ENERGY AND BUFFER AWARE APPLICATION MAPPING FOR NETWORKS ON CHIP

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

COŞKUN ÇELİK

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF DOCTOR OF PHILOSOPHY
IN
ELECTRICAL AND ELECTRONICS ENGINEERING

FEBRUARY 2013

Approval of the thesis:

ENERGY AND BUFFER AWARE APPLICATION MAPPING FOR NETWORKS ON CHIP

submitted by **COŞKUN ÇELİK** in partial fulfillment of the requirements for the degree of **Doctor of Philosophy in Electrical and Electronics Engineering Department, Middle East Technical University** by,

Prof. Dr. Canan Özgen _____
Dean, Graduate School of **Natural and Applied Sciences**

Prof. Dr. İsmet Erkmen _____
Head of Department, **Electrical and Electronics Engineering**

Assoc. Prof. Dr. Cüneyt F. Bazlamaçcı _____
Supervisor, **Electrical and Electronics Engineering Dept., METU**

Examining Committee Members:

Prof. Dr. Semih Bilgen
Electrical and Electronics Engineering Dept., METU _____

Assoc. Prof. Dr. Cüneyt F. Bazlamaçcı
Electrical and Electronics Engineering Dept., METU _____

Prof. Dr. Müslim Bozyiğit
Computer Engineering Dept., METU _____

Assoc. Prof. Dr. Ece Schmidt
Electrical and Electronics Engineering Dept., METU _____

Assist. Prof. Dr. Süleyman Tosun
Computer Engineering Dept., Ankara University _____

Date: 15.02.2013

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last Name: COŞKUN ÇELİK

Signature :

ABSTRACT

ENERGY AND BUFFER AWARE APPLICATION MAPPING FOR NETWORKS ON CHIP

ÇELİK, COŞKUN

Ph.D., Department of Electrical and Electronics Engineering

Supervisor : Assoc. Prof. Dr. Cüneyt F. Bazlamaçcı

February 2013, 81 pages

Network-on-Chip (NoC) is a developing and promising on-chip communication paradigm that improves scalability and performance of System-on-Chips. NoC design flow contains many problems from different areas, for example networking, embedded design and computer architecture. Application mapping is one of these problems, which is generally considered as a communication energy minimization problem. This dissertation approaches to this problem from a networking point of view and tries to find a mapping solution which improves the network performance in terms of the number of packets in the buffers while still minimizing the total communication energy consumption. For this purpose an on-chip network traffic model is required. Self similarity is a traffic model that is used to characterize Ethernet and/or wide area network traffic, as well as most of on-chip network traffic. In this thesis, by using an on-chip traffic characterization that contains self similarity, an application mapping problem definition that contains both energy and buffer utilization concerns is proposed. In order to solve this intractable problem a genetic algorithm based model is implemented. Execution of the algorithm on different test cases has proved that such a mapping formulation avoids high buffer utilizations while still keeping the communication energy low.

Keywords: Networks-on-chip, application mapping, self similar traffic, genetic algorithms

ÖZ

YONGA ÜZERİ AĞLAR İÇİN ENERJİ VE ARABELLEĞE DUYARLI UYGULAMA EŞLEŞTİRME

ÇELİK, COŞKUN

Doktora, Elektrik ve Elektronik Mühendisliği Bölümü

Tez Yöneticisi : Doç. Dr. Cüneyt F. Bazlamaçcı

Şubat 2013 , 81 sayfa

Yonga üzeri ağlar, yonga üzeri sistemlerin ölçeklenebilirliğini ve performansını artıran yeni ve gelişen bir yonga üzeri haberleşme yöntemidir. Tasarım aşamasında karşılaşılan uygulama eşleştirme problemi genellikle toplam haberleşme enerjisi azaltma problemi olarak çözülür. Bu çalışma, problemi ağ tasarımı açısından ele alır ve toplam haberleşme enerjisini azaltırken arabelleklerdeki paket sayısı gibi ağ parametrelerini iyileştiren eşleştirme çözümleri bulmaya çalışır. Bunun için bir yonga üzeri trafik modeli gereklidir. Öz benzerlik, Ethernet trafiklerini olduğu gibi yonga üzeri haberleşmeyi de karakterize edebilen bir modeldir. Bu tezde yonga üzeri ağlar için bir öz benzer trafik modeli oluşturularak, hem enerji hem de arabellek kullanımı ölçütlerini içeren bir uygulama eşleştirme problem tanımı önerilmiş ve bu problemi çözmek için genetik algoritma temelli bir model geliştirilmiştir. Yapılan deneyler sonucunda bu problem tanımının haberleşme enerjisini düşük tutarken yüksek arabellek kullanımını önlediği görülmüştür.

Anahtar Kelimeler: Yonga üzeri ağlar, uygulama eşleştirme, öz benzer trafik, genetik algoritma

To my wife Ceyda.

ACKNOWLEDGMENTS

First of all, I would like to thank my advisor, Assoc. Prof. Dr. Cüneyt F. Bazlamaçcı for his guidance, advice, criticism and encouragements during my MSc and PhD studies. Without his inspiration, patience and friendship, this thesis would have never been possible.

I am also grateful to my thesis committee members Prof.Dr. Semih Bilgen, Prof.Dr. Müslim Bozyiğit, Assoc. Prof. Dr. Ece Schmidt, Asst. Prof. Dr. Süleyman Tosun for their insightful suggestions and comments on my thesis.

Most importantly, I would like to thank my parents Nursen and Süleyman Çelik, my brother M. Orkun Çelik and my wife Ceyda Er Çelik for their endless support throughout the thesis and all of my life. Without the encouragement, support and love of Ceyda, this thesis would have never been finished.

Finally, I would like to express my gratitude to TÜBİTAK-BİDEB for supporting this thesis under 2211 - PhD scholarship program.

TABLE OF CONTENTS

LIST OF TABLES

TABLES

LIST OF FIGURES

FIGURES

LIST OF ABBREVIATIONS

APCG	Application Characterization Graph
ASIC	Application Specific Integrated Circuit
BU_{max}	Maximum Buffer Utilization
DSP	Digital Signal Processor
EBAM	Energy and Buffer Aware Application Mapping
FARIMA	Fractional Autoregressive Integrated Moving Average
FBM	Fractional Brownian Motion
FFT	Fast Fourier Transform
FGN	Fractional Gaussian Noise
FPGA	Field Programmable Gate Arrays
GALS	Globally Asynchronous Locally Synchronous
ILP	Integer Linear Programming
IP	Intellectual Property
LAN	Local Area Network
LRD	Long Range Dependant
MBps	Megabytes per second
MILP	Mixed Integer Linear Programming
NF	Negative-first
NI	Network Interface
NIRGAM	NoC Interconnect RoutinG and Applications' Modeling
NL	North-last
NoC	Networks-on-Chip
PE	Processing Elements
QoS	Quality of Service
RMD	Random Midpoint Displacement
RTL	Register Transfer Language
SoC	Systems-on-Chip
WAN	Wide Area Network
WF	West-first

CHAPTER 1

INTRODUCTION

Networks-on-Chip (NoC) has been proposed as a novel communication paradigm for Systems-on-Chips (SoCs). NoC concept brings a networking perspective to on-chip communication and provides improvements in terms of performance, scalability and flexibility over traditional bus-based or point-to-point structures. An NoC architecture uses layered protocols and a packet-switched network, which consists of on-chip routers, links and network interfaces (NI) for connecting different processing elements (PEs) of an SoC. A PE can be a memory unit, an ASIC, an FPGA, a DSP or a general purpose processor core [2, 3]. A PE is also called an intellectual property (IP) core.

NoC has many advantages over traditional on-chip communication schemes. First one is the scalability. Since the communication is performed over a set of point-to-point links, increase in network size does not degenerate the system performance [1]. Besides, packet based communication makes concurrent transmissions possible, which increases the overall throughput of the communication system. By using NoC, processing elements of a SoC can have different clock frequencies. NIs are responsible for synchronization between processing elements and network. This paradigm, which is called Globally Asynchronous Locally Synchronous (GALS), brings some advantages like reducing power consumption and avoiding clock skews [4] [5].

Although it has many advantages, there are some difficulties of using NoC. The major one is network congestion, which increases the communication latency. This problem may be considered at different stages of NoC design flow. This study aims to avoid congestion problem at the application mapping step in a proactive manner.

A generic NoC design flow starts with the selection of the network topology. A NoC topology can be regular, such as mesh, torus or butterfly, or irregular which is tailored to the application under consideration. Once the topology is decided the application which is generally expressed as an IP core graph is mapped onto it by considering various constraints. Then, the routing protocol and the flow control mechanisms should be implemented. Microarchitecture of the routers and the network interfaces are some of the challenging design problems in NoC area.

1.1 Motivation

One of the major steps in NoC design flow is the placement of IP cores. This problem, called Application Mapping Problem in the literature, is generally solved to minimize the communication energy consumption or delay by minimizing the hop count between communicating cores.

Our computational studies on the effect of application mapping on network performance reveal that

mapping of cores by considering the energy consumption only may have a significant degenerative effect on buffer utilization. The results indicate that buffer utilization of mapping solutions, which have acceptable communication energy consumption, may differ significantly under self-similar traffic assumption. Even though the energy consumption is optimal, if a mapping solution cause one or more buffers to operate in congestion state, this will decrease the run time performance of the on-chip network.

The main motivation of this dissertation is to solve the application mapping problem by considering a quality of service point of view also, i.e. buffer utilization and end-to-end delay. Our aim is to avoid possible high input loads on buffers at the mapping stage by using a priori traffic characteristics of the application. Obviously, a traffic characterization of the application under concern is necessary for this purpose. Self similarity is already an accepted model in local and wide area networks. Many on-chip applications are also proved to have self-similar traffic. We therefore perform a simple buffer utilization analysis under self-similar traffic assumption for on-chip networks and formulate a novel application mapping solution.

1.2 Contributions

The contributions of the thesis work can be summarized as follows:

- An on-chip network traffic model is required to formulate a traffic aware application mapping. Besides, self similarity has been proved to model on-chip traffic accurately. Therefore, a buffer utilization model under self-similar traffic is implemented.

- The self similar traffic model contains three parameters. The effect of self-similar parameters on buffer utilization is analyzed. The results emphasize the necessity of a mapping formulation that addresses network dynamics besides energy.

- A novel application mapping formulation, which minimizes the energy consumption while considering the network dynamics, is proposed. This is the first study which aims to lower the buffer utilization at mapping stage by using a priori traffic information, while still minimizing the communication energy consumption. In order to solve this intractable problem, a genetic algorithm based solution is implemented. The energy overhead of the proposed algorithm is also analyzed.

- The model and the performance of the proposed mapping algorithm are verified by means of a simulation study. A self-similar traffic generator, which is consistent with the analytical model used in the problem formulation, is implemented. Execution of simulations showed that NoCs mapped with the proposed algorithm have better performance than the mapping algorithms in the literature, in terms of buffer utilization or end-to-end delay.

- The proposed method is evaluated under various routing algorithms and observed to improve the buffer utilization of the basic NoC routing algorithms.

Results of the computational and the simulation studies show that mappings found by the proposed algorithm have better performance in terms of buffer utilization and end-to-end delay compared to the ones found by energy-only mapping algorithms.

1.3 Thesis Organization

Outline of the thesis is as follows;

Chapter 2 presents the related work. It contains a brief introduction to NoC concepts. After defining NoC design problems, recent solutions to the application mapping problem are reviewed. The fundamental concepts of self similarity which is the underlying traffic model of this study is also presented in this chapter.

Chapter 3 introduces the energy and buffer aware application mapping concept by presenting the network and queueing analysis under self-similar traffic. Then the computational work performed to evaluate the effect of self-similar traffic parameters on performance of a NoC system is presented. The chapter concludes with the novel problem formulation of energy and buffer aware application mapping.

Chapter 4 contains the details of our genetic algorithm implemented to solve energy and buffer aware application mapping problem. It starts with a brief overview of genetic algorithms and then presents the implemented method to solve energy and buffer aware application mapping. It also presents an Integer Linear Programming based solution of the same problem.

In Chapter 5, the proposed application mapping problem formulation is evaluated under different routing protocols. A brief introduction to NoC routing protocols and computational study are also presented in this chapter.

Chapter 6 contains the simulation study performed to evaluate the proposed method. Generation of self-similar traffic traces for a open source NoC simulator is presented. Details of the simulator and extensions made on it are stated. The chapter concludes with simulation results and a discussion on them.

Finally, Chapter 7 concludes the dissertation and outlines possible future directions.

CHAPTER 2

BACKGROUND AND RELATED WORK

Networks-on-Chip has been a promising communication paradigm proposed for Systems-on-Chip after 2000. NoC design flow addresses many different problems from various areas like embedded systems, networking or computer architectures. This chapter presents basics and related works of NoC design flow. Especially, application mapping problem is studied in detail. Self similarity, which is used as underlying traffic model in this study is also presented.

2.1 Networks-on-Chip

The growth of the chip scale has changed the functions of a chip, marginally. Chips evolved from being a part of a system module (e.g. a bitslice in a bitslice processor) to a single module (e.g. a processor or memory unit) [1]. With todays semiconductor technology, a single chip can contain an entire system consisting of different processing elements or replicas of identical processing elements. This paradigm, which is called Systems-on-Chip (SoC), leads to new design problems like synchronization, power minimization, reusability or on-chip communication.

As the number of processing elements (which is also called Intellectual Property (IP) core) on a SoC increases, the volume of on-chip traffic also increases. Older on-chip communication schemes like point-to-point, bus or hierarchical bus cannot respond to this increasing traffic demand. Therefore, a packet-based communication paradigm, Networks-on-Chip (NoC) has been proposed in early 2000's [6, 7]. The main idea behind NoC is to transfer fundamental concepts of computer networking into chip and construct a relatively simple packet based network between IP cores of a SoC.

A generic NoC architecture consists of three basic components, namely Network Interface (NI), router and link. Each IP core connects to NoC through an NI. The network is composed of routers and links connecting those routers. Figure 2.1 shows a basic 2D grid NoC architecture and its basic components.

As mentioned before, NoC brings fundamental concepts of computer networking into chip. Of course, there are some differences in network conditions and design constraints of NoC and Computer Networks. Since embedded systems generally target battery-powered hand-held appliances, main design constraint of NoC is power consumption. Therefore NoC components (like NI or router) and protocols should be simple and have low power and area overheads. On the other hand, bandwidth of on-chip links are not restricted with the underlying network infrastructure, as in the case of computer networks. Adding more wires to an on-chip link may increase the bandwidth significantly (with power and area trade-off, of course) [8].

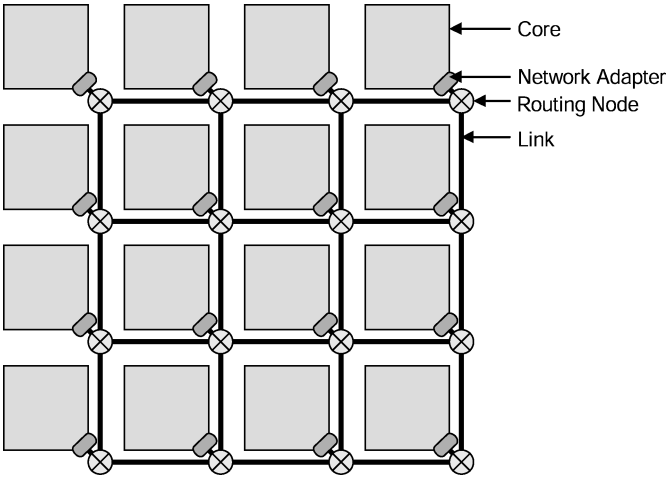


Figure 2.1: A generic 2D mesh NoC [1]

2.1.1 Basic NoC Design Issues

In this subsection, basic design issues of a generic NoC architecture are presented. First step of NoC design flow is to decide network topology. The network topology determines physical layout and connections between nodes. An NoC may have a regular topology like mesh, torus, tree or star, or an irregular topology which is generated to optimize a specific application [9, 10].

Once a topology is selected (or generated), a routing protocol is necessary to determine a path between communicating nodes. NoC routing protocols can be divided into three groups; deterministic, oblivious and adaptive [8]. Deterministic routing protocols are mostly used due to their simplicity. XY dimension-ordered routing is an example for deterministic routing, in which packets travel in X-dimension first and then in Y-dimension. In oblivious routing protocols, packets may travel in different paths. But path selection does not depend on network conditions. On the contrary, in adaptive routing, path of a transmission may change depending on the network congestion. Adaptive protocols are relatively difficult to implement and may require some additional control signaling between nodes.

The next issue to be considered is flow control. Flow control determines how packets are transferred in a router. Before presenting flow control schemes, packetization of NoC should be given. When an IP core has a *"message"* to send to another core, first, it is segmented into *"packets"* in NI. Each packet is tagged with a header field consisting destination IP's address and other necessary information. Then packets are divided into smaller flow control units, which are called *"flits"*. Only first flit (header flit) contains the addressing information of the packet. Flits are further divided into *"phits"*, which are physical units and correspond to width of the links. But in many NoC design phit size is equal to flit size. In another word, flit size is chosen as the data quantity that can be transferred in a single clock cycle [8]. Figure 2.2 shows composition of an NoC message.

There are three flow control mechanisms used in NoC architectures; *store-and-forward*, *virtual cut through* and *wormhole* . In *"store-and-forward"* flow control, a router waits until an entire packet is received, and then forwards packet to the next router. This mechanism requires buffer space for storing entire packet and has high latency since each node waits the entire packet before starting transmission. *"Virtual cut through"* flow control is similar to previous one, but in order to reduce latency a

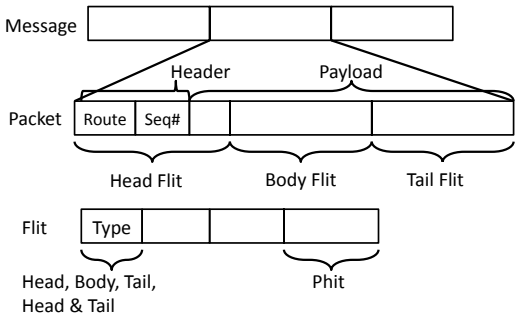


Figure 2.2: Message compositon for NoC

router can start transmission before receiving the whole packet, if there is enough space in next router. This scheme has relatively lower latency but it still requires high buffer space. To reduce buffer requirements, *"wormhole"* flow control mechanism has been proposed. It is a flit-based mechanism and states that a router forwards a flit as soon as there is enough space for that flit in the next router. Once the header flit is directed the body flits follow it. Wormhole flow control reduces the buffer requirements significantly. However, it may cause idle physical links, since a packet may span several routers.

2.2 Application Mapping Problem for Networks-on-Chip

NoC design problems are classified into four distinct areas in [11] and then fourteen different problems are defined in these areas from different perspectives such as networking, embedded design or computer architecture. Application mapping is one of these problems and is described as finding the optimal mapping of IP cores of the application under concern to routers of an on-chip network topology. In the above description, tasks of the application are assumed to be assigned and scheduled to a PE (IP Core) prior to the mapping stage. In other words, given an assigned and scheduled IP core graph, application mapping is the process of topological placement of these IP cores onto different tiles.

[12] is one of the first studies, which dealt with application mapping problem. The authors present a two step genetic algorithm based on a delay model for on-chip communication and an objective function that minimizes the overall execution time of the given task graph. Hence the paper addressed both application mapping and application scheduling problems simultaneously. The first step of the algorithm assigns each task in the task graph to an appropriate type of IP core using an average delay value for all edges and the second step of the algorithm maps the cores to the tiles of a 2D mesh topology by using actual delays.

Authors in [13] use a branch and bound algorithm to solve the application mapping problem from the perspective of communication energy minimization. An energy model which defines the required power to transmit a single bit through the network is used to construct the objective function. Authors extend their study in [14] in such a way that they obtain both the mapping of the cores and a deadlock free deterministic route for each edge of the IP core graph. Authors also propose a bandwidth reservation scheme to guarantee the design constraints.

A bandwidth-constrained application mapping problem is defined in [15] and a heuristic algorithm is

proposed to be applied to both mesh and torus network topologies. The paper also presents the novel idea of splitting traffic between communication pairs into smaller parts and transferring each part from a separate path. This avoids high bandwidth demand of single-path deterministic routing. This algorithm is also used in a NoC synthesis tool [16], named SUNMAP. This tool selects the best topology from a predefined topology set, and generates the mapping of the application onto that topology.

The application mapping problem is an important step of an NoC design flow. Therefore, some studies handle it with some other steps of the whole design process. For example, [17] presents an integrated approach that includes mapping of cores onto NoC topologies as well as physical planning of cores where the position and size of the cores and other network components are computed. Besides, an optimum topology is also selected from a topology library in this approach.

[18] proposes a heuristic to map IP cores of an application onto a mesh NoC topology and determine routes of traffics traces under both bandwidth and latency constraints. In [19] authors formulates custom NoC synthesis as a Mixed Integer Linear Programming (MILP) problem for minimizing power consumption of communication infrastructure. The method contains two steps. In the first step, the floorplan, i.e. the locations of cores and routers, are determined. By using the floorplan information, in the second step, a custom topology is generated and routes are determined. The paper also presents a cluster based heuristic for reducing run time of MILP formulation.

Almost all of the papers mentioned so far assume an ideal network infrastructure for on-chip communication, in which buffers in routers have infinite lengths and hence there is no packet drop and even no link contention. Such issues are generally addressed apart from the application mapping problem. A link contention aware application mapping problem has appeared only in [20]. The objective function in this paper has two parts; first one is a weighted communication distance term, which is a widely used objective for mapping problems, and a term emphasizing the link contention of on-chip communication. The authors claim that they place the IP Cores in a such way that there will be a minimum number of link contention between communication transactions, which will eventually decrease the packet latency.

[21] presents an approach to multi-objective exploration of the mapping space of a mesh-based networks-on-chip architecture by using a genetic algorithm based heuristic. The algorithm optimizes both performance and power consumption.

[22] presents a novel energy-aware scheduling algorithm which automatically assigns the application tasks onto different processing elements and then schedules their execution. Given the IP core graph and the NoC architecture, the method finds non-preemptive, static schedule which minimizes the communication energy consumption.

Temperature may have an impact on circuit behavior in terms of interconnect delay or leakage current. Therefore some studies handle NoC design problem from thermal-aware perspective. [23] proposes a method to obtain a thermal balanced placement of cores in order to reduce hotspot links or routers which cause higher temperature dissipation. [24] proposes a genetic algorithm based method for thermal and communication (in terms of hop count) aware mapping of cores to 3D NoC architectures. In [25] authors proposes ILP based mapping algorithms for 3D NoC architectures under thermal constraints.

