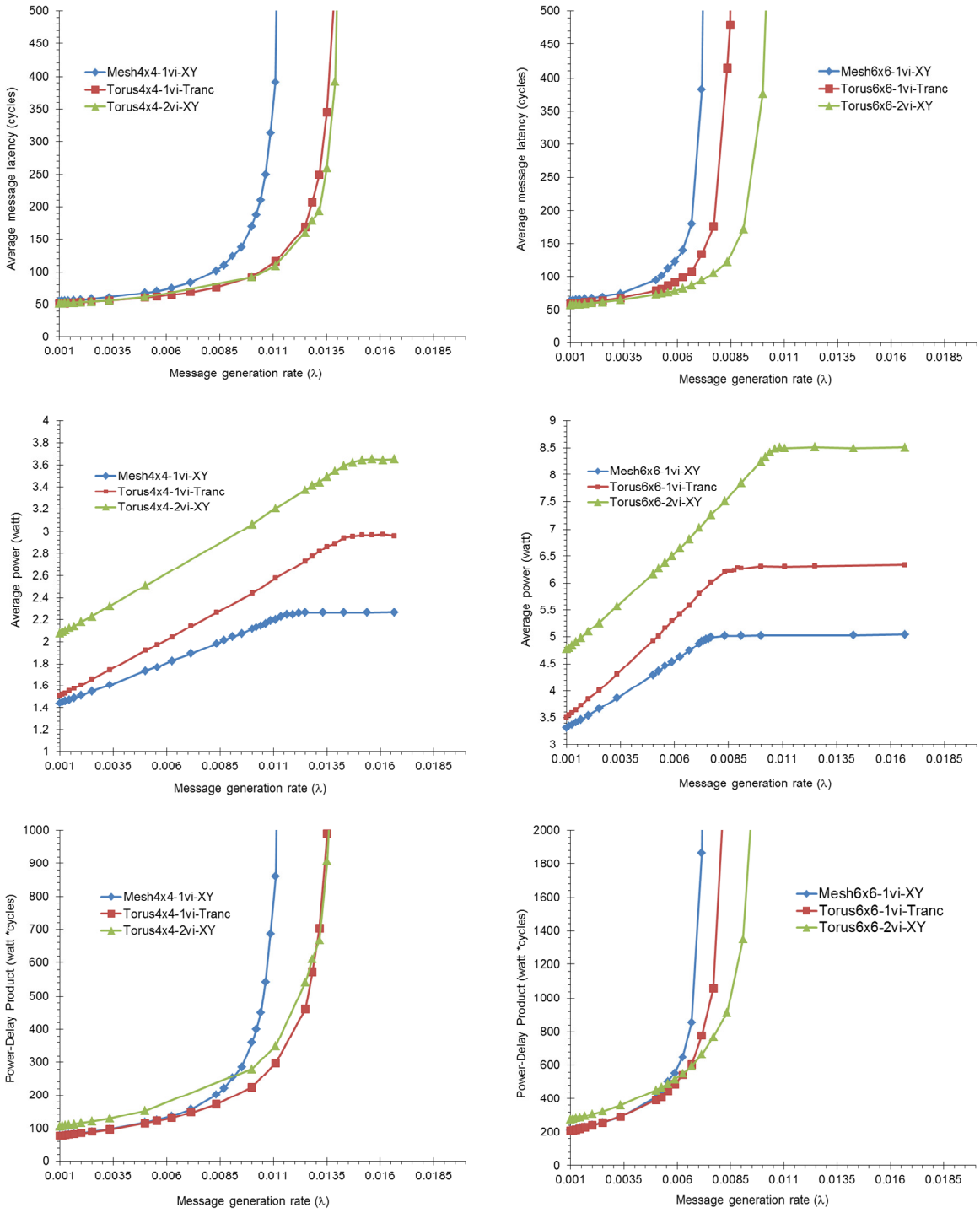


Figure 19. Performance, power consumption, and power-delay product of XY routing in the mesh and torus NoCs and TRANC routing algorithm in the torus NoC with radices 4 and 6.

As the second simulation scenario, to verify the efficiency of the algorithm on a real world application, streams of packets from some real applications *fft calculation* and *ocean simulation* from *splash benchmark* [37], have been generated and Fig. 20 shows the average packet latency for the implementation of the algorithms in the mesh network with XY-routing using one virtual channel, the torus network with (deterministic) TRANC routing algorithm using one virtual channel and the torus NoC with classic deterministic routing using two virtual channels, for 4x4 and 6x6 network configurations. As the figure shows, the TRANC in the torus outperforms XY

routing in the mesh and although TRANC has only one virtual channel compared to the torus with two virtual channels, its latency is close to the latter. The differences are more evident in the case of 4x4 network configuration, due to the smaller network size, the operating region of the network is closer to the saturation capacity of the network. Thus, in the 6x6 network, as the same amount of traffic is generated and the network is larger, the operation bias point is not close to the saturation region, and thus differences are less. This again describes that TRANC is more efficient when the network operates near the saturation region.

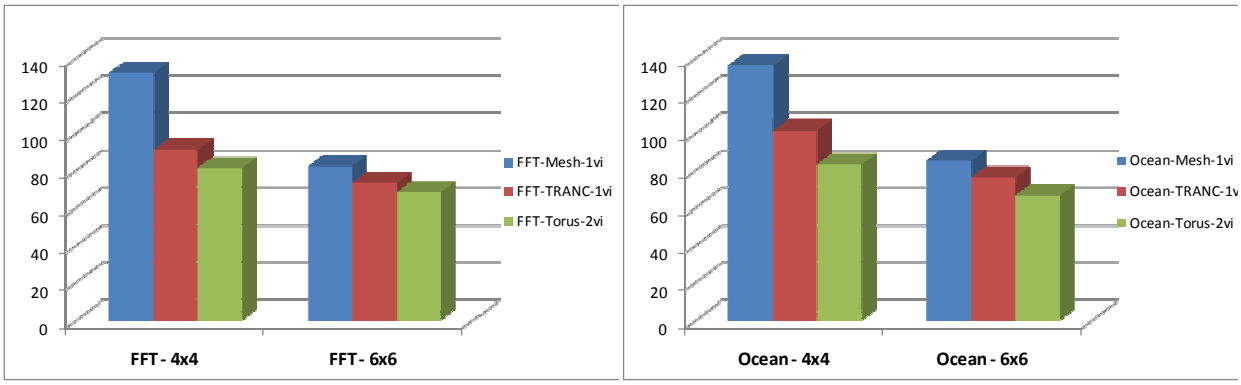


Figure 20: Average latency for Real Applications (FFT on left and Ocean simulator on right) implemented on different topologies and routings

The third simulation scenario, shown in Fig. 21, depicts the average latency, power consumption and power-delay product for partially adaptive West-First routing algorithm (Fig. 14) for the mesh, adaptive TRANC West-First for torus (Fig. 15), Duato's fully adaptive routing algorithm for the mesh with 2 virtual channels (Fig. 16) and TRANC fully adaptive routing algorithm with two virtual channels (Fig. 17) (extracted directly from Duato's algorithm). As the figure shows, power-delay product of the TRANC-based algorithms outperforms their counterparts for both partially and fully adaptive scenarios, which is a direct consequence of using wrap-around links and the features of the TRANC algorithm.

7. Conclusions

Current SoC designs have popularly employed point-to-point NoCs for inter-IP communication. The most popular NoCs employ mesh and torus topologies. The mesh topology enjoys its simple structure and the possibility of using XY routing algorithm with only one virtual channel per physical channel. However, when wrap-around links are used to form a torus NoC, two virtual channels should be used to ensure deadlock freedom to implement XY routing. On the other hand, adding virtual channels increase power dissipation, although performance is increased (compared to the mesh NoC) as a result of lower inter-IP distance due to wrap-around links in the torus.

In this paper, a new network routing notation and, based on it, a new deterministic and adaptive routing algorithms for torus NoCs were introduced. The TRANC routing algorithm can be easily implemented on a torus NoC for deterministic, partially and fully adaptive routing algorithms that traditionally were implemented on meshes, to utilize wrap-around links. The resulted algorithms consume lower energy compared to their counterpart algorithms by reducing the required number of virtual channels and better utilization of network resources. Also, the simplicity of the router (less buffering and switching hardware complexity) could well compensate for the slightly increased inter-IP distance and result in a slightly lower performance when compared to the torus counterparts. They also outperform their mesh-based counterparts in terms of performance while consuming almost the same power.

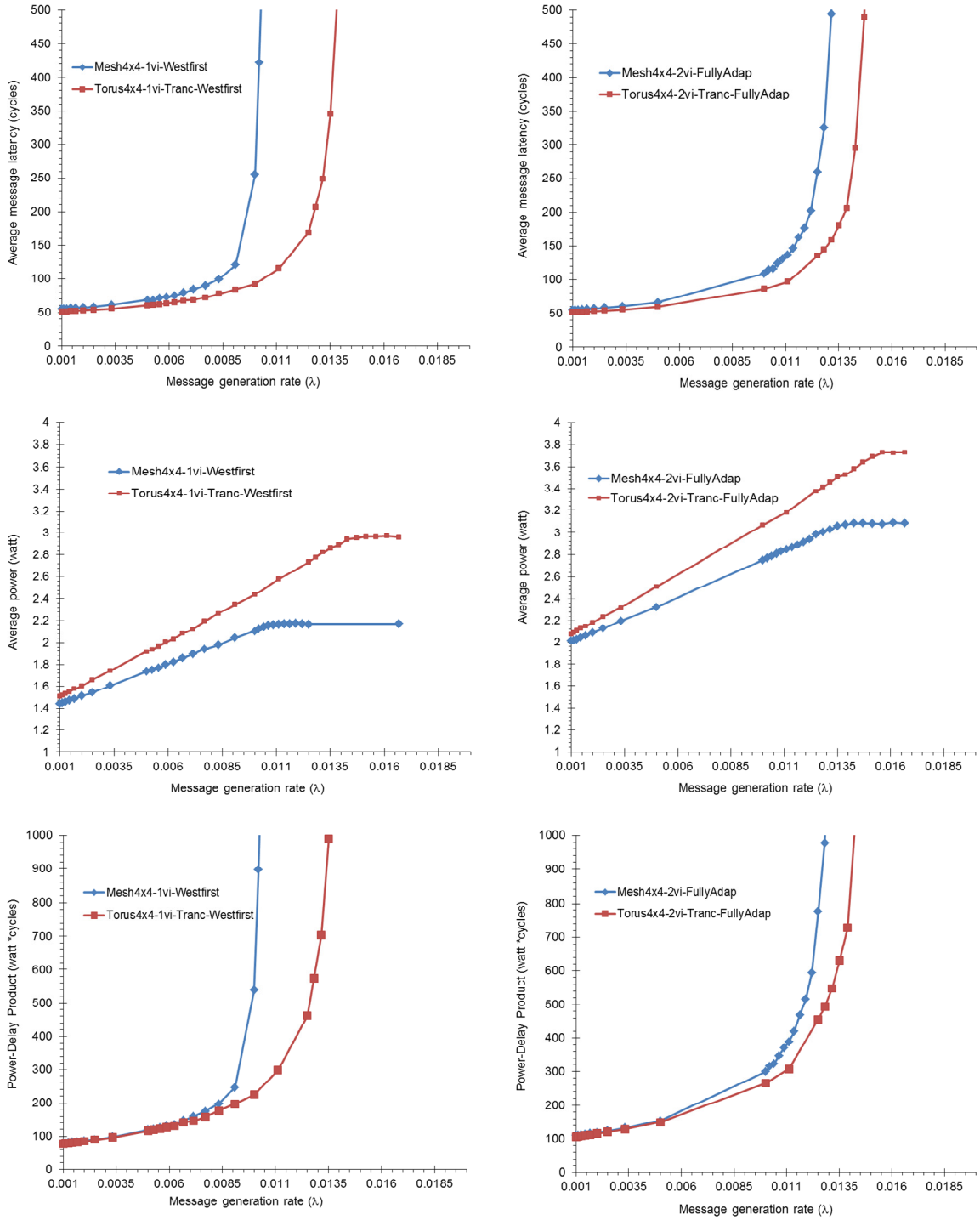


Figure 21: latency, power and power-delay product vs. message generation rate (left) for Mesh (4x4) with west-first and Torus (4x4) with TRANC west-first adaptive routings and (right) for Mesh with Duato's fully adaptive routing and TRANC fully adaptive routing for Torus

References

[9] S. Meraji, A. Nayebi and H. Sarbazi-Azad, "Simulation-Based Performance Evaluation of Deterministic Routing in Necklace Hypercubes," *IEEE*/ACS Int. Conference on

Computer Systems and Applications, AICCSA '07, pages 343-350, Amman, Jordan, May, 2007.

[10] H. Hashemi-Najafabadi, H. Sarbazi-Azad, and P. Rajabzadeh, "An accurate performance model of fully adaptive routing in wormhole-switched two-dimensional mesh multicomputers, "Journal of Microprocessors and Microsystems, vol. 31, no. 7, pages 445-455, 2007.

[11] P. Shareghi, H. Sarbazi-Azad, "The stretched network: properties, routing, and performance, " Journal of Information Science and Engineering 24, pages 361-378, 2008.

[12] P. Abad, V. Puente, and J. A. Gregorio, "Mrr: Enabling fully adaptive multicast routing for cmp interconnection networks, " In Int. Symp. On High Performance Computer Architecture, HPCA, pages 355-366, 2009.

[13] W.J. Dally and C.L. Seitz, "Deadlock-free message routing in multiprocessor interconnection networks," IEEE Trans. On Computers, vol. C-36, no. 5, pages 547-553, 1987.

[14] A. Khonsari, "Performance Modelling and Analysis of Deadlock Recovery Routing Algorithms in Multicomputer Interconnection Networks," PhD Thesis, Computing Science Department, Glasgow University, 2003.

[15] M. Ould-Khaoua, "A performance model for Duato's fully adaptive routing algorithm in k-ary ncubes, " IEEE Trans. on Computers, vol. 48, no. 12, pages 1297-1304, 1999.

[16] H. Sarbazi-Azad, "Performance Analysis of Wormhole Routing in Interconnection Networks, " PhD Thesis, Compting Science Department, Glasgow University, 2001.

[17] L. Schwiebert and D. N. Jayasimha, "Optimal Fully Adaptive Minimal Wormhole Routing for Meshes," Journal of Parallel and Distributed Computing, vol. 27, no. 1, pages 56-70, 1995.

[18] J. Upadhyay, V. Varavithya, and P. Mohapatra, "A Traffic-Balanced Adaptive Wormhole-Routing Scheme for Two-Dimensional Meshes," IEEE Transactions on Computers, vol. 46, no. 2, pages 190-197, 1997.

[19] W. J. Dally, H. Aoki, "Deadlock-Free Adaptive Routing in Multicomputer Networks Using Virtual Channels", IEEE Transactions on Parallel and Distributed Systems, vol. 4, no. 4, pages 466-475, 1993.

[20] J. H. Kim, Z. Liu, A. A. Chien, "Compressionless Routing: A Framework for Adaptive and Fault-tolerant Routing," IEEE Transactions on Parallel and Distributed Systems, vol. 8, no. 3, pages 229-244, 1996.

[21] W.J. Dally, C.L. Seitz, "Deadlock-free message routing in multiprocessor interconnection networks", IEEE Transactions on Computers, vol. C-36, no. 5, pages 547-553, 1987.

[22] J. Hu, R. Marculescu, "DyAD - Smart Routing for Networks on-Chip," Design Automation Conference, DAC, pages 260-263, 2004.

[23] G. M. Chiu, "The Odd-Even Turn Model for Adaptive Routing," IEEE Transactions on Parallel and Distributed Systems, vol. 11, no. 7, pages 729-38, 2000.

[24] R. V. Boppana, S. Chalasani, "A framework for designing deadlock-free wormhole routing algorithms," IEEE Transactions on Parallel and Distributed Systems, vol. 7, no. 2, pages 169-183, 1996.

[25] C. J. Glass, L. M. Ni, "The turn model for adaptive routing," Proceedings of the International Symposium on Computer Architecture (ISCA), pages 278-287, 1992.

[26] A. Singh, W. J. Dally, A. K. Gupta and B. Towles, "GOAL: A Load-balanced Adaptive Routing Algorithm for Torus Networks," in Proc. of the International Symp. on Comp. Arch., pages 194-205, 2003.

[27] A. M. Rahmani, I. Kamali, P. L. Kamran, A. Afzali-Kusha, and S. Safari, "Negative exponential distribution traffic pattern for power/performance analysis of network on chips, " In Proceedings of Int. Conference on VLSI Design, VLSID '09, pages 157-162, New Delhi, 2009.

[28] A. Patooghy, M. Fazeli, S.G. Miremadi, "Reducing Power Consumption in NoC Design with no Effect on Performance and Reliability," Int. Conf. on Electronics, Circuits and Systems, ICECS, Pages 886-889, 2007.
[29] T. Li, "Estimation of Power Consumption in Wormhole Routed Networks on Chip," Master Thesis IMIT/LECS, Sweden, 2005.
[30] P. P. Pande, C. Grecu, M. Jones, A. Ivanov, R. Saleh: "Effect of Traffic Localization on Energy Dissipation in NoC-based Interconnect," In Int. Sym. On Circuits and Systems, ISCAS (2), pages 1774-1777, 2005.
[31] S. Koohi, M. Mirza-Aghatabar, S. Hessabi "Evaluation of Traffic Pattern Effect on Power Consumption in Mesh and Torus Network-on-Chips," Integrated Circuits Symposium. ISIC '07, pages 512-515, 2007.
[32] M. Mirza-Aghatabar, S. Koohi, S. Hessabi, M. Pedram, "An Empirical Investigation of Mesh and Torus NoC Topologies under Different Routing Algorithms and Traffic Models," Int. Conference on Digital System Design Architectures, Methods and Tools, DSD, pages 19-26, Lubeck, Germany, 2007.
[33] W. J. Dally, B. Towles, "Route packets, not wires: on-chip interconnection networks," Proc. Of Design Automation Conference, DAC, pages 684–689, 2001.
[34] Terry T. Ye, Luca Benini, Giovanni De Micheli, "Analysis of Power Consumption on Switch Fabrics in Network Routers," In Proceedings of Design Automation Conference, DAC, pages 524-529, 2002.
[35] H. S. Wang, L. S. Peh, S. Malik, "Orion: A Power-Performance Simulator for Interconnection Network," In International Symposium on Microarchitecture, Micro 35, pages 294-305, Los Alamitos, CA, USA, 2002.
[36] D. L. Liu, C. Svensson, "Power consumption estimation in CMOS VLSI chips," IEEE Journal of Solid-State Circuits, vol. 29, no. 6, pages 663-670, 1994.
[37] W. J. Dally, C. Seitz, "The torus routing chip," In Journal of Distributed Computing, vol. 1, no. 4, pages 187-196, 1986.
[38] N. Banerjee, P. Vellanki, K. S. Chatha, "A Power and Performance Model for Network-on-Chip Architectures," In Proceedings of Design, Automation and Test in Europe, DATE, pages 1250-1255, France, 2004.
[39] K. Srinivasan, K. S. Chatha, "ISIS: A Genetic Algorithm based Technique for Custom On-Chip Interconnection Network Synthesis", In Int. Conference on VLSI Design, VLSID, pages 623-628, 2005.
[40] J. Duato et al., "Interconnection Networks," ISBN: 1-55860-852-4, 2003.
[41] D. Rahmati et al., "Power Efficient Routing Algorithm for Torus NoCs," Int. Conference on Contemporary Computing, IC3, pages 211-220, India, 2008.
[42] P. Pande, A. Ganguly, H. Zhu and C. Grecu, "Energy reduction through crosstalk avoidance coding in networks on chip, " Elsevier Journal of Systems Architecture, vol. 54, no. 3-4, pages 441-451, 2008.
[43] M. M. H. Rahman, Y. Inoguchi, F. Al Faisal and M. K. Kundu, "Symmetric and Folded Tori Connected Torus Network, " Journal of Networks, vol. 6, no. 1, pages 26-35, 2011.
[44] Srinivasan Murali, "Designing Reliable and Efficient Networks on Chips", Lecture Notes in Electrical Engineering, Issue No. 34, Springer, 2009.
[45] S. C. Woo, M. Ohara, E. Torrie, J. P. Singh, and A. Gupta, "The SPLASH-2 Programs: Characterization and Methodological Considerations," In Proceedings of Int. Symposium on Computer Architecture, ISCA, pages 24-36, 1995.

An Analytical Latency Model for Networks-on-Chip

Abbas Eslami Kiasari
Zhonghai Lu
Axel Jantsch

IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 21, no. 1, pp. 113-123, Jan. 2013.

Paper 9

An Analytical Latency Model for Networks-on-Chip

Abbas Eslami Kiasari, *Member, IEEE,* Zhonghai Lu, *Member, IEEE,*
and Axel Jantsch, *Member, IEEE*

***Abstract*—We propose an analytical model based on queueing theory for delay analysis in a wormhole-switched Network-on-Chip (NoC). The proposed model takes as input an application communication graph, a topology graph, a mapping vector, and a routing matrix, and estimates average packet latency and router blocking time. It works for arbitrary network topology with deterministic routing under arbitrary traffic patterns. This model can estimate per-flow average latency accurately and quickly, thus enabling fast design space exploration of various design parameters in NoC designs. Experimental results show that the proposed analytical model can predict the average packet latency more than four orders of magnitude faster than an accurate simulation, while the computation error is less than 10% in non-saturated networks for different system-on-chip platforms.**

***Index Terms*—Modelling and prediction, network-on-chip, performance analysis and design aids, queueing theory.**

1. INTRODUCTION

Latency is recognised as one of the most critical design characteristics for on-chip interconnection network architectures [17]. In this work, we propose a performance model which predicts the latency of flows in a Network-on-Chip (NoC) based system. Performance models are frequently employed by system designers for early architecture and design decisions. Typically, engineers construct a performance model, and then compare future technology options based on performance model projections. To this end, application and architecture models are first developed separately. Then, the application is mapped to the architecture and a performance model is used to evaluate the chosen application-architecture combination. Nowadays, most performance models of NoCs rely on simulations [2][19]. The use of simulation experiments makes the task of searching for efficient designs computationally intensive and does not scale well with the size of networks. Therefore, it is simply impossible to use the simulation in optimization loops.

An alternative approach is an analytical model which can estimate the desired performance metrics in a fraction of time. Analytical models can be used to prune the large design space in a very short time compared to simulation. Thus, it is justified to derive accurate analytical models for performance prediction of NoCs to eliminate the need for time consuming simulations. The information provided during the performance analysis step can be used in any optimization loop for NoCs such as topology selection, application mapping, and buffer allocation. Although the use of high-level models conceals a lot of complex technological aspects, it facilitates fast exploration of the NoC design space. Accurate simulations can be setup at later steps of design process when the design space is reduced to a few practical choices.

In this research a Performance Queueing (PQ) model, is proposed and evaluated for NoCs. The PQ model, which is based on a G/G/1 queueing model, has been developed for deterministic routing and wormhole switching. The proposed model is topology-independent and supports any kind of spatial and temporal traffic patterns. The estimated performance metrics such as average latency and router blocking time can be conveniently used for optimization purposes to find appropriate design parameters, as well as obtaining quick performance estimates. Our results show that the PQ model calculates quickly the latency of flows in the network with less than 10% error when compared to the simulation. This gives us confidence that we can utilize the model in the early design phase of high performance on-chip networks.

The rest of this paper is organized as follows. We start by reviewing previous studies and highlighting our contribution in Section II. Since our work is based on queueing theory, we give a very brief review of G/G/1 queues and priority queues in Section III. The proposed performance model is then described in Section IV, while Section V compares the modelling results and those obtained through accurate simulations. Finally, concluding remarks and future work plans are given in Section VI.

2. RELATED WORK

Much of the previous analytical latency models in wormhole-switched off-chip networks have been formulated for a specific topology and traffic pattern [12][13]. In [7], the authors utilized a queueing model and presented a performance model to overcome the problem of buffer allocation in NoC-based systems, but the approach cannot handle the wormhole-switched networks. The authors in [6] addressed the allocation of link capacities in NoCs through an analytical latency model. Their proposed model, however, only works for networks with single flit buffers and also ignores the queueing delays and network contentions. A more accurate analytical router model has been proposed in [16]. This work assumes that packet arrivals to the network follow the Poisson distribution. As a result, such models lack the accuracy for use in applications with bursty traffic such as multimedia application. In [11] a mathematical performance model for NoC-based systems was proposed to predict performance metrics in NoCs. However, the modelling approach was limited to *k*-ary *n*-cube networks with single flit buffers and dimension-order routing algorithm. A worst-case analysis of flow latency in the NoC-based systems was considered in [8]. This paper optimizes the traffic regulation parameters aiming for buffer optimization. Although this approach is proper for such a system with real-time requirements, many NoC-based systems have more relaxed timing constraints.

To the best of our knowledge, this work proposes the first average case analytical model for on-chip routers which takes into account the burstiness of the traffic. The proposed model can be used to develop a thorough performance analysis for arbitrary network topology with wormhole switching under arbitrary traffic pattern. Our proposed model, besides providing performance metrics such as average latency and router blocking time, gives useful feedbacks about the network behaviour which can be used in an optimization loop for NoCs such as topology selection, application mapping, and buffer allocation.

3. FOUNDATION

Queueing theory is an appropriate and useful modelling tool for system analysis and performance evaluation in computer and telecommunications network [14]. Since our proposed model has been constructed on the G/G/1 priority queue [3][22], in this section we give a quick review on the G/G/1 queue and priority queue concepts.

A. G/G/1 Queue

The G/G/1 model has a single service facility with one server, unlimited waiting room and the first-come first-served queue discipline. The service times are independent and identically distributed with a general distribution, the interarrival times of customers are also independent and identically distributed with a general distribution, and the interarrival times are independent of the service times. It is assumed that the general interarrival time and service time distributions are each partially specified by their first two moments. We should remind here that the *n*th moment of a random variable X is defined as the average of X^n $\left(\overline{X^n} = \sum_{i=1}^{k}(X_i)^n/k\right)$. All descriptions of this model thus depend only on the basic parameter 4-tuple $\left(\bar{a}, \overline{a^2}, \bar{s}, \overline{s^2}\right)$, where $\bar{a}$ and $\overline{a^2}$ are the first and second moments of the customers' interarrival time, and similarly, $\bar{s}$ and $\overline{s^2}$ are the first and second moments of the service time. Also in this work we consider the

arrival rate and service rate as $\lambda = 1/\bar{a}$ and $\mu = 1/\bar{s}$, respectively. The mean waiting time of a G/G/1 queueing system can be approximated by Allen-Cunneen formula [3].

$$\bar{W}_{G/G/1} \approx \frac{\rho(C_A^2+C_S^2)}{2\mu(1-\rho)} \quad (1)$$

where ρ is the utilization factor of the server and equal to λ/μ, and C_A and C_S are the coefficient of variation (CV) of the interarrival time and service time respectively [3]. We remind that the relationship between CV of random variable X and its moments is represented by $C_X^2 = \overline{x^2}/\bar{x}^2 - 1$.

B. Priority Queue

We consider a system with one server in which the customers have preferential treatment based on priorities associated with them. We assume that the priority of a customer is an integer fixed at arrival time, and a customer with priority i (i = 1, 2, …, p) belongs to class i. We say one customer has higher priority than another if it belongs to a priority class with lower index. In other words, the lower the index, the higher the priority. The priority queueing system to be studied is depicted in Figure 1, where the different queue levels correspond to the different priority classes. For the service discipline, we assume that whenever a customer is completed, the server is next assigned to that customer at the head of the highest priority nonempty queue. Once a customer begins on the server, it is allowed to run to completion; i.e., the service discipline is nonpreemptive. Independent and identically distributed arrivals and service times are assumed for the ith class with the arrival and service rate denoted by λ_i and μ_i, respectively. The mean waiting time of random arrivals to the ith queue, $\bar{W}_i$, can be written as [22]11:

$$\bar{W}_i = \frac{\bar{R}}{(1-\sum_{k=1}^{i-1}\rho_k)(1-\sum_{k=1}^{i}\rho_k)} \quad (2)$$

where $\bar{R}$ is the residual service time seen by an incoming customer. In a G/G/1 queueing system, $\bar{R}$ is approximated by [3]:

$$\bar{R} \approx \sum_{k=1}^{p} \frac{\rho_k}{2\mu_k}\left(C_{A_k}^2 + C_{S_k}^2\right) \quad (3)$$

where μ_k and ρ_k are average service rate and utilization factor of class k, respectively. Also, C_{A_k} and C_{S_k} are CV of interarrival time and service time of class k, respectively.

In all the analysis we have reviewed so far, the queue size of each class was infinite. However, in the case of wormhole switching this is not a true assumption, because in wormhole switching each buffer can hold only finite number of flits. Later in subsection V.B, we analyze such a queueing system.

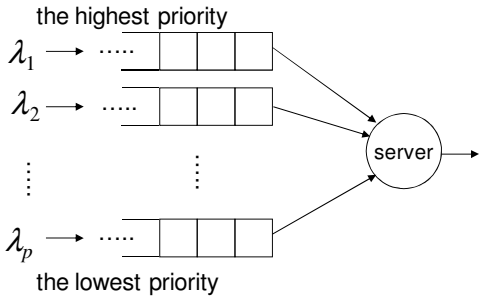


Figure 1: A typical priority queueing system

4. PERFORMANCE ANALYSIS

The following assumptions are made when developing the proposed performance model.

- The PQ model works for deterministic routing algorithms which may be minimal or non-minimal.

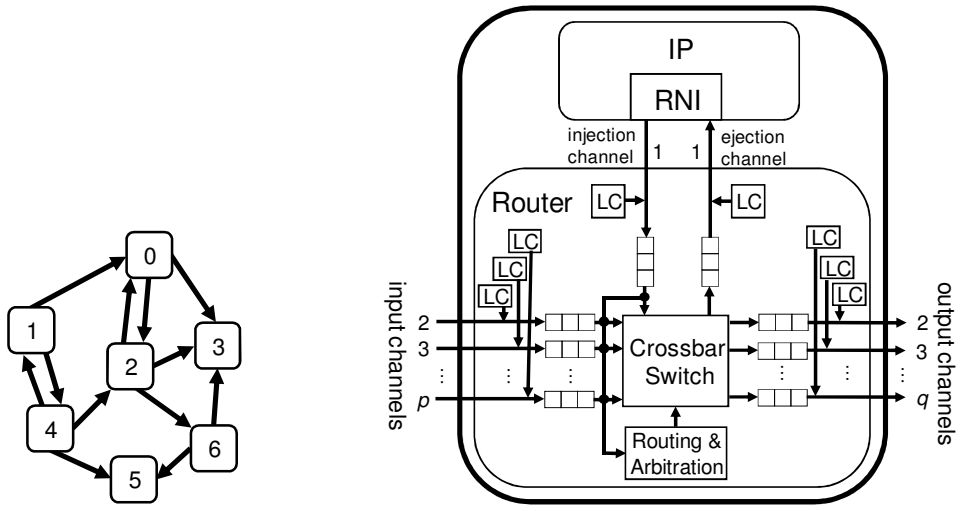
- The switching method is wormhole and messages are broken into packets.
- There is one finite FIFO queue per channel and channels are allocated per packet. It means that the channel is released when the whole packet has passed through the channel.
- Packets are consumed immediately by the destination node.

In order to characterize network performance, architecture and application models are essential.

A. Architecture Model

As shown in Figure 2.a, a directed graph can represent the topology of an NoC architecture. Vertices and edges of the graph show nodes and channels of the NoC, respectively. The structure of a single node is depicted in Figure 2.b. Every node contains an intellectual property (IP) core and a router with p input channels and q output channels. Each IP core performs its own computational, storage or I/O processing functionality, and is equipped with a Resource-Network-Interface (RNI). The RNI translates data between IP cores and routers by packing/unpacking data packets and also manages the packet injection process. Packets are injected into the network on the injection channel (input port 1) and leave the network from the ejection channel (output port 1). Generally, each channel connects output port *j* of node *N* to input port *i* of node *M*. Therefore, we denote this channel OC_j^N (*j*th output channel of router *N*) or IC_i^M (*i*th input channel of router *M*). We consider the general reference architecture for routers in [4] and it comprises the following major components.

- *Buffer*. This is a finite FIFO buffer for storing packets in transit. In the model shown in Figure 2.b, a buffer is associated with each input physical channel and each output physical channel. In alternative designs, buffers may be associated only with inputs (input buffering) or outputs (output buffering).



(a) (b)

Figure 2: (a) A graph representation of a general NoC architecture, (b) Structure of a node in an NoC-based system

- *Link controller* (*LC*). The flow of packets across the physical channel between adjacent routers is implemented by the link controller. The link controllers on either side of a channel coordinate to transfer flits.
- *Crossbar switch*. This component is responsible for connecting router input channels to router output channels.
- *Routing and arbitration unit*. This component implements the routing algorithms, selects the

output channel for an incoming packet, and accordingly sets the crossbar switch. Routing is only performed with the head flit of a packet. If two or more packets simultaneously request the same output channel, the arbiter must provide for arbitration among them. In this work, we suppose that input channels have a descending order of priority in a clock-wise direction for each output channel. The incoming packets from injection channel have the highest priority in each priority group. Usually, a control mechanism prevents the network from being overloaded. Therefore, it is guaranteed that the router is never overloaded and incoming packets from lower priority channels do not face starvation. If the requested output channel is busy, the incoming head flit remains in the input buffer. It will be routed again after the channel is freed and if it successfully arbitrates for the channel.

Similar to the network model in [4], we suppose that the routing decision delay for a packet, crossing time of a flit over the crossbar switch, and transfer time of a flit across a wire between two adjacent routers are t_r, t_s, and t_w, respectively. Also the transfer times of a flit across the injection and ejection channels are considered to be t_{inj} and t_{ej}, respectively. Having looked at Figure 3, we can infer that the latency of a head flit of a one hop packet in the absence of contention includes injection channel delay (t_{inj}), first router delay ($t_r + t_s$), inter-node wire delay (t_w), second router delay ($t_r + t_s$), ejection channel delay (t_{ej}). Therefore, we can write it as $t_{inj} + (t_r + t_s) + t_w + (t_r + t_s) + t_{ej}$.

In this study, we consider the wormhole switching under deterministic routing algorithm. Although adaptive routing algorithms avoid congested channels and result in more balanced load on the network, they may cause out-of-order packet delivery. The re-order buffers needed at the destination for ordering the packets impose large area and power on system [15]. Deterministic routers not only are more compact and faster than adaptive routers, but also guarantee in-order packet delivery. Therefore, it is not surprising that designers would like to use deterministic routing algorithms in the NoCs which desire small silicon overheads. Thus, in this research we use the deterministic routing for deadlock-free routing.

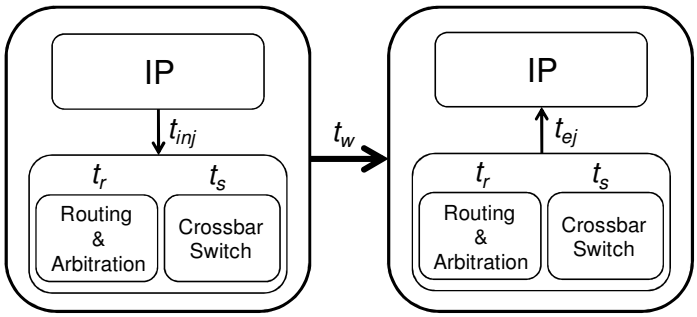


Figure 3: Delay of a one hop flow

B. Application Model

The target application can be specified by the *communication graph* [18]. The communication graph is a directed graph where each vertex represents an IP core, and the directed edge represents the communication between cores. The weight of the edge represents the communication rate between source and destination. In experimental results section, we consider the communication graph of a multimedia application (Table 3). Although generation of data packets in NoC nodes has dependence, especially in application-specific platforms, the studies in [1][21] show that compared to real traffic traces in NoCs, it will be still accurate to model their traffic generation separately as independent bursts of packets with statistical characteristics.

We assume that the packet injection process to the router N has a general distribution with mean value of λ^N packets/cycle and coefficient of variation of C_A. Also, the probability of packet transmission from the source node S to the destination node D is $P^{S \to D}$. This information can be easily extracted from the communication graph of application. Messages are broken into some

packets with arbitrary size distribution. m and σ_m represent the average and standard deviation of the packet size respectively, as listed in Table 1 along with other parameters.

5. COMMUNICATION ANALYSIS

To have a better view of the proposed model, the main idea of the analysis approach is summarized here.

a) To estimate the average latency of flows, it is essential to estimate the packet waiting times for network channels.
b) Each channel is modelled as a G/G/1 priority queue and the waiting time to access each channel is calculated based on the packet arrival rate and channel service time which are calculated in (c) and (d), respectively.
c) Given the communication volume among IP cores and routing algorithm, the packet arrival rate to each channel is determined.
d) The channel service time, which is part of the waiting time, is calculated recursively for each communication path starting from the destination node.

A. Latency Model

The average packet latency (L) is used as the performance metric. We assume that the packet latency spans the instant when the packet is created, to the time when the packet is delivered to the destination node. We also assume that the packets are consumed immediately once they reach their destination nodes.

In Figure 4, consider a flow which is generated in IP^S, and reaches its destination (IP^D) after traversing R^S, R^M, and R^D. The latency of this packet $(L^{S\to D})$ consists of two parts: the latency of head flit $(L_h^{S\to D})$ and the latency of body flits (L_b). In other words,

$$L^{S\to D} = L_h^{S\to D} + L_b \tag{4}$$

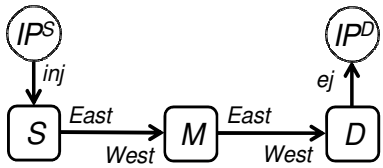


Figure 4: A two-hops flow from IP^S (source) to IP^D (destination)

$L_h^{S\to D}$ is the time since the packet is created in IP^S, until the head flit reaches the IP^D, including the queueing time spent at the source node and intermediate nodes. In Figure 4, $L_h^{S\to D}$ can be computed as

$$\begin{aligned} L_h^{S\to D} &= t_{inj} + \left(t_r + W^S_{inj\to East} + t_s\right) \\ &\quad + t_w + \left(t_r + W^M_{West\to East} + t_s\right) \\ &\quad + t_w + \left(t_r + W^D_{West\to ej} + t_s\right) + t_{ej} \end{aligned} \tag{5}$$

where $W^N_{i\to j}$ is the mean waiting time for a packet from IC^N_i to OC^N_j. Note that in Figure 4, the channel between S and M can be addressed with OC^S_{East} or IC^M_{West}.

Once the head flit arrives at the destination, the flow pipeline cycle time is determined by the maximum of the switch delay and wire delay. For an input-only or output-only buffered router, this cycle time would be given by the sum of the switch and wire delays [4]. In other words, in an input-output buffered router L_b is given by

$$L_b = (m-1) \times max(t_s, t_w) \tag{6}$$

and in an input-only or output-only buffered router it is

$$L_b = (m-1)(t_s + t_w) \tag{7}$$

The only unknown parameter for computing the latency is $W_{i\to j}^{N}$. This value can be calculated using a queueing model.

Table 1: Parameter notation.

t_r	Time spent for packet routing decision (*cycles*)	*Architecture parameters*
t_s	Time spent for switching (*cycles*)	
t_w	Time spent for transmitting a flit between two adjacent routers (*cycles*)	
m	Average size of packets (*flits*)	
σ_m	Standard deviation of packet size (*flits*)	
$L^{S\to D}$	Average packet latency from IP^S to IP^D (*cycles*)	
L	Average packet latency in the network (*cycles*)	
IP^N	The IP core located at address N	
R^N	The router located at address N	
IC_i^N	The ith input channel in router R^N	
OC_j^N	The jth output channel in router R^N	
IB_i^N	Capacity of the buffer in IC_i^N (*flits*)	
OB_j^N	Capacity of the buffer in OC_j^N (*flits*)	
$P^{S\to D}$	Probability of a packet is generated in IP^S and is delivered to IP^D $(\sum_S \sum_D P^{S\to D} = 1)$	*Application parameters*
λ^N	Average packet injection rate of IP^N (*packets/cycle*)	
$\lambda_{i\to j}^N$	Average packet rate from IC_i^N to OC_j^N (*packets/cycle*)	
λ_j^N	Average packet rate to OC_j^N (*packets/cycle*) $\left(\lambda_j^N = \sum_i \lambda_{i\to j}^N\right)$	
$P_{i\to j}^N$	Probability of a packet entered form IC_i^N to be exited from OC_j^N	
μ_j^N	Average service rate of the OC_j^N (*packets/cycle*)	
$\bar{R}_j^N$	Residual service time of OC_j^N seen by an incoming flow (*cycle*)	
$C_{B_j^N}$	Coefficient of variation (CV) for service time of the OC_j^N	
$C_{A_{i\to j}^N}$	CV for interarrival time of packets from IC_i^N to OC_j^N	
$\rho_{i\to j}^N$	The fraction of time that the OC_j^N is occupied by packets from IC_i^N	
$W_{i\to j}^N$	Average waiting time for a packet from IC_i^N to OC_j^N (*cycles*)	

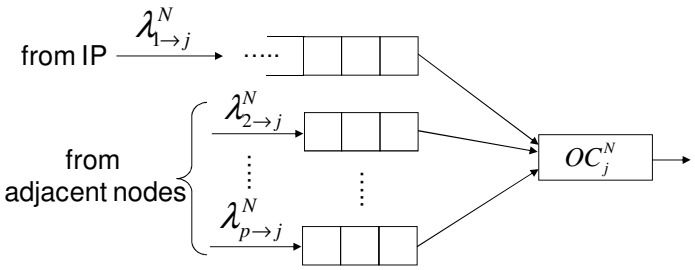


Figure 5: Queueing model of a channel of an arbitrary topology

B. Waiting Time Estimation

A router is primarily modelled based on nonpreemptive priority queueing system. Let us consider, for instance, the jth output channel of R^N $\left(OC_j^N\right)$. As can be seen in Figure 5, this

channel is modelled as a server in a priority queueing system with p classes (IC_1^N to IC_p^N), the arrival rate $\lambda_{i\to j}^N\ (1 \le i \le p)$, and served by one server $\left(OC_j^N\right)$ of service rate μ_j^N. Both interarrival and service times are independent and identically distributed with arbitrary distributions.

The queueing model for output channel represented in Figure 5 is different from traditional priority queue model in Figure 1.

Since in the wormhole switching each input buffer can hold finite number of flits, we cannot use Eq. (2) and we have to compute the average waiting time for the head of class *i* in this special case of priority queues. Using a technique similar to that employed in the literature for general priority queues [3]11[22], we can write

$$W_{i\to j}^N = \begin{cases} \bar{R}_j^N/\left(1-\rho_{1\to j}^N\right), & i = 1, \\ \bar{R}_j^N/\left(1-\sum_{k=1}^{i-1}\rho_{k\to j}^N\right)^2, & 2 \le i \le p \end{cases} \tag{8}$$

where $\rho_{k\to j}^N$ is the fraction of time that the OC_j^N is occupied by packets from IC_k^N and equals

$$\rho_{k\to j}^N = \lambda_{k\to j}^N/\mu_j^N \tag{9}$$

Also $\bar{R}_j^N$ is the residual service time of OC_j^N seen by an incoming head flit. Based on Eq. (3), in a G/G/1 queueing system the residual service time is approximated by [3]

$$\bar{R}_j^N \approx \sum_{i=1}^p \rho_{i\to j}^N \frac{C_{A_{i\to j}^N}^2 + C_{S_j^N}^2}{2\mu_j^N} \tag{10}$$

Since we do not have enough insight about the CV of interarrival time at each channel $\left(C_{A_{i\to j}^N}\right)$, we suppose that $C_{A_{i\to j}^N}$ is the same for all input channels in the network and equal to the coefficient of variation of the arrival process to network $\left(C_{A_{i\to j}^N} = C_A\right)$. Therefore, we can rewrite Eq. (10) as

$$\bar{R}_j^N \approx \frac{C_A^2 + C_{S_j^N}^2}{2\mu_j^N} \sum_{i=1}^p \rho_{i\to j}^N \tag{11}$$

Due to the definition of $\rho_{i\to j}^N$, we can write $\sum_{i=1}^p \rho_{i\to j}^N = \sum_{i=1}^p \left(\lambda_{i\to j}^N/\mu_j^N\right)$. It is obvious that the average packet rate to an output channel of R^N is equal to sum of the average packet rate from all input channel of R^N to this output channel. Therefore, we can write $\sum_{i=1}^p \lambda_{i\to j}^N/\mu_j^N = \lambda_j^N/\mu_j^N = \rho_j^N$. As a result, Eq. (11) can be rewritten as

$$\bar{R}_j^N \approx \rho_j^N \left(C_A^2 + C_{S_j^N}^2\right)/2\mu_j^N \tag{12}$$

By substituting $\bar{R}_j^N$ in Eq. (8) we can write

$$W_{i\to j}^N = \begin{cases} \dfrac{\rho_j^N\left(C_A^2 + C_{S_j^N}^2\right)}{2\left(\mu_j^N - \lambda_{1\to j}^N\right)}, & i = 1, \\ \dfrac{\lambda_j^N\left(C_A^2 + C_{S_j^N}^2\right)}{2\left(\mu_j^N - \sum_{k=1}^{i-1}\lambda_{k\to j}^N\right)^2}, & 2 \le i \le p \end{cases} \tag{13}$$

Therefore, to compute the $W_{i\to j}^N$ we have to calculate the arrival rate from IC_i^N to OC_j^N $\left(\lambda_{i\to j}^N\right)$, and also first and second moments of the service time of OC_j^N $\left(\bar{s}_j^N, \overline{\left(s_j^N\right)^2}\right)$. In the following two subsections, packet arrival rate and channel service time are computed.

C. Packet Arrival Rate Calculation

Assuming the network is not overloaded, the arrival rate from IC_i^N to OC_j^N can be calculated using the following general equation

$$\lambda_{i\to j}^N = \sum_S \sum_D \lambda^S \times P^{S\to D} \times R\left(S \to D, IC_i^N \to OC_j^N\right) \quad (14)$$

In Eq. (14), the routing function $R\left(S \to D, IC_i^N \to OC_j^N\right)$ equals 1 if a packet from IP^S to IP^D passes from IC_i^N to OC_j^N; it equals 0 otherwise. Note that we assume a deterministic routing algorithm, thus the function of $R\left(S \to D, IC_i^N \to OC_j^N\right)$ can be predetermined, regardless of topology and routing algorithm. After that, the average packet rate to OC_j^N can be easily determined as

$$\lambda_j^N = \sum_i \lambda_{i\to j}^N \quad (15)$$

D. Channel Service Time Estimation

After estimating the packet arrival rates, now we focus on the estimation of the moments of channel service times. At first, we assign a positive integer index to each output channel. Let D_j^N be the set of all possible destinations for a packet which passes through OC_j^N. The index of OC_j^N is equal to the maximum of distances among N and each M where $M \in D_j^N$. Obviously, the index of a channel is between 1 and diameter of the network. In addition, the index of all ejection channels is supposed to be 0. After that, all output channels are divided into some groups based on their index numbers, so that group k contains all channels with index k.

Determination of the channel service time moments starts at group 0 (ejection channels) and works in ascending order of group numbers. Therefore, the waiting time from lower numbered groups can then be thought of as adding to the service time of packets on higher numbered groups. In other words, to determine the waiting time of channels in group k, we have to calculate the waiting time of all channels in group $k-1$. This approach is independent of the network topology and works for all kinds of deterministic routing algorithm, whether minimal or non-minimal.

In the ejection channel of R^N, the head flit and body flits are accepted in $t_s + t_w$ and L_b cycles, respectively. Therefore, we can write $\bar{s}_1^N = t_s + t_w + L_b$ and since the standard deviation of packet size is known, we can easily compute $C_{s_1^N}$. Now, by using Eq. (13), the waiting time of input channels for ejection channel, $W_{i\to 1}^N$, can be determined for all nodes in the network, where $2 \le i \le p$.

Although the moments of service time can be computed simply for all ejection channels, service time moments of the other output channels cannot be computed in a direct manner by a general formula, and we have to use a more complicated approach. Consider flow *f* in Figure 6.a which passes through routers *M*, *N* and *O*. We suppose that the average service time of OC_k^N $(\bar{s}_k^N)$ with index *x* has been computed before, and now we want to compute the average service time of OC_i^M $(\bar{s}_i^M)$ with index *x*+1. At the first glance, it seems that the average service time of OC_i^M is equal to $t_s + t_w + t_r + W_{j\to k}^N + \bar{s}_k^N$. However, we should ponder the effect of buffer spaces in input and output port of a router on channels service time. In the Figure 6.a, when the tail flit of the passing packet through OC_k^N reaches position 2, the service time of OC_k^N is finished and similarly the service time of OC_i^M is finished, when the tail flit of the packet reaches position 1. Therefore, the preceding equation should be decreased by the spent time for reaching position 2 from position 1. Therefore, we can write $t_s + t_w + t_r + W_{j\to k}^N + \bar{s}_k^N - \left(IB_j^N + OB_k^N\right) \times max(t_s, t_w)$ where IB_j^N and OB_k^N are the capacity of the buffer in IC_j^N and OC_k^N, respectively.

Although the effect of buffer size on the channel service time is considered in this equation, it does not work in all cases. Because, as shown in Figure 6.b, there might be several paths for

different flows in OC_i^M, so we should consider the possibility of using several output channels to make the next hop. Now, we can estimate the first moment or average service time of OC_i^M as

$$\bar{s}_i^M = \sum_{k=1}^{q} P_{j\to k}^N \begin{pmatrix} t_s + t_w + t_r + W_{j\to k}^N + \bar{s}_k^N \\ -\left(IB_j^N + OB_k^N\right) \times max(t_s, t_w) \end{pmatrix} \quad (16)$$

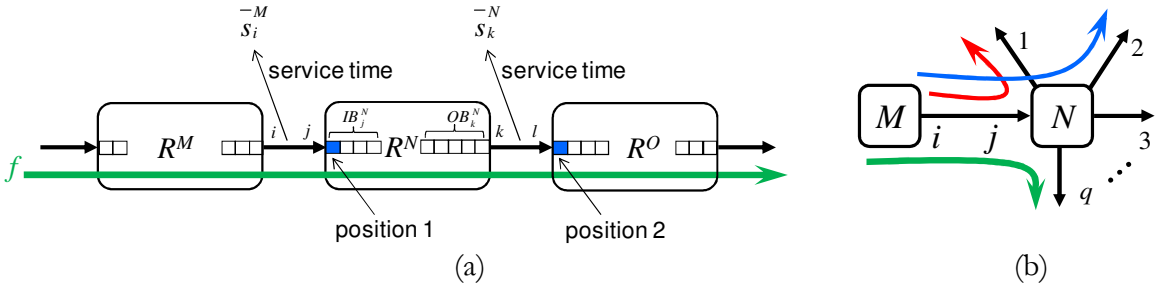


Figure 6: (a) A passing flow from R^M, R^N, and R^O, (b) Some possible path for an entering flow to R^N

where $P_{j\to k}^N$ is the probability of a packet entered form IC_j^N to be exited from OC_k^N and equals

$$P_{j\to k}^N = \lambda_{j\to k}^N / \lambda_i^M \quad (17)$$

Here, we should remind that to calculate $\bar{s}_i^M$, all values of $\bar{s}_k^N$ $(1 \le k \le q)$ must be computed before. Likewise, the second moment of service time of OC_i^M can be approximated by

$$\overline{(s_i^M)^2} = \sum_{k=1}^{q} P_{j\to k}^N \begin{pmatrix} t_s + t_w + t_r + W_{j\to k}^N + \bar{s}_k^N \\ -\left(IB_j^N + OB_k^N\right) \times max(t_s, t_w) \end{pmatrix}^2 \quad (18)$$

Finally, the CV of channel service time for OC_i^M can be given by

$$C_{s_i^M}^2 = \overline{(s_i^M)^2} / (\bar{s}_i^M)^2 - 1 \quad (19)$$

Now, we are able to compute the average waiting time of all output channels using Eq. (13). After computing $W_{i\to j}^N$ for all nodes and channels, the average packet latency between any two nodes in the network, $L^{S\to D}$, can be calculated. The average packet latency is the weighted mean of these latencies.

$$L = \sum_S \sum_D P^{S\to D} \times L^{S\to D} \quad (20)$$

where $P^{S\to D}$ is the probability of a packet is generated in IP^S and is delivered to IP^D.

E. Analysis Flow

To have a clear view of our proposed analysis approach, the flowchart description of the performance model is shown in Figure 7. Average packet latency in the network is computed in following steps.

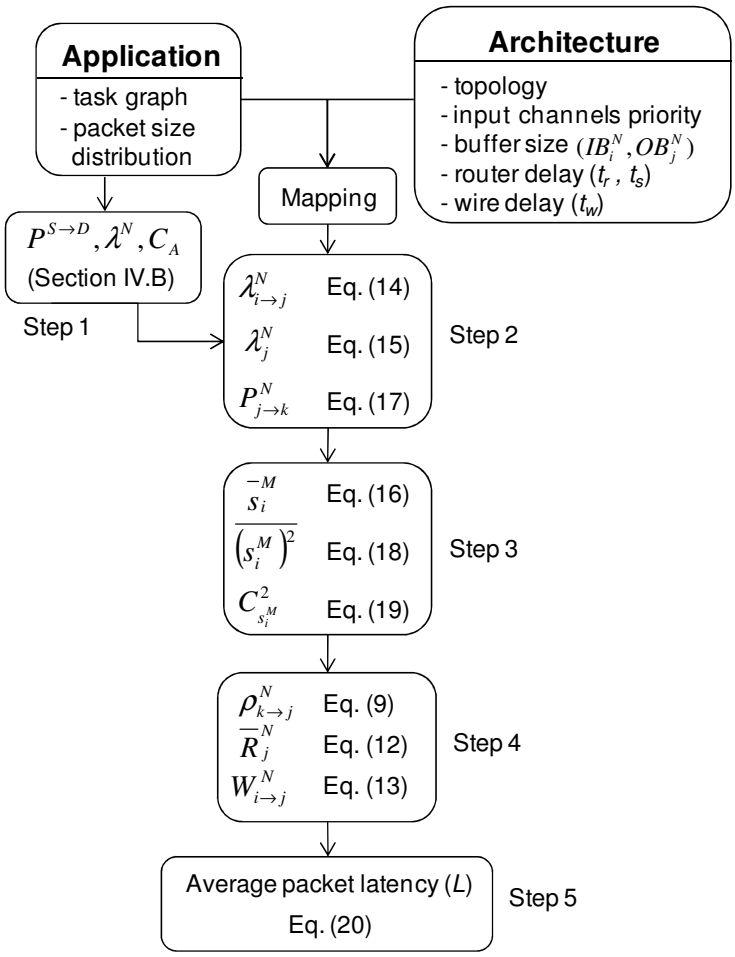


Figure 7: Flowchart of proposed analytical model

- Step 1. Given the application communication graph, we can easily extract the temporal and spatial features of communication among IP cores with the computational complexity of $O(n^2)$ where n is the number of nodes in the network.
- Step 2. After a mapping phase, the traffic input rates to network channels are computed. The computational complexity of this step is proportional to n^2 and d, where d is the diameter of the network. As a result the overall complexity of this step is obtained as $O(n^2d)$.
- Step 3. Statistical distribution of channel service times are partially computed using equations 16, 18, and 19. The computational complexity of this step is $O(nq^2)$, where q is the number of output ports per router.
- Step 4. After computing the channel service times, the average waiting time of packets are computed with the complexity of $O(np^2q)$, where p is the number of input ports.
- Step 5. The complexity of the average latency calculation using Eq. (20) is $O(n^2d)$ as with step 2.

As a result, the overall complexity of the PQ model is obtained as $O(n^2d) + O(np^3)$, if the number of input and output ports of routers are the same. More especially, in the case of 2D mesh network, a router is connected to maximum four neighbouring routers and also a local IP core through injection and ejection channels. Therefore, p and q equal 5 and d is proportional to $\sqrt{n}$. As a result, the proposed model has time requirement $O(n^{5/2})$ for 2D mesh networks.

6. EXPERIMENTAL RESULTS

The proposed analytical model has been validated through a discrete-event simulator that mimics the behaviour of the routing algorithm in the network at the flit level. The simulator uses the same assumptions as the analytical model. To achieve a high accuracy in the simulation results, we use the batch means method [20] for simulation output analysis. There are 10 batches and each batch includes 1,000 up to 80,000,000 packets depending on the workload type, traffic injection rate, packet length, and network size. Statistics gathering was inhibited for the first batch to avoid distortions due to the startup transient. The standard deviation of latency measurements is less than 1.8% of the mean value. As a result, the confidence level and confidence interval of simulation results are 0.99 and 0.02, respectively [20]. In other words, the probability of $0.98\mu_X \leq \bar{X} \leq 1.02\mu_X$ is 0.99 where μ_X is the real average value and $\bar{X}$ is the estimated average value by simulator [20].

For the sake of comprehensive study, numerous validation experiments have been performed for several combinations of workload types, network sizes and packet lengths. In what follows, the accuracy of PQ model will be assessed in Multi-processor system-on-chip and Application-specific system-on-chip platforms. Since their applications differ starkly in purpose, these classes of NoCs have substantially different traffic patterns.

A. Multi-processor System-on-Chip Platform

We have considered a 9x9 mesh on-chip interconnect and input-output buffered router with 4 flits in each input and output channel. It takes 2 clock cycles to pass a flit within a router and 1 clock cycle to transmit a flit between neighbouring routers. We also consider the XY routing algorithm to route the data packets among IP cores. Packet destinations are uniformly distributed across the network nodes. Following a Poisson process, nodes generate packets independently of each other. It means that the time between two successive packet generations in an IP core is distributed exponentially. The Poisson model widely used in many performance analysis studies, and there are a large number of papers in many application domains that are based on this stochastic assumption [7].

Figure 8.a depicts latency results predicted by the PQ model explained in the previous section, plotted against those provided by the simulator for the two different fixed packet lengths $m = 4$ and 64 flits. The horizontal axis in the figure shows the packet generation rate while the vertical axis shows the average packet latency. The figure reveals that in both cases the analytical model predicts the average latency with a good degree of accuracy. However, some discrepancies around the saturation point are apparent. These can be accounted for by the approximations made to facilitate the derivation of different variables, e.g. the approximation made to estimate CV of the interarrival time of each channels. Such an approximation greatly simplifies the model as it allows us to avoid computing the exact distribution of the interarrival time at a given channel, which is not a straightforward task due to interdependencies between successive arrival times at channels as wormhole switching relies on a blocking mechanism for flow control. However, the analytical model can still predict the average latency fairly accurately in almost all traffic regions which are appropriate for network operations.

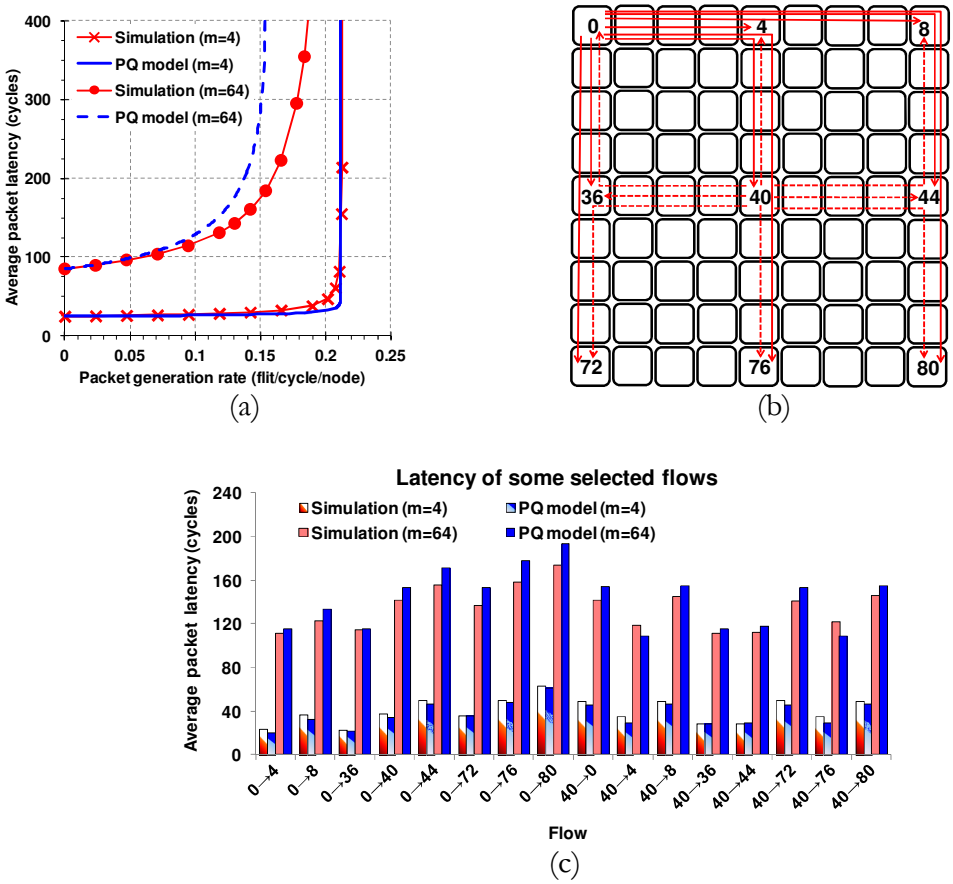


Figure 8: (a) The average packet latency of all flows against simulation results, (b) Some selected flows of uniform traffic in a 9X9 mesh network, (c) The average packet latency of the flows in Figure 8(b), predicted by the PQ model against simulation results

Also, we compare the average latency of some selected flows in the network predicted by the PQ model and the simulator. Figure 8.b shows these flows from node 0 in the corner of the network and node 40 in the centre of the network. Figure 8.c depicts the average latency of these flows when the flit injection rates are 0.18 and 0.12 flits/cycle/node for the packet length of 4 and 64 flits, respectively. The comparison results show that the model is in good conformity with the simulator with average relative error of 7.5%.

B. Application-specific System-on-Chip Platform

Analyzing the multimedia applications in NoCs shows bursty patterns of traffic over a wide range of time scales [23]. Since the Poisson process cannot model the bursty traffic very well, we use Markov-modulated Poisson process (MMPP) model [5] to model the temporal burstiness of traffic. MMPP has been widely employed to model the traffic burstiness in the temporal domain [5]. Figure 9 shows a two-state MMPP in which the arrival traffic follows a Poisson process with rate λ_0 and λ_1. The transition rate from state 0 to 1 is r_0, while the rate from state 1 to state 0 is r_1.

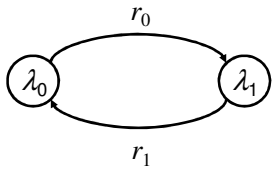


Figure 9: Two-state MMPP model

In this study, we use the notation MMPP(*k*) for the two-state MMPP in which $\lambda_1 = k\lambda_0$. Figure 10 shows the number of packet arrivals in a node against time for different values of *k* when the mean generation rate is 0.01 packet/cycle. Figure 10.a vividly shows that Poisson process (*k* =1) cannot model the traffic burstiness and figures 10.b – 10.f reveal that greater *k* results in greater intensity of packet burstiness.

The distribution of the interarrival times in the two-state MMPP is a second order hyper-exponential distribution [9]. Therefore, it is easy to compute the coefficient of variation of the interarrival time (C_A) which is reported in Table 2. We can infer that the greater the *k*, the greater the C_A. It means that C_A reflects the burstiness intensity very well. Here we recall that C_A is used in the proposed model to estimate the packet waiting time (Eq. 13).

Table 2: CV of packet interarrival time for different values of *k*.

Traffic model	$k = \lambda_1/\lambda_0$	C_A
MMPP(1)	1	1.00
MMPP(10)	10	1.55
MMPP(20)	20	2.04
MMPP(50)	50	3.08
MMPP(100)	100	4.28
MMPP(200)	200	6.00

To evaluate the capability of the proposed model to predict the performance of application-specific applications, we applied it to a generic multimedia system (MMS), which includes an H.263 video encoder, an H.263 video decoder, an mp3 audio encoder, and an mp3 audio decoder [7]. MMS includes 40 tasks and the tasks are assigned into 16 selected IPs. The communication volume requirements (in bytes) of this application are summarized in Table 3. In the next phase, we map theses 16 IPs into tiles of a 4x4 torus network randomly. Throughout the experiments, we considered an application-specific system-on-chip with 3 cycle router delay, 1 cycle wire delay and exponentially distributed packet size with average size of 16 flits. In addition, we supposed that input and output buffers of the routers have the capacity of 6 and 2 flits, respectively.

Latency of flows in this configuration is investigated in Figure 11.a and 11.b. Average latency of all packets generated by MMPP(10), MMPP(20) and MMPP(50) in different network throughput are compared in Figure 11.a. Figure 11.b depicts the average latency of all flows generated by MMPP(50) when the network operates at 0.02 flits/cycle/node.

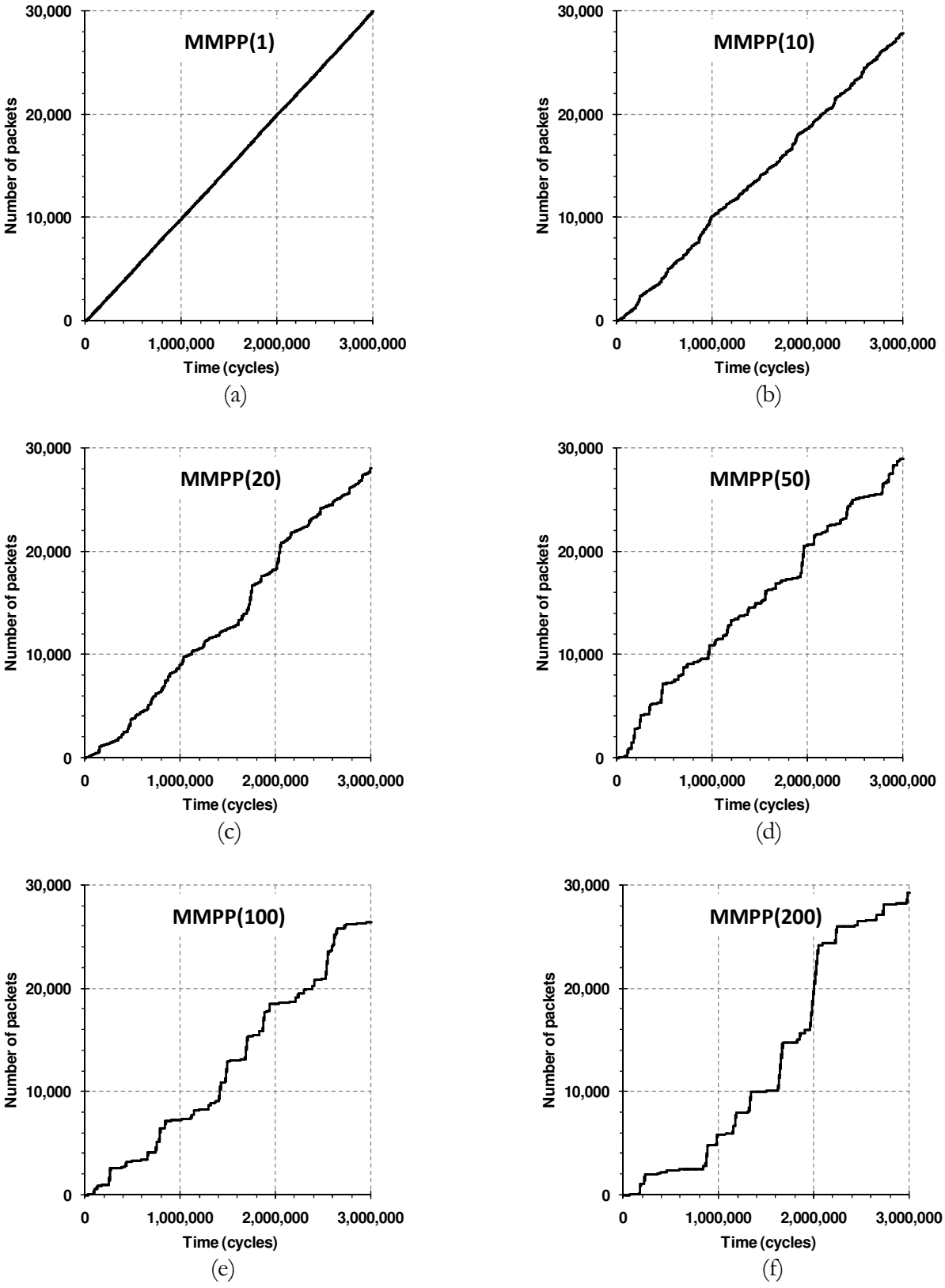


Figure 10: Number of packets against time in the MMPP model for (a) k =1 (Poisson model), (b) k =10, (c) k =20, (d) k =50, (e) k =100, (f) k =200.

Table 3: MMS application traffic requirement [7].