[26] proposes a heuristic for mapping an application onto mesh topology and determining routing paths between cores with the objective function of communication energy minimization. The heuristic selects its initial cores with the help of *symmetric cores of a mesh topology* concept.

Table 2.1 summarizes the related work on application mapping problem in a tabular form. As seen from the table, there are no application mapping algorithm which assumes buffer utilization in its objective function and uses a traffic model, at the same time. The current study aims to address these issues. A detailed survey study on the application mapping problem of on-chip networks can also be found in [27].

Table2.1: Related work on Application Mapping of NoCs

	Inputs				Outputs					Objective Function					Models used					Methods			
Reference	IP Core Graph	Task Graph	Topology	Initial Mapping	Mapping	Schedule	Topology	Physical Plan of Cores	Routes	Execution Time	Energy / Power	Communication Delay	Area	Thermal balance	Delay	Energy	Area	Traffic	Thermal	Genetic Algorithm	Branch&Bound	Integer Linear Programing	Heuristic
[12]	X		X		X	X				X					X					X			
[13]		X		X	X						X					X					X		
[14]		X		X	X				X		X					X					X		
[15]	X		X		X							X			X								X
[16]	X		X		X		X				X	X	X				X						X
[17]	X		X		X		X	X				X						X				X	
[22]		X		X		X					X					X							X
[21]		X		X	X						X				X	X				X			
[23]		X	X			X		X				X		X					X	X			
[18]	X				X				X		X					X							X
[19]	X				X		X									X						X	
[20]	X		X		X							X										X	
[24]		X	X		X	X						X		X	X				X	X			
[25]		X	X		X						X			X		X			X			X	
[26]	X		X		X				X		X					X							X

2.3 Self Similarity

2.3.1 Definition

Self similarity can be defined as the phenomenon where a certain property of an object (e.g. a natural image or a mathematical time series) is preserved with respect to scaling in space and/or time [28]. If an object is self-similar, when its parts are zoomed in, they resemble the whole image. For example, in Figure 2.3 a two-dimensional (2D) Cantor set is shown. A 2D Cantor set is obtained by starting a single square, scaling it by 1/3, then placing four copies of the scaled square at the four corners of original size, and repeating this process to infinity. Self similarity is a typical property of fractals. Many objects in real world, such as coast lines or leaves of some kind of trees, show self similarity.

2D Cantor set (Figure 2.3) is an example for deterministic self similarity. But in the context of Network Traffic Engineering a more flexible form of similarity is necessary, which is called Stochastic Self Similarity.

Unlike deterministic one, the Stochastic Self Similarity does not possess an exact resemblance of their rescaled samples. Instead, that means the similarity of second-order statistics like autocovariance or autocorrelation functions. Second-order statistics are statistical properties that capture burstiness, variability or scale invariance. [13]

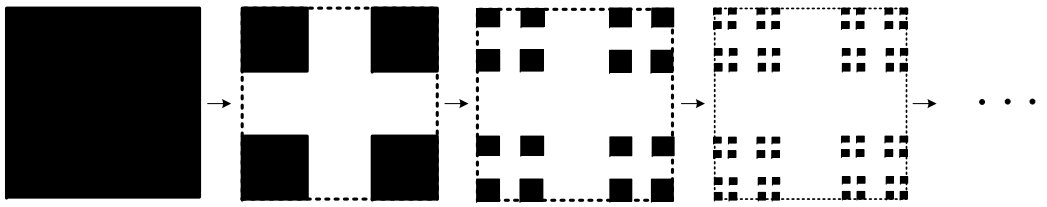


Figure 2.3: Two-Dimensional Cantor set

Although it is not possible to observe exact stochastic properties of given time series, Figure 2.4 gives an idea about the burstiness of the traffic. The top left graphic in Figure 2.4 shows a traffic trace, where time granularity is 100 sec, i.e. a single data point is the aggregated traffic volume over a 100 second interval. Top right figure gives the trace of the first 1000 sec of the same traffic. This time, the time granularity is 10 sec. The remaining two figure zoom in further on the initial segments by rescaling by factors of 10.

Formal definition for stochastic self similarity is given below:

Consider a discrete time stochastic process or time series, $X(t)$, $t \in Z$, where $X(t)$ is interpreted as the traffic volume measured in packets, bytes or bits, at time instance t.

Some properties and definitions on $X(t)$ are given below:

- Stationarity: $X(t)$ is assumed to be stationary in the sense that its behavior or structure is invariant with respect to shifts in time. In other words, $X(t)$ and the k-shifted process $X(t + k)$ have the same statistical properties.
- First moment, Mean: $\mu = E\{X(t)\}$
- Second moment, Variance: $\sigma^2 = E\left\{(X(t) - \mu)^2\right\}$
- Autocovariance Function:
$$C(r, s) = E\{(X(r) - \mu)(X(s) - \mu)\}$$
- Due to stationarity, $C(r, s) = C(r - s, 0) = C(k)$

Aggregate process $X^{(p)}$ of X at aggregation level p is defined as

$$X^{(p)}(i) = \frac{1}{p} \sum_{t=p.(i-1)+1}^{p.i} X(t) \tag{2.1}$$

In other words, $X(t)$ is partitioned into non-overlapping blocks of size p and their values are averaged while i is used to index these blocks. Let $C^{(p)}(k)$ denote the autocovariance function of $X^{(p)}$. Then the definition of second-order self similarity is as follows:

$X(t)$ is exactly second-order self-similar with Hurst parameter H, where $1/2 < H < 1$, if

$$C^{(p)}(k) = \frac{\sigma^2}{2}\left((k+1)^{2H} - 2k^{2H} + (k-1)^{2H}\right) = C(k) \tag{2.2}$$

and $X(t)$ is asymptotically second-order self-similar with Hurst parameter H, where $1/2 < H < 1$, if

$$\lim_{p\to\infty} C^{(p)}(k) = \frac{\sigma^2}{2}\left((k+1)^{2H} - 2k^{2H} + (k-1)^{2H}\right) = C(k) \tag{2.3}$$

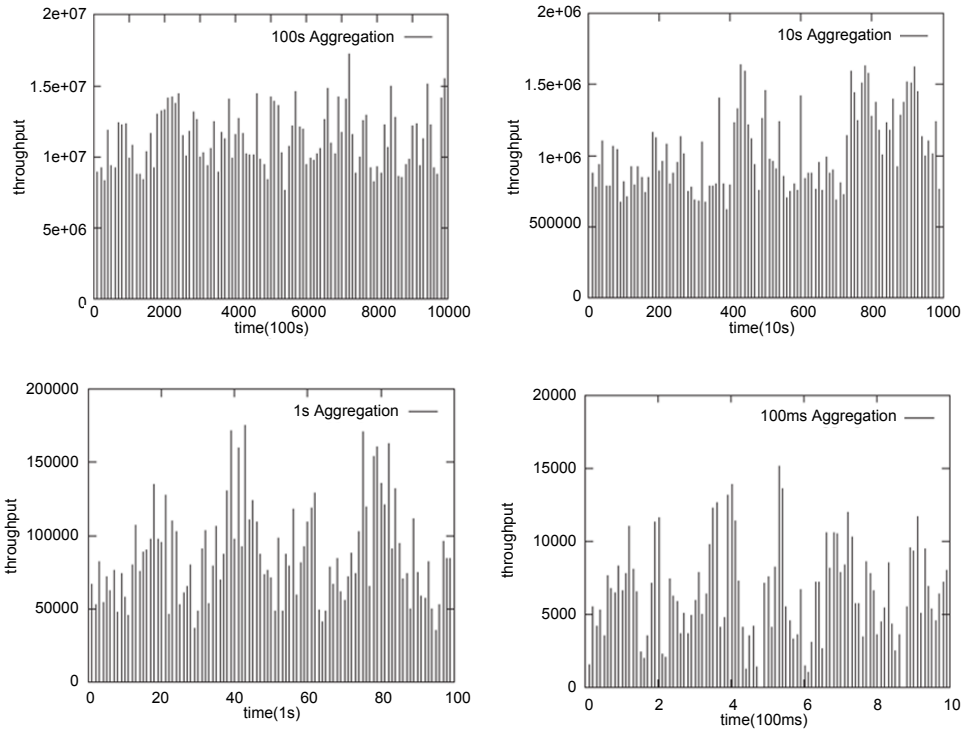


Figure 2.4: Example for Stochastic Self similarity in terms of burstiness preservation

2.3.2 Self Similarity in Network Traffic

Although the concept of self similarity goes back to 17th century, the modern definition of the term is given by Mandelbrot in 1975. In the second half of 1990s, researchers in computer networking area started to use self similarity widely after [29]. The study in [29] analyzes two real Ethernet local area network (LAN) traffic traces, recorded in Bellcore Laboratories in 1989 and 1992, and realizes that aggregating the traces at different time scales (i.e. from a few milliseconds to minutes and hours) does not smooth the traffic, which is the case in a Poisson process. In other words, burstiness is observed at all time scales. After this paper Poisson-like models, although being simple for mathematical derivations, are considered to be inaccurate for modeling local area and wide area network (WAN) traffics. Instead, mathematical models that contain self similarity are accepted to be more suitable for representing network traffic.

2.3.3 Self Similarity in NoC Traffic

Self similarity has also been proven to be a fundamental property of on-chip network traffic especially for multimedia applications. For example, in [30], authors showed the existence of self similarity in a typical MPEG-2 video application. In this study, the authors present methods for characterizing the degree of self similarity (i.e. Hurst parameter) and try to find optimal buffer length distributions under self-similar traffic, which is one of the main problems of router design for on-chip networks. Authors also deal with estimation of Hurst parameter by using statistical methods in [31].

[32] proposes an empirically-derived statistical traffic model for homogeneous on-chip networks. The

bursty nature of traffic is also represented in this work by the help of Hurst parameter and a method for generating synthetic self-similar on-chip network traffic is presented.

[33] extends the study in [30] by investigating on-chip network applications, which exhibit a self-similar characteristics. The authors, in [33], analyze the communication traces of processor IP cores at a cycle-accurate level and demonstrate the presence of self similarity. They also show that the impact of self similarity on networks-on-chip is highly correlated to the low level communication protocol used.

In [34] and [35] authors apply network calculus methods to analyze the delay and backlog bounds for self-similar traffic in networks-on-chips. They use the Fractional Brownian Motion model proposed by Norros [36] and also deal with estimation of self similarity parameters.

[37] shows the effects of Long Range Dependant (LRD) traffic (i.e. self-similar) on NoC performance by means of simulation. Authors utilize a Fractional Gaussian Noise (FGN) model to generate simulation traffic. The paper concludes with the statement the bursty behaviour of the LRD traffic has a strong impact on the NoC performance, and the greater the Hurst parameters is, the sooner network contention occurs.

In [38] authors propose a generic self-similar traffic model for NoC based on traffic traces obtained from system simulation or real system devices. They characterize the self-similar traffic with three parameters namely, Hurst parameter, injection rate and spatial distribution of the traffic. They compare the synthetic traffic with real NoC traffic.

In many papers, self-similar traffic is generated by using ON/OFF Pareto distributions, because of its simplicity. [39] show that aggregation of a large number of ON/OFF Pareto sources generates a self-similar traffic. By using this, authors perform a performance comparison of different NoC architectures under self-similar traffic. [40] proposes a new congestion-aware NoC architecture and evaluate it with a cycle accurate simulation platform with ON/OFF Pareto sources.

CHAPTER 3

ENERGY AND BUFFER AWARE APPLICATION MAPPING FOR NOC

Application mapping is one fundamental step in an NoC design flow and there exist a lot of different mapping algorithms in the literature. Most of these handle the application mapping problem as of minimizing the average hop count between communicating cores, which in effect means minimizing the communication energy consumption and the communication delay. Although it seems to be a reasonable approach, it is possible to note that such methods omit the dynamics of packet based network communications.

The effect of application mapping on NoC performance in terms of buffer utilization has been analyzed in [41]. It is shown in [41] that mapping of cores by considering only the energy consumption may have a significant degenerative effect on network performance in terms of buffer utilization of the NoC system under consideration. The mapping solutions, which have acceptable communication energy consumption, may differ significantly when number of packets in buffers is taken into account under self-similar traffic assumption.

Therefore our main motivation in this study is to propose, by using a priori traffic information, a cost function that minimizes the communication energy while also considering the number of packets waiting in buffers. The new formulation is likely to generate a mapping, which uses network resources fairly and decreases average network delay due to packet drops or high queue waiting times.

Formal definition of our energy and buffer aware application mapping problem is presented in the next subsections. A genetic algorithm based solution will be presented in the next chapter.

3.1 Network Analysis under Self-similar Traffic

The mathematical analysis of networks under self-similar traffic is more difficult than Poisson traffic because there is no simple, closed form formula, such as Little's theorem, for different network parameters. For modeling self-similar traffic (i.e. for queuing analysis or for generating synthetic traffic traces), different mathematical models, some of which are listed below, can be used [42]:

- Fractional Gaussian Noise (FGN)
- Fractional Brownian Motion (FBM)
- Fractional Autoregressive Integrated Moving Average (FARIMA)

- On/Off Processes
- Wavelet-based Models
- Markovian Model

Each model has some pros and cons in terms of simplicity or mathematical tractability. Within those, the Fractional Brownian Motion (FBM) model is used by Norros [43] for analyzing a network buffer under self-similar traffic. We review this analysis and present further results on buffer utilization in the following subsection.

There are some studies addressing loss probability of finite queues under self similar traffic. For example, [44] uses Markovian input model to approximate the loss probability, and [45] utilizes on/off processes models. In the current study, we use FBM model, which can express a self similar traffic by using only three parameters. To the best of our knowledge, there are no study dealing with loss probability of a finite queue under self similar traffic, by using FBM model. In the next subsection we derive a loss model for infinite buffers. This model can easily be used for formulating an application mapping problem, since it has only three distinct parameters. Using such a model can be interpreted as minimizing number of packets waiting in a very long buffer.

3.1.1 Buffer Utilization Analysis

In order to model a self-similar process, Norros [43] used a Fractional Brownian Motion process, $Z(t)$. Before presenting the self-similar process, it is necessary to state some properties of $Z(t)$ as below:

- $Z(t)$ has stationary increments
- Mean, $E\{Z(t)\} = 0$
- Variance, $E\{Z^2(t)\} = |t|^{2H}$, where H is the Hurst parameter.

After defining the Fractional Brownian Motion process, Z(t), a self-similar process A(t) can be given as follows:

$$A(t) = mt + \sqrt{ma}Z(t) \tag{3.1}$$

where $m > 0$ is the mean input rate, $a > 0$ is the variance coefficient. In this model, a self-similar traffic is described by the three parameters (m, a, H). $H \in [0.5, 1]$ is the Hurst (self similarity) parameter of the process $Z(t)$.

By using this model, Norros obtained two important results. One is the lower bound for the probability of the queue length exceeding a certain buffer size under the assumption of having an infinite buffer. Second is the effective bandwidth of a queue for satisfying a loss rate [36]. In this thesis, we used the former result to calculate the buffer utilization of an on-chip network.

Although self similarity has a considerable impact on queueing performance of an on-chip network, very few analytical results are available in the literature. Norros [43] has already used Fractional Brownian Motion (FBM) model to capture self similarity effects. This model finds out a lower bound

for the probability of the queue length Q to exceed a certain buffer size x under the assumption of having an infinite buffer. Mathematically,

$$P(Q > x) = e^{-K \cdot x^T} \tag{3.2}$$

where $K = \frac{(C-m)^{2H}}{2amH^{2H}(1-H)^T}$ and $T = 2 - 2H$

H, m and a are the parameters characterizing the traffic, C is the service rate of the queue.

Using (3.2), buffer occupancy probabilities (probability density function) can be written as

$$\begin{aligned} P(Q = 1) &= P(Q > 0) - P(Q > 1) \\ P(Q = 2) &= P(Q > 1) - P(Q > 2) \\ &... \\ P(Q = n) &= P(Q > (n-1)) - P(Q > n) \end{aligned} \tag{3.3}$$

Then by using (3.3) the average number of packets in buffer, l, can be derived as

$l = \sum_{i=1}^{b} i \cdot P(Q = i)$
$l = 1.P(Q = 1) + 2.P(Q = 2) + ... + b.P(Q = b)$
$l = 1.(P(Q > 0) - P(Q > 1)) + 2.(P(Q > 1) - P(Q > 2)) + ... + b.(P(Q > (b-1)) - P(Q > b))$
$l = P(Q > 0) + (2-1).P(Q > 1) + (3-2).P(Q > 2) + ... - b.P(Q > b)$
$l = P(Q > 0) + P(Q > 1) + P(Q > 2) + ... + P(Q > (b-1)) - b.P(Q > b)$

Finally, the average number of packets in buffer is,

$$l = \sum_{i=0}^{b} P(Q > i) - b \cdot P(Q > b) \tag{3.4}$$

If $b \to \infty$ then

$$l = \sum_{i=0}^{\infty} P(Q > i) = \sum_{i=0}^{\infty} e^{-K \cdot i^T} \tag{3.5}$$

Using (3.2), the probability of the number of packets exceeds the buffer size, b, (under infinite buffer assumption) can be written as

$$P(Q > b) = e^{-K \cdot b^T} \tag{3.6}$$

Throughout the thesis expression in (3.6) will be called the buffer utilization of an on-chip router. The lower value of the probability figure means less number of packets exist in the buffer, and eventually the utilization is low. This probability figure will be used one of two terms of the cost function of our combined optimization problem.

3.2 Effect of Self-Similar Traffic Parameters on Network Performance

The following sections aim to quantize the effect of self similarity parameters, namely Hurst parameter, H, variance coefficient, a, and mean input rate m on the average number of packets and the buffer utilization.

3.2.1 Effect of Mean Input Rate on Average Packet Number and Buffer Utilization

Figure 3.1 gives the effect of mean input rate, m, on average packet number and buffer utilization for different values of H. As expected for high values of input rate the queue enters to congestion region. For higher values of H (in other words, when self-similar property exists) this effect is more obvious, which may prove the potentially the negative effect of self similarity on network performance. Another point that should be noted is that, for very low values of input rate short range traffic (H=*0.5*) has a better performance in terms of average packet number. This is due to a phenomenon called *"resetting and truncating effect"*, which states that, the effects of long-range dependence (self similarity) are significant only if long range dependence causes the busy periods to be long enough for the long lags to come into play. However, when the input rate is too low, the buffer stays empty for most of the time (resetting). On the other hand, if the buffer length is too low, the buffer is full for long periods and incoming packets are lost (truncating). In both cases, the effect of self similarity (or long range dependence) is diminished. [46, 47]

3.2.2 Effect of Variance Coefficient on Average Packet Number and Buffer Utilization

Effect of variance coefficient on performance is shown in Figure 3.2 which illustrates that variance coefficient has a negative effect on performance. Therefore, keeping variance coefficient low earlier at the mapping stage might improve on-chip network performance considerably. The resetting and truncating effect is also observed in this figure.

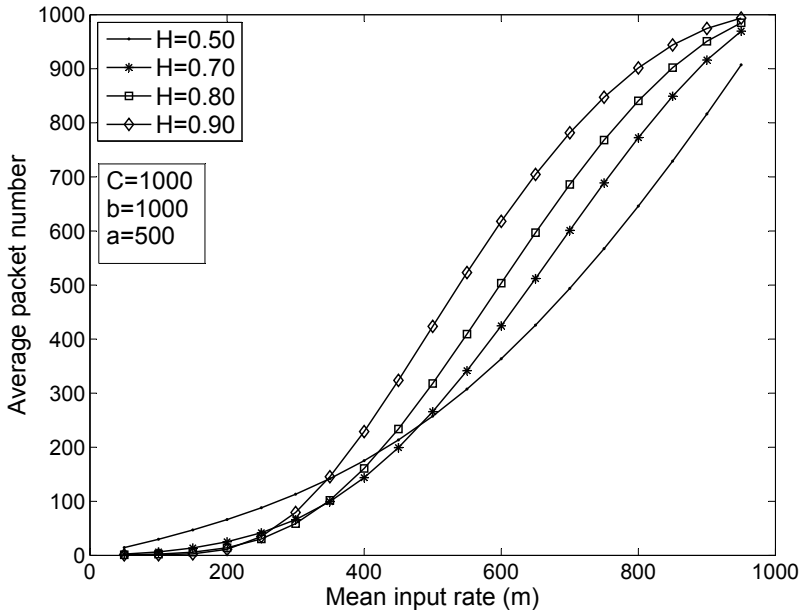
3.2.3 Computational Study

In order to analyze the effect of application mapping on NoC performance, we used a 6-core test graph. The density and traffic demand of the core graph are varied in order to reflect various network conditions ranging from a sparse network with low traffic to a dense one with high traffic. The density of the graph is selected as sparse, normal and dense, which have *9, 15, 23* links, respectively. The traffic demands of communicating cores are selected randomly with a uniform distribution between *50* and *250 MBps* (for low demand case) and *100* and *500 MBps* (for high demand case). These bounds are chosen such that traffic rates are in the lower half of the mean input rate axes of Figure 3.1. For each case, the service rate of routers is *1000 MBps* and buffer lengths are *1000 bits*. Each edge of the graph is also assigned with a Hurst parameter, H, between *0.5* and *1*, and a variance coefficient, a, between *50* and *500*, which are chosen in consistence with [48]. In Figure 3.3, core graph of the sparse network with high traffic demand (test case-2) is presented. Other test cases are generated by adding more links to the same graph. All test cases are listed in Table 3.1.

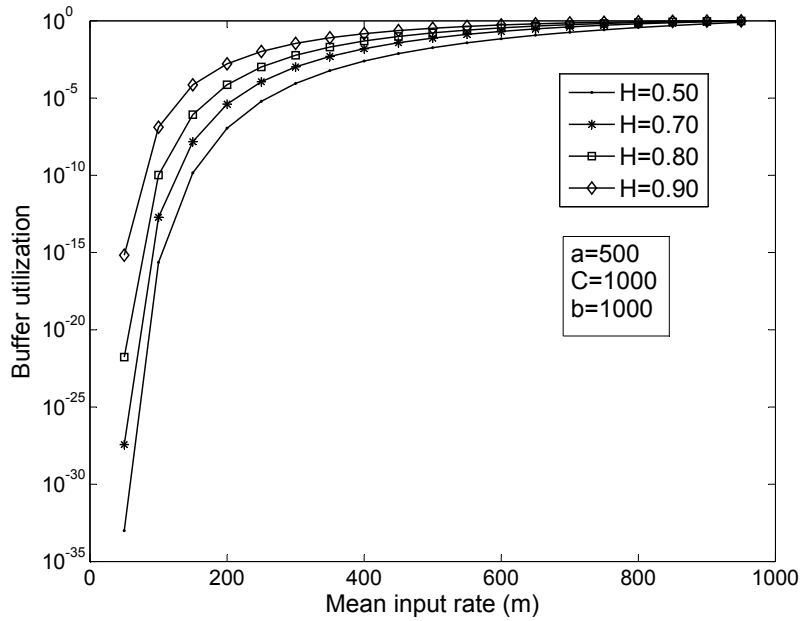
We assumed a simple on-chip network organized in the form of a 3x3 grid topology. Deterministic XY routing is chosen for simplicity but any other routing policy can also be adapted easily.

For each test case, we then iterated all possible mappings in a brute-force manner. Then for each possible mapping we considered communication energy overhead and buffer utilization, separately.

In calculation of each different mapping, incoming traffic for each queue is aggregated by using the method in [49]. The method states that when two self-similar traffic, with parameters (H_1, m_1, a_1) and (H_2, m_2, a_2), are combined in the same queue the parameters of aggregated input traffic (H_{agg}, m_{agg}, a_{agg}) are calculated as;

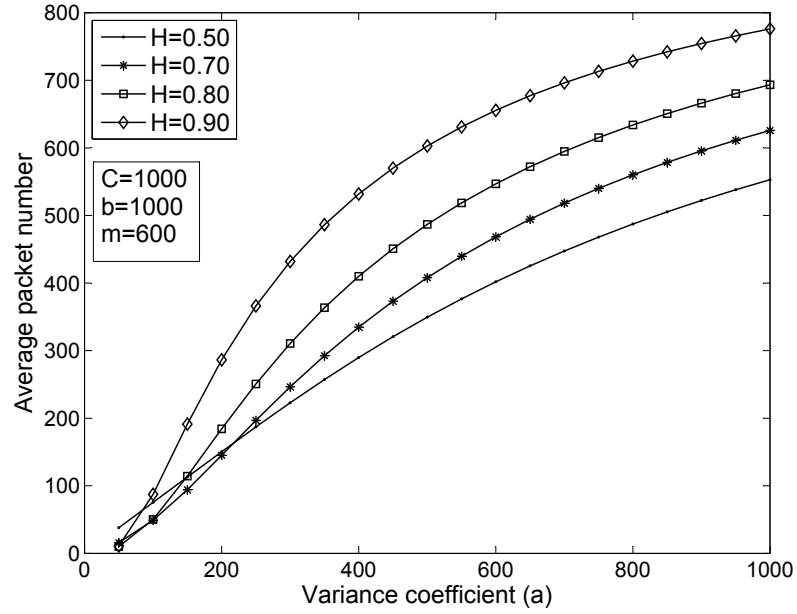


(a)

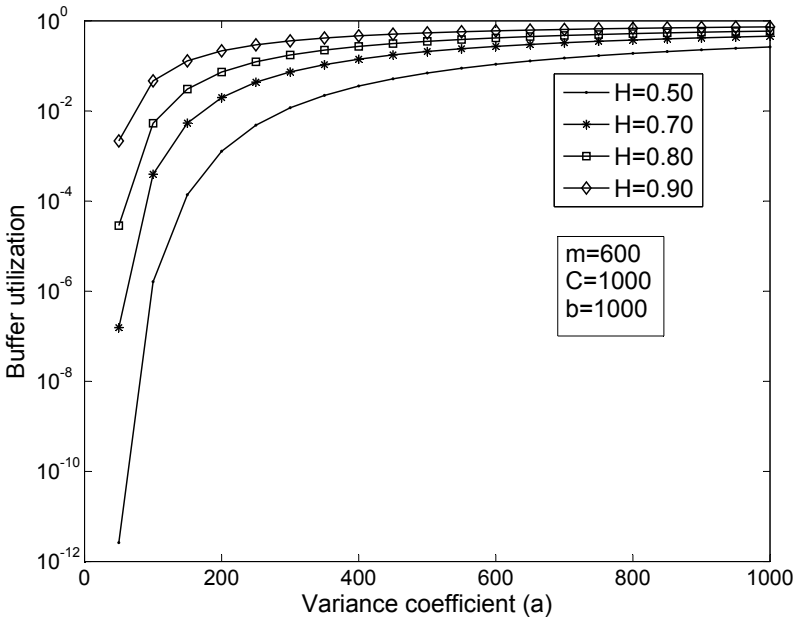


(b)

Figure 3.1: Effect of Mean Input Rate on (a) Average Packet Number (b) Buffer Utilization



(a)



(b)

Figure 3.2: Effect of Variance Coefficient on (a) Average Packet Number (b) Buffer Utilization

Table3.1: Test cases

	Graph Density	Traffic Demand
Test Case - 1	Sparse (9 links)	Low, between [50, 250] MBps
Test Case - 2	Sparse (9 links)	High, between [100, 500] MBps
Test Case - 3	Normal (15 links)	Low, between [50, 250] MBps
Test Case - 4	Normal (15 links)	High, between [100, 500] MBps
Test Case - 5	Dense (23 links)	Low, between [50, 250] MBps
Test Case - 6	Dense (23 links)	High, between [100, 500] MBps

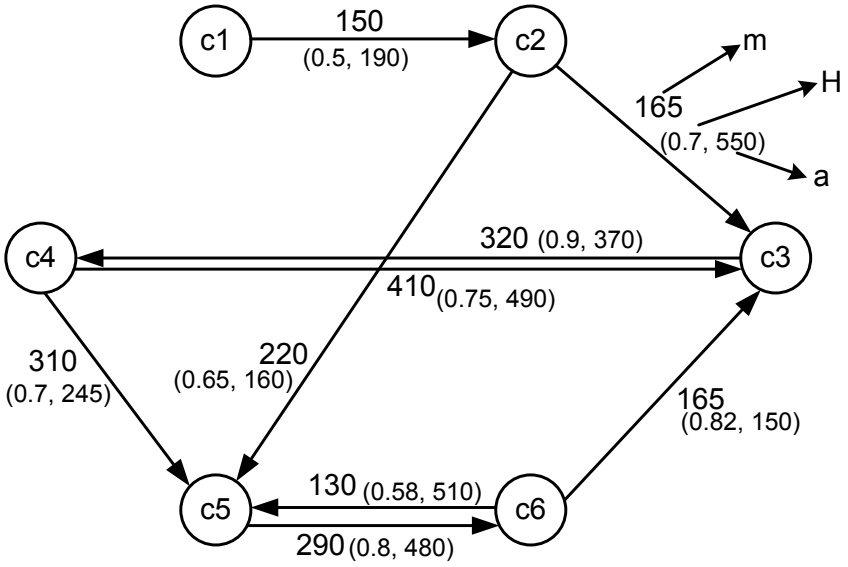


Figure 3.3: Core Graph for Test Case-2

$$H_{agg} = max(H_1, H_2) \tag{3.7}$$

$$m_{agg} = m_1 + m_2 \tag{3.8}$$

$$a_{agg} = \frac{m_1.a_1 + m_2.a_2}{m_1 + m_2} \tag{3.9}$$

The resulting aggregated input traffic is used in buffer utilization calculations of each buffer. In order to see the effect of application mapping on NoC performance, two different evaluation metrics are defined. "*Maximumbuffer utilization*" is defined as the highest utilization seen on the buffers of on-chip routers. Similarly, "*average buffer utilization*" is the average value of all utilizations. In this study, it is assumed that in case of congestion incoming packets are dropped. Therefore buffer utilization is very important and is used as the main evaluation metric. Similarly, same metric can be used as "blocking probability" for on-chip networks where a congested buffer blocks the traffic by a backpressure mechanism.

Table3.2: Proximity of worst short list solution to optima wrt. energy

	Proximity to optima
Test Case - 1	13%
Test Case - 2	12%
Test Case - 3	10%
Test Case - 4	9%
Test Case - 5	6%
Test Case - 6	6%

3.2.4 Effects of Application Mapping on NoC Performance

For each test case, the overall communication energy consumption and buffer utilization of the routers are calculated separately for each possible valid mapping.

After that, each mapping is labeled with a percentage value which indicates ratio of the value of evaluation metric under concern to the maximum value of that metric. These labels show the relative improvement of a mapping solution in terms of the evaluation metric (namely, maximum buffer utilization and average buffer utilization).

Then all mappings are sorted with respect to ascending overall communication energy consumption and the top 1% of this list is considered separately called as "*short list*". For example, for test case 1, the short list contains solutions that have at most 13% higher energy consumption value than the optimal solution. Proximity ranges of all test case short lists are presented in Table 3.2. It may be concluded that for all test cases, our short lists contain near optimal solutions of the mapping problem when communication energy is the only concern for minimization. Therefore analyzing only these sets is enough for evaluating the effect of application mapping on Networks-on-Chip performance.

For analysis purposes, the labels of this short list, calculated in the previous step, are counted and presented in a histogram form. In other words, buffer utilization values of best 1% mapping with respect to communication energy consumption are visualized on histograms. Figure 3.4 to 3.9 show these histograms for different test cases. In Figure 3.5-a, for example, test case 3 has 310 mappings in its short list. 121 of these mappings are in the best 1% with respect to average buffer utilization. 57 mappings are in 1%-2% interval, 33 are in 2%-3% interval and so on. All other figures can be interpreted in the same manner. In these figures, both average buffer utilization and maximum buffer utilization metrics are presented separately. Maximum buffer utilization is a stricter criterion than the average value for the evaluation of a mapping solution, since it considers the buffer that has the highest utlization. But in extreme cases, such as dense network with high traffic demand (Figure 3.9-b), this evaluation metric is not meaningful, since in many solutions there is at least one congested buffer. In such cases, average buffer utilization metric gives more information about the distribution of solutions in our short list, with respect to maximum buffer utilization. (Figure 3.6-b)

The main observation obtained from figures is that near-optimal solutions of energy minimization problem can vary significantly with respect to utilization. For example, in Figure 3.8-b the near optimal solutions range from highest buffer utilization solution, i.e. 100%, to 20% of it. That means minimum energy mapping solutions may perform a low utilization performance, which eventually degenerate the overall NoC performance. This effect cannot be observed in sparse network cases, as expected. But as

the number of communicating cores and the traffic demand increase, the buffer utilization performance of mapping solutions differ, representing the effect of application mapping on network performance.

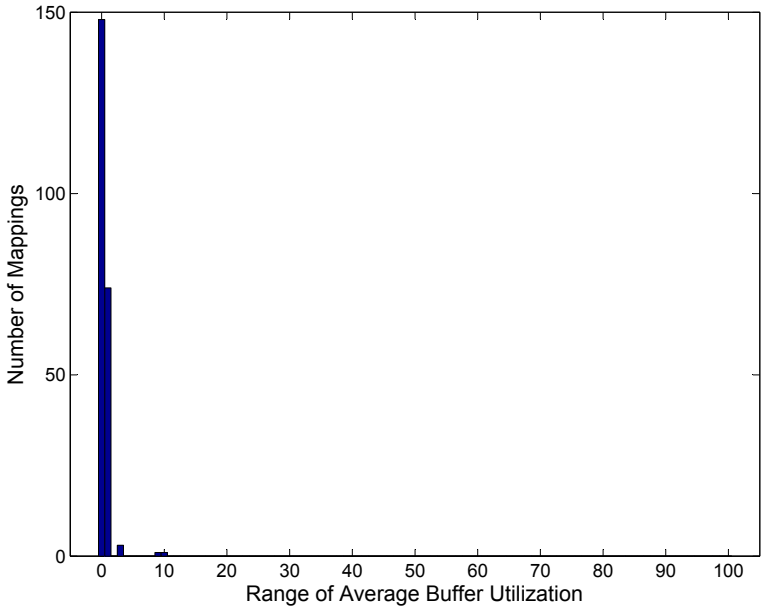
Another important observation from Figure 3.6-b and Figure 3.9-b is that the short list, which contains near-optimal solutions of energy minimization problem does not include the mapping solutions, which is under 20% with respect to buffer utilization. That presents a trade-off between energy consumption and utilization of the buffers. Some mapping solutions which have slightly higher energy consumption than near-optimal solutions may have a better networking performance in terms of buffer utilization and consequently support the QoS requirements of an application. Same effect can be observed in figures 3.5-b, 3.6-a and 3.8-b, in a smaller scale.

As a result, minimizing communication energy consumption during the application mapping stage does not necessarily minimize the buffer utilization of the routers. Therefore, although it is a network related issue, buffer utilization of on-chip routers can be decreased on the mapping stage in a proactive manner.

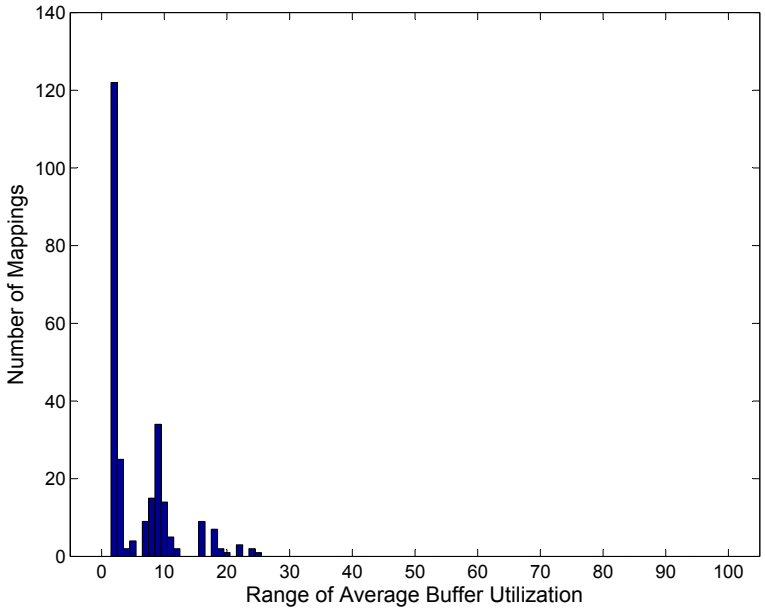
3.2.5 A New Approach to the Application Mapping Problem

Section 3.2.4 reveals that mapping of cores by considering the energy consumption only may have a significant degenerative effect on network performance in terms of buffer utilization for an on-chip network system. The results indicate that mapping solutions, which have acceptable communication energy consumption, may differ significantly when buffer utilization observed on buffers are taken into account under self-similar traffic assumption. Even though the energy consumption is optimal, if a mapping solution cause one or more buffers to operate in congestion state, this will decrease the run time performance of the on-chip network.

The above discussion shows that mapping of applications, which generate self-similar data traffic between its components, to an on-chip network topology by only considering the energy minimization may have a negative effect on the run time performance of the on-chip network. Therefore, if a priori traffic information of the application is available, a new kind of application mapping problem formulation can be proposed, which has a cost function that minimizes the communication energy while considering the utilization of network buffers. Such a formulation will generate a mapping that uses network resources fairly and decreases network delays caused by packet drops or high queue waiting times. Next section presents the formal definition of energy and buffer aware application mapping.

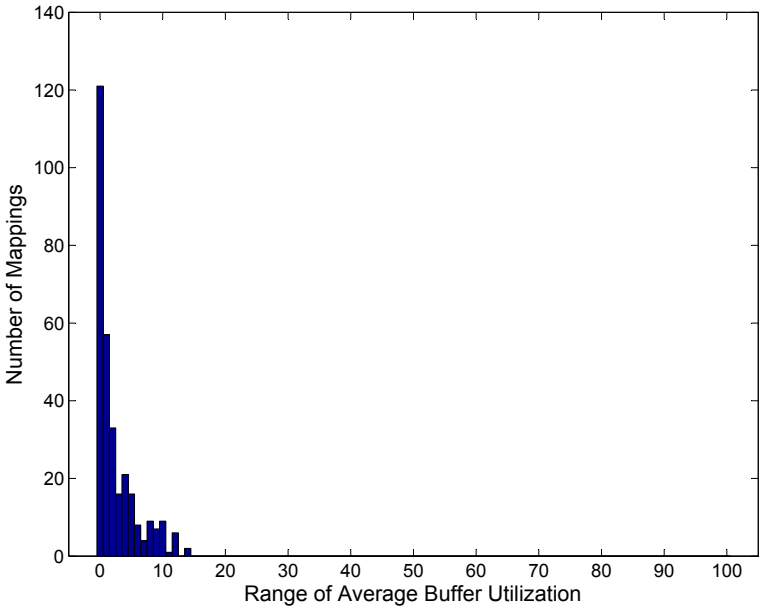


(a)

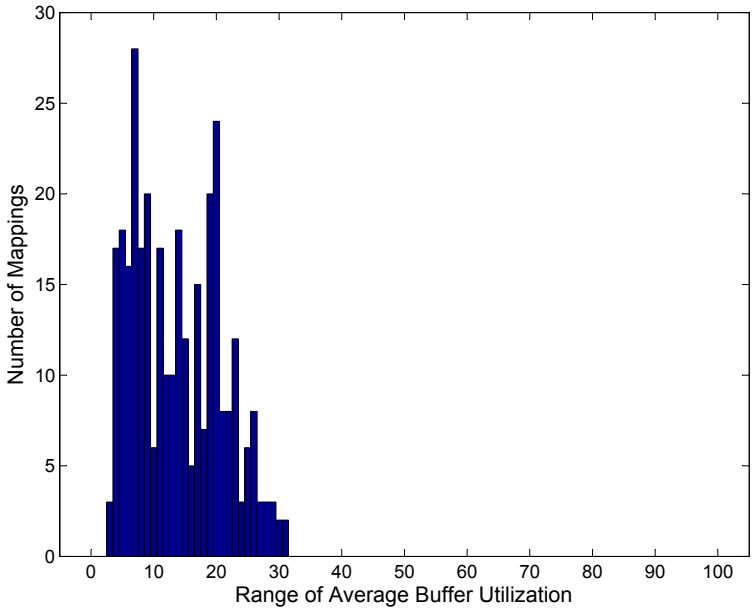


(b)

Figure 3.4: Histograms for average buffer utilization metric: (a) Sparse network with low traffic, (b) Sparse network with high traffic.

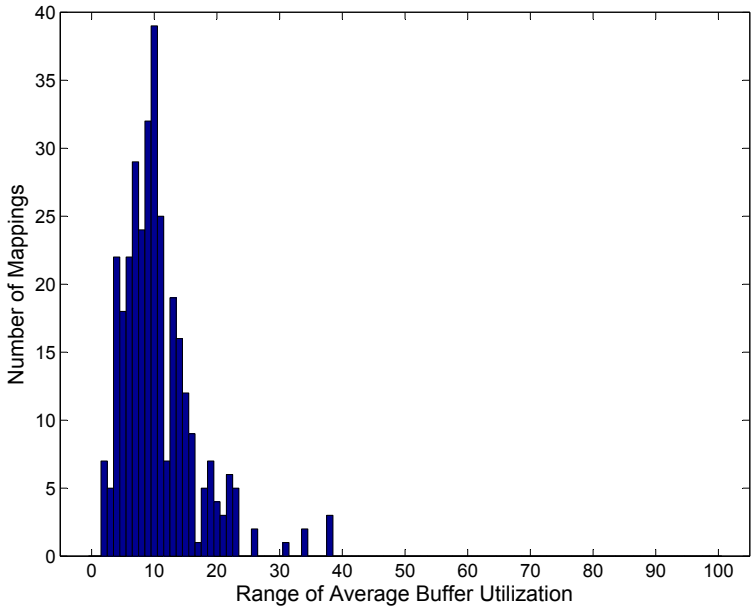


(a)

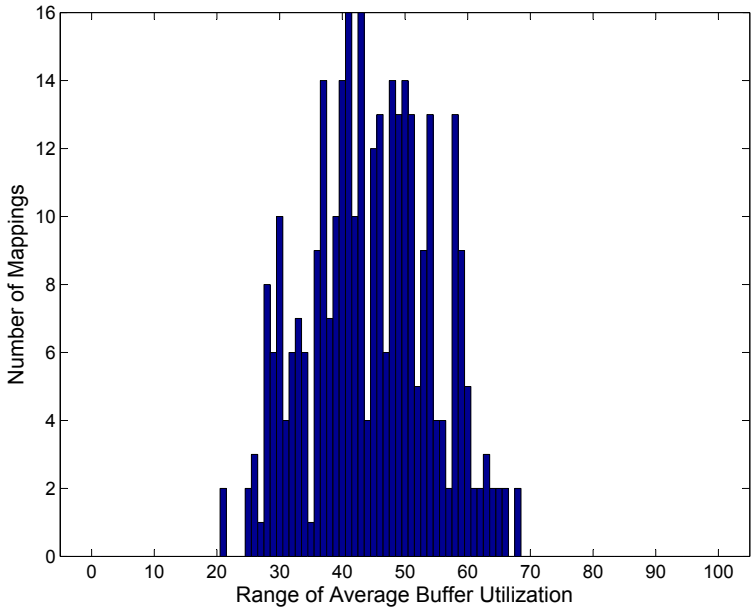


(b)

Figure 3.5: Histograms for average buffer utilization metric: (a) Normal network with low traffic, (b) Normal network with high traffic.

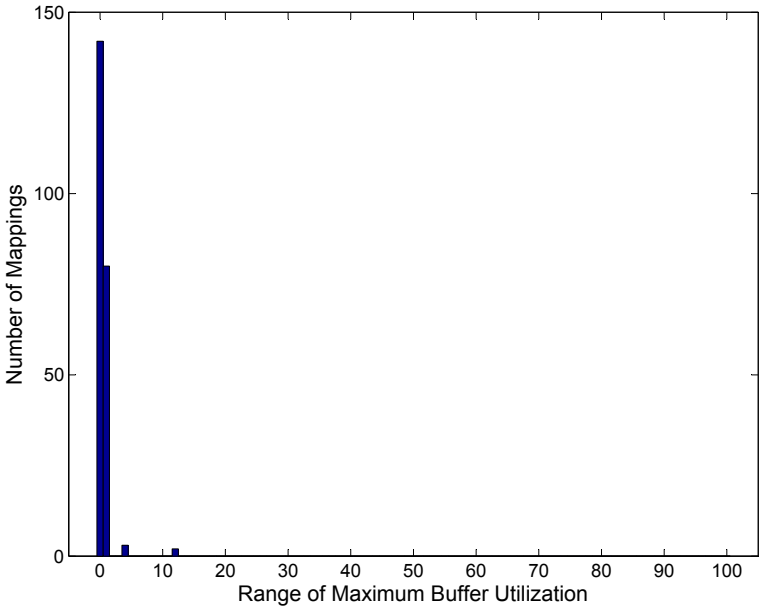


(a)

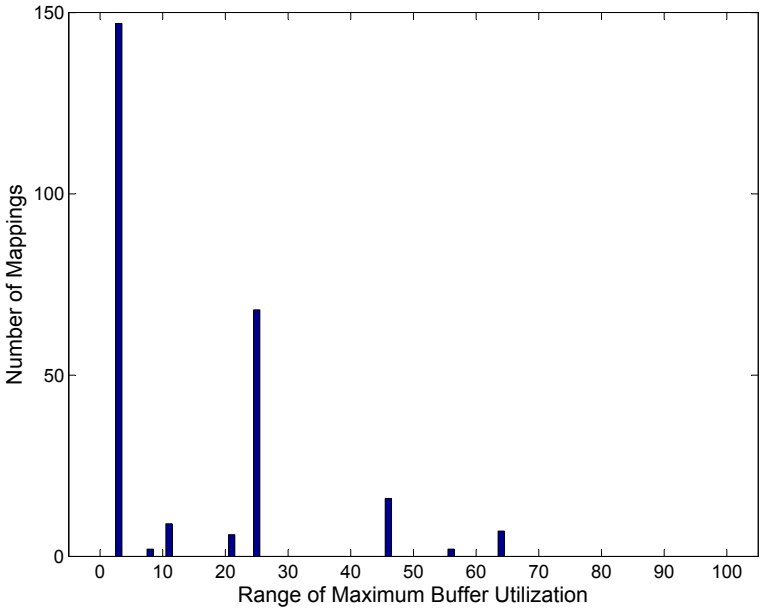


(b)

Figure 3.6: Histograms for average buffer utilization metric: (a) Dense network with low traffic, (b) Dense network with high traffic.