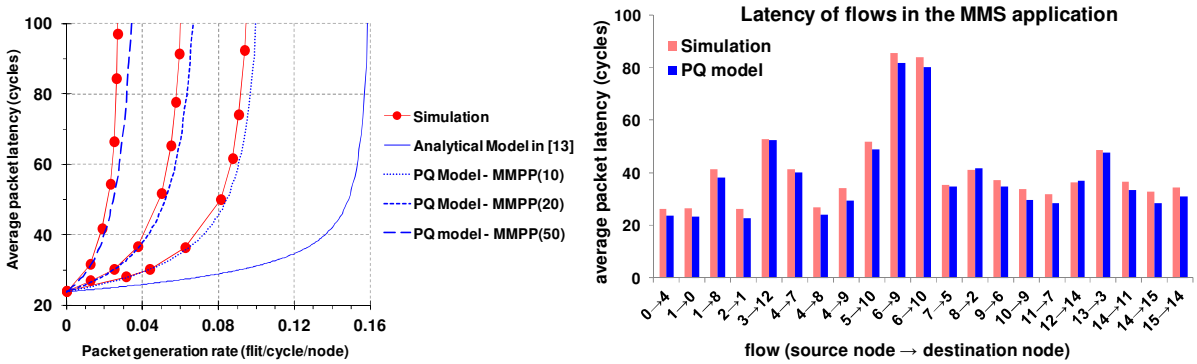
src	*dst*	*vol.*	*src*	*dst*	*vol.*
ASIC1	ASIC2	25	DSP2	DSP1	20363
ASIC1	DSP8	25	DSP3	ASIC4	38016
ASIC2	ASIC3	764	DSP3	DSP6	7061
ASIC2	MEM2	640	DSP3	DSP5	7061
ASIC2	ASIC1	80	DSP4	DSP1	3672
ASIC3	DSP8	641	DSP4	CPU	197
ASIC3	DSP4	144	DSP5	DSP6	26924
ASIC4	DSP1	33848	DSP6	ASIC2	28248
ASIC4	CPU	197	DSP7	MEM2	7065
CPU	MEM1	38016	DSP8	DSP7	28265
CPU	MEM3	38016	DSP8	ASIC1	80
CPU	ASIC3	38016	MEM1	ASIC4	116873
DSP1	DSP2	33848	MEM1	CPU	75205
DSP1	CPU	20363	MEM2	ASIC3	7705
DSP2	ASIC2	33848	MEM3	CPU	75584

As can be seen again, the model has a fairly good degree of accuracy in comparison to the simulation results with average relative error of 4.7%. We also implement the proposed model in [13] and compare it with the PQ model. Figure 11.a also shows that using the model in [13] to design a system with bursty traffic may lead us to less trusted decisions.

C. Arbitrary Topology

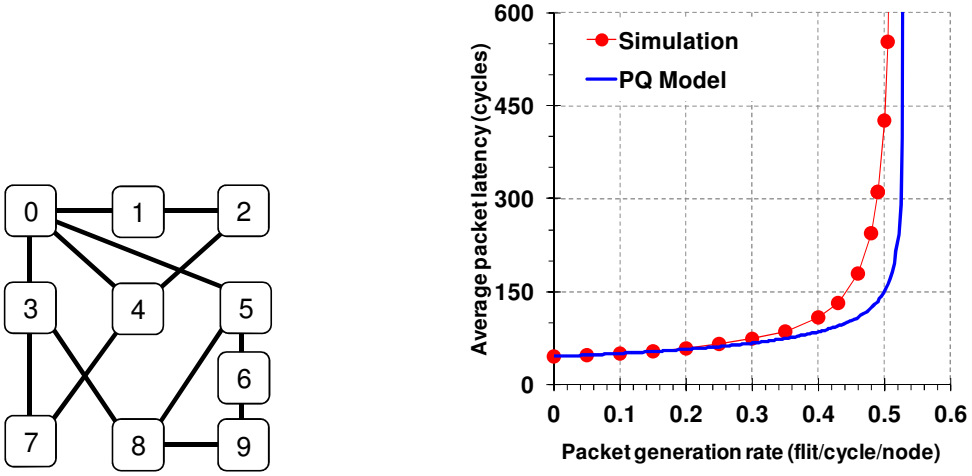
To show the capability of PQ model to predict the average latency in an arbitrary network, we consider the topology shown in Figure 12.a with the uniform workload and 32 flits packets. We used the CAR framework [10] to find the deadlock-free routes in this network. CAR constructs the channel dependency graph based on the network topology and application, and then deletes some edges from the channel dependency graph to guarantee the deadlock freedom. After that, CAR creates the routing space by finding all possible shortest paths for each flow. Finally, the simulated annealing heuristic is used to find congestion-aware routes.

Figure 12.b reveals that the proposed analytical model predicts the average packet latency accurately in almost all traffic regions which are appropriate for network operation.



(a) (b)

Figure 11: (a) The average packet latency of all flows in case of bursty traffic, (b) The average packet latency of each flow predicted by the PQ model against simulation results.



(a) (b)

Figure 12: (a) A custom topology, (b) The average packet latency of all flows.

Furthermore, to assess the proposed model for large networks, we compare the PQ model and simulation results for 16x16 mesh network and 8-dimensional hypercube network with 256 nodes. Dimension-order routing algorithms are used to route the data packets among IP cores. We choose the hypercube network for this experiment because it has totally different topological properties compared to the mesh network. Figure 13 shows the comparison result when the packet length is 32 flits, input buffers of the routers have the capacity of 8 flits and there are no output buffers.

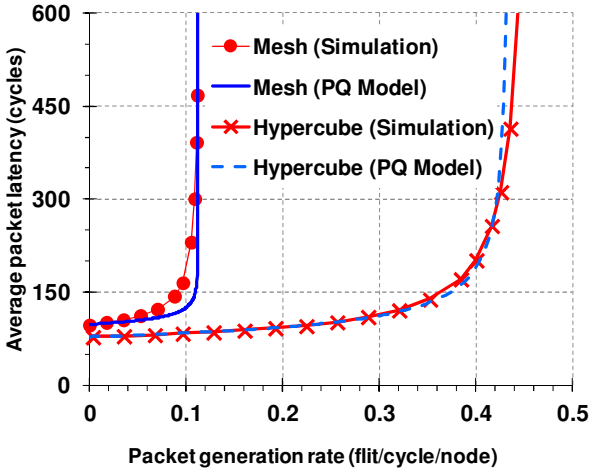


Figure 13: The average packet latency for a 16x16 mesh network and an 8-dimensional hypercube network with dimension-order routing.

D. Execution Time Comparison

Finally, the execution time of the proposed analytical model and simulation are compared. We implement both the PQ model and the simulator in C++ and run on the same computer. The execution times of the PQ model and simulation for mesh networks with various sizes from 9 (3x3) to 400 (20x20) nodes are compared in Figure 14. We simulate different size networks for 4 flits input and output buffers and 32 flits packets under uniform traffic. In such a traffic pattern, the number of flows are considerably increased with $O(n^2)$ where n is the number of nodes in the network.

As we mentioned previously in this section, a simulation run is divided into 10 batches. To reduce the simulation time we suppose that the simulator generates only three packets for each flow and averages the latency of these three packets to estimate the average latency of flows in each batch. Figure 14 shows that the proposed approach is much faster than the simulation and the overall speed-up due to the analytical model is more than 60,000 for small networks and more than 260,000 for large networks. Also, the simulation execution time grows faster for larger buffer size, longer packet length, heavier traffic, and more bursty traffic, while the execution time of the analytical approach is constant for the same platform under different operation conditions. Furthermore, we observe that the model accuracy fluctuates randomly with the network size and does not confirm any specific trend.

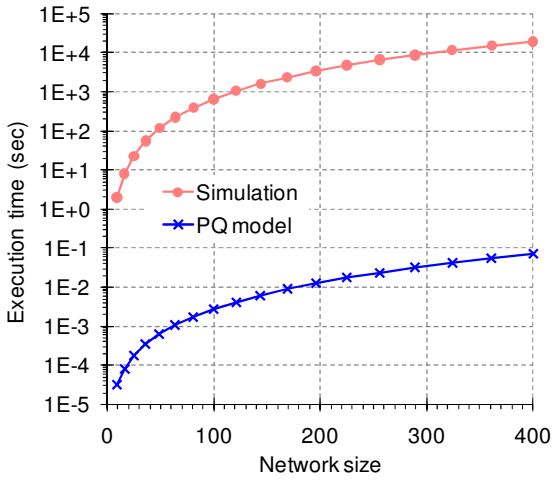


Figure 14: The execution time comparison of the PQ model and simulation for different size of mesh networks.

7. Conclusion and Future Work

Usually, system designers address the design problems by exploring the design space using detailed simulations. However, this approach has high run-time overhead and lacks of insights. Like in other disciplines of science and engineering, the use of analytical models can potentially address these limitations under certain assumptions. To this end, we propose the PQ model for predicting the communication performance of wormhole-switched NoC platforms. This queueing theory based model takes as input (1) an application communication graph, (2) a topology graph, (3) a mapping vector, and (4) a routing matrix, and estimates some performance metrics of the system such as average packet latency and router blocking time. The proposed model is validated through simulation experiments, and we have shown that the proposed model achieves a good degree of accuracy (< 10% error) making it a practical and useful evaluation tool that can be used by researchers in the field to gain insight into the performance behaviour of the designed system. The model independency on network topology and workload type makes it a robust tool to explore the huge design space of NoC-based systems.

In many applications such as real-time systems, the worst case execution time is of particular concern since it is important to know how much time might be needed in the worst case to guarantee that the task will always finish its jobs before the predetermined deadline. Therefore, we plan to advance this research by integrating the proposed average case model with an analytical worst case model. Finally, we would like to utilize the integrated performance model to find a near optimal solution for some design problems such as topology selection, module placement and buffer allocation problems in the network-based systems.

References

[1] J. H. Bahn and N. Bagherzadeh, “A Generic Traffic Model for On-Chip Interconnection

Networks," *The International Workshop on Network-on-Chip Architectures (NoCArc), Held in conjunction with the IEEE/ACM International Symposium on Microarchitecture* (*MICRO-41*), pp. 2008.

[2] D. Bertozzi, A. Jalabert, S. Murali, R. Tamhankar, S. Stergiou, L. Benini, and G. De Micheli, "NoC Synthesis Flow for Customized Domain Specific Multiprocessor Systems-on-chip," *IEEE Transactions on Parallel and Distributed Systems*, vol. 16, no. 2, pp. 113-129, 2005.

[3] G. Bolch, S. Greiner, H. de Meer, and K.S. Trivedi, *Queueing Networks and Markov Chains: Modeling and Performance Evaluation with Computer Science Applications*, 2nd Edition, John Wiley and Sons, 2006.

[4] J. Duato, C. Yalamanchili, and L. Ni, "Interconnection Networks: An Engineering Approach," IEEE Computer Society Press, 2003.

[5] W. Fischer and K. Meier-Hellstern, "The Markov-Modulated Poisson Process (MMPP) Cookbook", Performance Evaluation, vol. 18, no. 2, pp. 149-171, 1993.

[6] Z. Guz, I. Walter, E. Bolotin, I. Cidon, R. Ginosar, and A. Kolodny, "Network Delays and Link Capacities in Application-Specific Wormhole NoCs" *Journal of VLSI Design*, vol. 2007, Article ID 90941, 2007.

[7] J. Hu, U. Y. Ogras, and R. Marculescu, "System-level Buffer Allocation for Application-Specific Networks-on-Chip Router Design," *IEEE Transaction on Computer-Aided Design of Integrated Circuits and Systems*, vol. 25 no. 12, pp. 2919 – 2933, 2006.

[8] F. Jafari, Z. Lu, A. Jantsch, and M. H. Yaghmaee. "Buffer Optimization in Network-on-Chip Through Flow Regulation". *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 29, no. 12, pp 1973-1986, 2010.

[9] S. H. Kang and D. K. Sung, "Two-State MMPP Modelling of ATM Superposed Traffic Streams Based on the Characterisation of Correlated Interarrival Times", *In the Proceedings of the IEEE Global Telecommunications Conference* (*GLOBECOM*), vol. 2, pp. 1422-1426, 1995.

[10] A. E. Kiasari, A. Jantsch and Z. Lu, "A Framework for Designing Congestion-Aware Deterministic Routing," *In the Proceedings of the International Workshop on Network-on-Chip Architectures (NoCArc), Held in conjunction with the IEEE/ACM International Symposium on Microarchitecture* (*MICRO*-43), pp. 45-50, 2010.

[11] A. E. Kiasari, H. Sarbazi-Azad, and S. Hessabi, "Caspian: A Tunable Performance Model for Multi-Core Systems," *Euro-Par 2008 Parallel Processing*, E. Luque, T. Margalef, and D. Benitez, eds., Lecture Notes in Computer Science, Springer-Verlag, pp. 100-109, 2008.

[12] A. E. Kiasari, H. Sarbazi-Azad, and M. Ould-Khaoua, "An Accurate Mathematical Performance Model of Adaptive Routing in the Star Graph," *Future Generation Computer Systems*, vol. 24, no. 6, pp. 461-474, 2008.

[13] J. Kim and C. R. Das, "Hypercube communication delay with wormhole routing", *IEEE Transactions on Computers*, vol. 43, no. 7, pp. 806-814, 1994.

[14] L. Kleinrock, *Queueing Systems*, vol. 1, John Wiley, New York, 1975.

[15] S. Murali, T. Theocharides, N. Vijaykrishnan, M. Jane Irwin, L. Benini, G. De Micheli, "Analysis of Error Recovery Schemes for Networks on Chips", *IEEE Design & Test of Computers*, vol. 22, no. 5, pp. 434-442, 2005.

[16] U. Y. Ogras, P. Bogdan, R. Marculescu, "An Analytical Approach for Network-on-Chip Performance Analysis," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems,* vol. 29, no. 12, pp. 2001-2013, 2010.

[17] J. D. Owens, W. J. Dally, R. Ho, D. N. Jayasimha, S. W. Keckler, and L. S. Peh, "Research Challenges for On-Chip Interconnection Networks," *IEEE Micro*, vol. 27, no. 5, pp. 96-108, 2007.

[18] M. Palesi, R. Holsmark, S. Kumar, and V. Catania, "Application Specific Routing Algorithms for Networks on Chip," *IEEE Transactions on Parallel and Distributed Systems*, vol. 20, no. 3, pp. 316-330, 2009.

[19] P. P. Pande, C. Grecu, M. Jones, A. Ivanov, and R. Saleh, "Performance Evaluation and

Design Trade-offs for Network-on-Chip Interconnect Architectures," *IEEE Transactions on Computers*, vol. 54, no. 8, pp. 1025-1040, 2005.
[20]K. Pawlikowski, "Steady-State Simulation of Queueing Processes: A Survey of Problems and Solutions," *ACM Computing Surveys*, vol. 22, no. 2, pp. 123–170, 1990.
[21]V. Soteriou, H. Wang, L.-S. Peh, "A Statistical Traffic Model for On-Chip Interconnection Networks," *In the Proceedings of the IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems,* pp. 104-116, 2006.
[22]H. Takagi, *Queueing Analysis, vol.* 1*: Vacation and Priority Systems*, Amsterdam, 1991.
[23]G. V. Varatkar and R. Marculescu, "On-chip traffic Modeling and Synthesis for MPEG-2 Video Applications*," IEEE Trans. Very Large Scale Integration Systems*, vol. 12, no. 1, pp. 108–119, 2004.

Abbas Eslami Kiasari (S'11) received his B.Sc. degree in electrical engineering from the Ferdowsi University, Mashhad, Iran, in 2003, and his M.Sc. degree in computer engineering from the Sharif University of Technology, Tehran, Iran, in 2005. He is currently pursuing the Ph.D. degree in electronic systems under supervision of Prof. Axel Jantsch at the Royal Institute of Technology (KTH), Stockholm, Sweden. His research interests include design methodologies and performance analysis of network-based systems. He is a member of the IEEE Computer Society.

Zhonghai Lu (M'05) received the BSc. degree in Radio & Electronics from Beijing Normal University, Beijing, China, in 1989, the MSc. degree in System-on-Chip Design, and the Ph.D. degree in Electronic and Computer Systems Design from the Royal Institute of Technology (KTH), Stockholm, Sweden, in 2002 and 2007, respectively.

From year 1989 to 2000, he was an engineer in the area of electronic and embedded systems. He is currently an Associate Professor with the Department of Electronic Systems, School for Information and Communication Technology, KTH. His research interests include network-on-chip/system-on-chip, many-core computing architectures, cyber-physical systems, performance analysis, and design automation. He has published about 100 papers in those areas.

Axel Jantsch (M'97) received the Dipl.Ing. and Dr. Tech. degrees from the Technical University of Vienna, Vienna, Austria, in 1988 and 1992, respectively.

He was with Siemens Austria, Vienna, Austria, as a System Validation Engineer from 1995 to 1997. Since 1997, he has been an Associate Professor with the Royal Institute of Technology (KTH), Kista, Stockholm, Sweden. Since 2000, has been a Docent, and since December 2002, a Full Professor of Electronic System Design with the Department of Electronic Systems. He has published over 200 papers in international conferences and journals, and one book in the areas of very large scale integration design and synthesis, system level specification, modeling and validation, HW/SW codesign and cosynthesis, reconfigurable computing, and networks on chip.

Dr. Jantsch received the Alfred Schrödinger Scholarship from the Austrian Science Foundation while a Guest Researcher with KTH between 1993 and 1995. He has served on a large number of technical program committees of international conferences, such as FDL, DATE, CODES+ISSS, SOC, NOCS, and others. He has been the TPC Chair of SSDL/FDL 2000, the TPC Co-Chair of CODES+ISSS 2004, the General Chair of CODES+ISSS 2005, and the TPC Co-Chair of NOCS 2009. From 2002 to 2007, he was a Subject Area Editor for the *Journal of System Architecture*. At KTH, he is heading a number of research projects involving a total number of ten Ph.D. Students, in two main areas: system modeling and networks-on-chip.

Mathematical Formalisms for Performance Evaluation of Networks-on-Chip

Abbas Eslami Kiasari
Axel Jantsch
Zhonghai Lu

ACM Computing Surveys, vol. 45, no. 3, Article no. 38, Jun. 2013

Paper 10

Mathematical Formalisms for Performance Evaluation of Networks-on-Chip

Abbas Eslami Kiasari, KTH Royal Institute of Technology, Sweden
Axel Jantsch, KTH Royal Institute of Technology, Sweden
Zhonghai Lu, KTH Royal Institute of Technology, Sweden

This paper reviews four popular mathematical formalisms – *queueing theory, network calculus, schedulability analysis*, and *dataflow analysis* – and how they have been applied to the analysis of on-chip communication performance in Systems-on-Chip. The paper discusses the basic concepts and results of each formalism and provides examples of how they have been used in Networks-on-Chip (NoCs) performance analysis. Also, the respective strengths and weaknesses of each technique and its suitability for a specific purpose are investigated. An open research issue is a unified analytical model for a comprehensive performance evaluation of NoCs. To this end, the paper reviews the attempts that have been made to bridge these formalisms.

Categories and Subject Descriptors: **C.4 [PERFORMANCE OF SYSTEMS]**: Modeling techniques

General Terms: Design, Performance

Additional Key Words and Phrases: System-on-Chip (SoC), Network-on-Chip (NoC), performance evaluation, analytical modeling

1. INTRODUCTION

It is essential to gain a solid understanding of a system's performance as far as possible in advance of the system being implemented in detail and built. Therefore, performance models have been deployed in system design for many decades and, more recently, have been adopted for the study of System-on-Chip (SoC). In modern SoCs, the on-chip communication infrastructure or network-on-chip (NoC) is a dominant factor for design, validation, and performance analysis. SoC designers are interested in performance evaluation, given that their goal is either to provide the highest performance at a given cost or to provide a minimum level of performance at the lowest possible cost. In both cases, a reliable measure of performance is indispensible. However, the focus in the first case is typically on average performance, while the worst-case performance is the main metric in the latter case. In real-time systems such as automotive or avionic applications, the worst-case execution time is of particular concern; it is important to know how much time might be needed in the worst case in order to guarantee that the task will always finish its jobs before the predetermined deadline. However, the worst-case-based design results in resource over-dimensioning. Therefore, average-case-based design methods are usually used for non-time critical applications in order to achieve a more efficient system.

Performance estimation tools can be classified in simulation model and mathematical model. SoC designers have explored the design space using detailed simulations in order to tackle performance analysis. Although simulation tools are flexible and accurate, the complexity of modern SoCs imposes a firm limit on what can reasonably be simulated. Another disadvantage of a simulation-based design process is that the non-linear and non-monotonic behavior of system performance makes it difficult to draw conclusions from the simulation results regarding how to

adapt the system hardware or its programming. It is also difficult to determine the worst-case behavior of the SoC. An alternative approach is to build an abstract, analytical model for the architecture of the on-chip communication. An appropriate analytical model can estimate the desired performance metrics very early in the design phase, in a fraction of the time that simulation would take. Although the use of high-level models conceals a lot of complex technological aspects, it facilitates rapid exploration of the NoC's design space. Also, the analytical models provide not only the timing properties of the system, but also useful feedback about the system's behavior. Consequently, such models can be invoked in any optimization loop for NoCs in order to obtain fast and accurate performance estimations. Therefore, analytical models have a place alongside simulation in SoC performance analysis, and their importance is likely to grow as SoC communication architectures become increasingly complex and irregular. Several popular analysis methods, developed in other context years or even decades ago, have recently been adapted to NoC analysis.

The purpose of this survey is to recapitulate the results from the mathematical formalisms – *queueing theory, network calculus, schedulability analysis*, and *dataflow analysis* – and their application to the analysis of NoCs. For each of them, we review the basic concepts and results in Section 2 to 5. Section 6 considers a simple application and show how these formalisms can be used to evaluate the performance of a system. Section 7 presents attempts to combine these methods and finally their respective strengths, weaknesses, and suitability for a specific purpose are summarized in Section 8.

2. QUEUEING THEORY

2.1. Overview

Queueing theory is a branch of probability theory. As shown in Figure 1, in a queueing system a population of *customers* at some time enters a *service facility*, which includes one server or multiple servers, in order to obtain service. If a new customer arrives and all servers are busy, it enters a *queue* and waits until one server becomes available. Therefore, In order to analyze such a system, we must identify the arrival process as well as the structure and discipline of the service facility.

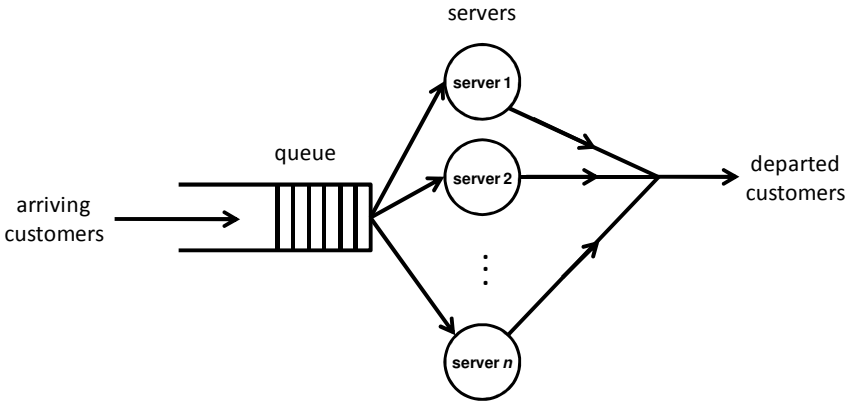


Figure 1: Model of a queueing system.

In queueing theory, the arrival process and service time are specified *probabilistically*. Generally, the arrival process is described in terms of the *cumulative distribution function* (*CDF*) of the *interarrival times* of customers (the time between two successive arrivals) and is denoted $A(t)$ where

$$A(t) = P\{\text{interarrival time} \leq t\} \quad (1)$$

The notation $P\{X\}$ denotes the probability of the event X. $A(t)$ is a non-negative and non-decreasing function of t. Also, the *probability density function* (*pdf*) of interarrival times is

$$a(t) = \frac{d}{dt}A(t) \tag{2}$$

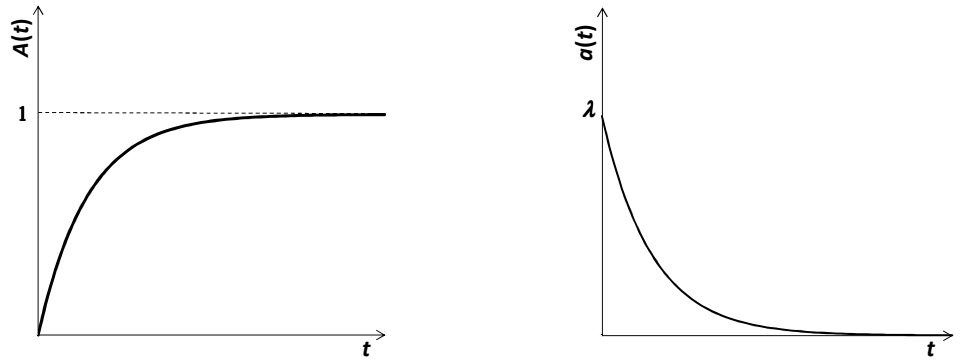
For instance, if the interarrival time of customers has exponential distribution with parameter $\lambda\ (\lambda > 0)$, then its CDF is given by

$$A(t) = \begin{cases} 1 - e^{-\lambda t} & t \geq 0 \\ 0 & t < 0 \end{cases} \tag{3}$$

which is sketched in Figure 2.a. The corresponding pdf of the interarrival time is

$$a(t) = \begin{cases} \lambda e^{-\lambda t} & t \geq 0 \\ 0 & t < 0 \end{cases} \tag{4}$$

which is sketched in Figure 2.b.



(a) (b)

Figure 2: (a) CDF and (b) pdf of an interarrival time with exponential distribution.

The assumption in queueing systems is that these interarrival times are independent and identically distributed random variables. Similarly, *service time,* the length of time that a customer spends in the service center, is considered as another continuous random variable whose CDF and pdf respectively are

$$B(x) = P\{\text{service time} \leq x\} \tag{5}$$

$$b(x) = \frac{d}{dx}B(x) \tag{6}$$

Regarding the structure and discipline of the service facility, a variety of additional quantities must be specified such as the extent of storage capacity available to hold waiting customers, the number of service stations available, the queueing discipline (FCFS, LCFS, and random order of service), etc.

In addition to interarrival and service times distributions, queueing systems may differ in the number of servers, the capacity of queue (infinite or finite), and the service discipline. Some common service disciplines are:

- *FCFS* (First-Come, First-Served): A customer that finds the service center busy goes to the end of the queue.

- *LCFS* (Last-Come, First-Served): A customer that finds the service center busy proceeds immediately to the head of the queue. It will be served next, given that no further customers arrive.
- *RS* (Random Service): The customers in the queue are served in random order.
- *RR* (Round Robin): Every customer gets a time slice. If its service is not completed, it will re-enter the queue.
- *PR* (Priority): Every customer has a (static or dynamic) priority, the server selects always the customers with the highest priority. This scheme can use preemption or not.

The *Kendall notation* is used for a short characterization of queueing systems [Bolch et al. 2006]. A queueing system description looks as $A/B/m/K - S$ where A denotes the distribution of the customer interarrival time, B denotes the distribution of the service time, m denotes the number of servers, K denotes the maximum capacity of queue in the finite case (if $K = \infty$, then this letter is omitted) and the optional S denotes the service discipline used. If S is omitted, the service discipline is always FCFS. For A the following abbreviations are very common:

- *M* (Markov property): this denotes the exponential distribution with average arrival rate of λ customers/time unit. In other words, the number of customers follows a Poisson distribution with the average of 1 customer per $1/\lambda$ time unit.
- *D* (Deterministic): The interarrival times are constant and have the same value.
- *G* (General): General distribution, not further specified. In most cases at least the mean and the variance are known.

Similarly, B can be specified by these notations (*M, D, and G*) to describe the distribution of service time. For instance, the $M/G/2/10$-*RS* queueing system can be described as follows:

- The customer interarrival times are exponentially distributed (with specified average).
- The service time distribution is arbitrary (with specified average and variance).
- There are two servers in the system.
- The queue has room for at most 10 customers.
- The customers in the queue are served in random order.

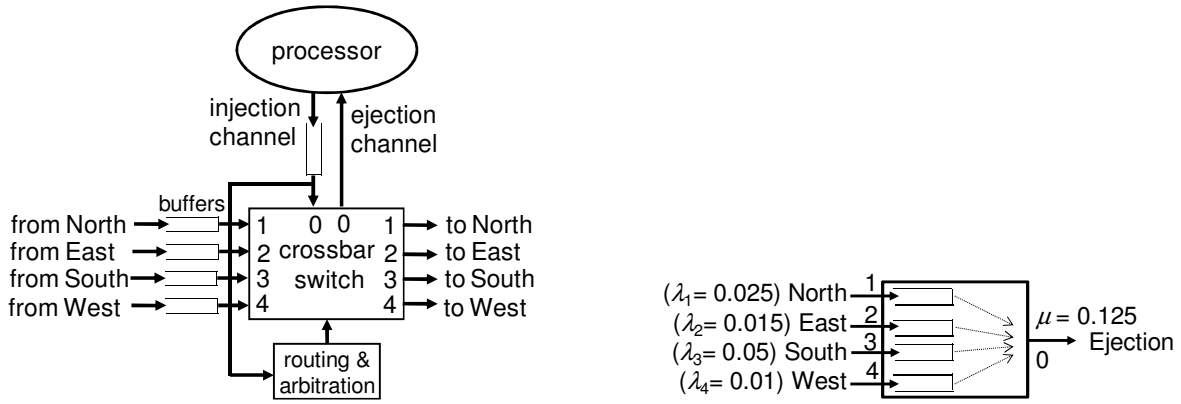
After specifying a queueing system, it is appropriate that we identify the measures of performance and effectiveness that we shall obtain by analysis. Basically, we are interested in the waiting time for a customer, the number of customers in the queue, the length of busy and idle periods of the server (the continuous interval during which the server is busy or idle), and the current work backlog (unfinished work) expressed in units of time. All these quantities are random variables and thus we seek their complete probabilistic description such as their pdf. However, in most applications it is enough to calculate the first few moments (mean, variance, etc.). Also within the scope of queueing theory is the case where several servers are arranged in a network and customers move through the network to visit several servers.

2.2. An Example

As an example, consider a packet-switched mesh network that packet routing is carried out by a router at each node. Every node contains a processor and a router. Packets are injected into the network on crossbar input port 0 (injection channel) and leave on output port 0 (ejection channel) as shown in Figure 3.a. In the following, we utilize the queueing theory to estimate the average waiting time to access the ejection channel. The following assumptions are made when developing the queueing model.

- The packet arrivals to the northern, eastern, southern, and western input channels are independent and follow Poisson processes with mean rate of $\lambda_1 = 0.025, \lambda_2 = 0.015, \lambda_3 = 0.05$, and $\lambda_4 = 0.01$ packets/cycle, respectively.

- An infinite FIFO buffer is associated only with each input channels for storing packets in transit.
- Messages are broken into some packets of fixed length. When a packet arrives on an ejection channel, it is accepted by the processor in 8 network cycles. Therefore, we can model the ejection channel as a constant-rate server with service rate of $\mu = 1/8 = 0.125$ packets/cycle.
- We consider an ejection channel as a server in which the packets have preferential treatment based on priorities associated with them. We assume that the priority of a packet is an integer fixed at arrival time, and packets with priority index 1, 2, 3, and 4 come from northern, eastern, southern, and western input channel, respectively. We say one packet has higher priority than another if it belongs to a priority class with lower index. For the service discipline, we assume that whenever a packet is traversed ejection channel completely, the ejection channel is next assigned to that packet at the head of the highest priority nonempty queue.



(a) (b)

Figure 3: (a) The structure of a router in 2D mesh network, (b) Queueing model of the ejection channel.

To calculate the average waiting time for ejection channels, we model the ejection channel as an $M/D/1$ priority queue as shown in Figure 3.b. The average waiting time of random arrivals to the ith queue of $M/D/1$ system, $\bar{W}_i$, can be written as [Bolch et al. 2006]

$$\bar{W}_k = \frac{\frac{1}{2\mu}\sum_{i=1}^{4}\rho_i}{(1-\sum_{i=1}^{k-1}\rho_i)(1-\sum_{i=1}^{k}\rho_i)} \tag{7}$$

where $\rho_i = \lambda_i/\mu$. In this example, $\rho_1 = 0.2, \rho_2 = 0.12, \rho_3 = 0.4$ and $\rho_4 = 0.08$. Thus, the waiting times can be computed as

$\bar{W}_1 = \frac{3.2}{1-0.2} = 4.0$ cycles
$\bar{W}_2 = \frac{3.2}{(1-0.2)(1-0.32)} = 5.9$ cycles
$\bar{W}_3 = \frac{3.2}{(1-0.32)(1-0.72)} = 16.8$ cycles
$\bar{W}_4 = \frac{3.2}{(1-0.72)(1-0.8)} = 57.1$ cycles

Using Little's theorem [Kleinrock 1975], the average number of packets in each input port can be computed as follows:

$\bar{N}_1 = \lambda_1 \bar{W}_1 = 0.10$ packet
$\bar{N}_2 = \lambda_2 \bar{W}_2 = 0.09$ packet
$\bar{N}_3 = \lambda_3 \bar{W}_3 = 0.84$ packet
$\bar{N}_4 = \lambda_4 \bar{W}_4 = 0.57$ packet

2.3. Applications in NoCs

Similar to other networks, traffic patterns play an important role in the performance of NoCs; consequently, traffic models are critically needed for effectively evaluating existing and new NoC designs. As a result, network traffic modeling is a first step towards understanding of the design space of NoC architectures, protocols and implementations. Soteriou et el. [2006] proposed an on-chip traffic model for homogeneous NoCs. The model is based on three statistical parameters: temporal burstiness, spatial hop distribution, and spatial injection distribution. The authors showed that their model captures the characteristics of NoC traffic accurately when compared to actual NoC application traces gathered from full system simulations of chip platforms. Varatkar and Marculescu [2004] have found long-range dependent behavior in communications traffic between different parts of the MPEG-2 video decoding application. They presented an approach for analyzing such a traffic pattern based on self-similar processes. They showed that characterizing the degree of self-similarity via the Hurst parameter helps in finding the optimal buffer-length distribution. However, Scherrer et al. [2009] showed that long-range dependence is not an ubiquitous property of the traffic produced by on-chip processors running multimedia applications. Using cycle-accurate simulator of a complete SoC, they showed that long-range dependence impact on the network-on-chip is highly correlated with the low level communication protocol used.

Performance evaluation techniques for NoCs have been inherited from parallel and distributed processing research groups. Many of the previous analytical latency models in off-chip networks have been formulated for a specific topology and traffic pattern [Kim and Das 1994; Kiasari et al. 2008a]. Queueing theory has been used to estimate average performance metrics such as average packet latency, average throughput, average energy and power consumption, and average resource utilization. System designers utilize these metrics to make decisions for solving problems such as module placement [Kiasari et al. 2008d], routing decision [Kiasari et al. 2010], buffer configuration [Hu et al. 2006], and link capacity [Guz et al. 2007].