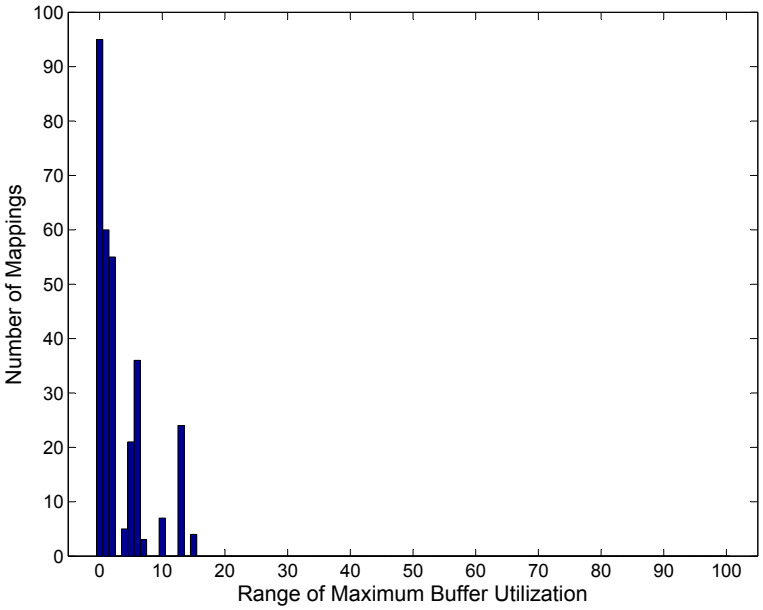


(a)

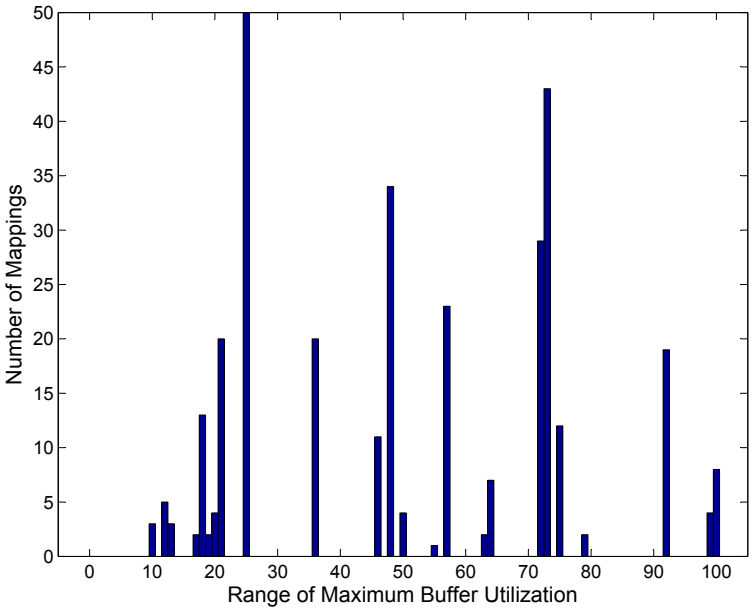


(b)

Figure 3.7: Histograms for maximum buffer utilization metric: (a) Sparse network with low traffic, (b) Sparse network with high traffic.

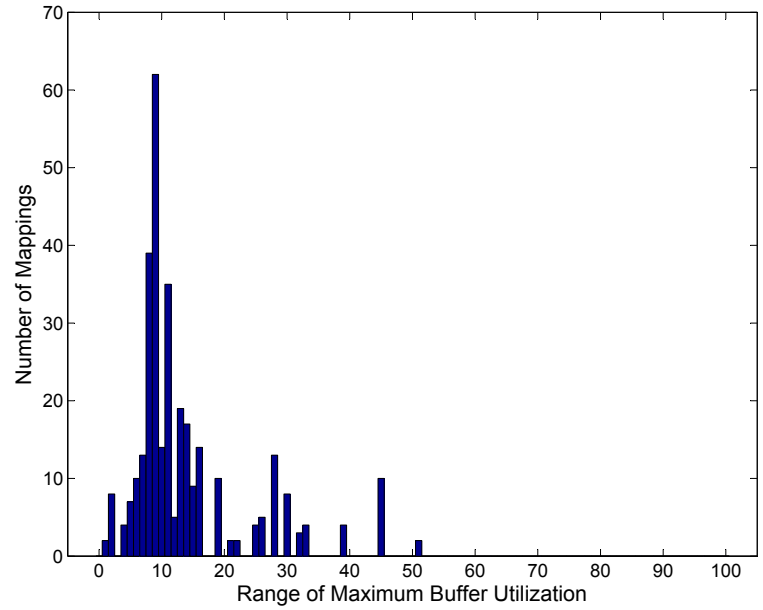


(a)

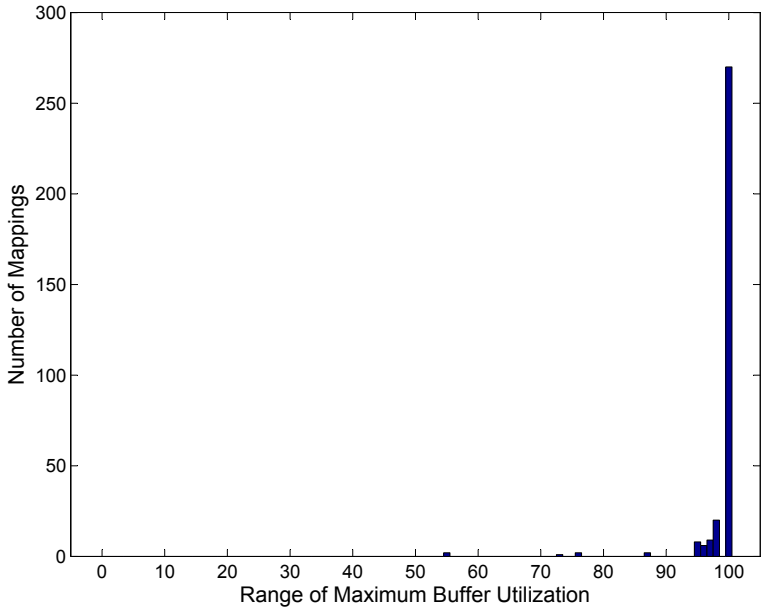


(b)

Figure 3.8: Histograms for maximum buffer utilization metric: (a) Normal network with low traffic, (b) Normal network with high traffic.



(a)



(b)

Figure 3.9: Histograms for maximum buffer utilization metric: (a) Dense network with low traffic, (b) Dense network with high traffic.

3.3 Problem Formulation

Our novel Energy and Buffer aware Application Mapping (EBAM) formulation consists of two models, namely the energy and the traffic models. The energy model, which is encountered commonly in the literature [13], calculates the total communication energy consumption between IP core pairs in an application, which is to be mapped onto the NoC whose topology is given. Besides, the traffic model calculates the buffer utilization of the routers, as described in Section 3.1.

The following set of definitions will be used in stating the overall problem formulation:

Definition 1: An application, which is going to be mapped onto an NoC architecture, is described as an Application Characterization Graph (APCG), $G = G(C, E)$. $G(C, E)$ is a directed graph, where each vertex $c_i \in C$ represents an IP core and each arc $e_{i,j} \in E$ represents the communication from vertex c_i to vertex c_j.

Definition 2: $v(e_{i,j})$ is the communication volume on $e_{i,j}$, i.e. between cores c_i and c_j, in bits.

Definition 3: Network topology is described by the graph $T(R, L)$, where R is the set of routers and L is the set of links in the network.

Definition 4: P_R defines the routing protocol, which determines the path between any source and destination router.

Definition 5: The energy required to send one bit from router r_i to router r_j can be calculated as

$$E_{bit}(r_i, r_j) = n_{hop} \times E_S + (n_{hop} - 1) \times E_L \tag{3.10}$$

where n_{hop} is the number of routers along the path between r_i and r_j, E_S is the energy consumed on routers, E_L is the energy consumed on links (in μJ/Mb).

In this study, we assume that the SoC under consideration has 100-nm technology. In [19], energy consumption of the on chip router and the links in 100-nm have been studied. According to this, the energy consumption of the input port of an router is 328 nJ/Mb and it is 65.5 for an output port. In total, we take E_S as 393.5 nJ/Mb. On the other hand, we assume that links are length of 3 mm. In [19], energy consumption of a physical link is assumed as 79.6 nJ/Mb/mm. Therefore in this study, we assume energy consumption of each link (E_L) is 238.8 nJ/Mb.

Definition 6: $B_i = \left\{b_i^k\right\}_{j=1}^{|B_i|}$ is the set of buffers in router r_i. Here, b_i^k denotes the k^{th} buffer of router r_i and so $\left|b_i^k\right|$ is the length of b_i^k, in flits.

$|B_i|$ is the number of buffers in a router. It is two for corner routers, three for edge routers and four for intermediate routers for a 2D grid topology.

Definition 7: $BU_i^k = P(Q > \left|b_i^k\right|)$ is the buffer utilization of b_i^k, where $P(Q > x)$ is referred to (3.6).

Definition 8: By using definition 7, the maximum buffer utilization of a networks-on-chip architecture, BU_{max} may be defined as

$$BU_{max} = max\{BU_i^k\},\ \forall r_i \text{ and } \forall b_i^k$$

Using the above definitions, our energy and buffer aware application mapping problem can be formulated as follows:

Given

- an application characterization graph, $APCG$
- a topology, T
- a routing protocol, P_R
- a self-similar traffic model using (H, a, m), where H, a and m are defined for every arc
- buffer lengths, $|b_i^k|$, of each router

Find

- a mapping function $map() : C \rightarrow R$ (from set of cores to set of routers), which minimizes

$$\min\{\frac{1}{\alpha}\sum_{\forall e_{i,j}} v(e_{i,j}) \times E_{bit}(map(c_i), map(c_j)) + \frac{1}{\beta}BU_{max}\} \tag{3.11}$$

Such that

- each core is mapped to a router,
- no two cores are mapped to the same router,
- QoS constraints (buffer utilization or end-to-end delay) are satisfied

The first term in (3.11) is the total energy consumed on the communication infrastructure. The second term in (3.11) is the maximum buffer utilization of all routers on the network. The variables α and β are the normalization coefficients, which will be calculated as part of the solution.

The application mapping problem presented above is an instance of *quadratic assignment problem* [14], which has been proven to be NP-hard [50]. The search space of a problem increases *factorially* with the size of the problem. Therefore, in the next chapter, we proposed a genetic algorithm based solution for our proposed mapping problem.

CHAPTER 4

AN EVOLUTIONARY METHOD FOR ENERGY AND BUFFER AWARE APPLICATION MAPPING

The problem formulation of energy and buffer aware application mapping proposed in previous chapter is an NP-hard one. Therefore a genetic algorithm based solution have been implemented to solve this intractable problem.

This chapter starts with a brief introduction to genetic algorithms. Then the genetic algorithm implemented for energy and buffer aware application mapping problem formulation is presented in detail. In order to evaluate the proposed method an Integer Linear Programming (ILP) model has also been implemented and used as a lower bound for the problem. The chapter concludes with the computational study performed to evaluate the proposed genetic algorithm based method.

4.1 Genetic Algorithms

Application mapping problem has already been proven to be NP-hard [50]. Hence our problem formulation proposed in previous chapter is also NP-hard. Therefore, a genetic algorithm based solution methodology is adopted in this work.

Genetic algorithm is a method for searching a large solution space using principles of evolution and genetics. In genetic algorithms, each solution in space is represented with an individual called *chromosome*. Chromosomes are composed of fundamental units called *genes*. Each chromosome may be evaluated using a fitness value that is related to the cost function of the problem.

Search process in genetic algorithms starts with an initial population containing randomly selected chromosomes. In each iteration new individuals, i.e. *offsprings*, are born by crossover operations. *Crossover* operation takes two individuals as input and generates two offsprings. It is desired that these new individuals have better fitness values, while still keeping some of the characteristics of their parents.

Search space may have some local optima and it is obviously necessary to avoid being stucked in such points. *Mutation* operation is used for this purpose of changing some characteristics of chromosomes, which may help in jumping to other parts of the search space.

At the end of an iteration, old and new chromosomes are sorted with respect to their fitness values and the ones with low fitness values are discarded in the next iteration, which represents the natural selection mechanism. A good feasible solution for the problem under consideration is found by iteratively repeating the above process.

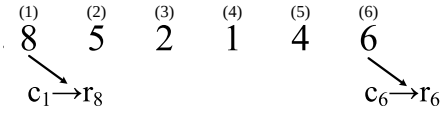


Figure 4.1: Permutation encoding example

4.2 Genetic Algorithm for Energy and Buffer Aware Application Mapping

In order to solve the problem given in Section 3.3 using a genetic algorithm, one should design the required mechanisms and operations. This subsection presents below the details of the algorithm designed for solving our application mapping problem.

4.2.1 Encoding

Permutation encoding, which is suitable for most mapping problems, is chosen here also. In this encoding the value of a gene is the router number and the place of the gene in the chromosome represents the IP core that is assigned to this router. For example, the example chromosome in Figure 4.1 represents a solution for a six core graph. From the values of the genes, it is possible to deduce that core 1 (c_1) is mapped to router 8 (r_8), core 2 (c_2) is mapped to router 5 (r_5), and so on.

In such an encoding, duplicated genes produce infeasible solutions since one core cannot be mapped to more than one router.

4.2.2 Population

Initial population is generated randomly in a genetic algorithm. From our experiments, we observed that an initial population of 30 chromosomes is sufficient to solve our problem. Final population contains the best solutions encountered during all iterations. The best solution of the final population is accepted as the solution of the problem.

4.2.3 Crossover

Crossover is the key operation in genetic algorithms for traversing the solution space efficiently. Since we use a permutation type encoding, it is not possible to use a basic crossover operation, in which a crossover point for parent chromosomes are randomly selected and the second parts of each parent are exchanged. Such a crossover may generate infeasible solutions and instead, a mechanism called partially matched crossover is used. In this scheme, two crossover points are selected randomly. The inner parts of these points are exchanged between two parents. Also, this inner part gives the exchange matching of genes. Figure 4.2 illustrates partially matched crossover operation on two sample six genes chromosomes.

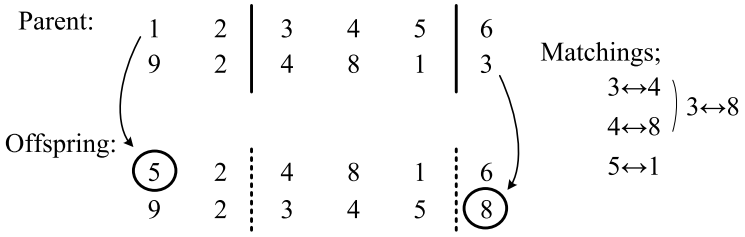


Figure 4.2: Partially matched crossover operation.

4.2.4 Mutation

The mutation for our algorithm is implemented by selecting two genes of a chromosome randomly and simply exchanging their values as illustrated below.

Initial chromosome: 1 2 3 4 5 6;

Chromosome after mutation: 1 5 3 4 2 6.

Mutation rate is kept at 5% in order to avoid changing the characteristics of the population drastically.

4.2.5 Elitism

Elitism is a mechanism to keep the best solution found so far within the population during iterations. If all of the population is replaced by newly generated offsprings, a good solution may be lost. This may occur due to mutation also. Therefore, in our algorithm only twenty new chromosomes are generated at each iteration and are replaced with the worst 20 out of 30 chromosomes. In other words, in every iteration the best ten solutions are kept within the population. Elitism may obviously increase the probability of being stuck in a local optimum.

The details and the parameters of our genetic algorithm are summarized in Table 4.2.

Table4.1: Methods and parameters of the genetic algorithm

Paradigm	**Decision**
Encoding	Permutation encoding
Population	30 chromosomes,
	random initialization (for first phase only)
Selection	Random 20 (10x2) parents
Crossover	Partially matched crossover
Mutation	Exchange one gene, rate: 5%
Elitism	Yes. Offspring replace worst 20 chromosomes.
	10 out of 30 remain unchanged.
Stopping	After a fixed number of iterations
Fitness value	Reciprocal of cost function (3.11)

4.3 Energy and Buffer Aware Application Mapping Algorithm

The genetic algorithm proposed above can be used for solving the problem to minimize energy only or similarly buffer utilization only. For solving the combined optimization problem however, one needs the fix two normalization coefficients α and β. In this study, we first solve the problem to minimize energy only and then separately buffer utilization only. After that we perform the required normalization simply by dividing each separate term, that corresponds to energy and buffer utilization respectively, to their maximum values obtained during previous genetic algorithm steps. This, in effect, means scaling both energy and buffer utilization to unity.

Therefore our algorithm runs in three phases. The first phase calculates the maximum value of energy consumption (coefficient α) by using the energy part of (3.11) only. The second phase calculates the maximum value of buffer utilization (coefficient β) by using the related part of (3.11) only . The third phase uses equation (3.11) fully for computing the fitness value.

We observed that using the final population of a phase as the initial population of the next phase increases the success rate of the algorithm. Hence, following a random initialization of the first phase's population, the second and third phases use the final populations of their preceding phases as initial population.

There is no stopping criteria for our genetic algorithm. It is seen that fixed number of 50 iterations are enough to get good solutions. With this number, the time taken for executing largest test case is less than 20 minutes.

4.4 An ILP Model for Energy Aware Application Mapping

In this section, we present an Integer Linear Programming (ILP) model of application mapping problem with only energy minimization goal, which is similar to the one in [20]. Utilization of this model is twofold. First, for evaluating the success of our genetic algorithm model. For this purpose, the results of the first phase of our genetic algorithm model (which has only energy constraints) are compared with the results of ILP model. In another words, the ILP model is used as the lower bound of our

problem.

Second, we use ILP model in order to see the efficiency of our problem formulation and pitfalls of energy-only solutions, in terms of buffer utilization. This is performed by comparing the buffer utilization figures of mapping solutions obtained by ILP and three phases of our genetic algorithm.

Hereafter our ILP based application mapping model will be called as "*ILP_M*" . The ILP_M contains two types of binary variables and a set of equations reflecting constraints. An open source solver, lp-solve [51] is used to solve the problem, optimally. The solver is capable of assigning binary attributes to variables. Using this property, the number of equations and the execution time of the solver is reduced. The following paragraphs present the details of the ILP_M.

$m_{c_i}^{r_s}$ variables indicate whether the core c_i is assigned to router r_s, or not.

$$m_{c_i}^{r_s} = \begin{cases} 1, \ core\ c_i\ is\ assigned\ to\ router\ r_s, \\ 0, \ otherwise. \end{cases} \tag{4.1}$$

$p_{c_i c_j}^{r_s r_t}$ variables express a communication path between routers r_s and r_t. Its value is 1 if there is a communication between routers r_s and r_t, where cores c_i and c_j are mapped to those routers, and 0 otherwise [20].

$$p_{c_i c_j}^{r_s r_t} = \begin{cases} 1, if\ there\ is\ a\ communication\ between\ c_i\ and\ c_j, \\ 0, \ otherwise. \end{cases} \tag{4.2}$$

$D_{r_s r_t}$ is used for the distance between routers r_s and r_t. Since we consider XY deterministic routing in this study, it implies the Manhattan Distance between two routers.

The first constraint of ILP_M is one-to-one mapping, which states every core must be mapped to a router (4.3).

$$\forall c_i \in C, \sum_{\forall r_s \in R} m_{c_i}^{r_s} = 1 \tag{4.3}$$

Since in our NoC infrastructure there is only one port for IP core connection in each router, two different cores cannot be mapped to the same router (4.4).

$$\forall r_s \in R, \sum_{\forall c_i \in C} m_{c_i}^{r_s} \leqslant 1 \tag{4.4}$$

$m_{c_i}^{r_s}$ variable must be binary, i.e. either 0 or 1 (4.5).

$$\forall r_s \in R, c_i \in C, 0 \leqslant m_{c_i}^{r_s} \leqslant 1 \tag{4.5}$$

Another constraint is on communication path variables. $p_{c_i c_j}^{r_s r_t}$ is equal to 1 only if $m_{c_i}^{r_s} = 1$, $m_{c_j}^{r_t} = 1$ and there is a communication between c_i and c_j. This can be expressed in ILP_M formulation as in (4.6).

$$m_{c_i}^{r_s} + m_{c_j}^{r_t} - 1 \leqslant p_{c_i c_j}^{r_s r_t} \leqslant \frac{m_{c_i}^{r_s} + m_{c_j}^{r_t}}{2} \tag{4.6}$$

The last one is binary constraint on $p_{c_i c_j}^{r_s r_t}$ variables (4.7).

$$0 \leqslant p_{c_i c_j}^{r_s r_t} \leqslant 1 \tag{4.7}$$

By using the above variables and equations the objective function of the ILP_M is;

$$\min\{ \sum_{\forall (c_i, c_j) \in C} w_{c_i, c_j} \cdot (\sum_{\forall r_s, r_t \in R} D_{r_s r_t} \cdot p_{c_i c_j}^{r_s r_t})\} \tag{4.8}$$

By using *"assign binary variable"* property of *lp-solve*, the equations (4.5) and (4.7) can be omitted. This reduces the number of equations and consequently decreases the execution time and memory requirement of the solver, considerably.

4.5 Numerical Evaluation of the EBAM

This section presents the computational study we performed for evaluating our genetic algorithm proposal and for demonstrating the benefits of energy and buffer aware application mapping approach in terms of buffer utilization of the routers. Test cases in 15 main test network classes of different sizes with different number of cores and links were generated. For each test case, the problem is solved by using our genetic algorithm both with and without the buffer utilization term in the cost function (3.11). For evaluating the success of our algorithm, the ILP_M presented in previous section is executed for each test case and the results are compared.

4.5.1 Test Cases

In order to address different network conditions, 15 core graphs with various dimensions and communication requirements are considered. These graphs are divided into several groups.

- The first group (Case 1a, 1b and 1c) considers a small core graph. Number of links are chosen to create sparse, normal and dense networks (namely, 9, 15 and 23 links) and the graph is mapped to a 3x3 topology.
- In our previous studies [41], we observed that an energy and buffer aware application mapping could be more effective on dense networks. For evaluating our method also on sparse networks, we formed the second, third and fourth test case groups that has a degree of around 2.5 and below.
- Finally, test cases 5 to 7 contain larger core graphs, which are mapped to 4x4 grid topology.

The traffic between each communicating core pair is selected randomly from a uniform distribution between 100 and 500 MBps. For each case, the service rate of a router is assumed to be 1000 MBps and

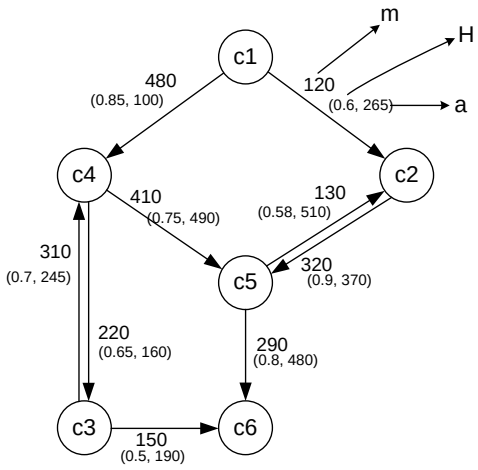


Figure 4.3: Core Graph for Case-1a

buffer lengths are chosen as 1000 bits. Each arc in the graph is also associated with a Hurst parameter, H (between 0.5 and 1), a variance coefficient, a (between 50 and 500), which are chosen in consistence with [52]. A detailed analysis of parameter selection can be found in [41]. Figure 4.3 shows the core graph for test case-1 as an example. The IP core graph (with (H, a, m) values), network topology, link and router capacities are the inputs of the problem. For simplicity, only deterministic XY routing has been considered but any routing policy can also easily be adapted.

4.5.2 Methodology

For evaluating the success of genetic algorithm and the efficiency of our problem formulation, results of different phases of genetic algorithm are compared with the results of the ILP_M. For each test case, we generated problem instances (core graphs) by selecting links of core graphs randomly with a uniform probability distribution and by assigning H, m and a parameters of each link in the core graph, again randomly with a uniform probability distribution. Each test case instance is solved by following three methods, separately.

- The first one is the our lower bound solution, ILP_M, which is described above. The cost function of this model is the total communication energy consumption, which is actually the first part of our problem formulation (3.11). This model is relatively slow and one needs some additional heuristics to make it run faster but such run time concerns are left out of the scope of this study.
- The second one is our proposed algorithm, which uses only the energy part of our formulation, which is equivalent to cost function of the ILP_M.
- The last one is the complete genetic algorithm including all three phases described. (Throughout the rest of the dissertation, the last two methods are called EBAM-energy and EBAM-complete, respectively.)

Then all results within a test case are averaged and statistical significance level is computed. The number of instances to solve before terminating the experiment is determined so that the obtained

results are within ±10% of actual values with a confidence level of 90%.

Computation of a single core graph by using EBAM takes 5 minutes for the smallest core graph (case-1a) and 20 minutes for the largest one (case-7) on a computer having Intel i5 processor with 4 GB of RAM. The ILP_M solves most of the cases in reasonable times but for case-6 and case-7 it times out at 30 minutes due to the high number of links of these test cases.

Table 4.2 presents the details of each test case and the corresponding results. Descriptions for each column are given below;

- Columns 2 to 4 give the number of cores and number of links in core graph, and size of network topology, respectively.
- Columns 5 and 6 present degree and density of IP core graph. Degree is the average number of links per core and density is the ratio of number of links to the number of links of a fully connected graph of that size.
- Columns 7 to 12 give total energy consumption (in μJ) and maximum buffer utilization (percentage) calculated by three different methods.
- Column 13 presents energy difference between ILP_M and EBAM-energy ($\Delta_E^{E-e}(\%)$) models. This column gives an opinion about **the success of our genetic algorithm** in solving the problem with energy only concern.

$$\Delta_E^{E-e}(\%) = \frac{E^{E-e} - E^{ILP\_M}}{E^{E-e}} \times 100$$

- Similarly, column 14 gives energy difference between ILP_M and EBAM-complete ($\Delta_E^{E-c}(\%)$), which implies **the energy overhead** of our proposed method

$$\Delta_E^{E-c}(\%) = \frac{E^{E-c} - E^{ILP\_M}}{E^{E-c}} \times 100$$

- Finally, column 15 gives **the buffer utilization improvement** achieved by EBAM-complete, Δ_L^{E-c}

$$\Delta_L^{E-c} = L^{E-c} - L^{ILP\_M}$$

where E^{ILP_M}, E^{E-e} and E^{E-c} are the communication energy consumptions of ILP_M, EBAM-energy and EBAM-complete, respectively and L^{ILP_M} and L^{E-c} are the maximum buffer utilization values of ILP_M and EBAM-complete models, respectively.

4.5.3 Performance of the Genetic Algorithm in solving EBAM

Table 4.2 contains many results about the value of our energy and buffer aware application mapping approach and the success of the proposed genetic algorithm.

Our first observation is about buffer utilization when only energy consumption is taken as the cost. Maximum buffer utilization values of ILP_M and EBAM-energy solutions are considerably high, which means that there is at least one bottleneck router in the network. Such results verify our statement that the minimum energy only solution does not necessarily generate an acceptable buffer utilization on routers, which in turn supports the main motivation of the thesis work.

Table4.2: Results of the computational study

						ILP_M		EBAM-energy		EBAM-complete				
	Cores	Links	Topology	Degree	Density	E (μJ)	BU_{max} (%)	E (μJ)	BU_{max} (%)	E (μJ)	BU_{max} (%)	$\Delta_E^{E-e}(\%)$	$\Delta_E^{E-c}(\%)$	Δ_L^{E-c}
Case1a	6	9	3x3	1.50	0.30	0.3695	44.67	0.3737	46.48	0.3873	24.32	1.13	4.59	-20.35
Case1b	6	15	3x3	2.50	0.50	0.6818	91.12	0.6876	91.16	0.7070	75.78	0.84	3.57	-15.33
Case1c	6	23	3x3	3.83	0.77	1.0743	98.73	1.0782	99.33	1.0821	98.10	0.36	0.72	-0.63
Case2a	7	12	3x3	1.71	0.29	0.5086	56.63	0.5226	62.82	0.5490	39.82	2.68	7.36	-16.82
Case2b	7	14	3x3	2.00	0.33	0.6210	76.50	0.6312	76.50	0.6692	54.05	1.63	7.21	-22.45
Case2c	7	18	3x3	2.57	0.43	0.8168	90.14	0.8337	91.49	0.8432	79.94	2.02	3.13	-10.19
Case3a	8	14	3x3	1.75	0.25	0.6051	70.05	0.6234	76.29	0.6466	51.20	2.94	6.41	-18.85
Case3b	8	16	3x3	2.00	0.29	0.6965	80.15	0.7129	80.69	0.7422	61.36	2.30	6.15	-18.79
Case3c	8	20	3x3	2.50	0.36	0.9189	92.05	0.9378	96.33	0.9639	83.12	2.01	4.66	-8.93
Case4a	9	16	3x3	1.78	0.22	0.6931	69.67	0.7120	78.29	0.7372	54.92	2.66	5.98	-14.75
Case4b	9	18	3x3	2.00	0.25	0.7970	83.06	0.8202	83.68	0.8468	65.67	2.83	5.89	-17.39
Case4c	9	23	3x3	2.56	0.32	1.0904	93.31	1.1051	92.98	1.1283	89.99	1.33	3.36	-3.31
Case5	12	30	4x4	2.50	0.23	0.9991	74.72	1.0537	76.63	1.1000	54.52	5.18	9.17	-20.20
Case6	14	35	4x4	2.50	0.19	1.2685	86.60	1.3173	92.20	1.3154	59.71	3.71	3.57	-26.88
Case7	16	40	4x4	2.50	0.17	1.4704	83.71	1.4995	94.23	1.5452	59.04	1.94	4.84	-24.67

The energy consumption values of ILP_M and EBAM-energy (two methods that use the same cost function) are observed to be very close (Δ_E^{E-e} values). Since the ILP_M solution is a lower bound of our mapping problem, we can conclude that the implemented genetic algorithm is efficient and successful. Figure 4.4 presents energy consumptions obtained by two different methods.

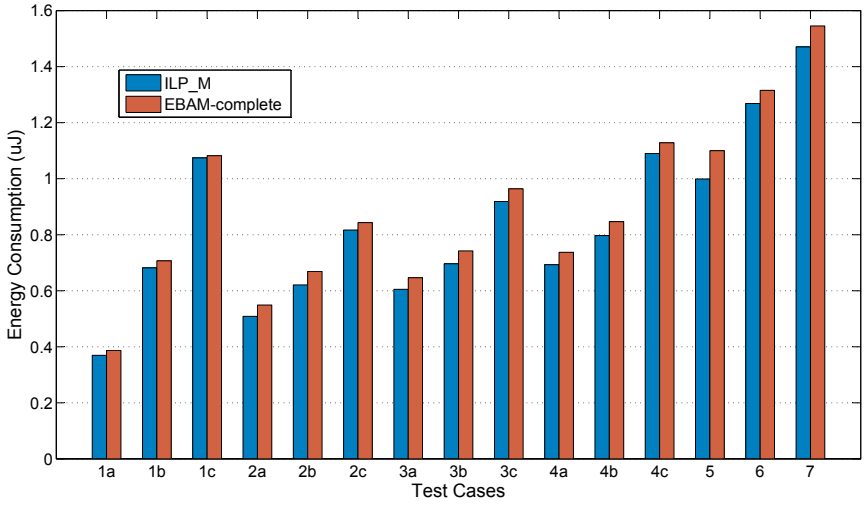


Figure 4.4: Energy overhead

Energy and buffer aware application mapping formulation naturally causes an increase in energy consumption. Results indicate that the energy trade-off (Δ_E^{E-c}) is reasonable, i.e., 9% at the most (case-5). On the other hand, quite high buffer utilization improvements (Δ_L^{E-c}) were achieved by the proposed algorithm (up to 26%). So that, probability of generating congested router will be decreased. Therefore the algorithm is suitable for applications where QoS requirements are strict such as real time multimedia applications.

In Table 4.2, test cases are divided into groups where each group has same number of cores but various number of links. Within each test case group, the test case with the highest degree (i.e. test cases 1c, 2c, 3c and 4c) has a lower buffer utilization improvement (Δ_L^{E-c}). This is because when the number of links increases, the number of intersecting paths also increases. This eventually increases the input traffic load of some buffers. But even in those cases, the proposed method has an important buffer utilization improvement (up to 10%). In other words, the method can find a mapping solution which can distribute the overall traffic more fairly.

Figure 4.5 presents maximum buffer utilization values of all test cases for both ILP_M and EBAM-complete. In each case, the gap corresponding to the chosen confidence interval obtained at the end of each experiment are also illustrated on the corresponding bar graph. It is observed for almost all test cases, that the possible worst value of the buffer utilization of EBAM-complete is better than the best value of the ILP_M solution.

Figure 4.6 shows some of maximum buffer utilization values, in groups having same number of links and different core numbers. We observed that for high number of cores, the buffer utilization value is low. The total traffic injected to the network is directly related to the number of links in the core graph. Therefore, two test cases with same number of links have same amount of traffic. On the other hand, increasing number of cores means more routers join to packet transmission. Since the number of routers that share same amount of traffic increases the input traffic per buffer decreases, which improves

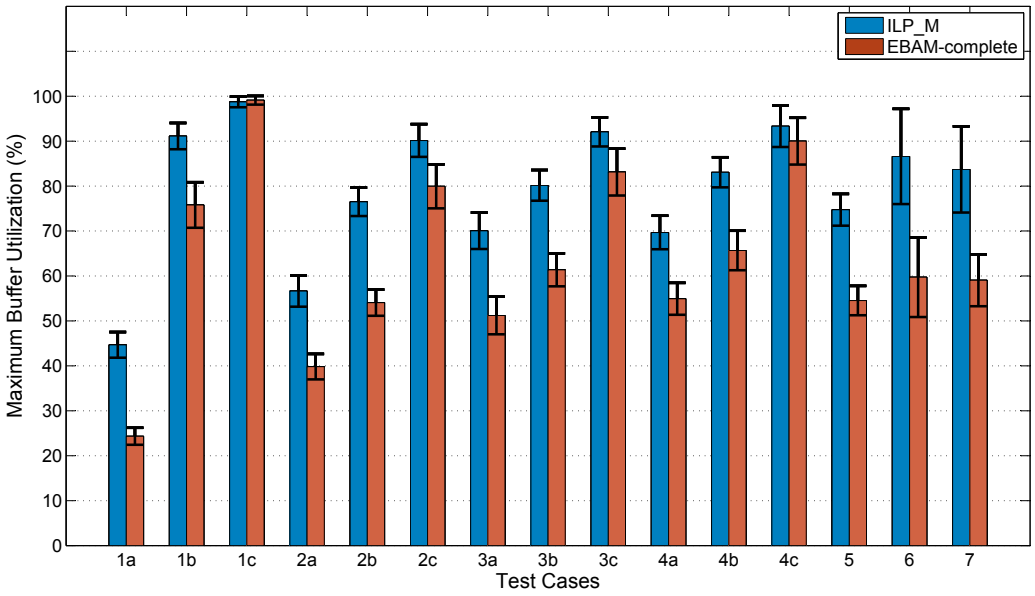


Figure 4.5: Maximum buffer utilization improvements

Table4.3: Results for increasing network resources

	ILP_M		EBAM-complete			
NRC*	Energy (μJ)	BU_{max} (%)	Energy (μJ)	BU_{max} (%)	Utilization improvement in dB	Energy overhead $\Delta_E^{E-c}(\%)$
×1	0.8168	90.136	0.8432	79.944	0.52	3.13
×1.5	0.8270	40.841	0.8923	18.696	3.39	7.32
×2	0.8271	9.160	0.8981	1.560	7.69	7.91
×3	0.8243	0.149	0.8563	0.004	16.13	3.73
×4	0.8245	6.21E-04	0.8937	2.18E-07	34.54	7.75

(*) NRC: network resource coefficient to multiple buffer size (b) and service rate (C)

the buffer utilization of the system. Therefore results presented in Figure 4.6 are quantitive proof of our initial statement which says that distributing the overall traffic across the network will improve the buffer utilization of the networks-on-chip.

All test case presented in Table 4.2 have low network resources in terms of buffer size (b) and service rate (C). Those cases results in high buffer utilization values showing the relative buffer utilization improvements for varying test cases. Another set of experiments has been carried out by increasing network resources in order to evaluate the efficiency of the proposed algorithm in more relaxed network conditions. Table 4.3 presents results using test case 2c but by multiplying the buffer size and service rate with a constant coefficient. Results indicate that the EBAM-complete method has better performance when network resources are increased. In some cases our proposal decreases maximum buffer utilization a few order of tens. Therefore, it can be concluded that our proposed method will be useful in satisfying more strict loss rate constraints of an application in relaxed network conditions.

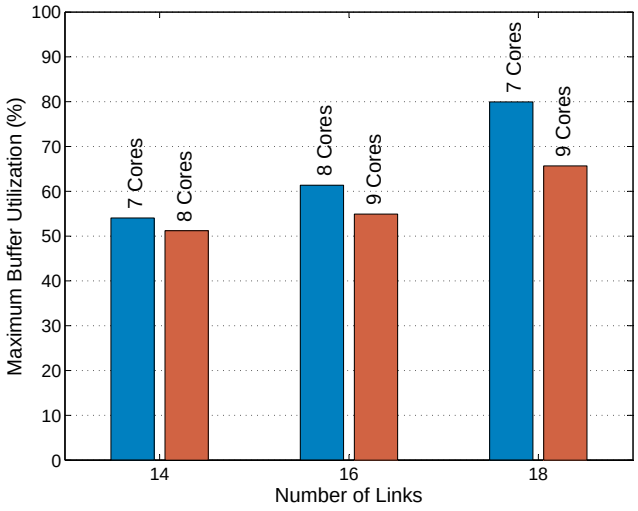


Figure 4.6: Maximum buffer utilization of EBAM-complete vs. Number of links in core graph.

4.6 Effect of Normalization Coefficients

Cost function of EBAM-Complete contains two terms, i.e. the energy consumption term and the maximum buffer utilization term (3.11). Combined optimization of energy and buffer utilization requires summation of these two terms, but since they have different units, direct summation is not applicable. Therefore a normalization process is necessary.

The cost function (given in Chapter 3 as (3.11)) can be simply written as;

$$CostFunction = \frac{1}{\alpha} f_1 + \frac{1}{\beta} f_2 \tag{4.9}$$

A simple normalization can be performed by dividing each term by its maximum value. In this way, each term gets a unitless value within [0, 1] interval. The summation will be in [0, 2]. After that comparison of costs of different mapping solutions will be meaningful.

For the reason described above, EBAM-Complete method contains three phases. First phase optimizes the problem with respect to energy consumption only and used to used to find maximum value of energy consumption for a given core graph (α). Similarly, second phase uses only maximum buffer utilization as the cost function and detects the solution with the highest maximum buffer utilization value (β). These two values are used as normalization coefficients in the third phase.

Values found in first two phases are not the exact maximum of the terms under concern. But an error made at this point only effects the normalization process. With a maximum value which is less than the actual maximum, normalization of the best solution will result slightly higher than unity. That means the weight of that term will be higher than the other one. The following paragraph quantizes this variation.

From results of our previous computational study (Table 4.2), it is observed that the difference between energy terms of the ILP_M and the EBAM-Energy (two methods having same cost function) is 5% at

most. Assuming that ILP_M is a lower bound for our problem, we can say that first phase will result with a value $E'_{max} = 0.95E_{max}$. (where E_{max} and E'_{max} are the actual and the computed maximums of energy term.) Assuming the second phase would find the exact maximum ($BU'_{max} = BU_{max}$), the cost of the best solution will be calculated as;

$$Cost = \frac{E_{max}}{E'_{max}} + \frac{BU_{max}}{BU'_{max}}$$
$$Cost = \frac{E_{max}}{0.95E_{max}} + \frac{BU_{max}}{BU_{max}} = 1.05 + 1 \tag{4.10}$$

It is seen from (4.10) that weight of the energy term will be 5% higher than that of buffer utilization term.

We can conclude that, this deviation is acceptable and cost function (3.11) can be used for combined optimization of energy consumption and maximum buffer utilization of a NoC system.

4.6.1 Adding Weights to Cost Function

Our cost function can be improved by adding a weight factor to each term. In the original cost function (3.11) both terms have same weights, i.e. 50%. But in some cases, it may be desired to give some priority to one of the terms against the other.

In (4.11), it is simply obtained by adding a factor, called gamma, γ, to energy term and (1-γ) to buffer utilization term. As the γ increases the weight of energy optimization increases, on the other hand low γ values means higher (dense) buffer utilization optimization. For γ=0.5, (4.11) is identical to (4.9).

$$CostFunction = \frac{\gamma}{\alpha} f_1 + \frac{1-\gamma}{\beta} f_2 \tag{4.11}$$

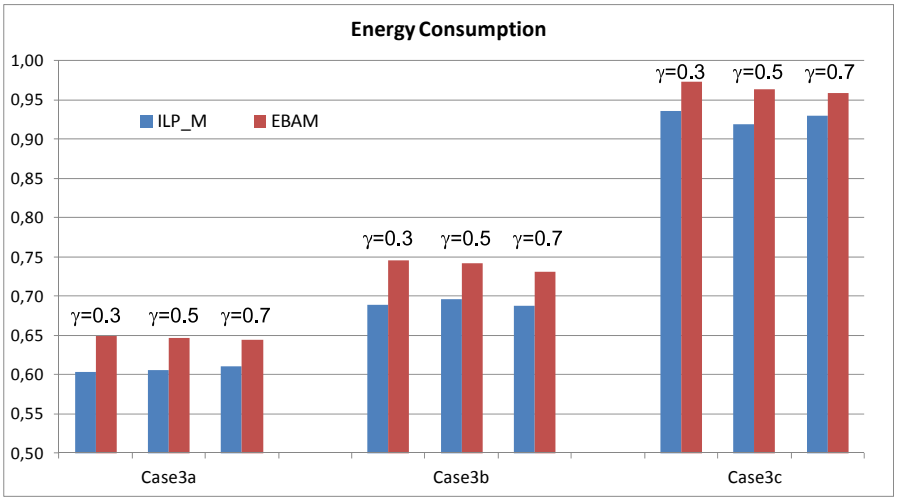
where f_1 and f_2 represent energy consumption and buffer utilization terms of the original cost function, respectively.

Effect of adding a weight factor can be examined in a new set of computational studies. For this purpose, Case 3a, 3b and 3c of previous computational study are repeated with γ=0.3 and γ=0.7 factors. Table 4.4 and Figure 4.7 present obtained results.

It is observed that, as γ increases the energy overhead of the method decreases. In other words, as γ increases the method gets more "*energy sensitive*". As the gamma goes to 1 the Δ_E^{E-c} value will decrease to Δ_E^{E-e}, i.e. no buffer utilization optimization. Similarly increasing gamma decreases the buffer utilization improvement, as expected. The methods ILP_M and EBAM-energy have only a single energy term and therefore they do not have a weight factor in their cost functions. But the variations within the results of same test cases for these two methods are due to the random generation of test core graphs and averaging according to confidence interval rules.