Guan et al. [1993] proposed an analytical model for a general topology with an exponential packet length distribution. Their approach has high complexity for high dimensional networks. Using queueing theory, Hu and Kleinrock [1997] presented a general analytical model for wormhole routing to estimate the average packet latency in interconnection networks. In order to provide fast performance estimates during the design cycle, Kim et al. [2005] developed a queueing theory-based model for quantifying the performance and energy behavior of on-chip networks. They assumed that packet arrivals at all input channels have Markov property. Hu et al. [2006] considered $M/M/1/K$ queueing models and solved a series of nonlinear equations to quickly analyze the current buffer size configuration and detect the performance bottlenecks in the router channels. This model was then used in buffer sizing problem in packet-switched NoCs. More precisely, given the traffic characteristics of the target application and the total budget of the available buffering space, the proposed model automatically assigns the buffer depth for each input channel, in different routers across the chip, such that the average packet latency is minimized in the system. Based on $M/M/1$ queueing model, an analytical delay model for virtual channeled wormhole networks was proposed for link capacity allocation in NoC-based

systems by Guz et al. [2007]. This assignment algorithm allocates network resources efficiently so that quality of service (QoS) and performance requirements are met.

Hur et al. [2008] presented a performance analysis of hard and soft on-chip networks for FPGAs. They applied the Jackson's queueing model [Jackson 1957] to analyze the performance of a multiprocessor SoC. They further used the Jackson's model to analyze circuit-switched NoCs and show that the hardwired networks perform significantly better than conventional soft NoCs. A Markovian performance model for torus on-chip networks with deterministic routing and wormhole switching was proposed by Kiasari et al. [2008b]. The model is then used to estimate the power consumption of all routers. This model is restricted to Poisson arrival process and uniform traffic pattern. Kiasari et al. [2008c] modeled each channel in an NoC with $G/G/1$-PR queueing model ($G/G/1$ queue with priority discipline) and estimated per-flow average packet latency. However, the modeling approach is limited to k-ary n-cube networks with single flit buffers and dimension-order routing algorithm. This model was used to map the processing cores onto an SoC architecture such that the average communication delay is minimized [Kiasari et al. 2008d]. A case-study of using an analytical method, based on Markov chain stochastic processes, for latency evaluation of an NoC arranged in a 2D mesh topology, with deterministic routing algorithm and uniform traffic pattern was presented by Foroutan et al. [2009].

Foroutan et al. [2010] proposed a generic analytical model to estimate communication latencies and link-buffer utilizations for wormhole-switched NoCs with a given application mapped on it. This work correctly models the resulting interdependencies between the routers. An analytical performance model for wormhole-switched NoCs has been proposed by Ogras et al. [2010]. Using $M/G/1$ queueing model, the average number of packets at each buffer is computed. This model provides three performance metrics, namely average buffer utilization, average packet latency, and network throughput. Another analytical model to estimate the communication performance of wormhole-switched NoCs is presented by Cheng et al. [2011]. This model supports arbitrary network topology with virtual channels. To resolve the inherent dependency of successive links occupied by a packet, the authors use routing path decomposition approach to generating a series of ordered link categories. Next, they used $M/M/1$ and $M/M/1/K$ queueing models to derive the transmission latency of network components. The analytical model proposed by Krimer et al. [2011] is inspired by industrial work-flow modeling techniques. The authors introduced a packet-level static timing analysis for wormhole-switched NoCs with virtual channels. It relies on a reduced Markov chain to represent the network state, including the occupancy of all buffers. The model handles any topology, link capacities, and buffer sizes and provides per-flow delay analysis. Wang et al. [2011] proposed a performance analytical model using semi-Markov process to estimate the average packet latency in NoCs. More precisely, semi-Markov process is used to describe the behavior of each link in the network and the header flit delay is calculated.

Kiasari et al. [2012] extended the proposed queueing model by Kiasari et al. [2008c] to support arbitrary topology, buffer size, and oblivious routing algorithm. This model is developed for wormhole switching and it supports any kind of spatial and temporal traffic patterns. Spatial traffic pattern refers to the distribution of packet destinations and temporal component of traffic is determined by the distribution of interarrival time of packets. It means that the model accepts both Poisson and non-Poisson arrival processes. The average packet latency (L) is used as the performance metric. The authors assume that the packet latency spans the instant when the packet is created, to the time when the packet is delivered to the destination node. They also assume that the packets are consumed immediately once they reach their destination nodes. In Figure 4, consider a flow which is generated in IP^S, and reaches its destination (IP^D) after traversing routers R^S, R^M, and R^D. The latency of this packet $(L^{S\rightarrow D})$ consists of two parts: the latency of head flit $(L_h^{S\rightarrow D})$ and the latency of body flits (L_b). $L_h^{S\rightarrow D}$ is the time since the packet is created in IP^S, until the head flit reaches the IP^D, including the transfer times of a flit across the injection and ejection channels (t_{inj} and t_{ej}), routing decision delay for a packet (t_r), crossing time of

a flit over the crossbar switch (t_s), and transfer time of a flit across a wire between two adjacent routers (t_w), and queueing time spent at the source node and intermediate nodes ($W_{i\to j}^{N}$, the mean waiting time for a packet from input port *i* of node *N* to output port *j* of the same node). Having looked at Figure 4, we can infer that the latency of a head flit of a two-hops packet includes injection channel delay (t_{inj}), first router delay $\left(t_r + W_{inj\to East}^{S} + t_s\right)$, inter-node wire delay (t_w), second router delay $\left(t_r + W_{West\to East}^{M} + t_s\right)$, inter-node wire delay (t_w), third router delay $\left(t_r + W_{West\to ej}^{D} + t_s\right)$, and ejection channel delay (t_{ej}). Therefore, it can be written as

$$\begin{aligned} L_h^{S\to D} &= t_{inj} + \left(t_r + W_{inj\to East}^{S} + t_s\right) \\ &+ t_w + \left(t_r + W_{West\to East}^{M} + t_s\right) \\ &+ t_w + \left(t_r + W_{West\to ej}^{D} + t_s\right) + t_{ej} \end{aligned} \tag{8}$$

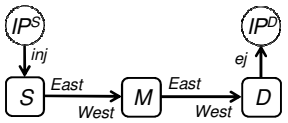


Figure 4: A two-hops flow from IP^S (source) to IP^D (destination).

Once the head flit arrives at the destination, the body flits follow the header flit in a pipelined fashion. Therefore, the only unknown parameter for computing the latency is $W_{i\to j}^{N}$. This value was calculated using a priority queueing model. Later, in Section 6.1, we show how to use this analytical model to estimate the average packet latency.

Studies done by Varatkar and Marculescu [2004] have demonstrated that the traffic of some multimedia applications exhibits a long-range dependent behavior which has a considerable impact on queueing performance. The assumption of the traditional Poisson arrival process is inherently unable to capture such a traffic pattern. Therefore, it is crucial to re-examine the performance properties of interconnection networks in the context of more realistic traffic models before practical implementations show their potential faults. Toward this end, Min and Ould-Khaoua [2004] proposed an analytical queueing model for wormhole-switched networks in the presence of self-similar traffic. This study reveals that the network suffers considerable performance degradation when subjected to self-similar traffic, stressing the great need for improving network performance to ensure efficient support for this type of traffic.

In queueing theory, generally, the average quantities in an equilibrium state are considered. Characterizing the transient behavior of queueing systems is known as a very difficult problem which has been addressed by either simplified analytical models or simulation [Odoni and Roth 1983; van As 1986; Bertsimas and Mourtzinou 1997; Yang and Liu 2010]. As an alternative approach, Bogdan and Marculescu [2007] proposed a statistical physics inspired framework to analyze the traffic dynamics in NoCs and show how the non-stationary effects of the system workload can be effectively captured. The temperature of a physical system is replaced by the NoC packet injection rate and the authors predicted that the buffer occupancy follows a power law distribution. Later, they addressed the buffer sizing problem under non-equilibrium conditions [Bogdan and Marculescu 2009]. The main idea in this model is that packets move from one node to another in a manner that is similar to particles moving in a Bose gas and migrating between various energy levels as a consequence of temperature variations. Bogdan and Marculescu [2010; 2011] also investigated the impact of non-stationary effects (as a function of packet injection rate) on buffer overflow probability and node-to-node latency exceedance probability.

In order to analyze a queueing system, it is necessary to know something about the laws governing the arrival pattern, the logic governing the behavior of the queue, and the

characteristics of the service facility. Queueing theory is concerned with the mathematical analysis of such systems subject to demands whose occurrences and lengths can, in general, be specified only probabilistically.

3. NETWORK CALCULUS

3.1. Overview

Network calculus is a mathematical framework to derive the worst-case bounds on maximum latency and backlog in a single node and a network of nodes. Therefore, it can be seen as a theory for analyzing performance guarantees in computer networks. Cruz [1991a; 1991b] has pioneered the network calculus and based on Cruz's foundation, Chang [2000] and Le Boudec and Thiran [2001] have further developed the network calculus theory and based it on *min-plus algebra*. The basic elements in this algebra are arrival curves as an abstraction of application traffic and service curves as an abstraction of network elements. Network calculus is similar to conventional system theory, in which a system consists of an input function, a transfer function and an output function. The difference to conventional system theory is that the min-plus algebra is used, where addition and multiplication are replaced by minimum and addition, respectively. As in conventional system theory, a key operation in network calculus is the *min-plus convolution*. The min-plus convolution of $f(t)$ and $g(t)$ is defined as

$$(f \otimes g)(t) = \inf_{0 \le s \le t} \{f(s) + g(t-s)\}. \tag{9}$$

The infimum (*inf*) is similar to the minimum; a minimum of a set is the smallest element of the set, and of course, is in the set. An infimum of a set is the *greatest lower bound* of the set, and need not to be in the set. The same applies for the maximum and supremum (*sup*).

In network calculus theory, cumulative functions $R(t)$ and $R^*(t)$ describe the *input function* and *output function*, respectively. They represent the number of bits (words, or packets) seen on the input and output data flow in time interval $[0, t]$. It is obvious that functions R and R^* are always monotonically increasing functions. System S receives input data and delivers the output data after a variable delay. System S might be, for example, a single buffer served at a constant rate, a complex communication node, or even a complete network.

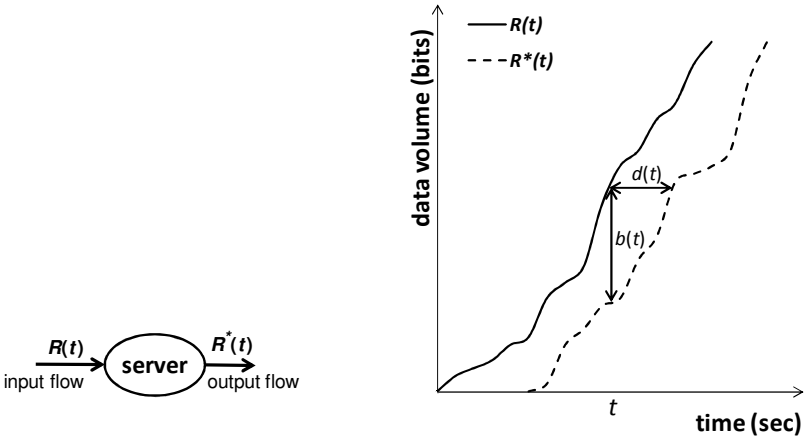
The *backlog* is the number of bits that are held inside the system; if the system is a single buffer, it determines the queue length. In contrast, if the system is more complex, then the backlog is the number of bits "in transit", assuming that we can observe input and output simultaneously. Therefore, for a lossless system the *backlog* at time t is

$$b(t) = R(t) - R^*(t). \tag{10}$$

The virtual delay at time t is the delay that would be experienced by a bit arriving at time t if all bits received before it are served before it. Hence, the virtual delay at time t is

$$d(t) = \inf_{\tau \ge 0}\{ R(t) \le R^*(t+\tau)\}. \tag{11}$$

In other words, $d(t)$ is the smallest value satisfying $R^*(t + d(t)) = R(t)$. As shown in Figure 5, the backlog and virtual delay are shown as the vertical and horizontal deviation between input and output functions, respectively. In network calculus theory, the input and transfer functions are referred to as *arrival curve* and *service curve*, respectively.



(a) (b)

Figure 5: Backlog and virtual delay of a flow at time *t*.

3.1.1. Traffic Model

Assume that we want to provide guarantees to traffic flows. This requires some specific support in the network to limit the traffic sent by sources. This is done by using the concept of *arrival curve*, illustrated in Figure 6.a. Given an increasing function $a(t)$ defined for $t \geq 0$, we say that an input flow $R(t)$ is constrained by $a(t)$ if and only if for all $s \leq t$:

$$R(t) - R(s) \leq a(t - s). \tag{12}$$

The min-plus representation of this equation is

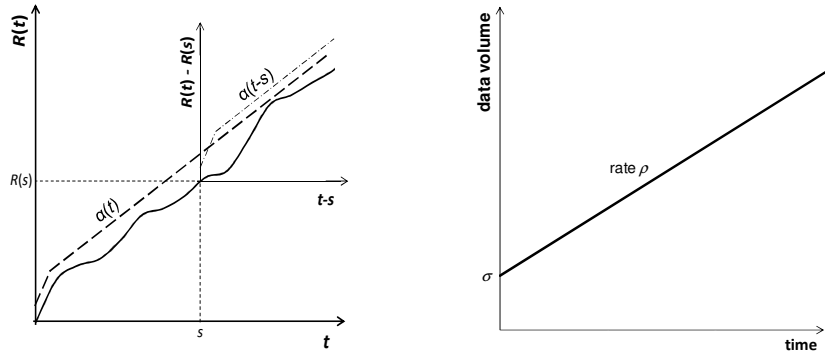
$$R \leq R \otimes a. \tag{13}$$

We say that R has a as an arrival curve, or also that R is a-smooth.

A common arrival curve is *leaky bucket* arrival curve (or affine arrival curve) defined by

$$\alpha(t) = \gamma_{\rho,\sigma} = \rho t + \sigma, \qquad t \geq 0. \tag{14}$$

where ρ is the rate of the flow (in units of data per time unit) and σ limits the burstiness of the flow (in units of data). Having such an arrival curve allows a source to send σ bits at once, but not more than ρ bits/second over the long run. The (σ, ρ) traffic characterization was initially proposed by Cruz [1991a] and the corresponding arrival curve is shown in Figure 6.b.



(a) (b)

Figure 6: (a) Input function $R(t)$ is constrained by an arrival curve $a(t)$, (b) Leaky bucket (affine) arrival curve.

3.1.2. Network Elements Model

Service curve describes minimal service levels of network elements (router, channel, etc). It often abstracts a scheduling policy. Consider a system S and a flow through S with input and output functions R and R^*. We say that S offers to the flow a service curve β if and only if

- β is an increasing function.
- $\beta(0) = 0$. (15)
- $R^* \geq R \otimes \beta$.

In other words, for all t, there exists some $s \leq t$ such that

$$R^*(t) \geq R(s) + \beta(t-s). \tag{16}$$

Figure 7.a shows a graphical representation of this condition. A well-defined service curve is latency-rate function $\beta_{R,T}$,

$$\beta_{R,T}(t) = R(t-T)^+ = \begin{cases} R(t-T), & t > T, \\ 0, & \text{otherwise.} \end{cases} \tag{17}$$

where R is the service rate and T the maximum response delay of the node [Stiliadis and Varma 1998]. Figure 7.b shows such a service curve which is widely used to model the routers in a network.

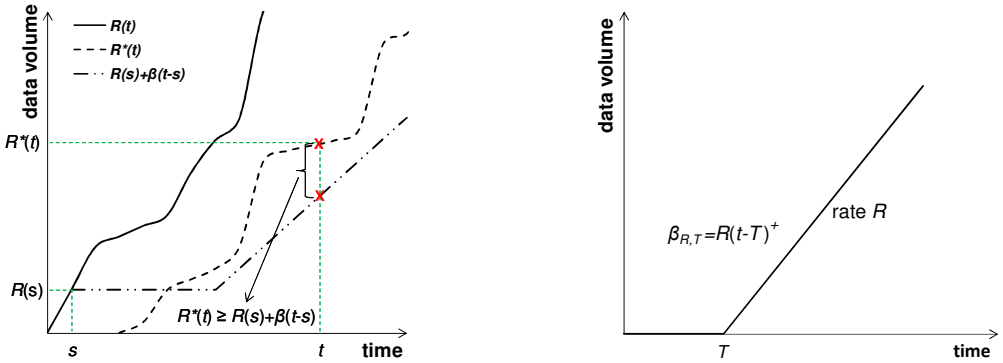


Figure 7: (a) Definition of service curve, (b) A latency-rate service curve.

3.1.3. Basic Bounds

Assume a flow, constrained by an arrival curve α, traverses a system that offers a service curve β.

- The backlog for all t satisfies

$$b(t) = R(t) - R^*(t) \leq \sup_{s \geq 0} \{\alpha(s) - \beta(s)\} \tag{18}$$

- The virtual delay for all t satisfies

$$d(t) \leq \sup_{s \geq 0} \left\{ \inf_{\tau \geq 0} \{\alpha(s) \leq \beta(s+\tau)\} \right\} \tag{19}$$

- The output flow is computed by the min-plus deconvolution operator $(\oslash)$ and constrained by the curve

$$\alpha^* = \alpha \oslash \beta = \sup_{u \geq 0} \{\alpha(t+u) - \beta(u)\} \tag{20}$$

For instance, in a system with leaky bucket arrival curve and latency-rate service curve shown in Figure 8, the maximum backlog, maximum delay, and output traffic characterization are [Le Boudec and Thiran 2001]

$$b_{max} = \sigma + \rho T, \tag{21}$$

$$d_{max} = T + \sigma/R, \tag{22}$$

$$\alpha^*(t) = \gamma_{\rho,\sigma+\rho T} = \rho t + \sigma + \rho T. \tag{23}$$

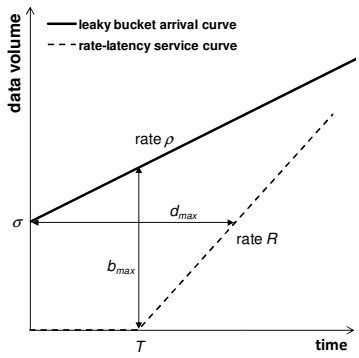


Figure 8: Maximum backlog and delay in a system with leaky bucket arrival curve and latency-rate service curve.

Using *superposition theorem*, *concatenation theorem*, and *leftover service theorem* [Jiang and Liu 2008], network calculus can be applied to a network of nodes.

Superposition: Consider the superposition of n flows R_i, $i = 1, \ldots, n$. If each flow R_i has an arrival curve a_i ,the aggregate flow R has an arrival curve $\alpha = \sum_{i=1}^{n} \alpha_i$.
Superposition property implies that the aggregate of individual flows can be represented by a single aggregate flow. For instance, the aggregate flow of two flows constrained by $\alpha_1 = \gamma_{\rho_1,\sigma_1}$ and $\alpha_2 = \gamma_{\rho_2,\sigma_2}$ is constrained by $\gamma_{\rho_1+\rho_2,\,\sigma_1+\sigma_2}$, because:

$$\alpha = \alpha_1 + \alpha_2 = \gamma_{\rho_1,\sigma_1} + \gamma_{\rho_2,\sigma_2} = (\rho_1 t + \sigma_1) + (\rho_2 t + \sigma_2) = \gamma_{\rho_1+\rho_2,\sigma_1+\sigma_2}$$

Concatenation: Similar to traditional system theory, the concatenation of a series of servers in tandem with service curves β_i ($i = 1, \ldots, n$) offers a service curve of $\beta = \bigotimes_{i=1}^{n} \beta_i = \beta_1 \otimes \beta_2 \otimes \ldots \otimes \beta_n$. As an example, consider two nodes offering each a latency-rate service curve β_{R_1,T_1} and β_{R_2,T_2}. A simple computation gives $\beta = \beta_{R_1,T_1} \otimes \beta_{R_2,T_2} = \beta_{\min(R_1,R_2),\,T_1+T_2}$

Leftover service: Consider a system offers a service curve β to the aggregate of flows R_1 and R_2. If R_1 has an arrival curve a_1, then $(\beta - \alpha_1)^+$ can be a service curve for flow R_2. For instance, assume a latency-rate server serves an aggregate of two flows as shown in Figure 9.a. If flow 1 is (σ_1, ρ_1) regulated flow, then the offered service to flow 2 is:

$$\beta_2 = (\beta - \alpha_1)^+ = (\, R(t-T)^+ - (\rho_1 t + \sigma_1)\,)^+ = (R - \rho_1)\left(t - \left(T + \frac{\rho_1 T + \sigma_1}{R - \rho_1}\right)\right)^+ =$$

$$\beta_{R-\rho_1,\,T+\frac{\rho_1 T+\sigma_1}{R-\rho_1}}$$

It means that the second flow is guaranteed a latency-rate service curve with parameters $R_2 = R - \rho_1$, and $T_2 = T + \frac{\rho_1 T + \sigma_1}{R - \rho_1}$ as shown in Figure 9.b.

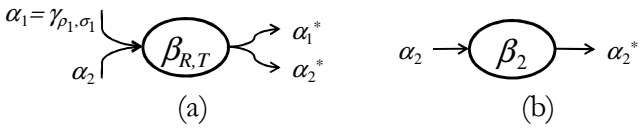


Figure 9: (a) Server offers a service curve β to the aggregate of two flows, (b) The second flow receives leftover service curve $\beta_2 = \beta_{R-\rho_1, T+\frac{\rho_1 T+\sigma_1}{R-\rho_1}}$.

Network calculus has been extremely successful when applied to ATM and IP networks with both differentiated and integrated services to achieve predictable performance [Le Boudec and Thiran 2001]. Recently, it has also been applied to wireless LAN [Kim and Hou 2009], sensor networks [Schmitt and Roedig 2005; She et al. 2009], and on-chip networks [Jafari et al. 2010]. Network calculus has been extended to a few directions. In the following, we briefly describe the real-time calculus and stochastic network calculus.

3.1.4. Real-Time Calculus

Chakraborty et al. [2003] proposed real-time calculus to model and analyze heterogeneous systems in a compositional manner. It is a framework based on network calculus and relies on the modeling of timing properties of event streams and available resources with curves called arrival curves and service curves. In real-time calculus, an arrival curve is function of relative time that constrains the number of events that can occur in an interval of time. For any sliding window of time of length Δ, the pair of arrival curves (α^l, α^u) gives the lower bound $\alpha^l(\Delta)$ and upper bound $\alpha^u(\Delta)$ on the number of events. Similarly, the processing capacity of a component is specified by a service curve (β^l, β^u). The number of events that may be processed in any time interval of size Δ is at least $\beta^l(\Delta)$ and at most $\beta^u(\Delta)$. In other words, arrival curve $\alpha = (\alpha^l, \alpha^u)$ and service curve $\beta = (\beta^l, \beta^u)$ are expressed in terms of numbers of events per time interval. *As an alternative representation*, Altisen et al. [2010] expressed arrival and service curves in terms of length of time interval. In this case, an arrival curve is represented by a pair of curves $\xi = (\xi^l, \xi^u)$. $\xi^l(k)$ and $\xi^u(k)$ respectively provide the lower and upper bounds on the length of the time interval during which any k consecutive events can arrive. Let t_i denote the arrival time of the i^{th} event; we have $\xi^l(k) \le t_{i+k} - t_i \le \xi^u(k)$ for all $i \ge 0$ and $k \ge 0$. Also, the processing capacity of a component is specified by a service curve $\psi = (\psi^l, \psi^u)$. The length of the time to process any k consecutive events for any potential stream is at least $\psi^l(k)$ and at most $\psi^u(k)$. Actually, ξ is a pseudo-inverse of α, satisfying $\xi^u(k) = min_{\Delta \ge 0}\{\Delta | \alpha^l(\Delta) \ge k\}$ and $\xi^l(k) = max_{\Delta \ge 0}\{\Delta | \alpha^u(\Delta) \le k\}$ (same for β and ψ). Also, the length of the time to process any k consecutive events for any potential stream is at least $\psi^l(k)$ and at most $\psi^u(k)$.

Based on the results from network calculus, the maximum delay experienced by an event and the maximum number of backlogged events from the stream that waiting to be processed can be given by the following inequalities:

$$delay \le \sup_{t \ge 0}\left\{\inf_{\tau \ge 0}\{\alpha^u(t) \le \beta^l(t+\tau)\}\right\} \quad (24)$$

$$backlog \le \sup_{t \ge 0}\{\alpha^u(t) - \beta^l(t)\} \quad (25)$$

Furthermore, real-time calculus gives exact bounds on the output stream of a component as a function of its input stream. This result can then be used as input for the next component. An event stream entering a processing or communication resource gets processed or transmitted, thereby generating an outgoing event stream which might enter another resource. As a result, the processing capability (such as the processor or bus bandwidth) of the resource, as specified by its upper and lower service curves gets modified.

Given an event stream which is specified by its arrival curves $\alpha = (\alpha^l, \alpha^u)$ and a resource which processes this event stream and its processing capability being specified by its service curves $\beta = (\beta^l, \beta^u)$. Let $\alpha' = (\alpha^{l'}, \alpha^{u'})$ denote the outgoing arrival curve of the (processed) event stream and $\beta' = (\beta^{l'}, \beta^{u'})$ denote the remaining service curves of the resource. By generalizing ideas from network calculus, these curves can be calculated as follows [Chakraborty et al. 2003]:

$$\alpha^{l'}(\Delta) = min\left\{\beta^l(\Delta), inf_{0\le\mu\le\Delta}\{sup_{\lambda\ge 0}\{\alpha^l(\mu+\lambda) - \beta^u(\lambda)\} + \beta^l(\Delta-\mu)\}\right\} \quad (26)$$

$$\alpha^{u'}(\Delta) = min\left\{\beta^u(\Delta), sup_{\lambda\ge 0}\left\{inf_{0\le\mu\le\lambda+\Delta}\{\alpha^u(\mu) + \beta^u(\lambda+\Delta-\mu)\} - \beta^l(\lambda)\right\}\right\} \quad (27)$$

$$\beta^{l'}(\Delta) = sup_{0\le\lambda\le\Delta}\{\beta^l(\lambda) - \alpha^u(\lambda)\} \quad (28)$$

$$\beta^{u'}(\Delta) = max\left\{0,\ inf_{\lambda\ge\Delta}\{\beta^u(\lambda) - \alpha^l(\lambda)\}\right\} \quad (29)$$

3.1.5. Stochastic network calculus

Stochastic network calculus [Jiang and Liu 2008] is the probabilistic version of the (deterministic) network calculus. Providing deterministic service guarantees often results in low resource utilization in the network. However, in some applications, such as multimedia applications, excess delay and loss for a small amount of data can be tolerated. For such applications, providing stochastic service guarantees can give better utilization of resources in the network without jeopardizing performance. Furthermore, in some networks, such as wireless networks, the service offered by a communication channel may vary randomly over time due to channel contention and impairment. Such networks can only provide stochastic services and guarantees [Jiang and Liu 2008]. Also, several stochastic versions of arrival curve have been proposed by extending the concept of arrival curve to the stochastic case based on the traffic amount property or virtual backlog property. In contrast to the deterministic arrival curves, stochastic arrival curves envelop traffic tighter, but have higher implementation complexity. An arrival process *R* is said to be constrained by a stochastic arrival curve *a*(*t*) with bounding function $f(x)$ if for all $0 \le s \le t$ and $x \ge 0$ there holds [Jiang and Liu 2008]

$$P\{sup_{0\le s\le t}\{R(t) - R(s) - a(t-s)\} > x\} \le f(x) \quad (30)$$

where notation $P\{Z\}$ means the occurrence probability of event *Z*. Also, a system *S* is said to provide a stochastic service curve β with bounding function $g(x)$, if for all $t \ge 0$ and $x \ge 0$ there holds [Jiang and Liu 2008]

$$P\{sup_{0\le s\le t}\{R\otimes\beta(s) - R^*(s)\} > x\} \le g(x) \quad (31)$$

Similar to network calculus, the probabilistic versions of backlog and delay bounds are computed based on stochastic arrival and service curves.

3.2. Applications in NoCs

Zhang [1995] surveyed several service disciplines proposed in the literature to provide per-connection end-to-end performance guarantees in packet-switching networks. Various issues and trade-offs in designing service disciplines for guaranteed performance service are discussed, and a general framework for studying and comparing these disciplines are presented. This work gives an excellent overview of guaranteed service in networks that can be applicable to NoCs.

Network calculus can be used to estimate the worst-case flow delays and backlogs in a given system. Qian et al. [2009c] investigated per-flow flit and packet worst-case delay bounds in on-chip wormhole networks. The authors first proposed analysis models for flow control, link and buffer sharing, and then based on these analysis models, they obtained an open-ended service analysis model capturing the combined effect of flow control, link and buffer sharing. With the service analysis model, they computed leftover service curves for individual flows, and then derived their flit and packet delay bounds.

Lu et al. [2009] defined a regulation spectrum for lossless flow regulation and used it to reduce delay and backlog bounds in SoC architectures. Based on the regulation spectrum, Jafari et al. [2010] then formulated optimization problems for minimizing total buffers and buffer variations under QoS constraints. The regulation analysis was performed for best-effort networks. Bakhouya et al. [2011] presented a methodology to analyze and evaluate on-chip interconnects in terms of performance and cost metrics such as latency, energy consumption and area requirements. The 2D mesh, spidergon, and WK-recursive topologies were compared using a given traffic pattern. The authors showed that WK-recursive outperforms the mesh and spidergon in all considered metrics. Lu [2011] used network calculus to analyze and determine the delay and buffer bounds for TDM virtual circuits crossing synchronous clock domains.

Qian et al. [2009a] applied network calculus to NoCs in order to analyze delay and backlog bounds for self-similar traffic. The authors first showed that self-similar traffic cannot be constrained by any deterministic arrival curve. Then they proved that self-similar traffic can be constrained by leaky bucket arrival curves if an additional parameter, excess probability, is used to capture its burstiness exceeding the arrival envelope. Qian et al. [2010a] derived the worst-case delay bound for an individual flow on packet-switched best-effort NoCs. To derive the leftover service curve for the flows, the authors first constructed a *contention tree* [Lu et al. 2005] for each flow, which captures its contention with other interfering flows along its routing path, and then scanned the tree. A tagged flow directly contends with interfering flows. Also, interfering flows may contend with each other and then contend with the tagged flow again. This indirect contention may, in turn, influence the performance of the tagged flow. To decompose a complex contention scenario, they identified three primitive contention patterns. Figure 10 shows a tagged flow, $f(1,N)$, traverses a tandem of N routers from source to destination, and is multiplexed with contention flows. The contention scenarios the tagged flow may experience can be classified into three patterns, *nested*, *parallel*, and *crossed*. The authors analyzed the three scenarios and derived their basic analytical models with focus on the derivation of the service curve the tandem provides.

For nested, parallel, and crossed contention flows, the service curve of tandem $(1,N)$ for $f(1,N)$ are calculated as in Eqs. (32), (33), and (34), respectively. Interested readers can find more details in [Qian et al. 2010a].

$$\beta_{(1,N)} = \left(\otimes_{i=1}^{g-1} \beta_i\right) \otimes \beta_{(1,N)}^{\{g \to k\},eq} \otimes \left(\otimes_{i=k+1}^{N} \beta_i\right) \tag{32}$$

$$\beta_{(1,N)} = \left(\otimes_{i=1}^{g-1} \beta_i\right) \otimes \beta_{(1,N)}^{\{g \to h\},eq} \otimes \left(\otimes_{i=h+1}^{j-1} \beta_i\right) \otimes \beta_{(1,N)}^{\{j \to k\},eq} \otimes \left(\otimes_{i=k+1}^{N} \beta_i\right) \tag{33}$$

$$\beta_{(1,N)} = \left(\otimes_{i=1}^{g-1} \beta_i\right) \otimes \beta_{(1,N)}^{\{g \to h-1\},eq} \otimes \beta_{(1,N)}^{\{h \to k\},eq} \otimes \left(\otimes_{i=k+1}^{N} \beta_i\right) \tag{34}$$

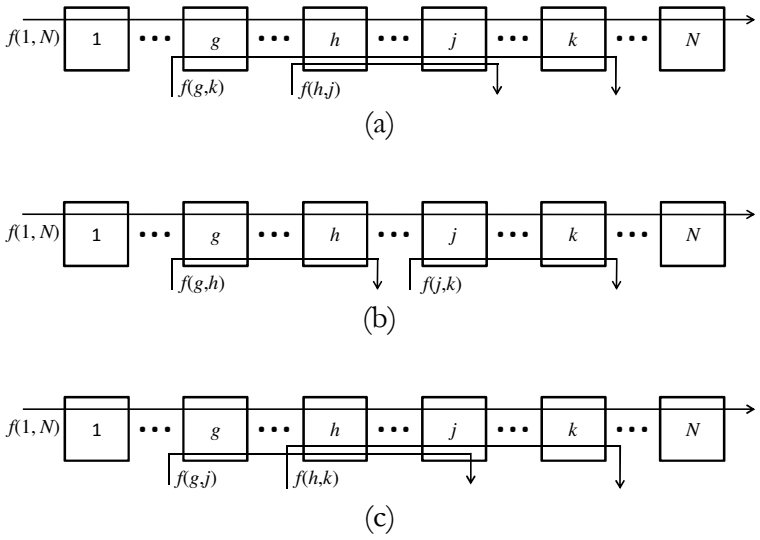


Figure 10: Three basic contention patterns for a tagged flow: (a) nested, (b) parallel, and (c) crossed [Qian et al. 2010a] © IEEE 2010.

After obtaining the tandem service curve, the authors derived closed-form formulas to calculate the delay bound and output arrival curve based on Eqs. (19) and (20), respectively. The authors showed that the simulated delays are totally constrained by the calculated delay bounds and the bounds are all tight. In this work, the authors have assumed big enough buffers in routers. Bounded buffers and virtual channels were considered in [Qian et al. 2009b] and [Qian et al. 2010b], respectively.

Based on real-time calculus, Hamann et al. [2004] presented a framework for design space exploration and system optimization for heterogeneous SoCs and distributed systems using SymTA/S, a software tool for formal performance analysis. SymTA/S takes the hierarchical structure of the design space of heterogeneous SoCs and distributed systems into account, allowing the designer to control the exploration process. The authors showed that optimization potential through traffic shaping in complex SoCs and distributed systems is very high. Therefore, the central aspect in their proposed framework is traffic shaping. Although traffic shaping is a promising approach for implementing real time systems, it is not suitable for non-real time and best-effort systems; since it makes the system non-work-conserving. SymTA/S allows designers to control the exploration process and provides them with insights on system-level performance dependencies. Based on this knowledge, designers can identify interesting design sub-spaces, worthy to be searched in-depth or even completely.

Soft real-time applications, such as a video decoder, may miss some deadlines without much of a detriment to their perceived performance. In these instances, Nelson et al. [2010] proposed a conservative simulation approach as an alternative to formal modeling for soft real-time applications. The authors introduced a hybrid simulation method which enables performance guarantees on a per-trace basis, without any modeling effort. Furthermore, they evaluate an implementation of the described technique and compare it with an actual MPSoC instance implemented on an FPGA.

Network calculus emerged as a new theory for the analysis of performance bounds in network-based systems. In contrast to queueing theory, network calculus deals with worst-case analysis instead of average-case analysis. Hence, it has been a promising formalism for quality of

service analysis. With network calculus, we are able to derive the worst-case bounds on maximum latency, backlog, and minimum throughput.