Table4.4: Obtained results for varying gamma

				ILP_M		EBAM-Energy		EBAM-Complete				
	Core	Link	Gamma	Energy	BU_{max}	Energy	BU_{max}	Energy	BU_{max}	$\Delta_E^{E-e}(\%)$	$\Delta_E^{E-c}(\%)$	Δ_L^{E-c}
Case3a	8	14	0.3	0.60	70.1%	0.62	76.2%	0.65	49.0%	2.8%	7.0%	-21.1%
Case3a	8	14	0.5	0.61	70.1%	0.62	76.3%	0.65	51.2%	2.9%	6.4%	-18.8%
Case3a	8	14	0.7	0.61	69.8%	0.63	74.5%	0.64	53.0%	2.6%	5.3%	-16.8%
Case3b	8	16	0.3	0.69	81.7%	0.70	81.2%	0.74	58.2%	2.3%	7.6%	-23.5%
Case3b	8	16	0.5	0.70	80.1%	0.71	80.7%	0.74	61.4%	2.3%	6.2%	-18.8%
Case3b	8	16	0.7	0.69	81.5%	0.70	81.0%	0.73	62.4%	2.2%	5.9%	-19.0%
Case3c	8	20	0.3	0.94	93.5%	0.96	93.5%	0.97	81.7%	2.4%	3.8%	-11.8%
Case3c	8	20	0.5	0.92	92.0%	0.94	96.3%	0.96	83.1%	2.0%	4.7%	-8.9%
Case3c	8	20	0.7	0.93	92.4%	0.95	94.9%	0.96	86.1%	2.3%	3.0%	-6.3%



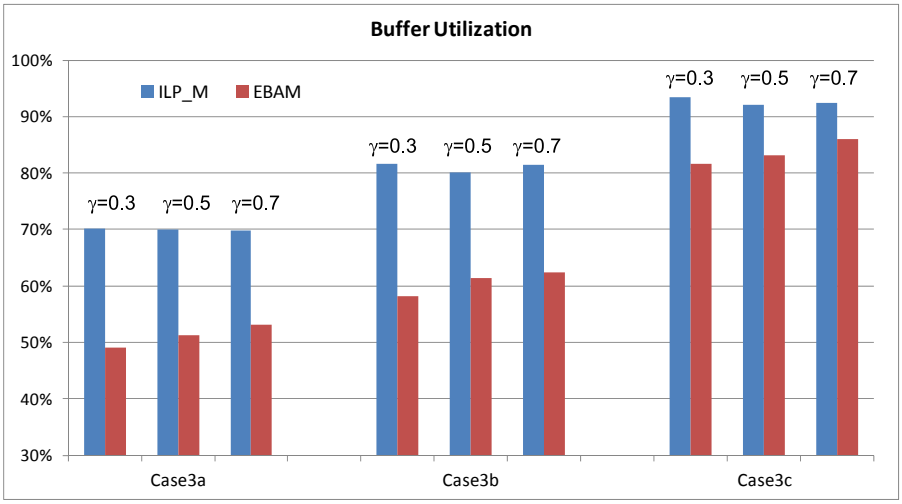


Figure 4.7: (a) Energy Overhead, (b) Buffer Utilization Improvement for varying weight factor

CHAPTER 5

EVALUATION OF EBAM UNDER VARIOUS ROUTING ALGORITHMS

In this chapter, the proposed mapping algorithm, EBAM, will be evaluated under various routing algorithms. After introducing NoC routing algorithms briefly, a computational study will be presented which evaluates the effect of minimizing buffer utilization at mapping stage on the performance of routing protocol.

5.1 Routing Protocols for Networks-on-Chip

In all packet-based network systems, a routing protocol is used to determine the path that each packet should traverse form its source to destination. Since energy and area constraints of SoC are very strict, a routing protocol for NoC should be simple, easy to implement and should not consume too much energy.

Routing protocols can be divided into two groups. The first group is deterministic routing protocols, in which all packets between two nodes always traverse the same path determined according to a specific rule. For example, in XY deterministic dimension-ordered routing, packets are sent in X-direction first and then in Y-direction. Deterministic routing protocols are widely used in NoC's because of their simplicity.

The second group is adaptive routing protocols. Adaptive routing protocols detect the state of the network and avoid selecting congested nodes while routing packets. Although they have better performance in terms of delay and throughput, adaptive routing protocols are prone to deadlock.

A deadlock occurs when a packet is waiting for an output link which is hold by a second packet and the second packet is waiting for a link which is hold by the first packet. Deadlocks may occur when a cycle exists among different routes. Therefore in order to avoid deadlocks in a routing protocol some turns are prohibited, in a way that the remaining turns cannot generate a cycle. In general, this is called turn model routing [8]. Figure 5.1.a shows all possible 8 turns in a 2D mesh network. Actually, XY deterministic routing is obtained by deleting 4 of these turns (Figure 5.1.b). Therefore, deterministic XY is a deadlock free routing protocol. But it can be shown that deleting only 2 appropriate turns is enough to avoid cycle generation. Following three turn model routing protocols are obtained in this manner.

- *West First Routing:* Figure 5.2.a shows possible turns of West First routing. In this protocol North-to-West and South-to-West turns are eliminated. That means, a packet going to west must

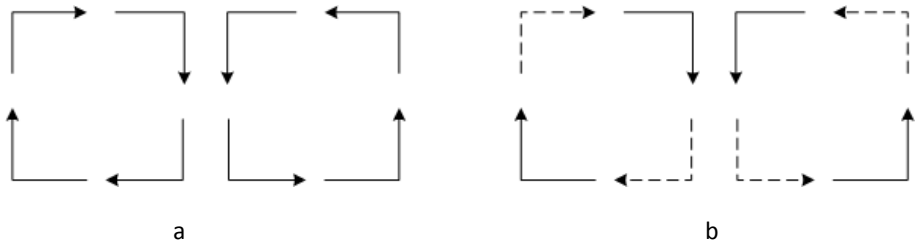


Figure 5.1: (a)All possible turns in 2D mesh network. (b) Allowed turns in XY routing (dashed lines are forbidden.)

first be transmitted to west until it reaches the same horizontal coordinate with its destination. After that a last turn (to the North or South) can be performed.

- *North Last Routing:* In this protocol, North-to-West and North-to-East turns are not allowed (Figure 5.2.b). Therefore any packet which turns to North cannot change its direction anymore. In this scheme, packets going to North are routed deterministically, other can be routed adaptively.
- *Negative First Routing:* This protocol allows all turns except the ones from positive directions to negative directions. Therefore packets should be routed in negative directions first. (Negative directions refer to South and West on a 2D coordinate plane.)

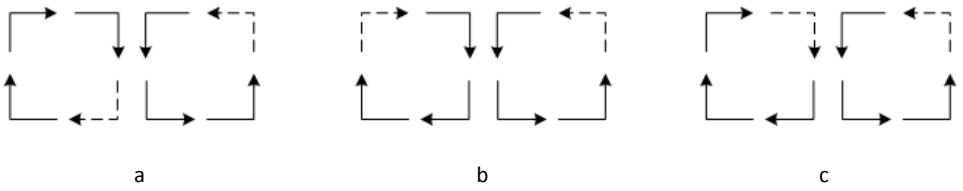


Figure 5.2: Turn Model routing protocols. (a) West First Routing, (b) North Last Routing, (c) Negative First Routing (dashed lines are forbidden.)

5.2 Computational Study

In this section performance of our proposed mapping method is analysed under various routing protocols with the help of a computational study which is similar to the one described in Section 4.5. The performance metrics are the total communication energy consumption and the maximum utilization of the buffers. Four routing protocols are considered, which are deterministic XY routing and adaptive west-first, north-last and negative-first routing protocols.

5.2.1 Test Cases

For comparison purposes, test cases with different number of cores and links were generated. For each test case, the problem is solved by using our proposed algorithm and the ILP_M, which can be considered as a lower bound of our optimization problem. In order to address different network conditions, 6 core graphs with various dimensions and communication requirements are considered. These are;

- Case 1a and 1b: small core graph (6 cores), normal and dense cases, respectively (9 and 15 links).
- Case 2a and 2b: core graph with 7 cores, normal and dense cases, respectively (14 and 18 links).
- Case 3: core graph with 8 cores and 16 links,
- Case 4: core graph with 9 cores and 18 links,

All test core graphs are considered to be mapped to a 3x3 2D mesh topology. The self-similar traffic between each communicating core pair is selected randomly from a uniform distribution between 100 and 500 MBps. For each case, the service rate of a router is assumed to be 1000 MBps and buffer lengths are chosen as 1000 bits.

5.2.2 Methodology

Mapping solutions for all test core graphs are found by both the ILP_M and the EBAM. For each solution the total communication energy consumption and the maximum buffer utilization values are recorded. In order to have statistical confidence each computation is repeated many times and the results are averaged. The number of computations is determined in a way to satisfy that the obtained results are within ±10% of actual values with a confidence interval of 90%.

5.2.3 Results

Figure 5.3 contains maximum buffer utilization values obtained by two different mapping techniques and with four different routing protocols, which are deterministic XY and adaptive west-first (WF), north-last (NL), negative-first (NF), for six different test core graphs.

Maximum buffer utilization values for the mapping obtained by the ILP_M are almost the same for each routing protocol in each case. That means, when only energy is considered as the cost function of a mapping formulation, deterministic and adaptive routing protocols have identical buffer utilization.

The main observation from Figure 5.3 is that the deterministic XY protocol has better performance compared to adaptive ones. For example in Figure 5.3.a the difference between XY and WF is 17% and the one between XY and NL is 23%. The reason for this result can be explained as follows: the main objective in buffer aware mapping is to distribute source destination pairs to NoC topology in a way that there are minimum number of intersecting paths. The method is successful on this when the paths between communicating pairs are deterministic (like XY routing). But when paths are determined randomly the method cannot distribute cores fairly. From a different point of view, our method can be regarded as to perform the main task of a routing protocol. It places communicating core pairs close to each other for minimizing the energy consumption which is identical to finding the minimum path between two points. Additionally, it also tries to separate the communication paths for minimizing buffer utilization which is the main property of an adaptive routing protocol. As a result our mapping method decreases the effect of routing protocol on NoC performance.

Total communication energy consumption values are almost the same for all routing protocols. This is because all routing protocols studied in this computational study are “minimum-path” protocols, and the fact that in our energy model, the energy consumption is only a function of hop count.

As a result, the computational study has shown that with such a mapping formulation with XY routing, which is a simple and widely used protocol, shows better buffer utilization performance compared to adaptive protocols. Adaptive protocols generate extra control traffic and are more complicated, i.e. require more energy and hardware resources.

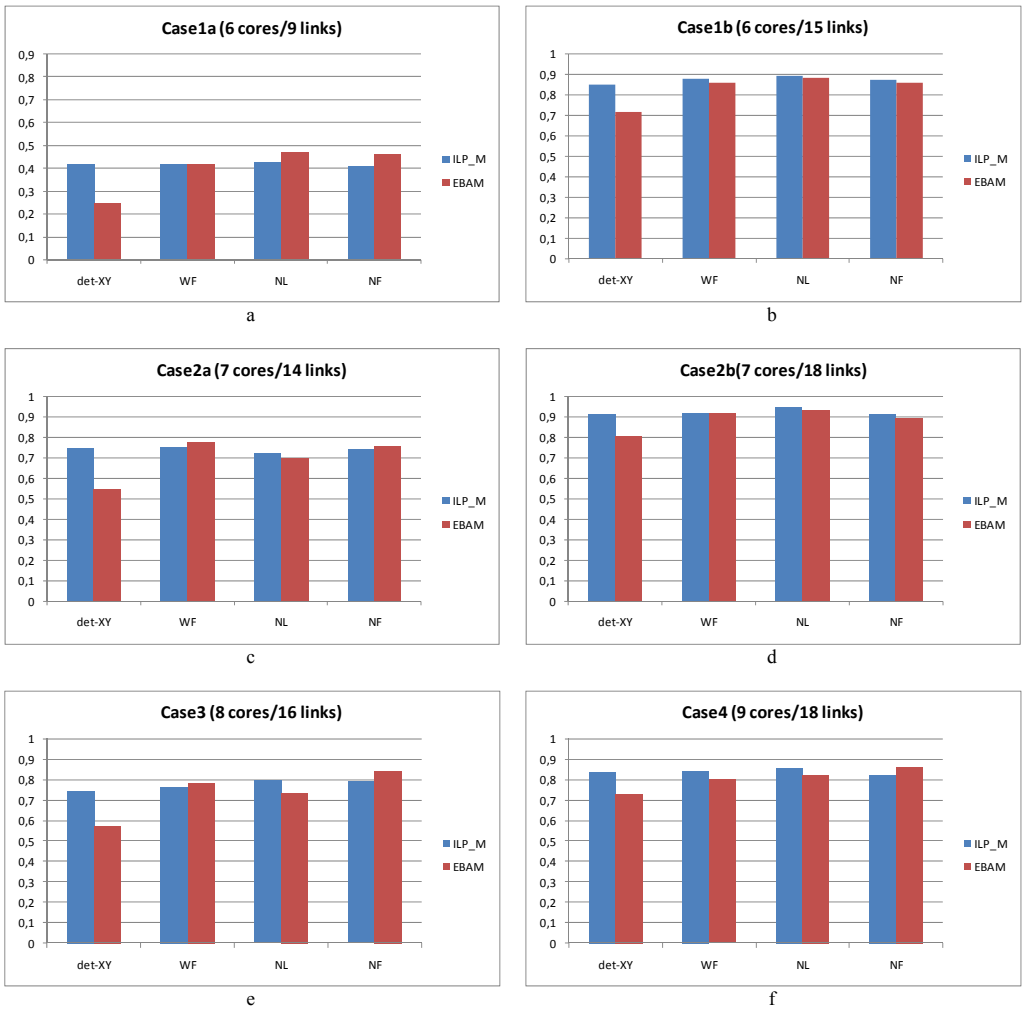


Figure 5.3: Maximum buffer utilization values obtained by two methods for various routing protocols

CHAPTER 6

SIMULATION STUDY

In previous chapters, the energy and buffer aware application mapping has been defined and it is solved by the help of genetic algorithm. The efficiency of the algorithm and the improvements of the proposed mapping technique, in terms of maximum buffer utilization, have been analyzed analytically. This section evaluates the proposed mapping algorithm by means of a simulation study and computationally. A self-similar traffic generator module is implemented on a well known NoC simulator (i.e. Nirgam [53]). The simulator is used to observe the buffer utilization of the routers in a given core graph and application mappings. Our energy and buffer aware application mapping method is compared with the ILP_M, which minimizes communication energy consumption only.

The main aim of the simulation study is to observe the buffer occupancies during the whole simulation time. By doing this, results such as the waiting times at queues and end-to-end delay will be obtained. By analyzing these results we can then compare the two mapping methods, in terms of packet drop or blocking rates that they cause, buffer utilization, probability of generating bottleneck routers, etc.

The chapter starts with a review of Fractional Brownian Motion process generation methods. Then the algorithm, which is used to generate self-similar traffic traces in our simulations, is presented in detail. The details of open source NoC simulator and our additions are given. Following the presentation of the steps of the simulation, the chapter concludes with the results of the simulation study.

6.1 Self-Similar Traffic Generation

The analytical model, on which our problem formulation is based, assumes a self-similar traffic on the on-chip network (Section 3.3). Therefore, the first and one of the most important steps of the simulation study is to generate a synthetic self-similar traffic. In some network simulators (for either on-chip or regular networks) self-similar traffic is generated by ON/OFF traffic models, which is not applicable to our case, since we have defined the self similarity using a three parameter model (namely H, a and m).

As was described in Section 3.1, mathematical models for expressing self similarity already exist in the literature. Some of these are Fractional Gaussian Noise, On/Off processes, Wavelet-based models and Fractional Brownian Motion (FBM). Buffer utilization model used in our study is based on FBM.

Generation of accurate FBM processes is crucial for the simulation of networks that does not obey traditional network models. In the literature, there are methods which can be used to generate FBM processes. These methods can be divided into two categories; exact and approximate ones. The following subsection briefly describes them.

6.1.1 Fractional Brownian Motion (FBM) Generation

There are some well known algorithms for generating stationary Gaussian processes with a given autocovariance function. The obtained process then can be used to generate an FBM sample by using different methods. These are the exact methods for generating FBM processes. The Hosking method [54] generates fractional Brownian motion samples from a fractional Gaussian noise sample by taking cumulative sums. The required computation time is of O(N^2). Another method uses the Cholesky decomposition of the covariance matrix, which results in an order O(N^3) algorithm [55]. Both methods compute the same matrix [56]. Another exact method is known as Davies and Harte method [57], which is based on finding "square root" of the covariance matrix of a stationary discrete-time Gaussian process. Its complexity is O($NlogN$) but it requires a positive definite covariance matrix.

The exact methods are not feasible due to their required CPU times [55]. There are some proposals in the literature for approximate synthetic FBM generation also. One of them, the spectral simulation [58] is a method to generate FBM processes using spectral analysis. The idea of the method is to generate a process in the frequency domain and then transform it to the time domain by using the Fast Fourier Transform (FFT). A wavelet-based generation method for FBM processes is also proposed in [59].

Another approximate method is Random Midpoint Displacement (RMD) [55]. The idea behind this algorithm is to obtain the values of the process at midpoints from the values at endpoints, by simply averaging and adding a random displacement. Beside its simple implementation and its wide usage in network traffic engineering domain, the RMD algorithm can directly be applied to our self-similar model, given in Section 3.3. Therefore, Random Midpoint Displacement algorithm is chosen for generating traffic traces of our simulation study.

6.1.2 The Random Midpoint Displacement (RMD) Algorithm

The idea behind RMD algorithm is to generate an FBM process inward, by obtaining midpoints using the values of the endpoints. For example, for generating a trace in the time interval [a, b], the value of the process at midpoint is

$$Z[\frac{a+b}{2}] = \frac{Z[a]+Z[b]}{2} + displacement. \tag{6.1}$$

The displacement is a random variable from a zero-mean Gaussian distribution, whose variance is updated at each iteration according to specific rules.

Figure 6.1 illustrates the operation of the algorithm for the first two steps. The point $d_{1,1}$ is obtained by averaging the values of d_0 and d_T and then adding a random displacement. Similarly all inner points are obtained by using their outer neighbors. For example, the value of $d_{2,1}$ is obtained by using d_0 and $d_{1,1}$. This process is repeated recursively until the desired number of points are obtained. The thick line in Figure 6.1 represents the FBM signal obtained at the end of two steps.

The obtained signal (Z(t) in Figure 6.1) is an approximate FBM trace. The cumulative arrival process, A(t), is derived as described in Section 3.1.1.

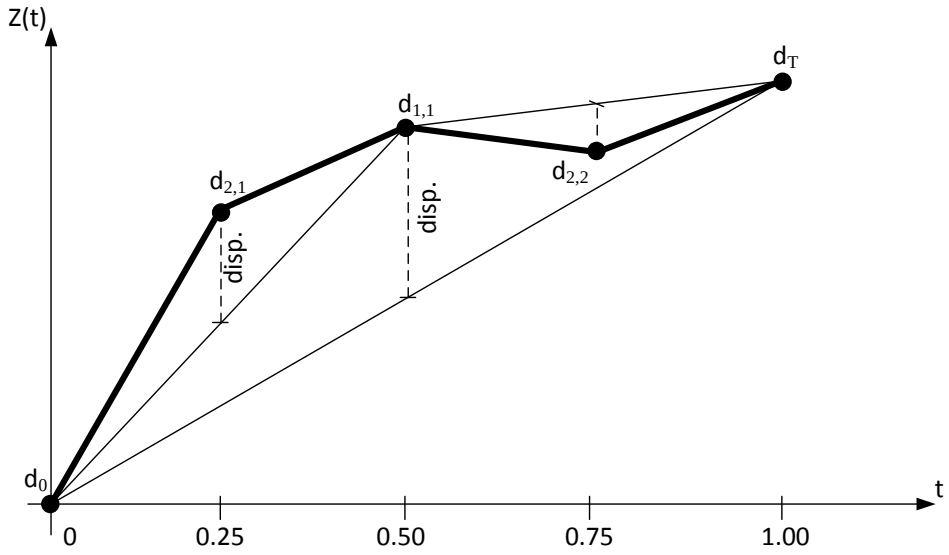


Figure 6.1: First two steps of RMD algorithm

$$A(t) = mt + \sqrt{ma}Z(t) \tag{6.2}$$

Then $\widetilde{A}(t)$, the number of arrivals in time interval [t, t+1], is given by,

$$\begin{aligned} \widetilde{A}(t) &= A(t+1) - A(t) \\ \widetilde{A}(t) &= (m(t+1) + \sqrt{ma}Z(t+1)) - (mt + \sqrt{ma}Z(t)) \\ \widetilde{A}(t) &= m + \sqrt{ma}((Z(t+1) - Z(t)) \end{aligned} \tag{6.3}$$

Finally, $\widetilde{A}(t)$ is our fundamental process constituting our packet injection task of source nodes. The process contains three parameters of our self-similar traffic model, namely, m, mean rate, a variance coefficient and H, the Hurst parameter (which is implicit in Z(t) process).

As a nature of FBM, some $\widetilde{A}(t)$ values may be negative. But since $\widetilde{A}(t)$ is interpreted as an arrival process, all values should be non-negative. Therefore, the truncated version of $\widetilde{A}(t)$ is used as described in [55]. Furthermore, since it is an approximate method, the obtained Hurst parameter of the algorithm may differ slightly from the given value, due to the extreme values of random displacements. Those cases are discarded by using the Hurst parameter estimator tool of Matlab. The generated signals whose H values are out of ±5% of the input value are not used in our simulation phase.

6.2 The NoC Simulator

Simulation is generally used for observing behavior of a network under various parameters such as communication protocols or traffic models. Similarly, in Networks-on-Chip research area also sim-

ulations are employed at different steps of the design flow. But since NoC design problems contain different aspects such as power consumption or area overhead of various entities (routers, network interfaces, etc.) well known network simulators (like ns2/ns3 or Opnet) are not suitable. Rather, a simulator implemented at Register Transfer Language (RTL) level is required.

In the last decade, several NoC simulators appeared in the NoC research area. Two of them, Noxim [60] and Nirgam [53], are widely used and observed to get lots of citations. Both of them are based on SystemC, a C++ library which is implemented for describing hardware components in simulation software. After analyzing both simulators, Nirgam is found to be more suitable for our simulation study, because of its simplicity for adding a new traffic generator.

NIRGAM (NoC Interconnect RoutinG and Applications' Modeling) is developed by University of Southampton, UK and Malaviya National Institute of Technology, India. It is a SystemC based discrete event, cycle accurate simulator which let the user experiment with different applications and routing algorithms [53]. The simulator supports mesh, torus, mesh with link failures and irregular topologies with wormhole switching mechanism. Some of the available routing algorithms are deterministic XY, source routing, deadlock free odd-even (OE), Q-routing, minimally adaptive XY and DyAd. The simulator has some synthetic traffic generators but lacks a self-similar traffic module, which is implemented and added to NIRGAM as part of this thesis work.

The Figure 6.2 shows the NIRGAM architecture [53]. The starred modules are updated for our simulation purposes. The additions made on the simulator are explained below;

- A new self-similar traffic generator module is implemented. The module takes as input the application core graph labeled with the three-parameters of our traffic model at each link. It then creates the packet inter arrival traces and injects packets according to these trace files.
- The simulator originally does not support the single source, multiple destinations case, which is frequently encountered in our test cases since number of links are greater than number of cores in our graphs. Therefore a new mechanism for sending packets from a source to multiple destinations is implemented. The transactions having the same source are combined prior to simulation run.
- Finally, during the simulation time entities that are related to our performance metrics are stored in separate log files. Those files are processed after the completion of simulations to obtain information about the number of packets waiting in buffers and end-to-end delay observed by each packet.

The simulations are run on a computer having Intel i5 processor with 4 GB of RAM and Ubuntu operating system. Each simulation lasts for at least 20000 clock cycles (equivalent to 20000 ns.) which takes 4-5 minutes for each run. Since increasing number of links in a core graph also increases the number of packets injected into the network, test cases with higher number of links (from Section 4.5) cause memory allocation problems. Therefore those cases are discarded in the simulation study.

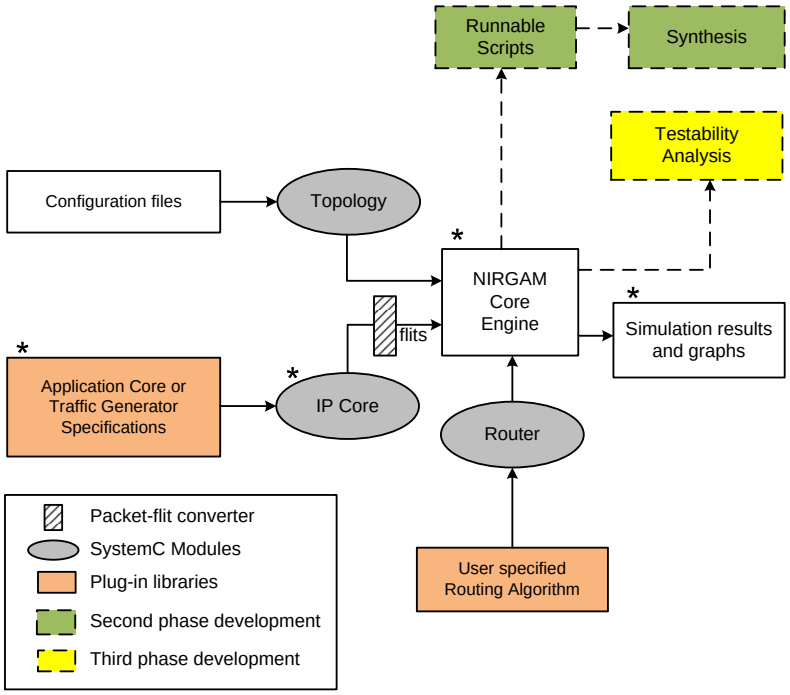


Figure 6.2: The Nirgam simulator architecture

6.3 Simulation Work

The main purpose of the simulation study is to simulate a core graph using identical traffic traces under two different mappings, i.e. a mapping found by an energy-only ILP model ("*ILP_M*") and the mapping found by our proposed algorithm. By doing this, queuing behaviors and packet delays of the same application under different mappings will be observed, and the main motivation and findings of this dissertation and the results of the computational study presented in Section 4.5.3 will be verified.

This subsection presents the details of the simulation study. The whole simulation work can be divided into 5 steps. These are

1. Generation of a core graph,
2. Mapping the core graph by using the two different algorithms under consideration,
3. Generation of synthetic traffic with the parameters obtained in step 1,
4. Running the simulation and collecting results for each mapping separately,
5. Combining the results.

Figure 6.3 visualize all simulation steps and the relation between them.

The first step of the simulation is to generate a core graph instance. The core graph contains application cores, communication links between them and the traffic parameters for each link. The number of cores

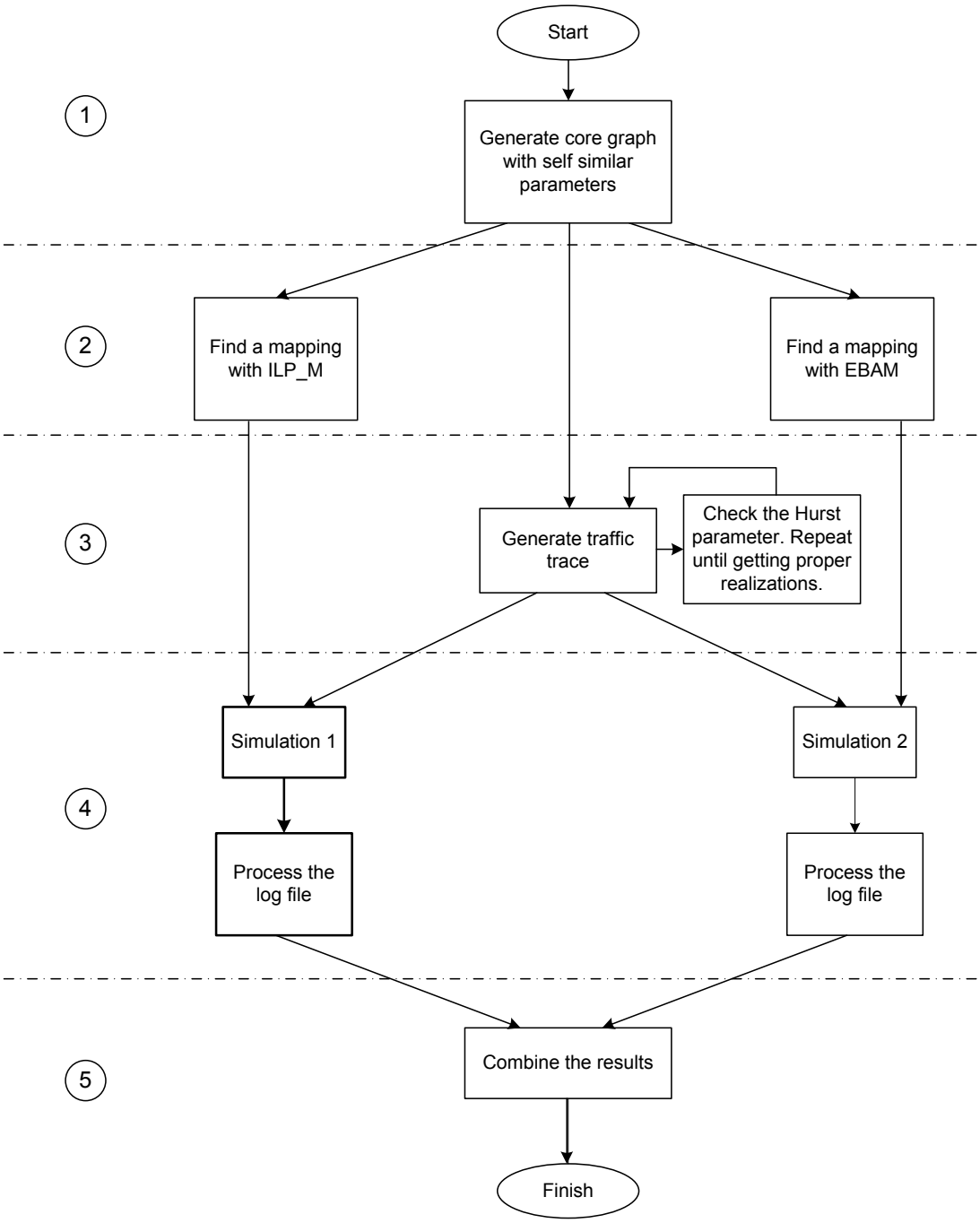


Figure 6.3: Outline of simulation work for a single core graph

and links are chosen to be consistent with the test cases presented in Section 4.5. Similarly, self-similar traffic parameters are assigned randomly within the intervals given in Section 4.5.

Once the core graph is generated, the next step is to map the graph to a 3x3 2D mesh topology, as in the case of previous chapters. At this point, again the ILP_M algorithm is assumed as the lower bound of the problem and the core graph is mapped by this algorithm. On the other hand, our energy and buffer aware application mapping algorithm is utilized to solve the same mapping problem. The two mapping solutions obtained are then used as the inputs of two separate simulation steps.

Before starting the simulations, we need to generate packet intervals. Due to the nature of the RMD algorithm, we cannot utilize it on-the-fly [55]. The packet inter arrival times are generated as was described in previous subsection by using the three parameters of the self-similar traffic as inputs. Once an FBM signal is obtained, its Hurst parameter is checked to be in ±5% of the given value. If it is not, corresponding FBM signal is discarded and a new one is generated. We observed that a proper H value is obtained in at most 10 trials.

Step 4 is the main simulation work whose inputs are

- the core graph generated at step 1,
- mapping solution found in step 2,
- traffic trace generated at step 3.

The simulation is run once for each of the two mappings. Then the simulator log files are processed separately. In the last step, results of these simulations are combined and number of packets in each source node buffer and the end-to-end delay seen by each individual packet are stored.

6.4 Results of the Simulation Study

The main aim of our application mapping algorithm is to distribute the on-chip traffic to the topology evenly and reduce the buffer utilization, while considering energy minimization, as well. The algorithm achieves this by lowering the maximum buffer utilization seen on the whole system. For doing this, we have defined a probability figure stating the probability of the number of packets in a buffer being greater than a specific value ($P(Q > b)$ in (3.6)).

In the simulation study, instead of trying to obtain a probability figure, we observe the number of packets in a buffer during the simulation period and compute the ratio of times that the packet number is greater than a specific threshold. Comparing those ratios against different mappings provides some information about the traffic distributions. The key point here is that the applied traffic is the same for both mappings. Different packet distributions under the same traffic pattern can be used to deduce the characteristics of the mapping solution.

The Nirgam simulator has a back pressure mechanism, which results in packet queues generated at source node. Although it is not the exact case in our analytical study, it provides an accurate comparison between the two mappings. Besides, observing only source nodes (nine nodes at most) makes our analysis of the simulation results easier.

In the simulator configuration, buffer size is set to 100 packets, which is 10 times greater than our

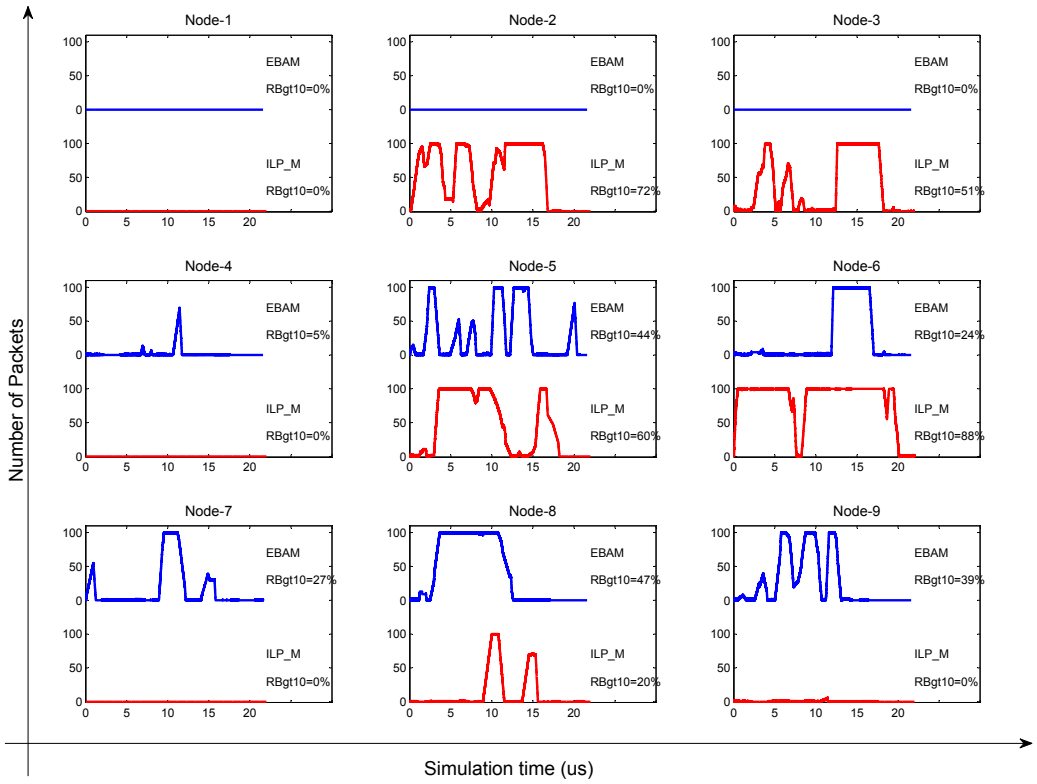


Figure 6.4: Number of packet distributions for a sample test case with 6 cores and 15 links

initial buffer size assumption. However, number of packets that is greater than 10 shall mean a buffer overflow at that router in our study.

6.4.1 Investigation of Buffer Utilization

With the help of the above discussion, first set of graphics are obtained for showing the number of packets in buffers against simulation time. Figure 6.4 gives an example for test case instances with 6 cores and 15 links. There are 9 figures, one for each tile of a 3x3 topology. Some of the figures have zero packet numbers during the whole simulation since those nodes do not have an application core (or traffic generator) assigned. Each figure has two packet distribution lines. The upper one is obtained with the EBAM mapping and the lower one is obtained with the ILP_M. Numbers next to each line gives the ratio of time portions that the packet number is higher than 10 to the whole simulation time. Hereafter this parameter will be called as *"RBgt10"* (Ratio of Buffer utilization greater than 10).

Average of all *RBgt10* values gives an insight about buffer utilization of the NoC with the mapping under concern. Table 6.1 summarizes those parameters for 10 instance of each test case group with two different mappings. These 10 instances are selected from a slightly larger set of simulation results (about 15-20 instances). At this selection, success rate of each test case group (i.e. the number of instances that EBAM gets better results) are kept constant. Results of all simulations are presented in Appendix A.1.

As was mentioned before, the aim of the simulation study is not to obtain a loss probability figure,

Table6.1: Average RBgt10 values of each test case. ((I): ILP_M, (II): EBAM)

	Core/	Instance 1		Instance 2		Instance 3		Instance 4		Instance 5		Instance 6		Instance 7		Instance 8		Instance 9		Instance 10		Average			
	Link	(I)	(II)	(I)	(II)	(I)	(II)	(I)	(II)	(I)	(II)	(I)	(II)	(I)	(II)	(I)	(II)	(I)	(II)	(I)	(II)	(I)	(II)	Diff.	Success
Case 1a	6/9	**24.3**	**17.3**	6.5	8.0	**22.3**	**18.5**	**39.0**	**21.1**	32.4	45.5	22.4	23.1	33.1	36.1	**5.1**	**4.0**	**42.1**	**17.8**	**28.9**	**15.8**	24.3	17.3	6.99%	60%
Case 1b	6/15	**35.4**	**31.5**	45.5	47.2	**23.7**	**17.4**	**56.6**	**34.5**	39.9	45.1	**29.8**	**28.4**	**34.3**	**18.8**	**75.6**	**59.3**	**48.6**	**30.9**	**26.1**	**24.5**	35.4	31.5	3.96%	80%
Case 1c	6/23	48.8	51.3	37.6	51.2	25.9	42.5	**67.5**	**61.9**	**60.0**	**51.8**	**51.2**	**50.9**	**64.2**	**38.8**	**70.7**	**70.0**	47.0	56.1	**69.6**	**60.6**	48.8	51.3	-2.55%	60%
Case 2a	7/12	**27.3**	**17.5**	**36.6**	**32.2**	**20.5**	**9.7**	**34.2**	**31.4**	**44.9**	**28.7**	**35.7**	**19.7**	**12.8**	**11.6**	**21.1**	**16.3**	16.8	23.8	**34.1**	**32.5**	27.3	17.5	9.77%	90%
Case 2b	7/14	**41.0**	**35.8**	**35.9**	**31.0**	**54.7**	**38.9**	20.8	25.2	**44.3**	**30.2**	**32.6**	**30.3**	**40.0**	**16.7**	**18.0**	**15.7**	**22.9**	**12.2**	**31.8**	**19.1**	41.0	35.8	5.16%	90%
Case 2c	7/18	**35.7**	**19.1**	60.1	77.8	**80.6**	**60.4**	**19.4**	**15.3**	42.3	43.9	**13.2**	**6.6**	31.7	37.1	**43.2**	**34.9**	**54.2**	**49.1**	**50.6**	**36.2**	35.7	19.1	16.56%	70%
Case 3a	8/14	**52.3**	**26.2**	**23.7**	**17.0**	17.5	19.4	**21.2**	**17.1**	**49.2**	**41.8**	**59.1**	**49.6**	28.5	31.1	**41.0**	**31.3**	**39.4**	**39.0**	23.9	26.9	52.3	26.2	26.14%	70%
Case 3b	8/16	**41.6**	**29.8**	**31.7**	**25.3**	34.6	50.2	**30.3**	**23.1**	37.0	51.1	**33.2**	**32.0**	**27.6**	**23.0**	**67.3**	**44.0**	**56.6**	**29.3**	**40.9**	**40.7**	41.6	29.8	11.72%	80%
Case 3c	8/20	**25.3**	**10.4**	42.2	56.1	**29.1**	**20.8**	23.5	26.5	**39.5**	**29.3**	**37.0**	**33.1**	**42.7**	**35.3**	**26.3**	**18.3**	28.2	30.3	**64.3**	**32.2**	25.3	10.4	14.97%	70%
Case 4a	9/16	**37.1**	**16.9**	**30.7**	**29.3**	**23.1**	**17.6**	**35.2**	**22.8**	**23.7**	**22.1**	26.1	27.3	37.1	52.5	**35.5**	**9.8**	37.6	39.7	31.9	34.6	37.1	16.9	20.20%	60%
Case 4b	9/18	**53.9**	**47.0**	**13.4**	**13.2**	29.1	29.4	26.1	29.0	**51.1**	**47.7**	**36.1**	**29.0**	**55.3**	**52.7**	51.6	53.8	42.3	46.2	**46.4**	**24.8**	53.9	47.0	6.89%	60%
Case 4c	9/23	**53.7**	**47.9**	**57.1**	**34.7**	30.7	31.6	**41.7**	**34.5**	53.9	58.1	**68.7**	**48.4**	29.9	35.3	**59.6**	**44.6**	**46.5**	**25.0**	56.7	61.2	53.7	47.9	5.73%	60%

which was the case in our analytical work. Instead, in this study we simulate an application core graph on an on-chip network twice with the same parameters (like the traffic traces and number of cores and links) except the mapping. Therefore the obtained results verify the positive effect of the proposed mapping solution on the buffer utilization of NoC. In Table 6.1, in most of cases the proposed method gets better results, meaning, buffer utilization of NoC's with mappings obtained by EBAM is lower than that of ILP_M.

The average RBgt10 values for all test cases decreases with EBAM mappings (difference is between 4%-26%) except case 1c. The core graphs generated in case 1c have 23 links connecting only 6 cores, i.e. the degree of the graph is near to 4, and 23 out of possible 30 links exist in the graph. This is an extreme case where the traffic load is very high. In this case, the average RBgt10 value increases only 2.5% in the EBAM case.

In all other cases there are considerable decreases in RBgt10. It is observed that in test cases with 7, 8 or 9 cores, the improvement of the proposed method is higher than that of 6-core graphs. This is a result of mapping a lower number of cores (6 cores) to a high number of routers (9 routers). In this case, a deeper analysis of mapping solutions shows that most of the paths are one-hop, since there are spare routers. In such a case, congestion on routers decreases naturally and the proposed method provides either no or very low improvements.

On the other hand, as the number of the cores gets closer to number of routers, the improvement level increases. For 7-, 8- and 9-core cases, it is possible to observe improvement levels that are greater than 20% (Table 6.1). For 9-core case, the number of successful simulations decreases to 6 (out of 10) but there are still improvements on the average RBgt10 value.

6.4.2 Investigation of End-to-End Delay

Another set of results is about the delay characteristics of the system. For doing this the end-to-end delay seen by each individual packet is extracted from the simulation log files and minimum, maximum and average values are stored. Since we utilize XY routing, a time difference between two packets of the same source-destination pair is due to queue waiting times. Therefore comparing these figures against the two mappings provides information about how the mapping technique effects the timing characteristics of an application.

Figure 6.5 is a sample for minimum and average end-to-end delays for two different mappings. The horizontal axes contains the index for source-destination pairs (i.e. links in the core graph). The bars represent the minimum and average value of the time passed between packet generation and arrival to destination node. The maximum values are not shown in figure because in case of high congestion this value reduces the readability of the figure. Besides, each figure is labeled with the average of all non-zero average end-to-end delays.

Average end-to-end delay values for 10 instances of each test case group are presented in Table 6.2. Again instances are selected in a way to provide success rate of all simulations. Results for all instances are presented in Appendix A.2. The values in Table 6.2 support the buffer utilization results found earlier (Table 6.1). Again average end-to-end packet delays obtained by EBAM mappings are observed to be lower than that of ILP_M.

The worst case is case 1c again. Even in this case the average end-to-end delay increase is only 0.8% with EBAM mappings, and in 5 out of 10 instances the proposed mapping decreases the end-to-end delays.