4. SCHEDULABILITY ANALYSIS

4.1. Overview

Schedulability analysis is a mathematical formalism to investigate the timing properties in real-time systems. It was originally proposed to analyze the computation systems [Liu and Layland 1973; Leung and Whitehead 1982; Lehoczky et al. 1989] and then applied to communication platforms such as multicomputers [Li and Mutka 1994] and NoCs [Shi and Burns 2008]. Usually in schedulability analysis, tasks are modeled with periodic and sporadic models. Periodic and sporadic tasks are released repeatedly. A periodic task is released at regular intervals and sporadic task is released at arbitrary times, but with a specified minimum time interval between releases. Given a set of periodic and sporadic tasks, their worst-case execution time, and a scheduling policy, schedulability analysis determines whether it is possible to schedule these tasks, such that deadline misses never occur. The earliest results in real-time scheduling and schedulability analysis have been obtained under restrictive assumptions about the task set and the underlying architecture: the task set is composed of a fixed number of independent tasks mapped on a single processor, the tasks are periodically released, each with a fixed period, the deadlines equal the periods, and the task execution times are fixed. Later works were done under more relaxed assumptions such as multi-processor systems, data dependency relationships among the tasks, deadlines less than or equal to the periods, and sporadic tasks.

Usually real-time systems are equipped with a *schedulability test* [Wu et al. 2010], which determines whether each of the admitted tasks can meet its deadline. A new task will not be admitted unless it passes the schedulability test. The schedulability test can be either *direct* or *indirect*. In a direct schedulability test the worst-case response time of the tasks is calculated and a task set is schedulable if and only if the worst-case response time of each task is less than or equal to its deadline. This type of test is accurate, but the computing cost in calculating the response times is very high. Audsley et al. [1993] proposed an iterative formula to compute the worst-case response time of a periodic task set. The complexity of this test is pseudo-polynomial, thus it may be unsuited for online admission control, especially in those real-time applications consisting of large task sets. Sjodin and Hansson [1998] proposed a few methods for reducing the number of iterations in computing the tasks response times. However, the worst-case complexity of their test is still pseudo-polynomial. Indirect schedulability tests do not compute the delays, but test another performance factor of the system to determine the task schedulability. The *utilization-based* test is the most common indirect schedulability test which tests system resource utilization to determine the task schedulability. A new task can be admitted only if the utilization is lower than a pre-derived bound. For utilization-based schedulability test, a task set is schedulable when the utilization of the task set is lower than a pre-derived bound.

In the seminal work of Liu and Layland [1973], the problem of multiprogram scheduling on a single processor is studied. They derived a utilization bound for rate monotonic (RM) scheduling policy in which a task with a shorter period is given a higher priority than a task with a longer period. They considered sets of periodic tasks on a uniprocessor system under the assumptions that all tasks start simultaneously at time $t = 0$, deadlines are equal to periods and tasks are independent. Under such assumptions, a set of *n* periodic tasks is schedulable by RM algorithm if

$$\sum_{i=1}^{n} \frac{C_i}{T_i} \leq n\left(2^{1/n} - 1\right) \tag{35}$$

where C_i is the worst-case execution time (WCET) of task *i*, T_i is the period of task *i,* and *n* is the number of tasks. The left-hand side of the inequality represents the maximum utilization of the

system and the right-hand side represents the utilization bound, a quantity which decreases monotonically from 0.83 when $n = 2$ to $\ln 2 \approx 0.69$ as $n \to +\infty$.

In the same paper, they analyzed the set of tasks in the case they are dynamically scheduled by a runtime scheduler according to a dynamic assignment of priorities to tasks. The assignment is made according to the earlier deadline first (EDF) algorithm (the closer the task deadline, the higher the task priority). Task preemption is allowed. Under these assumptions, they proved that a task set is schedulable if and only if

$$\sum_{i=1}^{n} \frac{C_i}{T_i} \leq 1 \tag{36}$$

Consequently, it can meet all the deadlines of all periodic tasks up to full processor utilization. They also proved that in a uniprocessor system RM and EDF are the optimal static and dynamic priority assignment algorithms, respectively. In other word, if a task set is not schedulable by RM (EDF), then it cannot be scheduled by any other static (dynamic) priority assignment. As an example, consider a processor with 3 tasks which their WCET and period are shown in Table 1.

Table 1: Time properties of tasks in a real-time system.

task	WCET (C_i)	period (T_i)
task 1	1	5
task 2	3	9
task 3	2	10

The maximum utilization of the processor is $1/5 + 3/9 + 2/10 = 0.73$ and the utilization bound for 3 tasks is $U = 3(2^{1/3} - 1) = 0.78$. Since $0.73 < 0.78$ the system is surely schedulable by the RM algorithm.

In spite of the dominance of the EDF over the RM, the RM algorithm is more common in practical real-time systems, because it is easier to implement [Sha et al. 1986]. The typical motivations that are usually given in favor of RM state that RM introduces less runtime overhead, it is easier to analyze, it is more predictable in overload conditions, and causes less jitter in task execution. However, Buttazzo [2005] compared RM against EDF under several aspects, using theoretical results and simulation experiments to show that many common beliefs are either false or only restricted to specific situations.

Based on Liu and Layland's result, any periodic task set of any size will be able to meet all deadlines all of the time if the rate monotonic algorithm is used and the total utilization is not greater than 69%. It is worth mention that this condition is sufficient and not necessary. In practical systems, the RM algorithm can often successfully schedule task sets having total utilization higher than 69%. Based on stochastic analysis, Lehoczky et al. [1989] performed an average-case study and showed that for randomly generated task sets consisting of a large number of tasks whose periods are drawn from a uniform distribution, 88% is a good approximation to the threshold of schedulability for the RM algorithm. This implies that due to better resource utilization, the average-case is substantially better than the worst-case. Exact schedulability tests for RM yielding to necessary and sufficient conditions have been independently derived by Joseph and Pandya [1986], Lehoczky et al. [1989], Audsley et al. [1993], and Manabe and Aoyagi [1995].

Leung and Whitehead [1982] considered the case of deadlines smaller than periods and they proved that the deadline monotonic priority assignment is optimal. Also, arbitrary deadline assignment schemes are studied in [Lehoczky et al. 1989; Lehoczky 1990; Peng and Shin 1993]. Xuan et al. [2000] and Abdelzaher et al. [2004] derived some utilization bounds for non-periodic

systems. Pop et al. [2000] proposed solutions to the schedulability analysis of hard real-time systems with control and data dependencies.

Oh and Bakker [1998], Liu [2000], and Bini et al. [2003] proposed approaches for deriving polynomial time tests with better acceptance ratio. For instance, the hyperbolic bound proposed in [Bini et al. 2003] improves the acceptance ratio by a factor of $\sqrt{2}$ for large *n*, compared with the Liu and Layland test. According to hyperbolic bound method, a set of periodic tasks is schedulable by RM if

$$\prod_{i=1}^{n}\left(\frac{C_i}{T_i}+1\right) \leq 2 \tag{37}$$

The authors also extended this test in the case of resource constraints and aperiodic servers. Bini and Buttazzo [2004] derived a schedulability test for periodic task sets under an arbitrary fixed priority assignment which can be tuned through a parameter to balance complexity versus acceptance ratio, so that it can be used online to better exploit the processor, based on the available computational power.

Oh and Son [1995] studied the problem of allocating a set of periodic tasks on a multiprocessor system such that tasks are scheduled to meet their deadlines on individual processors by the RM scheduling algorithm. Utilization bounds of RM in multiprocessor systems are derived in [Oh and Bakker 1998; Andersson and Jonsson 2000; Andersson et al. 2001; Funket al. 2001; Baker 2003]. Davis and Burns [2011] surveyed hard real-time scheduling algorithms and schedulability analysis techniques for homogeneous multiprocessor systems. The survey provides a taxonomy of the different scheduling methods, and considers the various performance metrics that can be used for comparison purposes.

4.2. Application in NoCs

The SoC communication platform needs to provide different levels of service for various application components on the same network. Real-time communication, has very stringent requirements, the correctness relies not only on the communication result but also the completion time bound. A data packet received by a destination too late could be useless. The worst-case acceptable time metric is defined to be the deadline of the packet. A set of real-time traffic flows over the network are termed *schedulable* if all the packets belonging to these traffic flows meet their deadlines under any arrival order of the packet set. In such a system, schedulability analysis deals with investigation of the schedulability of flows in the network. This formalism uses an iterative approach to estimate the maximum end-to-end latency of flows in a network-based system.

To support real-time communication in interconnection networks, several flow control mechanisms have been proposed to explore the priority-based packet scheduling mechanism in the literature [Li and Mutka 1994; Song et al. 1997; Balakrishnan and Ozguner 1998]. Li and Mutka [1994] proposed a flow control mechanism in which there is the same number of virtual channels as the number of priority levels, and a packet can request only a virtual channel which is numbered lower than or equal to its priority. Song et al. [1997] proposed *throttle and preempt* flow control to avoid the priority inversion problem of traditional blocking flow control in wormhole routers. Priority inversion is referred to as a situation where a higher priority packet must wait for the transferring of a lower priority packet. Throttle and preempt flow control prohibits low priority packets from using input buffers beyond their allowed limit, so that high priority packets can always preempt the low priority packets to use the channels, if necessary. Hence, this flow control does not cause priority inversion. However, the upper bound of network latency for each packet in the network is not delivered by this method.

A few works address the packet delivery guarantee problem in communication platforms and proposed schedulability analysis methods to solve it. Since a link in the network is shared among several flows, it is too complex to use the utilization-based tests for determining the packet

schedulability. Hence, researchers had to use direct schedulability tests and proposed a few methods to calculate the worst-case delay of packets. Kandlur et al. [1994], Sathaye and Strosnider [1994], and Li and Mutka [1996] addressed scheduling of real-time communication on direct networks. Kandlur et al. [1994] analyzed interprocessor communication for real-time systems with a direct network using store-and-forward switching. They presented a method for guaranteeing the maximum end-to-end delivery time for packets, and a schedulability test to ensure that real-time packets meet their deadlines. Several flow control methods for real-time wormhole-routed networks were proposed by Li and Mutka [1996]. These methods increase the likelihood of real-time packets meeting their deadlines, but have no guarantees on the feasibility of packets. For this reason, they are more suited to soft-deadline systems than hard-deadline systems. Hary and Ozguner [1997] presented FT1, an offline feasibility test for real-time wormhole-routed packets. This test works for any static priority assignment method. Passing FT1 is a sufficient, but not a necessary condition for feasibility. All the links used to form a route for a flow are lumped as one shared resource (like a bus structure). Since they just considered the direct competitions and ignored indirect competition, their result was optimistic. Balakrishnan and Ozguner [1998] utilized the same model proposed by Hary and Ozguner [1997] and considered the indirect competitions. However, since direct and indirect contentions are considered the same, their result is pessimistic. They showed that the computation complexity of proposed schedulability analysis algorithm is $O(n^2)$ where n is the number of flows in the system. Kim et al. [1998] used a blocking dependency graph to express the contentions a flow may meet and derived the packet delivery upper bound. Lu et al. [2005] formulated a contention tree to take in to account the direct and indirect contentions and captures concurrent use of links.

Shi and Burns [2008] proposed an offline schedulability analysis approach to discuss a real-time on-chip communication with wormhole switching and fixed priority scheduling. The authors proved that the general problem of determining the exact schedulability of real-time traffic flows over the on-chip network is NP-hard. However, they gave a determinant upper bound on the schedulability of real-time traffic flows by evaluating diverse inter-relationships among the traffic flows. They proposed a method to predict the packet network latency based on direct and indirect contention from higher priority traffic flows. Although wormhole switching with fixed priority preemption is a possible solution for real-time on-chip communication, the hardware implementation cost is expensive. Shi and Burns [2009] proposed a solution by utilizing a priority share policy to reduce the resource overhead while still achieving the hard real-time service guarantees. However, the blocking introduced by priority share policy complicates the analysis process. To address this problem, Shi and Burns [2010] proposed a per-priority basis analysis scheme which computes the total time window at each priority level instead of each traffic flow. By checking the release instance of each flow at the corresponding priority window, they determined schedulability efficiently. Building on this static analysis, for a given set of tasks and network topology, the authors further proposed a task mapping and priority assignment algorithm, in such a way that the hard time bounds are met with a reduced hardware overhead.

The focus of schedulability analysis is on real time systems and it determines if a real-time system can meet its deadline or not. Furthermore, schedulability analysis tries to assign *priority to tasks so that* each task meets its deadline.

5. DATAFLOW ANALYSIS

5.1. Overview

Dataflow graph is a Model-of-Computation (MoC) where a number of concurrent processes communicate with each other via unbounded FIFO channels [Lee and Parks 1995]. Writing to these channels is non-blocking while reading from these channels is blocking [Jantsch and Sander 2005]. A dataflow program is a directed graph consisting of nodes (actors) that represent communication and arcs that represent ordered sequences (streams) of data units (tokens) as illustrated in Figure 11.a. Circles represent nodes, arrows represent streams and the dot

represents a token. Dataflow graphs can be hierarchical since a node can represent a dataflow graph. The execution of a dataflow graph is a sequence of firings. During each firing an actor consumes input tokens and produces output tokens. The number of tokens consumed and produced may vary for each firing and is defined in the firing rules of a dataflow actor. An important property of dataflow graphs is that an actor firing only depends on the availability of data. This implies that dataflow graphs are untimed, meaning that nothing is specified about points in time at which firings occur. Dataflow graphs have been shown to be very valuable in digital signal processing applications (e.g., audio and video applications) for concurrent implementation on parallel hardware. When running multiple actors on a single resource, a sequence of firings, also called a schedule, is required. For general dataflow models it cannot be decided whether such a schedule exists because it depends on the input data.

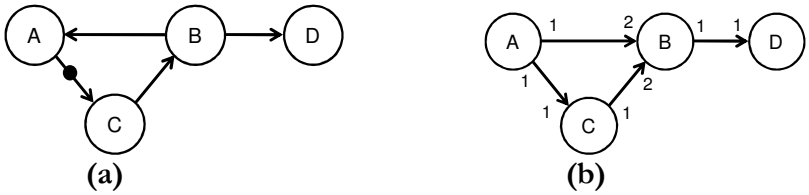


Figure 11: (a) Dataflow network, (b) A synchronous dataflow (SDF) network.

Depending on how the consumption, production, and the firing rules are specified, there exists a variety of different dataflow MoCs. They differ in their *expressiveness and succinctness*, *analyzability,* and *implementation efficiency* [Stuijk et al. 2011]. The expressiveness and succinctness of a MoC indicates how well it can explain characteristics of the system and how compact it is. The analyzability is determined by the availability of analysis and synthesis algorithms and the run-time needed for an algorithm on a graph with a given number of nodes. The implementation efficiency of a MoC is influenced by the complexity of the run-time scheduling problem. We shall now briefly describe the most important dataflow types and then compare them with regard to expressiveness, analyzability and implementation efficiency.

Synchronous dataflow (SDF) model is currently the most popular and widely studied dataflow model for streaming applications [Bekooij et al. 2005]. As shown in Figure 11.b, SDF puts further restrictions on the general dataflow model, since a process consumes and produces a fixed number of tokens for each firing [Lee and Messerschmitt 1987a]. With this restriction it can be tested efficiently, if a finite static schedule exists. If one exists, it can be effectively computed. The numbers on the arcs show how many tokens are produced and consumed during each firing. There exists an algorithm to construct a static periodic schedule for SDF models [Lee and Messerschmitt 1987b]. A possible schedule for the given SDF network is {A, A, C, C, B, D}. This allows determining a static firing sequence which returns the SDF graph into its initial state. Such a firing sequence can be repeated in a loop to statically schedule an SDF graph operating on a stream of data. Much of the existing work on SDF scheduling focuses on optimizing static and dynamic schedules for parallel execution, required sizes of buffers, and end-to-end throughput. There exist many analysis algorithms for SDFs which have polynomial complexity [Stuijk et al. 2011]. Hence, it is possible to derive efficient implementations based on SDF.

Since in SDF, the data rates over different channels are not the same, it is also called a multirate dataflow model [Horstmannshoff et al. 1997]. Single-rate or homogeneous SDF (HSDF) graph is a restricted form of SDF model in which the consumption and production on each edge is a single token [Rumbaugh 1977; Dennis 1980; Lee and Messerschmitt 1987a]. A token is fireable if there is at least one token on all its incoming edges. For any HSDF, a static schedule can be easily constructed by compile-time scheduling tools. Parhi [1989] described an algorithm that transforms any SDF graph into an HSDF graph.

One problem with the SDF model is that for algorithms with variation of the data rate, SDF model leads to use more memory than the application actually needs. For example, consider the graph in Figure 12. Here, implementing the up-sample actor as an SDF actor requires a large memory to hold all of the output tokens from a single firing. This problem has been addressed by extending SDF model to support *cyclo-static* actors in which rate conversion actors are implemented more efficiently to execute in multiple phases [Lauwereins et al. 1994].

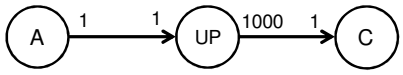


Figure 12: An SDF graph with a large sample rate change. C's Input requires excessive memory [BUCK 1994] © IEEE 1994.

In the cyclo-static dataflow (CSDF) model [Lauwereins et al. 1994; Bilsen et al. 1996] the number of consumed and produced tokens by an actor varies cyclically. There are fixed number of phases in a cycle and each actor produces or consumes fixed number of token in each phase, but different phases may have different behavior. As shown in Figure 13, the production of actor v_i on edge e, is represented as a sequence of constant integers $[x_i(1), x_i(2), \ldots, x_i(p_i)]$. The nth time that actor v_i is executed, it produces $x(1+ (n\text{-}1) \bmod p_i)$ tokens on edge e. The consumption of vertex v_j is analogous. The firing rule of a cyclo-static actor v_j is evaluated as "true" for its nth firing if and only if all input FIFOs contain at least $y_i(1+ (n\text{-}1) \bmod p_j)$ tokens. Lauwereins et al. [1994] showed that although the CSDF is more compact than the HSDF, it is as expressive as HSDF. Since the data rate of each channel is not fixed, analysis and scheduling of CSDF are more complex compared to SDF.

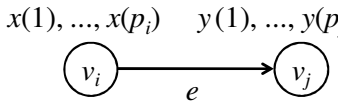


Figure 13: Cyclo-static dataflow (Adapted from [Bilsen et al. 1996] © IEEE 1996).

Due to some non-synchronous and data-dependent behavior, some streaming applications cannot be expressed by SDF and CSDF [Buck 1994]. This problem can be addressed by extending the SDF model to permit some actors with data-dependent behavior. A further generalization of SDF model is Boolean dataflow (BDF) [Buck 1993] where the numbers of consumed and produced tokens depends on the value of a token read from a dedicated control input. Since the token productions and consumptions depend on data values during run time, a BDF network is not completely statically schedulable. However, extending the SDF model to support some dynamic actors (SWITCH and SELECT actors) while preserving static scheduling as much as possible has been studied in [Lee 1991; Buck and Lee 1993]. By using SWITCH and SELECT actors, we can build conditional constructs like if-then-else and do-while loops. As shown in Figure 14.a and 14.b, the SWITCH actor gets a control token and then copies a token from the input to the appropriate output, determined by the Boolean value of the control token. Figure 14.c and 14.d show that the SELECT actor gets a control token and then copies a token from the appropriate input, determined by the Boolean value of the control token, to the output. These actors are not SDF compliant because the number of produced/consumed tokens is not fixed and depends on an input Boolean control.

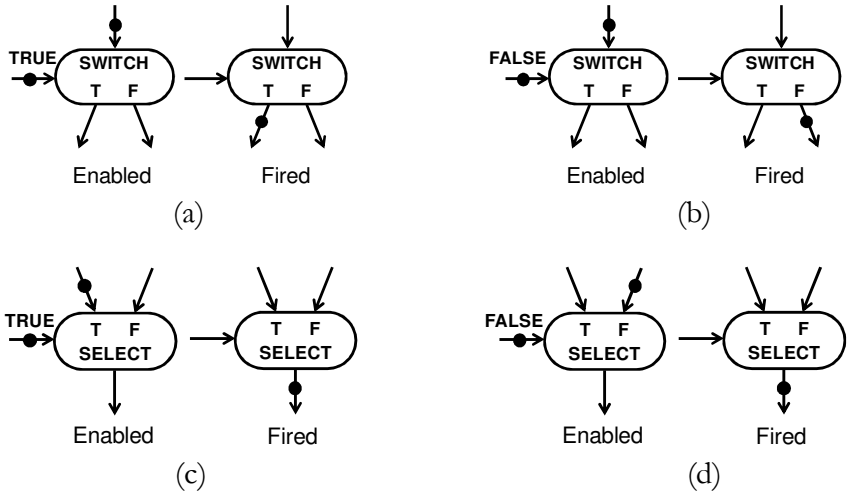


Figure 14: The behavior of SWITCH and SELECT actors for different input (Derived from [Buck 1994] © IEEE 1994).

Dynamic dataflow (DDF) model [Lee and Parks 1995] is a Boolean dataflow model with one additional variation: the control actors mentioned in the BDF model are able to read multiple token values and the data actors can be fired conditionally based on the control actors read. Because of the incomplete knowledge at compile time, BDF and DDF MoCs need a run-time scheduling mechanism to determine when an actor becomes executable. Moreover, it is not always possible to predict whether a schedule with bounded buffer lengths can be constructed. Consequently, run-time scheduling and deadlock detection mechanism are required to implement these MoCs. This makes their implementation less efficient compared to SDF and CSDF. To overcome this problem, several dataflow MoCs have been proposed in related literatures such as Parameterized Synchronous Dataflow (PSDF) [Bhattacharya and Bhattacharyya 2001], Scenario-Aware Dataflow (SADF) [Theelen et al. 2006], Variable Rate Dataflow (VRDF) [Wiggers et al. 2008], and Variable Phased Dataflow (VPDF) [Wiggers et al. 2011]. These MoCs provide a trade-off between analyzability and implementation efficiency. Consequently, they can express some dynamism while allowing design-time analysis and low overhead implementation.

Bhattacharya and Bhattacharyya [2001] proposed a parameterized dataflow framework to improve the expressive power of dataflow MoCs. The parameterized dataflow framework is compatible with many of the existing data flow models including SDF and CSDF. As an application of the parameterized modeling framework, formal semantics for parameterized SDF, PSDF, is developed in the same paper. In the PSDF MoC, channel rates are allowed to be parameterized rather than constant. Therefore, Parameterized schedules and buffer sizes can be computed. Although PSDF can model data-dependent and dynamic DSP systems, options to express dynamism are limited. Variable Rate Dataflow (VRDF) is proposed to model the data-dependent communication behavior. In VRDF, data rates on channels can vary arbitrarily within a specified range. Variable Phased Dataflow (VPDF) is a generalization of both VRDF and of CSDF MoCs where the number of repetitions of CSDF phases can vary in some finite interval. Also, the presented algorithm to compute buffer capacities under throughput constraint is a generalization of the algorithms presented in [Wiggers et al. 2007b; 2008] for CSDF and VRDF, respectively. Existing analyses of VRDF and VPDF are limited to computing buffer capacities that satisfy a throughput constraint. In the Scenario-Aware Dataflow (SADF) MoC, the dynamic behavior of an application is viewed as a collection of different scenarios (behaviors) [Theelen et al. 2006; Stuijk et al. 2011]. Each scenario is static and predictable in performance and resource usage. An SDF MoC models the behavior of each scenario. Since SDF model of different scenarios may differ in all aspects, it is possible to exploit the dynamic behavior of applications to derive an implementation with limited run-time overhead.

As shown in Figure 15, Stuijk et al. [2011] compared dataflow MoCs based on the previously mentioned aspects of expressiveness and succinctness, analyzability and implementation efficiency. Dataflow models are ordered in terms of their ability to capture dynamic behavior in a compact way in the expressiveness and succinctness axis. An overall conclusion is that expressiveness is typically traded off against analyzability and implementation efficiency.

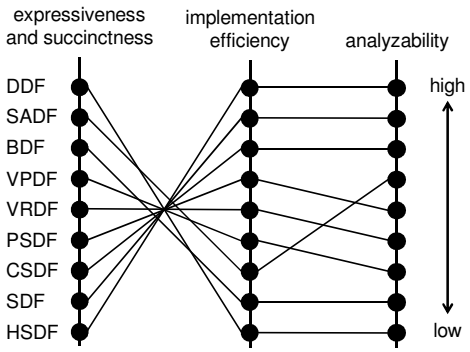


Figure 15: Comparison of dataflow MoCs (Adapted from [Stuijk et al. 2011] © IEEE 2011).

5.2. Applications in NoCs

The classical dataflow models are untimed. To address the timing properties of a system, a worst-case execution time can be associated with each actor [Sriram and Bhattacharyya 2009]. This extension allows us to assess the timing behavior of the NoC-based system such as throughput and latency. A worst-case execution time is added to each actor as shown in Figure 16. The specified number of tokens is consumed and produced within the execution time of the actor. A self-edge of an actor is used to model that the previous execution must be finished before the next execution can start. Scheduling policies can be modeled indirectly by transforming the worst-case execution time to the worst-case response time [Bekooij et al. 2005].

Throughput is an important performance indicator in streaming applications. It has been well studied in the literature on dataflow models [Dasdan and Gupta 1998; Dasdan 2004; Ghamarian et al. 2006]. All these studies focused on analysis of HSDFs and are applicable to SDFs only through a conversion to HSDF [Lee and Messerschmitt 1987a; Sriram and Bhattacharyya 2009]. Maximum Cycle Mean (MCM) analysis is then used to determine throughput. To determine the MCM, the maximum of the cycle means of all simple cycles in the HSDF graph needs to be determined, where the cycle mean (CM) of a cycle c is the sum of the response times of the actors on c divided by the number of initial tokens on the cycle c. The maximal attainable throughput of the graph relates to 1/MCM. Latency is another prominent performance metric. However, a little research has been done on latency. Sriram and Bhattacharyya [2009] studied the latency for the HSDFs. Although it is possible to compute the latency for an SDF through a conversion to a HSDF, the conversion may lead to an exponential increase in the number of nodes in the graph which makes it prohibitively expensive in predicting performance metrics [Stuijk et al. 2006]. Moreira and Bekooij [2007] presented a closed-form expression for the latency of HSDF graphs. The authors provide useful bounds on maximum latency for jobs with periodic, sporadic, and bursty sources, as well as a technique to check latency requirements. Ghamarian et al. [2007] proposed a latency minimization technique that works directly on SDFs. This technique computes the minimal achievable latency for an SDF and provides an execution scheme that gives the minimal latency.

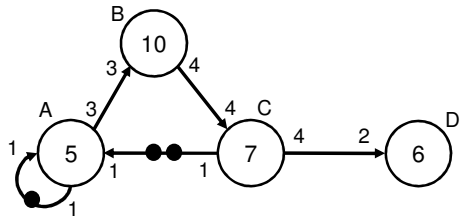
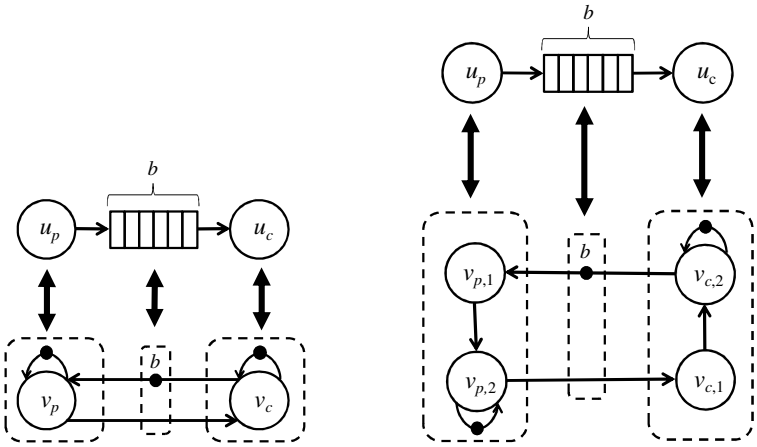


Figure 16: An SDF graph with execution time [Kumar et al. 2008].

Bekooij et al. [2004] proposed an NoC-based multiprocessor architecture and an HSDF model of the jobs which enables reasoning about the timing behavior of the system. The NoC provides virtual point-to-point connections with a guaranteed throughput and maximal latency. The authors modeled every task by one actor with a self edge as depicted in Figure 17.a. Wiggers et al. [2007a] have shown that Latency-Rate servers [Stiliadis and Varma 1998] can be included in a dataflow model by two actors as shown in Figure 17.b. One actor models the rate and the other one models the latency.



(a) (b)

Figure 17: A task model with (a) one actor and (b) two actors [Wiggers et al. 2007a].

Bekooij et al. [2005] used SDF models to derive the end-to-end temporal behavior of jobs in a real-time embedded multiprocessor system. Hansson et al. [2009] and Hansson and Goossens [2010] showed how to construct a CSDF model that conservatively models an NoC connection. Then they used the proposed dataflow model for dimensioning the buffer size in network interfaces to guarantee the system performance and showed that buffer sizes are determined with a run time comparable to analytical methods, and results comparable to exhaustive simulation. Wiggers et al. [2007b] proposed an algorithm that determines close to minimal buffer capacities for CSDF graphs such that the throughput requirement and constraints on maximum buffer capacities are satisfied. Also, they showed that a CSDF model can lead to reduced resource requirements compared to an SDF model.

6. NUMERICAL EXAMPLES

In this section, we consider a simple application mapped on an NoC and show how to estimate the performance metrics by using surveyed mathematical formalisms. Throughout these analyses, we assume the same topology and routing but different flow control mechanism, since applications of formalisms differ starkly in purpose. As an example, schedulability analysis is usually used to determine the worst-case delay bound in systems with hard real time constraints, so we assume the preemptive flow control. On the other hand, network calculus studies more

general systems, so we assume the nonpreemptive flow control. Figure 18 shows the task graph and also communication infrastructure for the on-chip network. Links and routers are organized in a 3x3 mesh structure as shown in Figure 18.b. The delay of links and routers are assumed 1 cycle and 2 cycles, respectively. Tasks, executing on different Intellectual Property (IP) modules, communicate with each other by transmitting packets through the NoC.

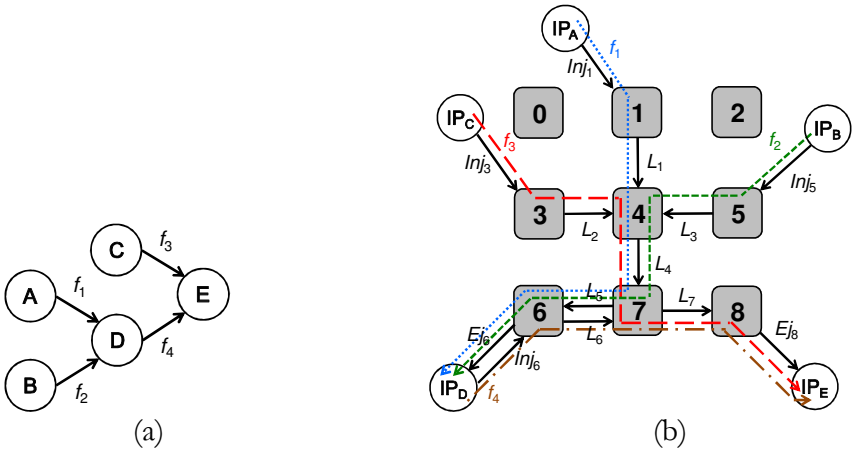


Figure 18: (a) Task graph of an application mapped on an (b) NoC platform.

Table 2 shows attributes of the traffic flows including flow priority, source and destination of the flow, packet length in flits and average packet generation rate in packet/cycle/IP, as well as route of the flow in the network. For instance, f_1 has the highest priority in the system and starts in IP_A and passes through injection channel 1, links 1, 4, 5, and ejection channel 6 before terminating in IP_D. All packets of this flow have the same length of the 8 flits and the average packet generation rate is 0.02 packet/cycle. In other words, on average, every 50 cycles a packet is generated in IP_A.

Table 2: Description of traffic flows.

flow	priority	source	destination	packet length (m_i)	packet generation rate (λ_i)	route
f_1	high	1	6	8	0.02	Inj_1, L_1, L_4, L_5 , Ej_6
f_2	low	5	6	16	0.01	Inj_5, L_3, L_4, L_5 , Ej_6
f_3	medium	3	8	12	0.02	Inj_3, L_2, L_4, L_7, Ej_8
f_4	low	6	8	8	0.03	Inj_6, L_6, L_7, Ej_8

6.1. Queueing Theory

In this section, the queueing theory-based analytical model proposed by Kiasari et al. [2012] is used to estimate the average latency of flows in Figure 18.b. We described this model briefly in Section 2.3. We assume that the flow control mechanism is wormhole switching and there is one flit buffer per input channel. Channels are allocated per packet. It means that the channel is released when the whole packet has passed through the channel. Also, we assume that nodes generate packets independently of each other following a Poisson process.

The basic packet latency, d_i, happens when there is no traffic contention. It consists of two parts: the latency of head flit and the latency of body flits. Latency of head flit is determined by

routing distance and router and wire delay. Once the head flit arrives at the destination, the body flits follow the header flit in a pipelined fashion. Hence, the body flit latency is a function of packet size and wire delay. For instance, according to Figure 18.b, head flit of f_1 passes through 4 routers and 5 links. Therefore, the head flit latency is $4t_{router} + 5t_{wire} = 13$ cycles and the body flit latency equals $(m_1 - 1)t_{wire} = 7$. As a result, the basic packet latency of f_1 is

$d_1 = 4t_{router} + 5t_{wire} + (m_1 - 1)t_{wire} = 8 + 5 + 7 = 20$ cycles

Average packet latency of f_1, D_1, is the time since the packet is created in IP_A, until the last flit reaches the IP_D, including the queueing time spent at the source node ($\overline{W}_{Inj_1 \to L_1}$) and intermediate nodes ($\overline{W}_{L_1 \to L_4}$). In Figure 18.b, D_1 can be computed as

$\overline{D}_1 = d_1 + \overline{W}_{Inj_1 \to L_1} + \overline{W}_{L_1 \to L_4} + \overline{W}_{L_4 \to L_5} + \overline{W}_{L_5 \to Ej_6}$

Note that, $\overline{W}_{L_4 \to L_5}$ and $\overline{W}_{L_5 \to Ej_6}$ are equal to zero. The input buffer of L_4 and L_5 only have space for one head flit. Hence, if the head flit holds the input buffer of L_4, it can access channel L_5 without any waiting time. Therefore,

$\overline{D}_1 = d_1 + \overline{W}_{Inj_1 \to L_1} + \overline{W}_{L_1 \to L_4}$

To estimate the $\overline{W}_{i \to j}$, the first moment (average) and second moment of channels service time should be computed. We remind here that the second moment of a random variable X is defined as the average of X^2 $\left(\overline{X^2} = \sum_{i=1}^{k} (X_i)^2 / k\right)$. Determination of the channel service time moments starts at ejection channels and works in the reverse order of routing towards to the source of the packet. It means that, to compute $\overline{D}_1$, we should compute the service time of Ej_6, L_5, L_4, and L_1 ($\bar{s}_{Ej_6}, \bar{s}_{L_5}, \bar{s}_{L_4}, \bar{s}_{L_1}$ respectively). Since the delay of all channels is considered 1 cycle, an ejection channel offers service time of m_i cycles to a packet of length m_i flits. According to Figure 18.b and Table 2, Ej_6 serves flows 1 and 2 with the length of 8 and 16 flits and the rate of 0.02 and 0.01 packet/cycle, respectively. Therefore, average service time of ejection channels 6 is

$$\bar{s}_{Ej_6} = \frac{0.02}{0.03} \times 8 + \frac{0.01}{0.03} \times 16 = 10.67$$

The waiting time for a channel closer to the destination (ejection channel) can be thought of as adding to the service time of channels farther from the destination. In other words,

$$\bar{s}_{L_5} = \bar{s}_{Ej_6} + \overline{W}_{L_5 \to Ej_6}$$
$$\bar{s}_{L_4} = \frac{3}{5}\left(\bar{s}_{L_5} + \overline{W}_{L_4 \to L_5}\right) + \frac{2}{5}\left(\bar{s}_{L_7} + \overline{W}_{L_4 \to L_7}\right)$$
$$\bar{s}_{L_1} = \bar{s}_{L_4} + \overline{W}_{L_1 \to L_4}$$

As stated before, $\overline{W}_{L_5 \to Ej_6} = 0$. Hence, $\bar{s}_{L_5} = \bar{s}_{Ej_6} = 10.67$. To compute the $\bar{s}_{L_4}$, we have to compute $\overline{W}_{L_4 \to L_7}$ in advance. The first and second moments of service time of L_7 can be given by

$$\bar{s}_{L_7} = \bar{s}_{Ej_8} = \frac{2}{5} m_3 + \frac{3}{5} m_4 = 9.6$$
$$\overline{s_{L_7}^2} = \frac{2}{5} (m_3)^2 + \frac{3}{5} (m_4)^2 = 96$$

After computing the moments of service time, the service rate and squared coefficient of variation (SCV) of service time of L_7 are computed.

$$\mu_{L_7} = 1/\bar{s}_{L_7} = 0.1042$$
$$C_{s_{L_7}}^2 = \overline{s_{L_7}^2}/\left(\bar{s}_{L_7}\right)^2 - 1 = 0.0417$$

Now, we are able to compute the waiting time for channel L_7. Flows f_3 and f_4 compete to access L_7 while f_3 has higher priority than f_4.

$$\bar{W}_{L_4 \to L_7} = \frac{1}{2}\left(C_A^2 + C_{s_{L_7}}^2\right)\frac{\lambda_{L_7}}{\mu_{L_7}^2} = 2.4$$

C_A^2 is the SCV of arrival process and for the Poisson process it equals 1. Similarly, for L_4 we can write

$$\bar{s}_{L_4} = \frac{3}{5}\left(\bar{s}_{L_5} + \bar{W}_{L_4 \to L_5}\right) + \frac{2}{5}\left(\bar{s}_{L_7} + \bar{W}_{L_4 \to L_7}\right) = 11.2$$
$$\overline{s_{L_4}^2} = \frac{6}{8}\left(\bar{s}_{L_5} + \bar{W}_{L_4 \to L_5}\right)^2 + \frac{2}{8}\left(\bar{s}_{L_7} + \bar{W}_{L_4 \to L_7}\right)^2 = 125.9$$
$$\mu_{L_4} = 1/\bar{s}_{L_4} = 0.0893$$
$$C_{s_{L_4}}^2 = \overline{s_{L_4}^2}/\left(\bar{s}_{L_4}\right)^2 - 1 = 0.0037$$

Waiting time for L_4 are given by

$$\bar{W}_{L_1 \to L_4} = \frac{1}{2}\left(C_A^2 + C_{s_{L_4}}^2\right)\frac{\lambda_{L_4}}{\mu_{L_4}^2} = 3.1$$

If we repeat the computation for L_1, $\bar{W}_{Inj_1 \to L_1}$ can be computed as

$$\bar{W}_{Inj_1 \to L_1} = \frac{1}{2}\left(C_A^2 + C_{s_{L_1}}^2\right)\frac{\lambda_{L_1}}{\mu_{L_1}\left(\mu_{L_1} - \lambda_1\right)} = 2.9$$

Finally, we can write

$$\bar{D}_1 = d_1 + \bar{W}_{Inj_1 \to L_1} + \bar{W}_{L_1 \to L_4} = 26 \text{ cycles}$$

Following the same approach, the average packet latency for other flows can be computed.

6.2. Network Calculus

In this section, we show how to apply the network calculus formalism for estimating the worst-case latency in the NoC described in Figure 18 and Table 2. Figure 19 shows the system model in network calculus.

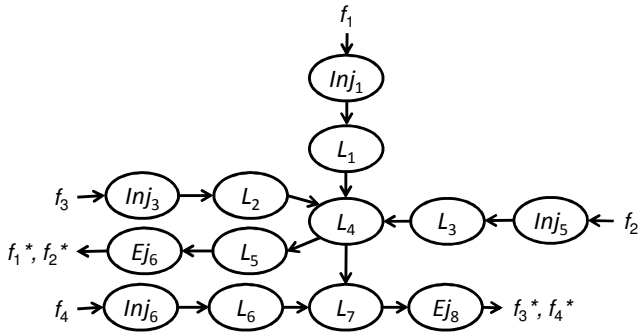


Figure 19: Network calculus model of the system in Figure 18.b.

We consider the virtual cut-through switching with nonpreemptive priority scheduling policy. It is also assumed that four leaky bucket controllers constrain the packet injection process in the system. In other word, f_i is a (σ_i, ρ_i) regulated flow which is constrained by arrival curve $\gamma_{\rho_i,\sigma_i} = \rho_i t + \sigma_i$. According to Table 2, average flit injection rate are calculated as

$\rho_1 = 0.02 \times 8 = 0.16$ flit/cycle
$\rho_2 = 0.01 \times 16 = 0.16$ flit/cycle
$\rho_3 = 0.02 \times 12 = 0.24$ flit/cycle
$\rho_4 = 0.03 \times 8 = 0.24$ flit/cycle

Let us further assume that the burstiness values are

$\sigma_1 =$ 5 packets = 40 flits
$\sigma_2 =$ 3 packets = 48 flits
$\sigma_3 =$ 4 packets = 48 flits
$\sigma_4 =$ 5 packets = 40 flits

Channels and routers are modeled with latency-rate servers, $\beta_{R,T}(t) = R(t-T)^+$. Average service rates are considered 1 flit/cycle and wire and router delay were supposed to be 1 cycle and 2 cycles, respectively. Therefore, we can model the system as shown in Figure 20.

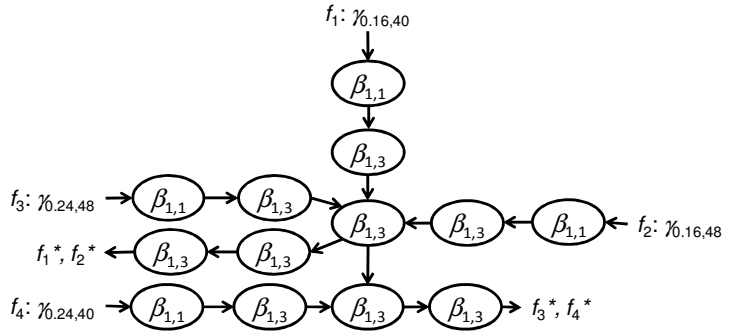


Figure 20: System model based on the leaky bucket arrival curves and latency-rate servers.

Applying the concatenation theory, the model shown in Figure 20 can be simplified to Figure 21. As we mentioned in Section 3.1.3, the concatenation of two latency-rate server results in a new latency-rate server.

$$\beta_{R_1,T_1} \otimes \beta_{R_2,T_2} = \beta_{\min(R_1,R_2),T_1+T_2}$$

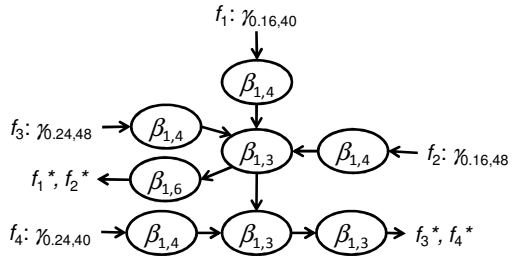


Figure 21: Simplified system model.

Figure 22 shows how to compute the leftover service curve for f_1 step by step. The node in the center of Figure 22.a guarantees the service curve $\beta_{1,3} = t - 3$ to the aggregate of the three flows where f_1 has the highest priority. Then f_1 is guaranteed a service curve $\beta_{1,3} = (t - 3) - 16$ because the maximum packet size for the lower priority flows is 16 [Le Boudec and Thiran 2001]. Applying the concatenation theory on Figure 22.b results in Figure 22.c.

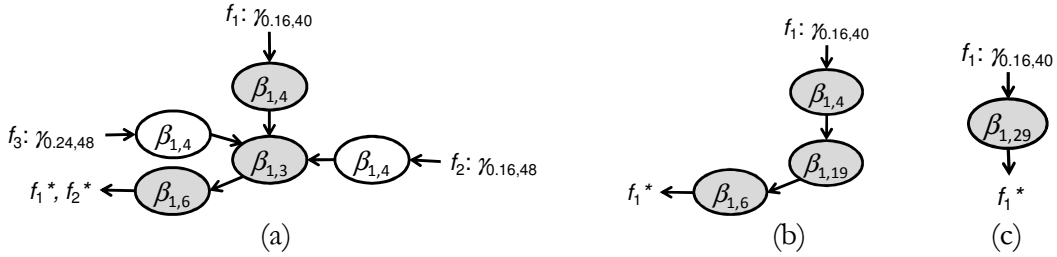


Figure 22: Leftover service curve for flow f_1.

By using the delay bound formula, Eq. (22), we can write

$D_1 \leq T + \sigma_1/R = 29 + 40/1 = 69$ cycles

The worst-case delay of other flows can be computed by the same approach.

6.3. Schedulability Analysis

In section 4.2, we reviewed FT1 proposed by Hary and Ozguner [1997], a feasibility test for real-time wormhole-routed systems. In this section, we describe it in more detail as a sample of schedulability analysis approach. Consider again the system described in Figure 18 and Table 2. We assume that packets are injected periodically in the network. The length of time between releases of successive packets of f_i is a constant, which is called the period T_i for this flow. Using packet generation rate in Table 2, period of each flow can be easily computed.

$T_1 = 1/0.02 = 50$ cycles
$T_2 = 1/0.01 = 100$ cycles
$T_3 = 1/0.02 = 50$ cycles
$T_4 = 1/0.03 = 33$ cycles

It is also assumed that routers architecture support preemptive priority scheduling and there are as many virtual channels per link as flows per link. Therefore, a packet cannot be blocked due to

its inability to access a virtual channel. Let S_i be the set of higher-priority flows that share at least one link with f_i.

$S_1 = \emptyset$
$S_2 = \{f_1, f_3\}$
$S_3 = \{f_1\}$
$S_4 = \{f_3\}$

A packet from f_i can only be blocked from accessing a link by higher priority packets that share a link with f_i (i.e., any packets from $f_j \in S_i$). f_i may be blocked by more than one instance of each $f_j \in S_i$, since flows are periodic. The maximum end-to-end latency of f_i is the sum of the blocking time and d_i. d_i is the basic packet latency and we showed how to calculate them in section 6.1.

$d_1 = 4t_{router} + 5t_{wire} + (m_1 - 1)t_{wire} = 8 + 5 + 7 = 20$ cycles
$d_2 = 4t_{router} + 5t_{wire} + (m_2 - 1)t_{wire} = 8 + 5 + 15 = 28$ cycles
$d_3 = 4t_{router} + 5t_{wire} + (m_3 - 1)t_{wire} = 8 + 5 + 11 = 24$ cycles
$d_4 = 3t_{router} + 4t_{wire} + (m_4 - 1)t_{wire} = 6 + 4 + 7 = 17$ cycles

f_1 do not suffer any contention and receives the worst-case network latency equal to their basic latency. Therefore, the worst-case delay of a packet from f_1 is 20 cycles.

The worst-case response time of other flows, f_i, at time t, $R_i(t)$, is given as

$$R_i(t) = d_i + \sum_j d_j \lceil t/T_j \rceil, \quad f_j \in S_i \tag{38}$$

where $\lceil t/T_j \rceil$, is the maximum number of instances of higher priority packets f_j that can occur up to time t. An iterative approach is used to solve Eq. (38) and the first iteration begins at $t = 0$. The value of t used for each iteration is the latency of the previous iteration. Eq. (38) converges when the latency of the current iteration is equal to the latency of the previous iteration. For instance, consider flow f_3 which shares at least one link with higher priority flow f_1. The worst-case latency for flow f_3 is given by

$$R_3(t) = d_3 + d_1 \lceil t/T_1 \rceil = 24 + 20 \lceil t/50 \rceil$$

$t = 0$: $R_3(t) = 24$
$t = 24$: $R_3(t) = 24 + 20\lceil 20/50 \rceil = 44$
$t = 44$: $R_3(t) = 24 + 20\lceil 44/50 \rceil = 44$

The worst-case latency of f_3 converges at $t = 44$ cycles. Therefore, based on FT1 test, if the deadline of f_3 is greater than 44, f_3 is schedulable, otherwise not.

6.4. Dataflow Analysis

We consider again the application and architecture in Figure 18. In this section, we assume that all packets are single-flit and they arrive strictly periodically. We use the proposed approach by Wiggers et al. [2007a] to find the minimum required buffer for f_1, if IP_A injects a packet to the network every 2 cycles ($throughput(f_1) = 0.5$). Similar to network model in Section 6.2, we can use the latency-rate servers to model the network elements which serve f_1. The result is shown in Figure 23.a. Also, Figure 23.b shows the HSDF model of the network for f_1.

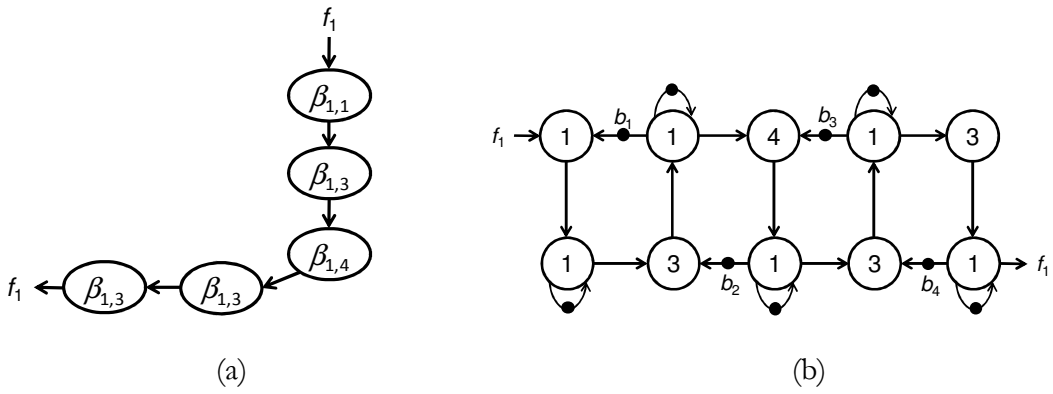


Figure 23: (a) Latency-rate model and (b) dataflow model of the network for flow f_1.

In the dataflow model shown in Figure 23.b, we need to guarantee that the throughput of the graph equals the throughput of flow f_1. Since $throughput(f_1) = 0.5$, it is required that the MCM of the graph to be maximally 2, as we described in Section 5.2. In other words,

$$MCM = max\left\{\frac{1}{1}, \frac{1+1+1+3}{b_1}, \frac{1+3+1+4}{b_2}, \frac{1+4+1+3}{b_3}, \frac{1+3+1+3}{b_4}\right\} \leq 2$$

which results in

$$\frac{1+1+1+3}{b_1} \leq 2$$

$$\frac{1+3+1+4}{b_2} \leq 2$$

$$\frac{1+4+1+3}{b_3} \leq 2$$

$$\frac{1+3+1+3}{b_4} \leq 2$$

Therefore, $b_1 = 3$ packets, $b_2 = 5$ packets, $b_3 = 5$ packets, and $b_4 = 4$ packets are the minimum number of buffers to guarantee $throughput(f_1) = 0.5$.

7. BRIDGING DIFFERENT FORMALISMS

With the advances of technology, SoCs characteristics and requirements are changing. Therefore, there has been a demand for theories to deal with heterogeneous architectures and applications encountered in such systems. The two front runner models for performance analysis and performance guarantees of SoCs are average-case and worst-case analytical models. An open research issue is a unified analytical model for performance. To overcome the weaknesses of individual formalisms, research should try to combine the components of these formalisms. A few attempts are made to link these formalisms together.

Based on network calculus, Schmitt [2003] derived bounds on delay and backlog per traffic class in non-preemptive priority queueing systems. There are known results for the average behavior of such a queueing system from queueing theory. By use of numerical investigations, worst-case bounds are compared to those average-case analysis results in order to give a feel as to how conservative the worst-case bounds are. Pandit et al. [2004] analyzed the impact of network calculus bounds on queueing theory results. More precisely, they studied the impact of traffic shaping and service curve enforcement on a single M/M/1 queue. They do not analyze the system analytically and the study was performed through simulation. The queue length distribution was compared with the original M/M/1 case and the authors showed how the probability mass of the higher buffer states (longer queues) of the M/M/1 queue distributes over the lower buffer states (shorter queues).

Another attempt to link queueing theory and network calculus was presented by Jiang [2009]. Based on the two network calculus principles, the min-plus and the max-plus convolutions, the author derived delay bounds for the single node case and showed that they are consistent with similar bounds derived based on Lindley's equation [Lindley 1952] for G/G/1 queues. Besides attempts to link network calculus and queueing theory, there is an effort to bridge a gap between network calculus and dataflow analysis formalisms. A relation between concepts from the network calculus and dataflow domains was described by Wiggers et al. [2007a], where it is shown that a latency-rate server, which is a concept from network calculus, can be included in the HSDF model. Figure 24 models a task u_x with one input FIFO and one output FIFO that executes on a latency-rate server with latency T_x and allocated rate R_x. r_y and r_z are the response time of actors y and z, respectively. The resulting dataflow model provides guarantees on the temporal behavior of the implementation.

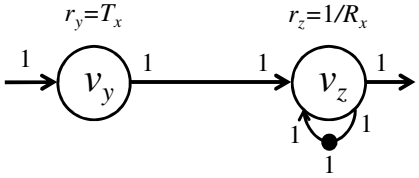


Figure 24: A dataflow component that models a latency-rate server (Derived from [Wiggers et al. 2007a]).

8. CONCLUSION

To summarize the discussion, we compare the presented formalisms based on the event model, node mode, and analysis output of the formalisms.

8.1. Event Model

The event model refers to the data packet representation. In queueing theory, the event model is the probability distribution of the interarrival time of packets. The interarrival times of different flows are independent and identically distributed random variables. In network calculus, an upper bound for the number of packets, called the arrival curve, models the events. An arrival curve is associated to each flow and they are assumed to be independent. In schedulability analysis, the events are modeled with periodic and sporadic models in which a flow is represented by its minimum interarrival times. Like in queueing theory and network calculus, there is no dependency between events. Dataflow analysis models events with tokens which are produced and consumed by nodes. The production of new tokens (events) in the output ports depends on the availability of tokens in the input ports. Hence, dataflow analysis is the only formalism that captures the dependency between the events.

8.2. Node Model

Nodes are modeled based on their service time. In queueing theory, service time is specified probabilistically. The node model is the probability distribution of the router service time. In

network calculus, nodes are modeled using the notion of a service curve, which is a function characterizing the minimum number of bits a node must transmit in any given time interval. In scheduling analysis, a node is modeled based on its worst-case delay and the scheduling policy. The worst-case node delay and scheduling policy represent a node in dataflow analysis. Note, that scheduling policy is modeled explicitly in scheduling theory and data flow models, but implicitly in network calculus and queueing theory as more abstract node models.

Transformation of one node model to another is not always possible without loss of information or accuracy. Since queueing theory deals with average-case analysis and the three other formalisms deal with worst-case analysis, transformations between them results in information loss. Many common distributions in queueing theory, e.g. exponential distribution, assign a non-zero probability to any positive number, even regions far from the mean value. Therefore, usually it is not possible to find a worst-case bound for the service time of a node in queueing theory. Also, it is not possible to estimate the shape of the service time distribution from the worst-case service time. Node models in the other formalisms (network calculus, schedulability analysis, and dataflow analysis) can be transformed to each other without information loss. For instance, it is easy to find the worst-case node delay in dataflow analysis from a service curve in network calculus or designers can estimate the shape of the service curve from the worst-case node delay.

8.3. Analysis Results

The four considered formalisms lead to different kinds of analysis results. Since queueing theory deals with probability models, it can compute average-case performance metrics such as average packet latency, average throughput, average energy and power consumption [Kim et al. 2005; Kiasari et al. 2008b], and average resource utilization. Network calculus computes the worst-case packet latency and maximum backlog in the system, and schedulability analysis estimates the worst-case latency of a flow to determine if it is schedulable or not. Finally, dataflow analysis determines the worst-case latency and throughput of a given system. Table 3 summarizes the input model, node model and output of the studied formalism.

Table 3: Input model, node model and output of the mathematical formalisms.

Formalism	Event model	Node model	Analysis Result
Queueing Theory	the probability distribution of the interarrival time of packets	probability distribution of the node service time	average packet latency, average throughput, average energy and power consumption, and average resource utilization
Network Calculus	an upper bound for the number of packets (arrival curve)	a representation of worst-case service time of the node (service curve)	worst-case latency and backlog
Schedulability Analysis	minimum interarrival times of periodic or sporadic packets	worst-case node delay and scheduling policy	worst-case latency
Dataflow Analysis	tokens which are produced and consumed by nodes	worst-case node delay and scheduling policy	throughput, buffer sizing, worst-case latency

8.4. Summary

Finally, we list the features and weaknesses of each formalism which are summarized in Table 4.

Table 4: Advantages and disadvantages of the formalisms.

Formalism	Feature	Weakness
queueing theory	• abstract model • average-case analysis	• hard to derive accurate models • cannot represent flow dependencies
network calculus	• abstract model • worst-case analysis	• hard to derive accurate models • cannot represent flow dependencies
schedulability analysis	• easy to setup event and node models	• cannot represent flow dependencies • limited accuracy
dataflow analysis	• can express flow dependencies and flow control	• must be used with restricted models such as SDF and CSDF

All formalisms can find a closed-form relationship between system parameters and system performance metrics. However, in case of queueing theory and network calculus, it is difficult to derive mathematical models of a given network, because they use complicated event models and node models. On the other hand, since schedulability analysis uses simpler event models, the performance model is easily extracted with less accuracy. A common problem of all models except dataflow is that they cannot capture well dependencies between data flows. The dataflow model is the only formalism that can model the dependent flows in a system accurately. The general trade-off between abstraction and accuracy can also be observed in the comparison between these four formalisms. Queueing theory, network calculus, and schedulability analysis can be considered more abstract than dataflow. As a consequence, details such as flow control, back-pressure and data dependencies are more difficult to captures in a natural way. Dataflow can easily model these details but for an efficient and precise analysis restricted models such as SDF and CSDF have to be used.

8.5. Outlook

Since each of the reviewed formalisms has different advantages and difficulties, and since they also partially differ in purpose, none of them can easily replace all others. There are definitely point problems for each formalism that are worthy for further studies, but research on integrated approaches to the problems of system performance analysis is most urgent. Although each formalism can be extended in various directions, these extensions typically run into problems of complex mathematics or they are perceived to be unnatural and cumbersome. Therefore, we believe that comprehensive frameworks that combine two or more formalisms would be most desirable. For instance, queueing theory and network calculus could be combined to offer both worst-case and average-case analysis. The result could be combined with dataflow analysis to naturally model event dependencies and lead a bridge to simulation. However, significant work to explore and understand the relations between these models and the possible and useful transformations between them is required.

ACKNOWLEDGMENT

The authors would like to thank the reviewers for their valuable comments and suggestions. This work is funded in part by Intel Corporation through a research gift.

REFRENCES

ABDELZAHER, T. F., SHARMA, V., AND LU, C. 2004. A utilization bound for aperiodic tasks and priority driven scheduling. *IEEE Trans. Comput.* 53, 3, 334-350.

ANDERSSON, B. AND JONSSON, J. 2000. Fixed-priority preemptive multiprocessor scheduling: to partition or not to partition. In *Proceedings of the 7th International Conference on Real-Time Systems and Applications (RTCSA'00)*. IEEE Computer Society, 337-346.

ANDERSSON, B., BARUAH, S., AND JONSSON, J. 2001. Static-priority scheduling on multiprocessors. In *Proceedings of the 22nd IEEE Real-Time Systems Symposium (RTSS'01)*. IEEE Computer Society, 193-202.

AUDSLEY, N. C., BURNS, A., TINDELL, K., AND WELLINGS, A. 1993. Applying new scheduling theory to static priority preemptive scheduling. *Softw. Eng. J.* 8, 5, 284-292.

ALTISEN, K., LIU, Y., AND MOY, M. 2010. Performance evaluation of components using a granularity-based interface between real-time calculus and timed automata. In *Proceedings of the 8th Workshop on Quantitative Aspects of Programming Languages (QAPL'10)*. 16-23.

BAKER, T. P. 2003. Multiprocessor EDF and deadline monotonic schedulability analysis. In *Proceedings of the 24th IEEE International Real-Time Systems Symposium (RTSS'03)*. IEEE Computer Society, 120-129.

BAKHOUYA, M., SUBOH, S., GABER, J., EL-GHAZAWI, T. A., AND NIAR, S. 2011. Performance evaluation and design tradeoffs of on-chip interconnect architectures. *Simulation Modeling Practice and Theory*. 19, 6, 1496-1505.

BALAKRISHNAN, S. AND OZGUNER, F. 1998. A priority-driven flow control mechanism for real-time traffic in multiprocessor networks. *IEEE Trans. Parallel Distrib. Syst.* 9, 7, 664-678.

BEKOOIJ, M. J. G., MORIERA, O., POPLAVKO, P., MESMAN, B., PASTRNAK, M., AND VAN MEERBERGEN, J. 2004. Predictable embedded multiprocessor system design. In *Proceedings of the International Workshop on Software and Compilers for Embedded Systems*, Lecture Notes in Computer Science, vol. 3199, Springer, 77-91.

BEKOOIJ, M. J. G., HOES, R., MOREIRA, O., POPLAVKO, P., PASTRNAK, M., MESMAN, B., MOL, J. D., STUIJK, S., GHEORGHITA V., AND VAN MEERBERGEN J. 2005. Dataflow analysis for real-time embedded multiprocessor system design, chapter 15, Dynamic and robust streaming between connected consumer-elecronic devices, Kluwer Academic Publishers.

BERTSIMAS, D. AND MOURTZINOU, G. 1997. Transient laws of non-stationary queueing systems and their applications. *Queueing Systems,*25, 1-4, 115-155

BHATTACHARYA, B. AND BHATTACHARYYA, S. S. 2001. Parameterized dataflow modeling for DSP systems, *IEEE Trans. Signal Process.* 49, 10, 2408-2421.

BILSEN, G., ENGELS, M., LAUWEREINS, R., AND PEPERSTRAETE, J. 1996. Cyclo-static dataflow. *IEEE Trans. Signal Process.* 44, 2, 397-408.

BINI, E., BUTTAZZO, G. C. AND BUTTAZZO, G. M. 2003. Rate monotonic scheduling: the hyperbolic bound, *IEEE Trans. Comp.* 52, 7, 933-942.

BINI, E. AND BUTTAZZO, G. C. 2004. Schedulability analysis of periodic fixed priority systems. *IEEE Trans. Comput.* 53, 11, 1462-1473.

BOGDAN, P. AND MARCULESCU, R. 2007. Quantum-like effects in network-on-chip buffers behavior. In *Proceedings of the 44th Annual Design Automation Conference (DAC'07)*. ACM Press, 266-267.

BOGDAN, P. AND MARCULESCU, R. 2009. Statistical physics approaches for network-on-chip traffic characterization. In *Proceedings of the 7th IEEE/ACM International Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS'09)*. ACM Press, 461-470.

BOGDAN, P. AND MARCULESCU, R. 2010. Workload characterization and its impact on multicore platform design. In *Proceedings of the 8th IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS'10)*. ACM Press, 231-240.

BOGDAN, P. AND MARCULESCU, R. 2011. Non-stationary traffic analysis and its implications on multicore platform design. *IEEE Trans. Computer-Aided Design Integr. Circuits Syst.* 30, 4, 508-519.

BOLCH, G., GREINER, S., DE MEER, H., AND TRIVEDI, K. S. 2006. *Queueing Networks and Markov Chains: Modeling and Performance Evaluation with Computer Science Applications*, 2nd Edition, John Wiley & Sons.

BUCK, J. T. 1993. Scheduling dynamic dataflow graphs with bounded memory using the token flow model. Ph.D. dissertation, Dept. of EECS, UC Berkeley, USA.

BUCK, J. T. AND LEE, E. A. 1993. Scheduling dynamic dataflow graphs with bounded memory using the token flow model. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing: Plenary, Special, Audio, Underwater Acoustics, VLSI, Neural Networks - (ICASSP'93)*, vol. I. IEEE Computer Society, 429-432.

BUCK, J. T. 1994. A dynamic dataflow model suitable for efficient mixed hardware and software implementations of DSP applications. In *Proceedings of the 3rd international workshop on hardware/software co-design (CODES'94)*. IEEE Computer Society, 165-172.

BUTTAZZO, G. C. 2005. Rate monotonic vs. EDF: Judgement day. *Real Time Systems 29*, 1, 5-26.

CHAKRABORTY, S., KUNZLI, S., AND THIELE, L. 2003. A general framework for analysing system properties in platform-based embedded system designs. In *Proceedings of the conference on Design, Automation and Test in Europe - Volume 1 (DATE'03)*. IEEE Computer Society, 10190-10195.

CHANG, C.-S. 2000. *Performance Guarantees in Communication Networks*. Springer-Verlag, London, UK.

CHENG, A.-L., PAN, Y., YAN, X.-L., HUAN, R.-H. 2011. A general communication performance evaluation model based on routing path decomposition. *J. Zhejiang Univ. - Sci. C (Comput. & Electron.)* 12, 7, 561-573.

CRUZ, R. L. 1991a. A calculus for network delay, part I: Network elements in isolation. *IEEE Trans. Inf. Theory.* 37, 1, 114-131.

CRUZ, R. L. 1991b. A calculus for network delay, part II: Network analysis. *IEEE Trans. Inf. Theory.* 37, 1, 132-141.

DASDAN, A. 2004. Experimental analysis of the fastest optimum cycle ratio and mean algorithms. *ACM Trans. Des. Autom. Electron. Syst.* 9, 4, 385-418.

DASDAN, A. AND GUPTA, R. 1998. Faster maximum and minimum mean cycle algorithms for system performance analysis. *IEEE Trans. Computer-Aided Design Integr. Circuits Syst.*17, 10, 889-899.

DAVIS, R. I. AND BURNS, A. 2011. A survey of hard real-time scheduling for multiprocessor systems. ACM Comput. Surv. 43, 4, article 35.

DENNIS. J. B. 1980. Data flow supercomputers. *Computer.* 13, 11, 48-56.

FOROUTAN, S., THONNART, Y., HERSEMEULE, R., AND JERRAYA, A. 2009. Analytical computation of packet latency in a 2D-mesh NoC. In *Proceedings of the Joint IEEE North-East Workshop on Circuits and Systems and TAISA Conference*, 1-4.

FOROUTAN, S., THONNART, Y., HERSEMEULE, R., AND JERRAYA, A. 2010. An analytical method for evaluating network-on-chip performance. In *Proceedings of the 13th Conference on Design, Automation and Test in Europe (DATE'10).*1629-1632.

FUNK, S., GOOSSENS, J., AND BARUAH, S. 2001. On-line scheduling on uniform multiprocessors. In *Proceedings of the 22nd IEEE Real-Time Systems Symposium (RTSS'01).* IEEE Computer Society, 183-192.

GHAMARIAN, A. H., GEILEN, M. C. W., STUIJK, S., BASTEN, T. THEELEN, B. D., MOUSAVI, M. R., MOONEN, A. J. M., AND BEKOOIJ, M. J. G. 2006. Throughput analysis of synchronous data

flow graphs. In *Proceedings of the 6th International Conference on Application of Concurrency to System Design (ACSD'06)*. IEEE Computer Society, 25-36.

GHAMARIAN, A. H., STUIJK, S., BASTEN, T., GEILEN, M. C. W., AND THEELEN, B. D. 2007. Latency minimization for synchronous data flow graphs. In *Proceedings of the 10th Euromicro Conference on Digital System Design Architectures, Methods and Tools (DSD'07)*. IEEE Computer Society, 189-196.

GUAN, W., TSAI, W., AND BLOUGH, D. 1993. An analytical model for wormhole routing in multicomputer interconnection networks. In *Proceedings of the International Parallel Processing Symposium*, 650-654.

GUZ, Z., WALTER, I., BOLOTIN, E., CIDON, I., GINOSAR, R., AND KOLODNY, A. 2007. Network delays and link capacities in application-specific wormhole NoCs. *J. VLSI Design*, 2007, article 90941, 15 pages.

HAMANN, A., JERSAK, M., RICHTER, K., AND ERNST, R. 2004. Design space exploration and system optimization with SymTA/S - symbolic timing analysis for systems. In *Proceedings of the 25th IEEE International Real-Time Systems Symposium (RTSS'04)*. IEEE Computer Society, 469-478.

HANSSON, A., WIGGERS, M., MOONEN, A., GOOSSENS, K., AND BEKOOIJ, M. J. G. 2009. Enabling application-level performance guarantees in network-based systems on chip by applying dataflow analysis. *IET* Comput. Digital Tech. 3, 5, 398 - 412.

HANSSON, A. AND GOOSSENS, K. 2010. *On-chip Interconnect with Aelite: Composable and Predictable Systems*, Springer.

HARY, S. L. AND OZGUNER, F. 1997. Feasibility test for real-time communication using wormhole routing. *IEE Proc. Comput. Digital Tech.* 144, 5, 273-278.

HORSTMANNSHOFF, J., GROTKER, T., AND MEYR, H. 1997. Mapping multirate dataflow to complex RT level hardware models. In *Proceedings of the IEEE International Conference on Application-Specific Systems, Architectures and Processors (ASAP'97)*. 283-292.

HU, P.-C. AND KLEINROCK, L. 1997. An analytical model for wormhole routing with finite size input buffers. In *Proceedings of the 15th International Teletraffic Congress*. 549-560.

HU, J., OGRAS, U. Y. AND MARCULESCU, R. 2006. System-level buffer allocation for application-specific networks-on-chip router design. *IEEE Trans. Computer-Aided Design Integr. Circuits Syst.* 25, 12, 2919-2933.

HUR, J. Y., GOOSSENS, K., AND MHAMDI, L. 2008. Performance analysis of soft and hard single-hop and multi-hop circuit-switched interconnects for FPGAs. In *Proceedings of the IFIP International Conference on Very Large Scale Integration*, 224-229.

JACKSON, J. R. 1957. Networks of waiting lines. *Operations Research*, 5, 518-521.

JAFARI, F., LU, Z., JANTSCH, A., AND YAGHMAEE, M. H. 2010. Buffer optimization in network-on-chip through flow regulation. *IEEE Trans. Computer-Aided Design Integr. Circuits Syst.* 29, 12, 1973-1986.

JANTSCH, A., AND SANDER, I. 2005. Models of computation and languages for embedded system design. *IEE Proceedings of Computers and Digital Techniques*, 152, 2, 114-129.

JIANG, Y. AND LIU, Y. 2008. *Stochastic Network Calculus*, Spriger-Verlag.

JIANG, Y. 2009. Network calculus and queueing theory: two sides of one coin. In *Proceedings of the 4th International Conference on Performance Evaluation Methodologies and Tools*, article 37, 12 pages.

JOSEPH, M. AND PANDYA, P. 1986. Finding response times in a real-time system. *The Computer J. - British Computer Society*, 29, 5, 390-395.

KANDLUR, D. D., SHIN, K. G., AND FERRARI, D. 1994. Real-time communication in multihop networks. *IEEE Trans. Parallel Distrib. Syst.* 5, 10, 1044-1056.

KIASARI, A. E., SARBAZI-AZAD, H., AND OULD-KHAOUA, M. 2008a. An accurate mathematical performance model of adaptive routing in the star graph. *Future Generation Computer Systems,* 24, 6, 461-474.

KIASARI, A. E., RAHMATI, D., SARBAZI-AZAD, H. AND HESSABI, S. 2008b. A Markovian performance model for networks-on-chip. In *Proceedings of the 16th Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP'08)*, 157-164.

KIASARI, A. E., SARBAZI-AZAD, H., AND HESSABI, S. 2008c. Caspian: a tunable performance model for multi-core systems, In *Proceedings of the 14th European Conference on Parallel and Distributed Computing (Euro-Par'08)*, Lecture Notes in Computer Science, vol. 5168, 100-109.

KIASARI, A. E., HESSABI, S., AND SARBAZI-AZAD, H. 2008d. PERMAP: A performance-aware mapping for application-specific SoCs. In *Proceedings of the International Conference on Application-Specific Systems, Architectures and Processors (ASAP'08)*. IEEE Computer Society, 73-78.

KIASARI, A. E., JANTSCH, A., AND LU, Z. 2010. A framework for designing congestion-aware deterministic routing. In *Proceedings of the 3rd International Workshop on Network-on-Chip Architectures (NoCArc'10)*. ACM Press, 45-50.

KIASARI, A. E., LU, Z., AND JANTSCH, A. 2012. An analytical latency model for networks-on-chip. *Accepted for publication in the IEEE Trans. on Very Large Scale Integration (VLSI) Systems*. doi: 10.1109/TVLSI.2011.2178620.

KIM, J. AND DAS, C. R. 1994. Hypercube communication delay with wormhole routing. *IEEE Trans. Comput.* 43, 7, 806-814.

KIM, B., KIM, J., HONG, S. J., AND LEE, S. 1998. A real-time communication method for wormhole switching networks. In *Proceedings of the International Conference on Parallel Processing (ICPP'98)*. IEEE Computer Society, 527-534.

KIM, J., PARK, D., NICOPOULOS, C., VIJAYKRISHNAN, N., AND DAS, C. R. 2005. Design and analysis of an NoC architecture from performance, reliability and energy perspective. In *Proceedings of the ACM Symposium on Architecture for Networking and Communications Systems (ANCS'05)*. ACM Press, 173-182.

KIM, H. AND HOU, J. C. 2009. Enabling network calculus-based simulation for TCP congestion control. *Comput. Netw.* 53, 11-24.

KLEINROCK, L. 1975. *Queueing Systems, vol. 1*, John Wiley.

KRIMER, E., KESLASSY, I., KOLODNY, A., WALTER, I., AND EREZ, M. 2011. Static timing analysis for modeling QoS in networks-on-chip. *J. Parallel Distrib. Comput.* 71, 5, 687-699.

KUMAR, A., MESMAN, B., THEELEN, B. D., CORPORAAL, H., HA, Y. 2008. Analyzing composability of applications on MPSoC platforms. *Journal of Systems Architecture - Embedded Systems Design*. 54, 3-4, 369-383.

LAUWEREINS, R., WAUTERS, P., ADE, M., PEPERSTRAETE, J. A. 1994. Geometric parallelism and cyclo-static data flow in GRAPE-II. In *Proceedings of the International Workshop on Rapid System Prototyping*, 90-107.

LE BOUDEC J.-Y. AND THIRAN, P. 2001. *Network Calculus: A Theory of Deterministic Queuing Systems for the Internet*. Springer-Verlag.

LEE, E. A. AND MESSERSCHMITT, D. G. 1987a. Synchronous data flow. *Proceedings of the IEEE*, 75, 9, 1235-1245.

LEE, E. A. AND MESSERSCHMITT, D. G. 1987b. Static scheduling of synchronous data flow programs for digital signal processing. *IEEE Trans. Comput.* 36, 1, 24-35.

LEE, E. A. 1991. Consistency in dataflow graphs. *IEEE Trans. Parallel Distrib. Syst.* 2, 2, 223-235.

LEE, E. A. AND PARKS, T. M. 1995. Dataflow process networks. *Proceedings of the IEEE*, 83, 5, 773-799.

LEHOCZKY, J. P., SHA, L., AND DING, Y. 1989. The rate monotonic scheduling algorithm: exact characterization and average case behavior. In *Proceedings of the IEEE Real-Time Systems Symposium*, 166-171.

LEHOCZKY, J. P. 1990. Fixed priority scheduling of periodic task sets with arbitrary deadlines. In *Proceedings of the IEEE Real Time Systems Symposium*, 201-209.

LEUNG, J. Y. T. AND WHITEHEAD, J. 1982. On the complexity of fixed priority scheduling of periodic, real-time tasks. *Performance Evaluation*, 2, 4, 237-250.

LI, J.-P. AND MUTKA, M. W. 1994. Priority based real-time communication for large scale wormhole networks. In *Proceedings of the 8th International Symposium on Parallel Processing*, IEEE Computer Society, 433-438.

LI, J.-P. AND MUTKA, M. W. 1996. Real-time virtual channel flow control. *J. Parallel Distrib. Comput.* 32, 1, 49-65.

LINDLEY, D. V. 1952. The theory of queues with a single server. *Proceedings of the Cambridge Philosophical Society.* 48, 2, 277-289.

LIU C. L. AND LAYLAND, J. W. 1973. Scheduling algorithms for multiprogramming in a hard-real-time environment. *J. ACM* 20, 1, 46-61.

LIU, J. W. S. 2000. *Real-Time Systems (1st ed.)*. Prentice Hall.

LU, Z., JANTSCH, A., AND SANDER, I. 2005. Feasibility analysis of messages for on-chip networks using wormhole routing. In *Proceedings of the Asia and South Pacific Design Automation Conference (ASP-DAC'05)*. ACM Press, 960-964.

LU, Z., MILLBERG, M., JANTSCH, A., BRUCE, A., VAN DER WOLF, P., AND HENRIKSSON, T. 2009. Flow regulation for on-chip communication. In *Proceedings of the Design, Automation and Test in Europe Conference (DATE'09)*. 578-581.

LU. Z. 2011. Cross clock-domain TDM virtual circuits for networks on chips. In *Proceedings of the 5th ACM/IEEE International Symposium on Networks-on-Chip (NoCS'11)*. ACM Press, 209-216.

MANABE, Y. AND AOYAGI, S. 1995. A feasibility decision algorithm for rate monotonic scheduling of periodic real-time tasks. In *Proceedings of the Real-Time Technology and Applications Symposium (RTAS'95)*. IEEE Computer Society, 212-218.

MIN, G. AND OULD-KHAOUA, M. 2004. A performance model for wormhole-switched interconnection networks under self-similar traffic. *IEEE Trans. Comput.* 53, 5, 601-613.

MOREIRA, O. M. AND BEKOOIJ, M. J. G. 2007. Self-timed scheduling analysis for real-time applications. *EURASIP J. Advances in Signal Processing*, 2007, id: 083710.

NELSON, A., HANSSON, A., CORPORAAL, H., GOOSSENS, K. 2010. Conservative application-level performance analysis through simulation of MPSoCs. In *Proceedings of the 8th IEEE Workshop on Embedded Systems for Real-Time Multimedia,* 51-60.

ODONI, A. AND ROTH E. 1983. An empirical investigation of the transient behavior of stationary queueing systems. *Oper. Res.* 31, 432-455.

OGRAS, U. Y., BOGDAN, P., AND MARCULESCU, R. 2010. An analytical approach for network-on-chip performance analysis. *IEEE Trans. Comp.-Aided Des. Integ. Cir. Sys.* 29, 12, 2001-2013.

OH D. AND BAKKER, T. P. 1998. Utilization bounds for n-processor rate monotone scheduling with static processor assignment. *Real Time Systems J.*, 15, 1, 183-193.

OH, Y. AND SON, S. H. 1995. Allocating fixed-priority periodic tasks on multiprocessor systems. *Real-Time Syst.* 9, 3, 207-239.

PANDIT, K., SCHMITT, J., AND STEINMETZ, R. 2004. Network calculus meets queueing theory - a simulation based approach to bounded queues. In *Proceedings of the IEEE International Workshop on Quality of Service*, pp. 114-120.

PARHI, K. K. 1989. Algorithm transformation techniques for concurrent processors. *Proceedings of the IEEE*, 77, 12, 1879-1895.

PENG, D.-T. AND SHIN, K. G. 1993. A new performance measure for scheduling independent real-time tasks. *J. Parallel Distrib. Comput.* 19, 1, 11-26.

POP, P., ELES, P., AND PENG, Z. 2000. Schedulability analysis for systems with data and control dependencies. In *Proceedings of the 12th Euromicro Conference on Real-Time Systems (RTS'00)*. IEEE Computer Society, 201-208.

QIAN, Y., LU, Z., AND DOU, W. 2009a. Applying network calculus for performance analysis of self-similar traffic in on-chip networks. In *Proceedings of the 7th IEEE/ACM International Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS'09)*. ACM Press, 453-460.

QIAN, Y., LU, Z., AND DOU, W. 2009b. Analysis of worst-case delay bounds for best-effort communication in wormhole networks on chip. In *Proceedings of the 3rd ACM/IEEE International Symposium on Networks-on-Chip (NOCS'09)*. IEEE Computer Society, 44-53.

QIAN, Y., LU, Z., AND DOU, W. 2009c. Worst case flit and packet delay bounds in wormhole networks on chip. *IEICE Trans. Fundamentals of Electronics, Communications and Computer Sciences, Special Section on VLSI Design and CAD Algorithms*, E92-A, 12, 3211-3220.

QIAN, Y., LU, Z., AND DOU, W. 2010a. Analysis of worst-case delay bounds for on-chip packet-switching networks. *IEEE Trans. Comp.-Aided Des. Integ. Cir. Sys.* 29, 5, 802-815.

QIAN, Y., LU, Z., AND DOU, W. 2010b. QoS scheduling for NoCs: strict priority queueing versus weighted round robin. In *Proceedings of the 28th International Conference on Computer Design (ICCD'10)*, 52-59.

RUMBAUGH. J. 1977. A data flow multiprocessor. *IEEE Trans. Comput.* 26, 2, 138-146.

SATHAYE, S. S. AND STROSNIDER, J. K. 1994. A real-time scheduling framework for packet-switched networks. In *Proceedings of the IEEE International Conference on Distributed Computing*, IEEE Computer Society, 182-191.

SCHERRER, A., FRABOULET, A., AND RISSET, T. 2009. Long-range dependence and on-chip processor traffic. *Microprocess. Microsyst.* 33, 1, 72-80.

SCHMITT, J. B. AND ROEDIG, U. 2005. Sensor network calculus: a framework for worst case analysis. *Distributed Computing in Sensor Systems,* Lecture Notes in Computer Science, vol. 3560, Springer, 141-154.

SCHMITT, J. 2003. On average and worst case behaviour in non-preemptive priority queueing. In *Proceedings of the International Symposium on Performance Evaluation of Computer and Telecommunication Systems*. 197-204.

SHA, L., LEHOCZKY, J. P., AND RAJKUMAR, R. 1986. Solutions for some practical problems in prioritized preemptive scheduling. In *Proceedings of the IEEE Real-Time Systems Symposium.*181-191.

SHE, H., LU, Z., JANTSCH, A., ZHOU, D., AND ZHENG, L. 2009. Analytical evaluation of retransmission schemes in wireless sensor networks. In *Proceedings of the 69th IEEE Vehicular Technology Conference* (*VTC'09-Spring)*.

SHI, Z. AND BURNS, A. 2008. Real-time communication analysis for on-chip networks with wormhole switching. In *Proceedings of the 2nd ACM/IEEE International Symposium on Networks-on-Chip (NOCS'08)*. IEEE Computer Society, 161-170.

SHI, Z. AND BURNS, A. 2009. Real-time communication analysis with a priority share policy in on-chip networks. In *Proceedings of the 21st Euromicro Conference on Real-Time Systems (ECRTS)*. IEEE Computer Society, 3-12.

SHI, Z. AND BURNS, A. 2010. Schedulability analysis and task mapping for real-time on-chip communication. *Real-Time Systems*. 46, 3, 360-385.

SJODIN, M. AND HANSSON, H. 1998. Improved response-time analysis calculations. In *Proceedings of the IEEE Real-Time Systems Symposium (RTSS'98)*. IEEE Computer Society, 399-409.

SONG, H., KWON, B., AND YOON, H. 1997. Throttle and preempt: a new flow control for real-time communications in wormhole networks. In *Proceedings of the international Conference on Parallel Processing (ICPP'97)*. IEEE Computer Society, 198-202.

SOTERIOU, V., WANG, H., AND PEH, L.-S. 2006. A statistical traffic model for on-chip interconnection networks. In *Proceedings of the 14th IEEE International Symposium on Modeling, Analysis, and Simulation (MASCOTS'06)*. IEEE Computer Society, 104-116.

SRIRAM, S., AND BHATTACHARYYA, S. S. 2009. *Embedded Multiprocessors: Scheduling and Synchronization* (2nd ed.). CRC Press.

STILIADIS, D. AND VARMA, A. 1998. Latency-rate servers: a general model for analysis of traffic scheduling algorithms. *IEEE/ACM Trans. Netw.* 6, 5, 611-624.

STUIJK, S., GEILEN, M. C. W., AND BASTEN, T. 2006. Exploring trade-offs in buffer requirements and throughput constraints for synchronous dataflow graphs. In *Proceedings of the 43rd Annual Design Automation Conference (DAC'06)*. ACM Press, 899-904.

STUIJK, S., GEILEN, M. C. W., THEELEN, B. D., AND BASTEN, T. 2011. Scenario-aware dataflow: modeling, analysis and implementation of dynamic applications. In *Proceedings of the International Conference on Embedded Computer Systems,* 404-411.

THEELEN, B. D. GEILEN, M. C. W., BASTEN, T., VOETEN, J. P. M., GHEORGHITA, S. V., AND STUIJK, S. 2006. A Scenario-aware data flow model for combined long-run average and worst-case performance analysis. In *Proceedings of the International Conference on Formal Methods and Models for Co-Design,* 185-194.

H. R. VAN AS. 1986. Transient analysis of Markovian queueing systems and its application to congestion-control modeling. *IEEE J. Selected Areas in Communications*. 4, 6, 891-904.

VARATKAR, G. V. AND MARCULESCU, R. 2004. On-chip traffic modeling and synthesis for MPEG-2 video applications. *IEEE Trans. Very Large Scale Integr. Syst.* 12, 1, 108-119.

WANG, J., LI, Y., AND PENG, Q. 2011. A novel analytical model for network-on-chip using semi-Markov process. *Advances in Electrical and Computer Engineering*, 11, 1, 111-118.

WIGGERS, M. H., BEKOOIJ, M. J. G., AND SMIT, G. J. M. 2007a. Modeling run-time arbitration by latency-rate servers in dataflow graphs. In *Proceedings of the International Workshop on Software and Compilers for Embedded Systems*, 11-22.

WIGGERS, M. H., BEKOOIJ, M. J. G., AND SMIT, G. J. M. 2007b. Efficient computation of buffer capacities for cyclo-static dataflow graphs. In *Proceedings of the 44th Annual Design Automation Conference (DAC'07)*. ACM Press, 658-663.

WIGGERS, M. H., BEKOOIJ, M. J. G., AND SMIT, G. J. M. 2008. Buffer capacity computation for throughput constrained streaming applications with data-dependent inter-task communication. In *Proceedings of the IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS'08)*. IEEE Computer Society, 183-194.

WIGGERS, M. H., BEKOOIJ, M. J. G., AND SMIT, G. J. M. 2011. Buffer capacity computation for throughput-constrained modal task graphs. *ACM Trans. Embed. Comput. Syst.* 10, 2, article 17, 59 pages.

WU, J., LIU, J.-C., AND ZHAO, W. 2010. A general framework for parameterized schedulability bound analysis of real-time systems. *IEEE Trans. Comput.* 59, 6, 776-783.

XUAN, D., BETTATI, R., CHEN, J., ZHAO, W., AND LI, C. 2000. Utilization-based admission control for real-time applications. In *Proceedings of the International Conference on Parallel Processing (ICPP'00)*. IEEE Computer Society, 251-262.

YANG, F. AND LIU, J. 2010. Transient analysis of general queueing systems via simulation-based transfer function modeling, In *Proceedings of the Winter Simulation Conference (WSC)*. IEEE Computer Society, 1110-1122.

ZHANG, H. 1995. Service disciplines for guaranteed performance service in packet-switching networks. *Proceedings of the IEEE*, 83, 10, 1374-1396.

A Heuristic Framework for Designing and Exploring Deterministic Routing Algorithm for NoCs

Abbas Eslami Kiasari
Axel Jantsch
Zhonghai Lu

M. Palesi and M. Daneshtalab, editors, Routing Algorithms in Networks-on-Chip, Springer, 2013, ISBN 978-1-4614-8273-4.

Paper 11

A Heuristic Framework for Designing and Exploring Deterministic Routing Algorithm for NoCs

Abbas Eslami Kiasari, Axel Jantsch, and Zhonghai Lu

Abstract In this chapter, we present a system-level framework for designing minimal deterministic routing algorithms for Networks-on-Chip (NoCs) that are customized for a set of applications. To this end, we first formulate an optimization problem of minimizing average packet latency in the network and then use the simulated annealing heuristic to solve this problem. To estimate the average packet latency we use a queueing-based analytical model which can capture the burstiness of the traffic. The proposed framework does not require virtual channels to guarantee deadlock freedom since routes are extracted from an acyclic channel dependency graph. Experiments with both synthetic and realistic workloads show the effectiveness of the approach. Results show that maximum sustainable throughput of the network is improved for different applications and architectures.

1. INTRODUCTION

Thanks to high performance and low power budget of ASICs (application specific integrated circuits), they have been common components in the design of embedded systems-on-chip. Advances of semiconductor technology facilitate the integration of reconfigurable logic with ASIC modules in embedded systems-on-chip. Reconfigurable architectures are used as new alternatives for implementing a wide range of computationally intensive applications, such as DSP, multimedia and computer vision applications [1]. In the beginning of the current millennium, network-on-chip (NoC) emerged as a standard solution in the on-chip architectures [8][10][11]. In network-based systems, the performance of the communication infrastructure is critical, as it can represent the overall system performance bottleneck. The performance of networks depends heavily on the routing algorithm effectiveness, since it impacts all network metrics such as latency, throughput, and power dissipation.

Routing algorithms are generally categorized into deterministic and adaptive. A deterministic routing algorithm is oblivious of the dynamic network conditions and always provides the same path between a given source and destination pair. In contrast, in adaptive routing algorithms, besides source and destination addresses, network traffic variation plays an important role for selecting channels to forward packets. However, adaptive routing may cause packets to arrive out-of-order since they may be routed along different paths. The re-order buffers needed at the destination for ordering the packets impose large area and power on system [13]. Deterministic routers not only are more compact and faster than adaptive routers [4], but also guarantee in-order packet delivery. Therefore, it is not surprising that designers would like to use deterministic routing algorithms in the NoCs which suffer from limited silicon resources. However, in deterministic routing a packet cannot use alternative paths to avoid congested channels along its route; this leads to degraded performance of the communication architecture at high levels of network throughput.

A well-designed routing algorithm utilizes the network resources uniformly as much as possible and avoids the congested channels, even in the presence of non-uniform traffic patterns, which are

A.E. Kiasari (✉) • A. Jantsch • Z. Lu
KTH Royal Institute of Technology,
Stockholm, Sweden
e-mail: kiasari@kth.se

usual in the embedded systems. In this paper, we propose a system-level Latency-Aware Routing (LAR) framework for designing minimal deterministic routing algorithms for network-based platforms. Especially, LAR is appropriate for reconfigurable embedded systems-on-chip which host several applications with high computational requirements and static workloads. To the best of our knowledge, the proposed framework is the first one to deal with traffic burstiness. Before the execution of a new application, the routing tables are configured with pre-computed routes, as well as other components in the system. After selecting the route and adding it to the packet, no further time is needed on routing at the intermediate nodes along the path. Due to advantages of table-based routing, it is one of the most widely used routing methods for implementing deterministic routing algorithm, e.g., IBM SP1 and SP2 [5].

LAR uses a recently proposed analytical model in [14] to calculate the average packet latency in the network. The results obtained from simulation experiments confirm that the proposed routing framework can find efficient routes for various networks and workloads.

The rest of the paper is organized as follows. We start by reviewing previous studies in Section 2. The proposed heuristic framework is proposed in Section 3. Experimental results in Section 4 show that our proposed approach can improve the system performance. Finally, concluding remarks are given in Section 5.

2. RELATED WORK

Turn model for designing partially adaptive routing algorithms for mesh and hypercube networks was proposed in [7]. Prohibiting minimum number of turns breaks all of the cycles and produces a deadlock-free routing algorithm. Turn model was used to develop the Odd-Even adaptive routing algorithm for meshes [3]. This model restricts the locations where some turns can be taken so that deadlock is avoided. In comparison with turn model, the degree of routing adaptivity provided by the Odd-Even routing is more even for different source-destination pairs.

DyAD routing scheme, which combines deterministic and adaptive routing, is proposed in [9] for NoCs, where the router works in deterministic mode when the network is not congested, and switches to adaptive mode when the network becomes congested. In [17] the authors extend routers of a network to measure their load and to send appropriate load information to their direct neighbors. The load information is used to decide in which direction a packet should be routed to avoid hot-spots. Recently, the authors in [14] present APSRA, a methodology to develop adaptive routing algorithms for NoCs that are specialized for an application or a set of concurrent applications. APSRA exploits the application-specific information regarding pairs of cores that communicate and other pairs that never communicate in the NoC platform to maximize communication adaptivity and performance.

Since all of these approaches are based on adaptive routing, they suffer from out-of-order packet delivery. Our proposed routing framework overcomes this problem while it minimizes the average packet latency across the network.

An application-aware oblivious routing is proposed in [11] that statically determines deadlock-free routes. The authors presented a mixed integer-linear programming approach and a heuristic approach for producing routes that minimize maximum channel load. However, in case of realistic workload, they did not study the effect of task mapping on their approach. Also, we have addressed the congestion-aware routing problem in [15]. With the analysis technique, we first estimated the congestion level in the network, and then embedded this analysis technique into the loop of optimizing routing paths so as to find deterministic routing paths for all traffic flows while minimizing the congestion level in the network. Since this framework cannot capture the traffic burstiness, in this work we utilize an analytical model [14] to deal with traffic burstiness.

3. LAR FRAMEWORK

The LAR framework consists of 5 steps as its flowchart is shown in Figure 1. At first, we represent the architecture and application using *topology graph* (TG) and *communication graph* (CG), respectively. Then we construct the *channel dependency graph* (CDG) based on TG and CG. In the

third step, an acyclic CDG is extracted by deleting some edges from CDG to guarantee the deadlock freedom. After that, we find all possible shortest paths for each flow to create the routing space. Finally, we formulate an optimization problem over the routing space and solve it. In the following subsections, each step is described in detail.

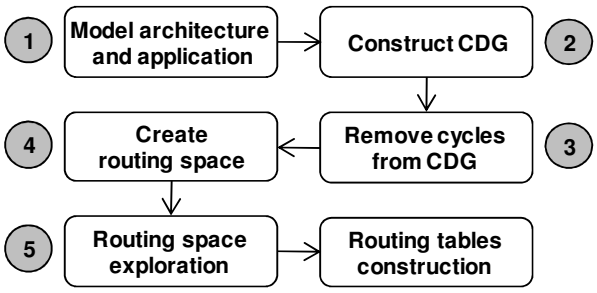


Figure 1: The flowchart of LAR framework.

3.1 Model Architecture and Application

In order to characterize network performance, a network model is essential. As shown in Figure 2, a directed graph, which is called *Topology Graph* (*TG*), can represent the topology of an NoC architecture. Vertices and edges of TG show nodes and links of the NoC, respectively. Every node in TG contains a core and a wormhole-switched router. Such cores are local computing or storage regions, which may contain a processor, a DSP core, a configurable hardware, a high-bandwidth I/O controller, or a memory block. Each core is equipped with a Resource Network Interface (RNI). The RNI translates data between cores and routers by packing/unpacking data packets and also manages the packet injection process. Packets are injected into the network on injection channel and leave the network from ejection channel. We consider the general reference architecture for routers [7], where a routing logic determines the output channel over which the packet advances. Routing is only performed with the head flit of a packet. After routing phase, a crossbar switch handles the connections among input and output channel.

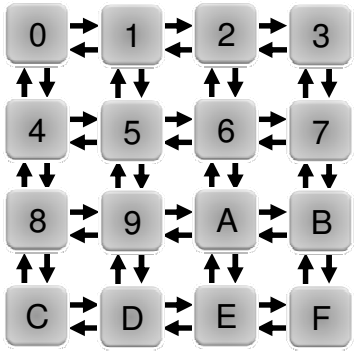


Figure 2: TG of a 4X4 mesh network.

An application can be modeled by a graph called *Communication Graph* (CG). CG is a directed graph, where each vertex represents one selected task, and each directed arc represents the communication volume from source task to destination task. As an example, the CG of a video object plane decoder (VOPD) is shown in Figure 3 [18]. Each node in the CG corresponds to a task and the numbers near the edges represent the bandwidth (in MBytes/sec) of the data transfer, for a 30 frames/sec MPEG-4 movie with 1920×1088 resolution [18].

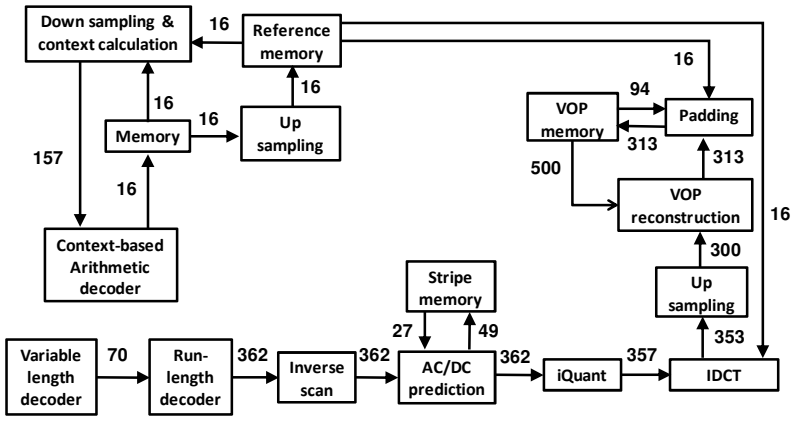


Figure 3: CG of a video object plane decoder (VOPD) application [18].

3.2 Construct Channel Dependency Graph

Dally and Seitz simplified designing deadlock-free routing algorithms with a proof that an acyclic channel dependency graph (CDG) guarantees deadlock freedom [5]. Each vertex of the CDG is a channel in TG. For instance, vertex 01 in Figure 4 corresponds to the channel from node 0 to node 1 in Figure 2. There is a directed edge from one vertex in CDG to another if a packet is permitted to use the second channel in TG immediately after the first one. To find the edges of a CDG, we use the *Dijkstra's algorithm* to find all shortest paths between source and destination of any flows in corresponding TG. CDG of a 4X4 mesh network (Figure 2) under minimal fully adaptive routing is shown in Figure 4.a, when any two nodes have the need to communicate such as in the uniform traffic pattern.

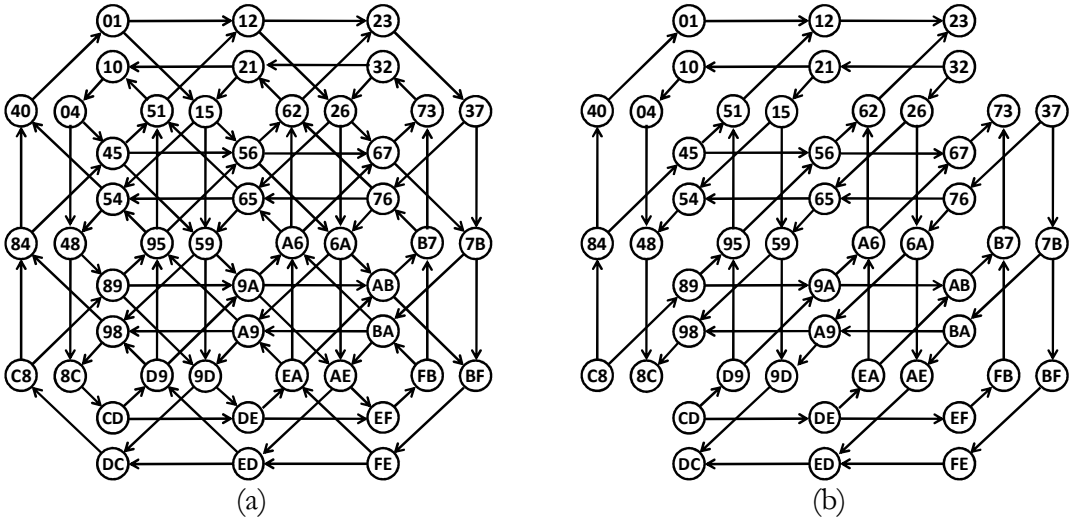


Figure 4: The CDG of 4X4 mesh network for minimal fully adaptive routing under (a) uniform and (b) transpose traffic patterns.

3.3 Remove Cycles from CDG

Traditional routing algorithms, such as *dimension-order routing* (DOR) and turn model, extract an acyclic CDG by systematically removing some edges from the CDG regardless of the traffic pattern. This may result in poor performance of routing algorithm due to prohibition of unnecessary turns. For instance, as shown in Figure 4.b, there is no cycle in CDG of 4X4 mesh network under transpose traffic pattern, which the node in row *i* and column *j* sends packets to the node in row *j* and column *i*. However, traditional routing algorithms conservatively remove some edges from the CDG.

We modify the *depth-first-search* (*dfs*) algorithm to find cycles in a given CDG. Since we want to remove minimum number of edges, we delete an edge from the CDG which is shared among more cycles. Note that, this edge is removed if the reachability of all flows is guaranteed. For example, in a CDG of 4x4 mesh network, shown in Figure 4.a, there are 6,982,870 cycles and the edge from vertex 40 to vertex 01 is shared among 5,041,173 cycles. Thus by removing this edge from the CDG, the number of cycles is considerably reduced to 1,941,697. These steps are repeated again while there is no cycle in the CDG. Table 1 shows the numbers of cycles found by LAR in the CDG of different mesh networks. As it can be vividly seen, number of cycles is exponentially grown with the size of the TG and it takes a long time to find all cycles in the CDG. Hence, we find cycles in the CDG till certain number of cycles, and then remove an edge from the CDG which is shared among more cycles.

Table1: Number of cycles in CDG of mesh networks.

TG	Number of cycles in corresponding CDG
Mesh (2x2)	2
Mesh (2x3)	8
Mesh (3x3)	292
Mesh (3x4)	14,232
Mesh (4x4)	6,982,870
Mesh (4x5)	3,656,892,444

3.4 Create Routing Space (RS)

In this step, we apply Dijkstra's algorithm to the acyclic CDG to find all shortest paths between source and destination of flows in corresponding TG and create a set of *f* flows $RS = \{F1, F2, \ldots, Ff\}$ where *f* is the number of all flows in the system. $Fi=\lambda i,CAi,ni,Pi$, where λi is the average packet generation rate and C_{A_i} is the coefficient of variation (CV) of packet interarrival time for flow *i*. We remind that the relationship between CV of random variable *X* and its moments is represented by $C_X^2 = \overline{x^2}/\bar{x}^2 - 1$. In [14], we show that CV of a random variable reflects the burstiness intensity very well. n_i is the number of available shortest paths for flow *i* and P_i is itself a set and includes all n_i routes for flow *i*.

Usually more than one shortest path is available between two nodes $(n_i > 1)$ in the routing space *RS*, so it is reasonable to choose a path such that the average packet latency is minimized. In the next subsection, we formulate an optimization problem over *RS* to find a suitable route for each flow and then use the simulated annealing heuristic to solve this problem.

3.5 Routing Space Exploration

3.5.1. Define Optimization Problem

In this subsection, we define an optimization problem to explore the routing space of *RS*. It is essential to define *decision variables* and *objective functions* in formulating an optimization problem. Our goal is to select a path for flow *i* $(1 \leq i \leq f)$ among n_i available paths to minimize the average packet latency. Therefore, we define $X = \{x_1, x_2, \ldots, x_f\}$ as decision variables in the space of *RS* where x_i refers to a path number for flow *i* $(1 \leq x_i \leq n_i)$ and the average packet latency as objective function.

The use of simulation experiments makes the task of searching for efficient designs computationally intensive and does not scale well with the size of networks since the search space of such a problem increases dramatically with the system size. Therefore, it is simply impossible to use the simulation in optimization loops. In the following subsection, we use an efficient analytical model to find nearly optimal solutions in reasonable time.

3.5.2. Analytical Latency Model

If the performance of a routing algorithm is measured in terms of average packet latency, then maximizing the performance means, in fact, minimizing the end-to-end packet latency. In this section, we briefly review a recently proposed analytical performance model which estimates the average packet latency in on-chip networks[14].

In a wormhole switched network, the end-to-end delay of a packet consists of two parts: the latency of the head flit and the latency of the body flits which follow the header flit in a pipelined fashion. The average latency of the head flit can be computed as the sum of delays at each hop, clearly, the link delays the head flit experienced and the residence times of the head flit in each of the routers along the path. Therefore, generally the only unknown parameter for computing the average packet latency is the mean waiting time for a packet from input channel *i* to output channel *j* in router *N* $\left(W_{i\to j}^{N}\right)$. Using a G/G/1 priority queueing model, we estimated this value by [14]

$$W_{i\to j}^{N}=\begin{cases}\dfrac{\rho_j^N\left(C_A^2+C_{S_j^N}^2\right)}{2\left(\mu_j^N-\lambda_{1\to j}^N\right)}, & i=1,\\[2ex] \dfrac{\lambda_j^N\left(C_A^2+C_{S_j^N}^2\right)}{2\left(\mu_j^N-\sum_{k=1}^{i-1}\lambda_{k\to j}^N\right)^2}, & 2\le i\le p\end{cases} \tag{1}$$

where the variables are listed in Table 2 along with other parameters used in this chapter.

Therefore, to compute the $W_{i\to j}^{N}$ we have to calculate the arrival rate from IC_i^N to OC_j^N $\left(\lambda_{i\to j}^N\right)$, and also first and second moments of the service time of OC_j^N $\left(\bar{s}_j^N,\overline{\left(s_j^N\right)^2}\right)$. In the following two subsections, packet arrival rate and channel service time are computed.

Assuming the network is not overloaded, the arrival rate from IC_i^N to OC_j^N can be calculated using the following general equation

$$\lambda_{i\to j}^N=\sum_S\sum_D\lambda^S\times P^{S\to D}\times R\left(S\to D,IC_i^N\to OC_j^N\right) \tag{2}$$

In Eq. (2), the routing function $R\left(S\to D,IC_i^N\to OC_j^N\right)$ equals 1 if a packet from IP^S to IP^D passes from IC_i^N to OC_j^N; it equals 0 otherwise. Note that we assume a deterministic routing algorithm, thus the function of $R\left(S\to D,IC_i^N\to OC_j^N\right)$ can be predetermined, regardless of topology and routing algorithm. After that, the average packet rate to OC_j^N can be easily determined as

$$\lambda_j^N=\sum_i\lambda_{i\to j}^N \tag{3}$$

After estimating the packet arrival rates, now we focus on the estimation of the moments of channel service times. At first, we assign a positive integer index to each output channel. Let D_j^N be the set of all possible destinations for a packet which passes through OC_j^N. The index of OC_j^N is equal to the maximum of distances among *N* and each M where $M\in D_j^N$. Obviously, the index of a channel is between 1 and diameter of the network. In addition, the index of all ejection channels is supposed to be 0. After that, all output channels are divided into some groups based on their index numbers, so that group k contains all channels with index *k*.

Determination of the channel service time moments starts at group 0 (ejection channels) and works in ascending order of group numbers. Therefore, the waiting time from lower numbered groups can then be thought of as adding to the service time of packets on higher numbered groups. In other words, to determine the waiting time of channels in group k, we have to calculate the waiting time of all channels in group $k-1$. This approach is independent of the network topology and works for all kinds of deterministic routing algorithm, whether minimal or non-minimal.

Table 2: Parameter notation.

t_r	Time spent for packet routing decision (*cycles*)
t_s	Time spent for switching (*cycles*)
t_w	Time spent for transmitting a flit between two adjacent routers (*cycles*)
m	Average size of packets (*flits*)
L_b	The latency of body flits
$L^{S\to D}$	Average packet latency from IP^S to IP^D (*cycles*)
L	Average packet latency in the network (*cycles*)
IP^N	The IP core located at address N
R^N	The router located at address N
IC_i^N	The ith input channel in router R^N
OC_j^N	The jth output channel in router R^N
IB_i^N	Capacity of the buffer in IC_i^N (*flits*)
OB_j^N	Capacity of the buffer in OC_j^N (*flits*)
$P^{S\to D}$	Probability of a packet is generated in IP^S and is delivered to IP^D $(\sum_S \sum_D P^{S\to D} = 1)$
λ^N	Average packet injection rate of IP^N (*packets/cycle*)
$\lambda_{i\to j}^N$	Average packet rate from IC_i^N to OC_j^N (*packets/cycle*)
λ_j^N	Average packet rate to OC_j^N (*packets/cycle*) $\left(\lambda_j^N = \sum_i \lambda_{i\to j}^N\right)$
$P_{i\to j}^N$	Probability of a packet entered form IC_i^N to be exited from OC_j^N
μ_j^N	Average service rate of the OC_j^N (*packets/cycle*)
$C_{S_j^N}$	Coefficient of variation (CV) for service time of the OC_j^N
C_A	CV for interarrival time of packets
ρ_j^N	The fraction of time that the OC_j^N is occupied
$W_{i\to j}^N$	Average waiting time for a packet from IC_i^N to OC_j^N (*cycles*)

In the ejection channel of R^N, the head flit and body flits are accepted in $t_s + t_w$ and L_b cycles, respectively. Therefore, we can write $\bar{s}_1^N = t_s + t_w + L_b$ and since the standard deviation of packet size is known, we can easily compute $C_{S_1^N}$. Now, by using Eq. (1), the waiting time of input channels for ejection channel, $W_{i\to 1}^N$, can be determined for all nodes in the network, where $2 \le i \le p$.

Although the moments of service time can be computed simply for all ejection channels, service time moments of the other output channels cannot be computed in a direct manner by a general formula, and we have to use a more complicated approach. Now, we can estimate the first moment or average service time of OC_i^M as

$$\bar{s}_i^M = \sum_{k=1}^{q} P_{j\to k}^N \left(t_s + t_w + t_r + W_{j\to k}^N + \bar{s}_k^N - \left(IB_j^N + OB_k^N\right) \times max(t_s, t_w)\right) \quad (4)$$

$$\overline{(s_i^M)^2} = \sum_{k=1}^{q} P_{j\to k}^N \left(t_s + t_w + t_r + W_{j\to k}^N + \bar{s}_k^N - \left(IB_j^N + OB_k^N\right) \times max(t_s, t_w)\right)^2 \quad (5)$$

where $P_{j\to k}^N$ is the probability of a packet entered form IC_j^N to be exited from OC_k^N and equals

$$P_{j\to k}^N = \lambda_{j\to k}^N / \lambda_i^M \quad (6)$$

Here, we should remind that to calculate $\bar{s}_i^M$ and $\overline{(s_i^M)^2}$ all values of $\bar{s}_k^N$ $(1 \le k \le q)$ must be computed before. Finally, the CV of channel service time for OC_i^M can be given by

$$C_{S_i^M}^2 = \overline{(s_i^M)^2}/(\bar{s}_i^M)^2 - 1 \quad (7)$$

Now, we are able to compute the average waiting time of all output channels using Eq. (1). After computing $W_{i\to j}^N$ for all nodes and channels, the average packet latency between any two nodes in the network, $L^{S\to D}$, can be calculated. The average packet latency is the weighted mean of these latencies.

$$L = \sum_S \sum_D P^{S\to D} \times L^{S\to D} \quad (8)$$

where $P^{S\to D}$ is the probability of a packet is generated in IP^S and is delivered to IP^D. LAR framework uses the simulated annealing heuristic to minimize the average packet latency L as described briefly in the next subsection.

3.5.3 Simulated Annealing

Simulated Annealing is a stochastic computational method for solving the global optimization problem in a large search space. It is often used when the search space is discrete. For instance, simulated annealing has been applied to some computer-aided design (CAD) problems such as module placement [21] and packet routing [15]. For such problems, simulated annealing may be more efficient than exhaustive enumeration. While simulated annealing is unlikely to find the optimum solution, it can often find an acceptably good solution in a fixed amount of time. Simulated annealing was independently proposed as an optimization technique in 1983 [12] and 1985 [3]. This technique stems from thermal annealing in metallurgy which aims to increase the size of crystals and reduce their defects by heating a material and then slowly lowering the temperature to give atoms the time to attain the lowest energy state.

To simulate the physical annealing process, the simulated annealing algorithm starts with an initial solution and then at each iteration, a trial solution is randomly generated. The algorithm accepts the trial solution if it lowers the objective function (better solution), but also, with a certain probability, a trial solution that raises the objective function (worse solution). Usually the Metropolis algorithm [2] is used as the acceptance criterion in which worse solution are allowed using the criterion that

$$e^{-\Delta E/T} > R(0,1), \quad (9)$$

where ΔE is the difference of objective function with current and trial solutions (negative for a better solution; positive for a worse solution), T is a synthetic temperature, and $R(0,1)$ is a random number in the interval [0,1]. Typically this step is repeated until the system reaches a state that is good enough for the application, or until a given computation budget has been exhausted. By accepting worse solutions, the algorithm avoids being stuck at a local minimum in early iterations and is able to explore globally for better solutions. Detailed information about simulated annealing approach can be found in [12].

As mentioned in subsection 3.5.1, objective function is the average packet latency and decision variables are represented by the routing set $X = \{x_1, x_2, \dots, x_f\}$ where x_i is the path number for flow i $(1 \le x_i \le n_i)$. Let $X = \{x_1, x_2, \dots, x_r, \dots, x_f\}$ be the initial routing set. To choose a trial routing set $X_{new} = \{x_1, x_2, \dots, x_r^{new}, \dots, x_f\}$, we generate a random number r where $1 \le r \le f$ to choose a flow, and then generate another random number x_r^{new} where $1 \le x_r^{new} \le n_r$ and $x_r^{new} \ne x_r$ to choose another path for flow r. Using analytical model describe in subsection 3.5.2, we estimate the average packet latency for current and trial routing set.

4. Experimental Results

To evaluate the capability of the proposed framework, we developed a discrete-event simulator that mimics the behavior of routing algorithm in the networks at the flit level. Due to the popularity of the mesh network in NoC domain, our analysis focuses on this topology but LAR framework can be equally applied for other topologies without any change. We compare the performance of LAR with DOR which becomes XY routing algorithm in 2D mesh networks.

To achieve a high accuracy in the simulation results, we use the batch means method [13] for simulation output analysis. There are 10 batches and each batch includes 1000 up to 1,000,000 packets depending on the workload type, packet injection rate, and network size. Statistics gathering was inhibited for the first batch to avoid distortions due to the startup transient. The standard deviation of latency measurements is less than 1.8% of the mean value. As a result, the confidence level and confidence interval of simulation results are 0.99 and 0.02, respectively.

For the sake of comprehensive study, numerous validation experiments have been performed for several combinations of workload types and network size. In what follows, the capability of LAR will be assessed for both synthetic and realistic traffic patterns. Since their applications differ starkly in purpose, these classes of NoC have substantially different traffic patterns.

4.1 Synthetic Traffic

Synthetic traffic patterns used in this research include *uniform, transpose, shuffle, bit-complement,* and *bit-reversal* [4]. After developing models describing spatial traffic distributions, we should use an appropriate model to model the temporal traffic distribution. In the case of synthetic traffics, we use the Poisson process for modeling the temporal variation of traffic. It means that the time between two successive packet generations in a core is distributed exponentially. The Poisson model widely used in many performance analysis studies, and there are a large number of papers in many application domains that are based on this stochastic assumption.

The average packet latencies in the 4X4 and 8X8 mesh networks are plotted against offered load in the network in Figure 5 and Figure 6, respectively. We observe that under uniform and bit-complement traffic patterns LAR converges to DOR, because in such traffic patterns the standard deviation of channels throughput is minimum for DOR. This result is consistent with other results reported in [3][7][9][14]. The main reason is that the DOR distributes packets evenly in the long term [7]. Previous works, Odd-Even [3], turn model [7], DyAD [9], and APSRA [14] indicate that in the case of uniform traffic, their proposed approaches underperform DOR. However, as can be seen in Figure 5.a and 6.a, our proposed framework has the same performance as DOR for different traffic loads.

Figure 5.b and 5.c compare the latency of DOR and LAR in 4X4 mesh network under transpose and bit-reversal workloads, respectively. It can be vividly seen that LAR considerably outperforms DOR. Also, in the case of 8x8 mesh network, LAR has better performance than DOR as shown in Figure 6.b and 6.c. Figure 5.d and 6.d reveal that under shuffle traffic pattern LAR slightly outperforms DOR. Table 3 shows the maximum sustainable throughput of the network for each workload and for each routing algorithm in 4X4 and 8X8 mesh networks. It also shows the percentage improvement of LAR over DOR and reveals that on average LAR outperforms DOR. The maximum load that the network is capable of handling using LAR is improved by up to 205%.

Also, the performance of LAR framework is compared against DyAD routing scheme [9] which combines deterministic and adaptive routing algorithms. We simulate the uniform and transpose workloads on the similar architecture (6X6 mesh network) and compare their improvement over DOR. Table 4 shows the percentage improvement of DyAD and LAR over DOR. In case of uniform workload, DyAD underperforms DOR while LAR has the same performance as DOR. In case of transpose traffic pattern, DyAD and LAR give about 62% and 60% improvement over DOR, respectively. This means that our deterministic routing policy can compete with adaptive routing policies (DyAD switches to adaptive mode under high traffic load) and meanwhile guarantees in-order packet delivery.

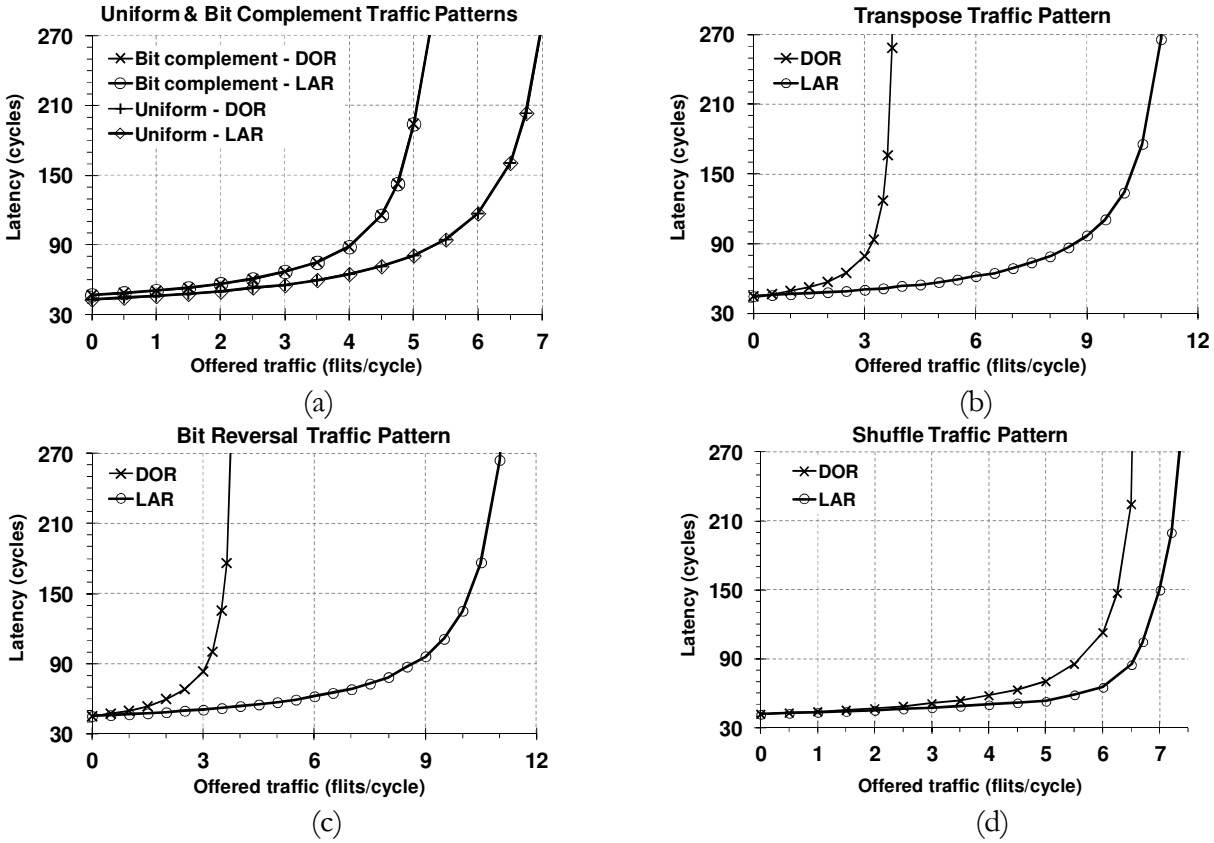


Figure 5: Average packet latency under (a) uniform and bit-complement, (b) transpose, (c) bit-reversal, and (d) shuffle traffic patterns in 4x4 mesh network.

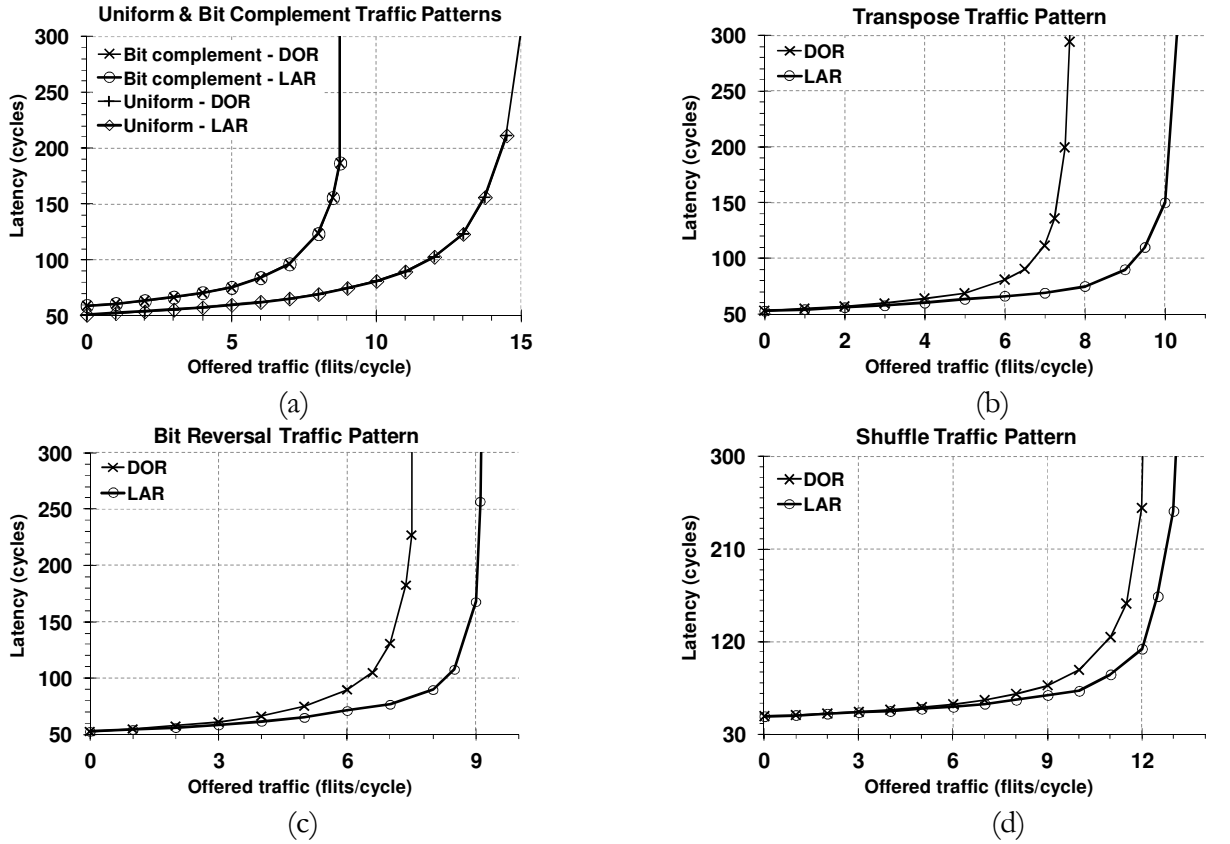


Figure 6: Average packet latency under (a) uniform and bit-complement, (b) transpose, (c) bit-reversal, and (d) shuffle traffic patterns in 8x8 mesh network.

Table 3: Improvement in maximum sustainable throughput of LAR as compared to DOR for different synthetic workloads.

Workload	4X4 mesh network			8X8 mesh network		
	DOR	LAR	Impr.	DOR	LAR	Impr.
Uniform	7.4	7.4	0	15.9	15.9	0
Transpose	3.8	11.6	205%	7.7	10.5	36%
Bit-comp.	5.6	5.6	0	8.8	8.8	0
Bit-rev.	3.8	11.6	205%	7.6	9.2	21%
Shuffle	6.6	7.4	12%	12.2	13.4	10%

Table 4: Improvement in maximum sustainable throughput of DyAD and LAR over DOR.

Workload	Improvement over DOR	
	DyAD	LAR
Uniform	-21%	0
Transpose	62%	60%

4.2 Realistic Traffic

In case of realistic traffic, we consider two virtual channels for links to show the consistency of proposed framework with multiple virtual channel routing. As realistic communication scenarios, we consider a generic multimedia system (MMS) and the video object plane decoder (VOPD) application. MMS includes an H.263 video encoder, an H.263 video decoder, an mp3 audio encoder, and an mp3 audio decoder [10]. The communication volume requirements of this application are summarized in Table 5. VOPD is an application used for MPEG-4 video decoding and its communication graph is shown in Figure 3. Several studies reported the existence of bursty packet injection in the on-chip interconnection networks for multimedia traffic [16][19].

Table 5: MMS application traffic requirement [10].

src	*dst*	*vol.*	*src*	*dst*	*vol.*
ASIC1	ASIC2	25	DSP2	DSP1	20363
ASIC1	DSP8	25	DSP3	ASIC4	38016
ASIC2	ASIC3	764	DSP3	DSP6	7061
ASIC2	MEM2	640	DSP3	DSP5	7061
ASIC2	ASIC1	80	DSP4	DSP1	3672
ASIC3	DSP8	641	DSP4	CPU	197
ASIC3	DSP4	144	DSP5	DSP6	26924
ASIC4	DSP1	33848	DSP6	ASIC2	28248
ASIC4	CPU	197	DSP7	MEM2	7065
CPU	MEM1	38016	DSP8	DSP7	28265
CPU	MEM3	38016	DSP8	ASIC1	80
CPU	ASIC3	38016	MEM1	ASIC4	116873
DSP1	DSP2	33848	MEM1	CPU	75205
DSP1	CPU	20363	MEM2	ASIC3	7705
DSP2	ASIC2	33848	MEM3	CPU	75584

Poisson process is not the appropriate model in case of bursty traffic; consequently, we used Markov-modulated Poisson process (MMPP) model as stochastic traffic generators to model the bursty nature of the application traffic [4][8]. MMPP has been widely employed to model the traffic burstiness in the temporal domain [8]. Figure 7 shows a two-state MMPP in which the arrival traffic follows a Poisson process with rate λ_0 and λ_1. The transition rate from state 0 to 1 is r_0, while the rate from state 1 to state 0 is r_1.

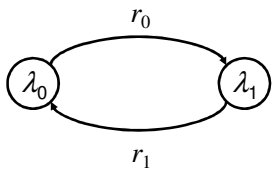
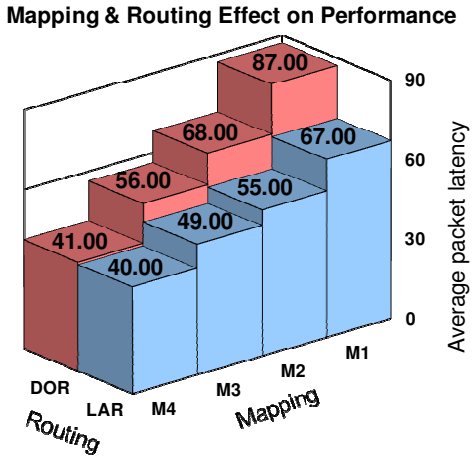


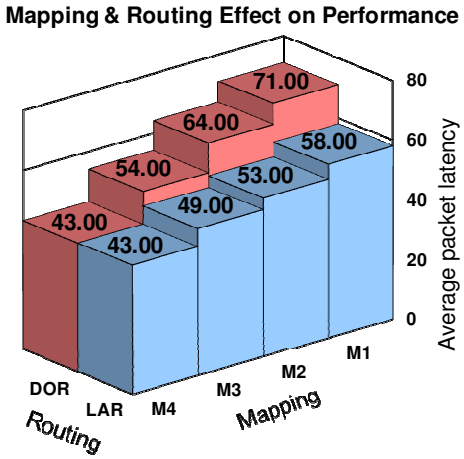
Figure 7: Two-state MMPP model

Since in such systems, there are various types of cores with different bandwidth requirements, placement of tasks on a chip has strong effect on the system performance. To find a suitable mapping of these applications, we formulate another optimization problem to prune the large design space in a short time and then again use the simulated annealing heuristic to find a suitable mapping vector. Initially, we map task *i* to node *i* and then try to minimize the average packet latency through the simulated annealing approach. Figure 8.a shows that in the case of MMS application and DOR, for the initial mapping M1, average packet latency equals 87 and after a certain number of tries, the mapping vector converges to the mapping M4 with average packet latency = 41. Furthermore, average packet latency values for mappings M2 and M3, which are two local minimum points in simulated annealing process, are shown in the figure.

After the mapping phase, we apply the LAR framework to these four mapping vectors. Figure 8.a reveals that in case of mapping M1, LAR can significantly reduce the average packet latency from 87 to 67. However, for more efficient mapping vectors (M2, M3, and M4), we achieve less improvement. Specially, in the case of best mapping (M4), average packet latency is reduced insignificantly from 41 to 40. It is reasonable that DOR is latency-aware for the best mapping, because during the mapping problem solving process, we fix the routing policy to DOR and strive to minimize average packet latency for this routing policy. Likewise, as shown in Figure 8.b, for the VOPD application, the analysis result is the same as MMS application.



(a)



(b)

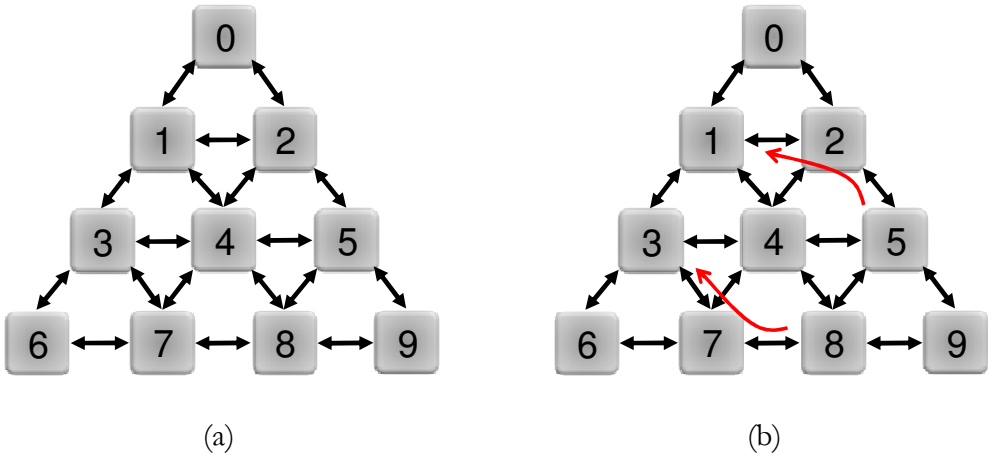
Figure 8: The effect of mapping and routing on the performance of (a) MMS application and (b) VOPD application.

Figure 8 reveals that in case of application-specific traffic patterns, the improvement in the performance of the routing schemes highly depends on how the application tasks are mapped to the topology. This fact was not considered in the related works such as [16]. Nowadays, in embedded systems-on-chip there are several different types of cores including DSPs, embedded DRAMs, ASICs, and generic processors which their places are fixed on the chip. On the other hand, such a system hosts several applications with completely different workload. Furthermore, modern embedded devices allow users to install applications at run-time, so a complete analysis of such systems is not feasible during design phase. As a result, it is not feasible to map all applications such that the load is balanced for all of them with specific routing algorithm and we should balance the load in routing phase.

In this section we used the LAR framework to find low latency routes in the mesh network. Due to simplicity, regularity, and low cost merits of 2D mesh topology, it is the most popular one in the field of NoC. However, for large and 3D NoCs, which will be popular in the future, the communication in mesh architecture takes a long time. In the next subsection we use LAR to find deadlock-free paths in an arbitrary topology.

4.3 Find Routes in an Arbitrary Topology

To show the capability of LAR framework to find deadlock-free routes in an arbitrary topology, we consider the topology shown in Figure 9.a. LAR reports that under uniform traffic pattern there are 2 cycles in the corresponding CDG and by prohibiting turns 52 to 21 and 87 to 73 (shown in Figure 9.b) the deadlock-freedom is guaranteed.



(a) (b)

Figure 9: (a) A custom topology and (b) prohibited turns.

Table 6 shows the routing table for node 0 of the topology in Figure 9.a. Each route in the table specifies a path from node 0 to a given destination as channels name. SE, SW, and EJ specify South East, South West, and ejection channels, respectively. To route a packet, the routing table is indexed by destination address to look up the pre-computed route by LAR. This route is then added to the packet. Since there are 7 channels in this network (E, S, NE, NW, SE, SW, and EJ), they can be encoded as 3-bit binary numbers. Also, there are techniques to reduce the size of routing tables [4][14].

Table 6: Routing table for node 0 of topology in Figure 8.a.

dst.	route	dst.	Route
0	No packet	5	SE, SE, EJ
1	SW, EJ	6	SW, SW, SW, EJ
2	SE, EJ	7	SE, SW, SW, EJ
3	SW, SW, EJ	8	SW, SE, SE, EJ
4	SW, SE, EJ	9	SE, SE, SE, EJ

5. Conclusion

On-chip packet routing is extremely crucial because it heavily affects performance and power. This calls for a great need of routing optimization. However, due to the diverse connectivity enabled by a network and the interferences in sharing network buffers and links, determining good routing paths, which are minimal and deadlock free for traffic flows, is nontrivial. In this paper, we have addressed the latency-aware routing problem. Using an analytical model, we first estimate the average packet latency in the network, and then embed this analysis technique into the loop of optimizing routing paths so as to quickly find deterministic routing paths for all traffic flows while minimizing the latency. Our experiments with both synthetic and realistic workloads show that we can extract high quality solutions with small computational time.

The proposed framework is appropriate for reconfigurable embedded systems-on-chip which run several applications with regular and repetitive computations on large set of data, e.g., multimedia and computer vision applications. LAR can not only design minimal and deterministic routing, but also can implement non-minimal routing without virtual channels in arbitrary topology.

Reference

[1] K. Bondalapati and V.K. Prasanna, "Reconfigurable Computing Systems," *Proceedings of the IEEE*, 90(7):1201-1217, 2002.

[2] O. Catoni, "Metropolis, Simulated Annealing, and Iterated Energy Transformation Algorithms, Theory and Experiments," *Journal of Complexity* 12(4):595-623, 1996.

[3] V. Cerny, "Thermodynamical Approach to the Traveling Salesman Problem: an Efficient Simulation Algorithm," *J. Opt. Theory Appl.*, vol. 45, pp. 41-45, 1985.

[4] G.-M. Chiu, "The Odd-Even Turn Model for Adaptive Routing," *IEEE Transactions on Parallel and Distributed Systems*, 11(7):729-738, 2000.

[5] W. J. Dally and B. Towles, Principles and Practices of Interconnection Networks, Morgan Kaufmann Publishers Inc., First edition, 2004.

[6] W.J. Dally and C.L. Seitz, "Deadlock-Free Message Routing in Multiprocessor Interconnection Networks," *IEEE Transactions on Computers*, 36(5):547-553, 1987.

[7] J. Duato, C. Yalamanchili, and L. Ni, "Interconnection Networks: An Engineering Approach," IEEE Computer Society Press, 2003.

[8] W. Fischer and K. Meier-Hellstern, "The Markov-Modulated Poisson Process (MMPP) Cookbook", *Performance Evaluation*, vol. 18, no. 2, pp. 149-171, 1993.

[9] C.J. Glass and L.M. Ni, "The Turn Model for Adaptive Routing," *Journal of the Association for Computing Machinery*, 41(5):874-902, 1994.

[10] P. Guerrier and A. Greiner, "A Generic Architecture for on-chip Packet-Switched Interconnections," *Proceedings of the Design, Automation, and Test in Europe*, pp. 250-256, 2000.

[11] A. Hemani, *et. al.*, "Network on a Chip: An Architecture for Billion Transistor Era," *Proceedings of the IEEE NorChip*, pp. 166-173, 2000.

[12] J. Hu and R. Marculescu, "DyAD - Smart Routing for Networks-on-Chip," *Proceedings of the Design Automation Conference,* pp. 260-263, 2004.

[13] J. Hu and R. Marculescu, "Energy- and Performance-Aware Mapping for Regular NoC Architectures," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 24(4):551-562, 2005.

[14] A. E. Kiasari, Z. Lu and A. Jantsch, "An Analytical Latency Model for Networks-on-Chip," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 21, no. 1, pp. 113-123, Jan. 2013.

[15] A. E. Kiasari, A. Jantsch and Z. Lu, "A Framework for Designing Congestion-Aware Deterministic Routing" In the Proceedings of the 3rd International Workshop on Network-on-Chip Architectures (NoCArc), Held in conjunction with the 43rd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO-43), pp. 45-50, 2010.

[16] M. A. Kinsy, *et. al.*, "Application-Aware Deadlock-free Obli- vious Routing," *Proceedings of the ISCA*, pp. 208-219, 2009.

[17] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. "Optimization by Simulated Annealing," *Science*, 220(4598):671–680, 1983.

[18] S. Murali, *et. al.,* "Analysis of Error Recovery Schemes for Networks on Chips", *IEEE Design and Test of Computers*, 22(5): 434-442, 2005.

[19] M. Palesi, *et. al.,* "Application Specific Routing Algorithms for Networks on Chip," *IEEE Transactions on Parallel and Distributed Systems*, 20(3):316-330, 2009.

[20] K. Pawlikowski, "Steady-State Simulation of Queueing Processes: A Survey of Problems and Solutions," *ACM Computing Surveys*, 22(2):123-170, 1990.

[21] C. Sechen and A. Sangiovanni-Vincentelli, "The TimberWolf placement and routing package," Journal for Solid State Circuits," vol. SC-20, pp. 510-522, 1985.

[22] V. Soteriou, H. Wang, L.-S. Peh, "A Statistical Traffic Model for On-Chip Interconnection Networks," *Proceedings of the MASCOTS*, pp. 104-116, 2006.

[23] W. Trumler, *et. al.*, "Self-optimized Routing in a Network-on-a-Chip," *IFIP World Computer Congress*, pp. 199-212, 2008.

[24] E.B. van der Tol and E.G. Jaspers, "Mapping of MPEG-4 Decoding on a Flexible Architecture Platform," *SPIE*, vol. 4674, pp. 1-13, 2002.

[25] G. Varatkar and R. Marculescu, "Traffic Analysis for On-chip Networks Design of Multimedia Applications," *Procee- dings of the Design Automation Conference,* pp. 795-800, 2002.