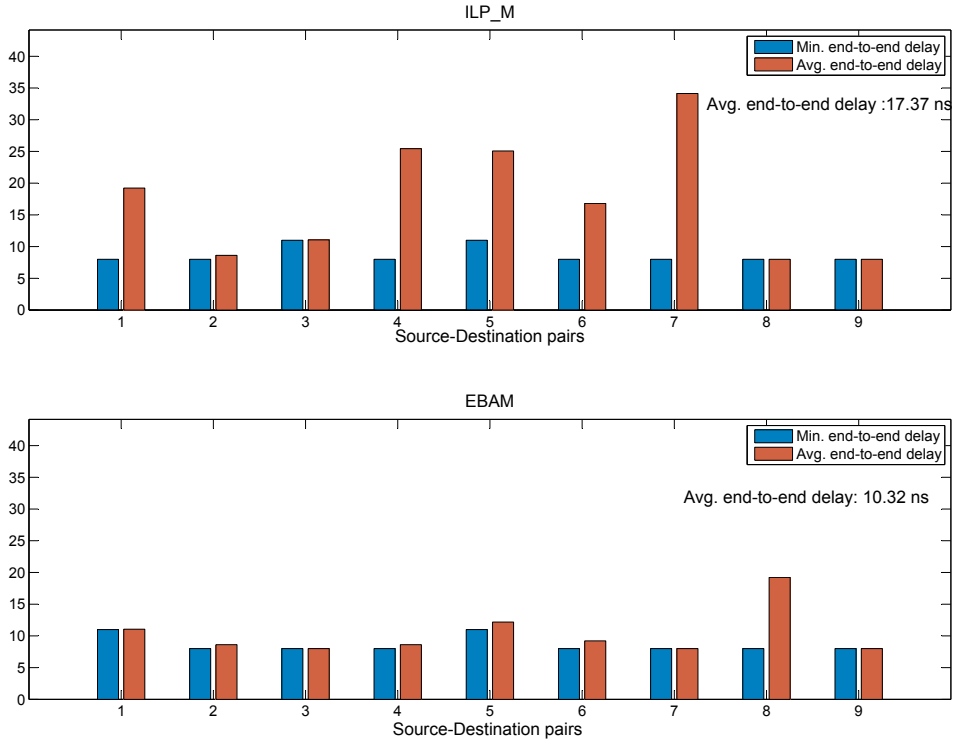


Figure 6.5: Minimum and average end-to-end packet delay for a sample core with 6 cores and 9 links

Table 6.2 shows that in two instances (instance 9 of case 3c and instance 6 of case 4b) the ILP_M mappings cause congestion somewhere in the network. In both of these cases, EBAM avoids those congestions and results in reasonable average end-to-end delays.

Table6.2: Average end-to-end delay for each test case. ((I): ILP_M. (II): EBAM)

	Core/	Instance 1		Instance 2		Instance 3		Instance 4		Instance 5		Instance 6		Instance 7		Instance 8		Instance 9		Instance 10		Average			
	Link	(I)	(II)	(I)	(II)	(I)	(II)	(I)	(II)	(I)	(II)	(I)	(II)	(I)	(II)	(I)	(II)	(I)	(II)	(I)	(II)	(I)	(II)	Impr.	Success
Case 1a	6/9	**109.5**	**72.6**	27.8	35.1	**88.5**	**52.7**	**94.7**	**51.6**	85.2	115.2	38.4	39.4	**68.4**	**53.5**	**186.0**	**98.3**	**163.8**	**55.2**	**52.3**	**32.7**	91.5	60.6	33.7%	70%
Case 1b	6/15	163.2	221.7	112.1	117.6	**67.4**	**49.2**	**755.9**	**190.8**	119.6	232.8	107.3	108.5	**120.8**	**69.4**	**230.6**	**159.9**	**280.5**	**122.1**	**101.2**	**73.4**	205.9	134.5	34.6%	60%
Case 1c	6/23	**276.8**	**189.5**	304.5	416.6	82.4	173.1	145.9	147.9	236.7	309.2	**218.5**	**174.8**	**244.3**	**198.4**	**227.4**	**214.0**	**221.5**	**151.0**	194.7	195.8	215.3	217.0	-0.8%	50%
Case 2a	7/12	**95.6**	**67.5**	**175.0**	**125.1**	**88.0**	**31.3**	**146.0**	**100.8**	**266.3**	**142.1**	**52.8**	**26.7**	**110.7**	**56.0**	28.8	29.2	61.7	67.9	**117.6**	**78.5**	114.2	72.5	36.5%	80%
Case 2b	7/14	**199.7**	**134.5**	153.9	160.5	**302.0**	**113.3**	**295.1**	**94.8**	60.3	70.4	69.5	79.7	**228.3**	**55.3**	65.6	72.2	**92.2**	**62.7**	107.4	115.3	157.4	95.9	39.1%	50%
Case 2c	7/18	**191.2**	**98.3**	237.3	316.0	**270.9**	**215.9**	33.1	36.6	174.5	198.3	104.0	119.5	**202.2**	**157.1**	**194.4**	**161.8**	**268.1**	**215.9**	**206.2**	**112.0**	188.2	163.1	13.3%	60%
Case 3a	8/14	46.5	51.9	**246.0**	**189.9**	**318.3**	**235.5**	68.7	82.9	**201.6**	**126.9**	**163.6**	**83.9**	80.6	86.3	**86.4**	**65.6**	**104.0**	**95.9**	**104.5**	**89.9**	142.0	110.9	21.9%	70%
Case 3b	8/16	**148.7**	**62.4**	85.9	90.4	**42.2**	**4.5**	100.7	116.9	132.6	157.9	**116.3**	**106.0**	**243.8**	**198.8**	**115.5**	**97.6**	**183.6**	**178.0**	114.4	150.2	128.4	116.3	9.4%	60%
Case 3c	8/20	**124.8**	**49.8**	**135.9**	**88.6**	**121.3**	**109.9**	233.9	446.7	**250.3**	**147.3**	**188.1**	**101.1**	**79.0**	**50.1**	95.6	96.5	**1431.6**	**118.6**	74.1	83.0	273.5	129.2	52.8%	70%
Case 4a	9/16	**199.5**	**64.2**	**166.1**	**110.2**	**75.3**	**52.7**	**78.7**	**64.3**	62.1	84.6	140.3	140.3	104.5	138.8	**111.3**	**41.0**	92.5	141.4	**165.3**	**116.9**	119.6	95.5	20.2%	60%
Case 4b	9/18	46.8	67.9	**251.2**	**147.4**	**169.9**	**136.0**	140.8	143.1	35.0	47.2	**1144.2**	**167.3**	63.0	85.1	**278.3**	**198.3**	95.7	108.6	**214.3**	**116.4**	243.9	121.7	50.1%	50%
Case 4c	9/23	167.2	184.7	**261.1**	**124.4**	124.9	142.7	**132.8**	**101.9**	**185.8**	**128.7**	182.7	187.2	**249.6**	**161.5**	**120.1**	**93.0**	**209.7**	**179.2**	207.2	235.6	184.1	153.9	16.4%	60%

CHAPTER 7

CONCLUSION

This chapter concludes the dissertation. A brief summary of the thesis work and its contributions are presented. It is then followed by future directions and extensions for current study and also for the application mapping problem of NoC design flow.

7.1 Conclusion

In this study, the application mapping problem of the NoC design flow is tackled from a networking point of view. The main motivation is to find a mapping solution such that the obtained on-chip network system has better characteristics in terms of buffer utilization or average number of packets in buffers. In other words, solutions that degenerate network performance are aimed to be avoided by using a priori traffic characteristics of the application under concern.

In order to implement traffic aware application mapping, first of all an on-chip network traffic model is required. Self similarity has been proved to model computer networks better than Poisson-based models. In previous works, it is also shown that on-chip traffics have self-similar characteristics. Therefore in this thesis, a buffer utilization analysis under self-similar traffic is performed. Self similarity is a phenomenon which is hard to express mathematically. Therefore FBM process is used to express self similarity and average number of packets in a buffer and buffer utilization figures are obtained under infinite buffer length assumption.

Before an application mapping problem formulation is obtained the effect of self-similar model parameters on network performance is analyzed first. Results showed that mapping solutions which minimize the energy consumption may cause some congested buffers and degenerate the network performance, especially in dense networks or under heavy traffic loads.

Seeing that possible effect, a novel application mapping formulation, which minimizes the energy consumption while considering the network dynamics, is proposed. In order to solve this intractable problem, a genetic algorithm based solution is implemented. Application of the method to various test cases containing different core graphs and communication requirements demonstrates potential improvements on network performance in terms of buffer utilization. In order to evaluate the proposed genetic algorithm, an ILP model is also implemented within the scope of this study and is used as a lower bound.

In order to evaluate the energy overhead of our algorithm, results of the ILP_M and the EBAM-energy (two methods that use the same cost function) are compared. The energy consumption of both methods are observed to be very close. In some cases both methods generate the same result, which shows that

the implemented genetic algorithm is efficient and successful.

After a series of executions, we observed that our method is capable of solving complex mapping problems successfully in a reasonable amount of time. We also observed that mapping solutions obtained by only energy minimization has worse network performance, which emphasizes the necessity of a mapping model that addresses network dynamics besides energy. The proposed method improves the buffer utilization figure with a reasonable energy trade-off and is suitable for applications with QoS requirements.

The initial motivation of the dissertation is also verified by mean of a simulation study. A self-similar traffic generator, based on a three-parameter FBM model, was implemented. Execution of simulations showed that NoCs mapped with the proposed algorithm have better performance in terms of buffer utilization or end-to-end delay. In some simulations the algorithm was proved to avoid congestion states that was created by energy-only mapping algorithm.

The proposed method is also evaluated under various routing algorithms. XY routing algorithm is a widely used one in NoC area due to its simplicity and low overhead. However, it has a degenerate performance when compared with adaptive routing protocols. Our computational study performed on various routing protocols showed that the proposed mapping algorithm increases the performance of XY routing. This is because of the twofold optimization mechanism in the algorithm; minimizing energy at mapping stage is identical to finding the minimum path between nodes. On the other hand, distributing the traffic evenly and avoiding congestion corresponds to the main task of an adaptive algorithm. Therefore, together the proposed algorithm makes the routing protocol's operation easier, and so, a simple routing protocol, such as XY, can show a better performance.

7.2 Future Work

Future directions for the dissertation contain application of the EBAM to different NoC topologies. The method should be extended to cover 3D NoC architectures, which is a promising research area. Efficiency of the proposed method should be evaluated on NoC benchmarks or real NoC applications. This requires some further study for the estimation of self-similar model parameters.

The present study is an example for taking network related issues into consideration at earlier stages of the NoC design flow. In this manner, some other steps of the design flow, such as buffer size selection, routing protocol or congestion/flow control, can be improved or simplified.

Most of the SoCs today perform multiple functions, i.e. the application core graph may change on-the-fly. This may also require the updating of the mapping dynamically. So, the work presented in this dissertation should be extended to be able to change the mapping dynamically, with respect to varying communication requirements or the core graph.

Self similarity is an essential phenomenon in both computer and on-chip networks. Therefore, self-similar traffic parameter estimation by means of simulation or emulation will be very useful for improving the current study and also for other network related issues of NoC design like routing, congestion or flow control.

REFERENCES

[1] T. Bjerregaard and S. Mahadevan. A survey of research and practices of network-on-chip. *ACM Computing Surveys*, 38(1), 2006.

[2] C.L. Chou, U.Y. Ogras, and R. Marculescu. Energy- and performance-aware incremental mapping for networks on chip with multiple voltage levels. *IEEE Tran. on Computer-Aided Design of Integrated Circuits and Systems*, 27(10):1866–1879, 2008.

[3] K. Lee, S. Lee, and H. Yoo. Low-power network-on-chip for high-performance SoC design. *IEEE Trans. on Very Large Scale Integration (VLSI) Systems*, 14(2):148–160, 2006.

[4] S. Kumar, A. Jantsch, J. Soininen, M. Forsell, M. Millberg, J. Öberg, K. Tiensyrjä, and A. Hemani. A network on chip architecture and design methodology. In *Proceedings of IEEE Computer SoCiety Annual Symposium on VLSI*, April 2002.

[5] H.J. Yoo, K. Lee, and J.K. Kim. *Low-power NoC for high-performance SoC design*. System-on-chip design and technologies. CRC Press, 2008.

[6] W. J. Dally and B. Towles. Route Packets, Not Wires: On-Chip Interconnection Networks. In *Proc. of the 38th Design Automation Conference (DAC)*, June 2001.

[7] L. Benini and G. De Micheli. Networks on chips: A new SoC paradigm. *Computer*, 35:70–78, 2002.

[8] N. E. Jerger and L. Peh. *On-Chip Networks*. Morgan and Claypool, 2009.

[9] W. J. Dally and B. Towles. *Principles and Practices of Interconnection Networks*. Morgan Kaufmann, 2003.

[10] P. Pratim 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*, 54(8):1025–1040, 2005.

[11] R. Marculescu and U. Y. Ogras. Outstanding research problems in NoC design: system, microarchitecture, and circuit perspectives. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, 28(1):3–21, January 2009.

[12] T. Lei and S. Kumar. A two-step genetic algorithm for mapping task graphs to a network on chip architecture. In *Proc. of the Euromicro Symp. on Digital System Design (DSD03)*, pages 180–187, 2003.

[13] J. Hu and R. Marculescu. Energy-aware mapping for tile-based NoC architectures under performance constraints. In *Proc. of the Asia and South Pacific Design Automation Conference*, pages 233–239, 2003.

[14] J. Hu and R. Marculescu. Energy- and performance-aware mapping for regular NoC architectures. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, 24(4), April 2005.

[15] S. Murali and G. De Micheli. Bandwidth-constrained mapping of cores onto NoC architectures. In *Proc. of the Design Automation and Test in Europe Conference and Exhibition (DATE 04)*, pages 896–901, 2004.

[16] S. Murali and G. De Micheli. Sunmap: A tool for automatic topology selection and generation for NoCs. In *Proc. of the 41st Design Automation Conference (DAC 04)*, pages 914–919, 2004.

[17] S. Murali, L. Benini, and G. De Micheli. Mapping and physical planning of NoC arch with qos guarantees. In *Proc. of the Design Automation Conference (DAC'05)*, pages 27–32, 2005.

[18] K. Srinivasan and K. S. Chatha. A technique for low energy mapping and routing in network-on-chip architectures. In *International Symposium on Low Power Electronics and Design*, pages 387–392, 2005.

[19] K. Srinivasan, K. S. Chatha, and G. Konjevod. Linear programming based techniques for synthesis of network-on-chip architectures. *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, 14(4):407–420, 2006.

[20] C.L. Chou and R. Marculescu. Contention-aware application mapping for NoC communication architecture. In *Proc. of the IEEE Int. Conf. on Computer Design*, pages 164–168, 2008.

[21] G. Ascia, V. Catania, and M. Palesi. Multi-objective mapping for mesh-based NoC architectures. In *Second IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis*, pages 182–187, Stockholm, Sweden, September 8–10 2004.

[22] J. Hu and R. Marculescu. Communication and task scheduling of application-specific networks-on-chip. In *IEE Proceedings of Computers and Digital Techniques*, pages 643–651, 2005.

[23] W. Hung, C. Addo-Quaye, T. Theocharides, Y. Xie, N. Vijaykrishnan, and M. J. Irwin. Thermal-aware IP virtualization and placement for networks-on-chip architecture. In *Proc. of the Intl. Conf. on Computer Design (ICCD)*, 2004.

[24] C. Addo-Quaye. Thermal-aware mapping and placement for 3-d NoC designs. In *IEEE International SoC Conference*, pages 25–28, 2005.

[25] P.K. Hamedani, S. Hessabi, H. Sarbazi-Azad, and N. E. Jerger. Exploration of temperature constraints for thermal aware mapping of 3d NoC. In *Proc. of the 20th Euromicro Int. Conf. on Parallel Distributed and Network-Based Computing (PDP'12)*, 2012.

[26] S. Tosun. New heuristic algorithms for energy aware application mapping and routing on mesh-based NoCs. *Journal of Systems Architecture [Special Issue On-Chip Parallel And Network-Based Systems]*, 57(1):69–78, 2011.

[27] P. K. Sahu and S. Chattopadhyay. A survey on application mapping strategies for network-on-chip design. *Journal of Systems Architecture*, 59(1):60–76, 2013.

[28] K. Park and W. Willinger. *Self-Similar Network Traffic and Performance Evaluation*. New York: John Wiley and Sons, 2000.

[29] W. Leland, M. Taqqu, W. Willinger, and D.V. Wilson. On the self similar nature of ethernet traffic (extended version). *IEEE/ACM Trans. Networking*, 2:1–15, February 1994.

[30] G. Varatkar and R. Marculescu. On-chip traffic modeling and synthesis for mpeg-2 video applications. *IEEE Trans. on Very Large Scale Integration (VLSI) Systems*, 12(1):108–119, January 2004.

[31] P. Bogdan and R. Marculescu. Statistical physics approaches for network-on-chip traffic characterization. pages 461–470, 2009.

[32] V. Soteriou, H. Wang, and L. Peh. A statistical traffic model for on-chip interconnection networks. In *IEEE Int. Symp. on Modeling Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS'06)*, pages 104–116, 2006.

[33] A. Scherrer, A. Fraboulet, and T. Risset. Long-range dependence and on-chip processor traffic. *Microprocessors and Microsystems*, pages 72–80, 2008.

[34] Y. Qian, Z. Lu, and W. Dou. Applying network calculus for performance analysis of self-similar traffic in on-chip networks. In *Proc. of the 7th IEEE/ACM international conference on Hardware/software codesign and system synthesis, CODES+ISSS '09*, pages 453–460, 2009.

[35] Y. Qian. A self-similar traffic model for network-on-chip performance analysis using network calculus. *Advanced Topics in Multimedia Research*, pages 3–19, 2012.

[36] I. Norros. On the use of fractional brownian motion in the theory of connectionless networks. *IEEE J. Sel. Areas Commun.*, 13(6), 1995.

[37] J. Sepulveda, M. Strum, W.J. Chau, and R. Pires. The lrd traffic impact on the NoC-based SoCs. In *Proceedings of the 23rd symposium on Integrated circuits and system design*, pages 97–102, 2010.

[38] J. H. Bahn and N. Bagherzadeh. A generic traffic model for on-chip interconnection networks. In *NoCArc International Workshop on Network on Chip Architectures*, pages 41–49, 2008.

[39] S. Kundu and K. Manna. A comparative performance evaluation of network-on-chip architectures under self-similar traffic. In *Proceedings of the International Conference on Advances in Recent Technologies in Communication and Computing*, pages 278–287, 2009.

[40] C. Wang, W. Hu, and N. Bagherzadeh. Area and power-efficient innovative congestion-aware network-on-chip architecture. *Journal of Systems Architecture*, 57(1):24–38, 2011.

[41] C. Çelik and C.F. Bazlamaçcı. Effect of application mapping on networks-on-chip performance. In *Proc. of the 20th Euromicro Int. Conf. on Parallel Distributed and Network-Based Computing (PDP'12)*, 2012.

[42] Z. Shuo, Z. Rongcai, and A. Ke. On generating self-similar network traffic using multi-core processors. In *Proc. of the Int. Symp. on Computer Science and Computational Technology*, pages 667–672, 2008.

[43] I. Norros. A storage model with self-similar input. *Queueing Systems*, 16(3-4):387–396, 1994.

[44] S. Kasahara. Internet traffic modeling: Markovian approach to self-similar traffic and prediction of loss probability for finite queues. *IEICE Transactions on Communications*, E84-B(8):2134–2141, 2001.

[45] S. Teymori and W. Zhuang. Finite buffer queue analysis and scheduling for heavy-tailed traffic in packet-switchingwireless networks. In *Quality of Service in Heterogeneous Wired/Wireless Networks, International Conference on*, page 19, 2005.

[46] D.P. Heyman and T.V. Lakshman. What are the implications of long range dependence for vbr video traffic engineering? *IEEE/ACM Transactions on Networking*, 4(3):301–317, 1996.

[47] G. Mao. Finite timescale range of interest for self similar traffic measurements, modelling and performance analysis. In *11th IEEE International Conference on Networks, ICON2003.*, pages 7–12, 2003.

[48] T. Xianhai, H. Yuanhui, and J. Weidong. Modeling and performance analysis of self-similar traffic based on fbm. In *IFIP International Conference on Network and Parallel Computing*, pages 543–548, 2007.

[49] B.N. Bashforth and C.L. Williamson. Statistical multiplexing of self-similar video streams: simulation study and performance results. In *Sixth International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems*, pages 119–126, 1998.

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

[51] http://lpsolve.sourceforge.net/5.5/ (last visited on December 2012).

[52] T. Xianhai, H. Yuanhui, and J. Weidong. Modeling and performance analysis of self-similar traffic based on fbm. In *Proc. of Int. Conf. on Network and Parallel Computing(IFIP)*, pages 543–548, 2007.

[53] M.S. Gaur V. Laxmi A. Narayanan L. Jain, B.M. Al-Hashimi. Nirgam: a simulator for NoC interconnect routing and application modeling. In *Workshop on Diagnostic Services in Network-on-Chips, Design, Automation and Test in Europe Conference (DATE' 07)*, pages 16–20, 2007.

[54] J.R.M. Hosking. Modeling persistence in hydrological time series using fractional differencing. *Water Resources Research*, pages 1898–1908, 1984.

[55] Wing-Cheong Lau, A. Erramilli, J.L. Wang, and W. Willinger. Self-similar traffic generation: the random midpoint displacement algorithm and its properties. In *Communications, 1995. ICC '95 Seattle, 'Gateway to Globalization', 1995 IEEE International Conference on*, volume 1, pages 466–472, 1995.

[56] A.B. Dieker and M. Mandjes. On spectral simulation of fractional brownian motion. *Probability in the Engineering and Informational Sciences archive*, 17(3):417–434, 2003.

[57] R.B. Davies and D.S. Harte. Tests for hurst effect. *Biometrika*, 74(1):95–101, 1987.

[58] V. Paxson. Fast, approximate synthesis of fractional gaussian noise for generating self-similar network traffic. *Computer Communication Review*, 27(5):5–18, 1998.

[59] P. Abry and F. Sellan. The wavelet-based synthesis for the fractional brownian motion proposed by f. sellan and y. meyer: Remarks and fast implementation. *Applied and Computational Harmonic Analysis*, 3(4):377–383, 1996.

[60] http://noxim.sourceforge.net/.

APPENDIX A

RESULTS OF SIMULATION STUDY

A.1 Results of Buffer Usage Analysis

In this section, simulation results on buffer usage are presented. Following tables contain average RBgt10 values of the ILP_M and the EBAM mappings for each simulation instances. The values of instances in which the proposed method improves the performance are presented in bold.

ID	RBgt10 ILP	RBgt10 ELAM
1000	**24,31**	**17,31**
1003	6,49	7,96
1004	**22,34**	**18,50**
1005	**39,00**	**21,12**
1006	32,37	45,53
1007	22,38	23,10
1009	21,90	24,42
1010	10,60	11,81
1011	**34,38**	**31,82**
1013	**29,94**	**22,67**
1015	**67,03**	**41,69**
1016	**51,48**	**38,97**
1018	33,12	36,10
1027	**5,09**	**3,97**
1028	**42,12**	**17,82**
1029	**28,90**	**15,82**
Average	29,47	23,66

Case 1a	
Number of simulations	16
Number of successful cases	10
Success rate	0,63

ID	RBgt10 ILP	RBgt10 ELAM
2001	**35,42**	**31,46**
2006	45,55	47,20
2011	**16,85**	**15,61**
2015	**23,67**	**17,37**
2016	**56,58**	**34,47**
2018	39,89	45,05
2022	**29,76**	**28,36**
2030	**34,27**	**18,78**
2036	**75,61**	**59,32**
2038	**48,56**	**30,92**
2047	**26,13**	**24,48**
Average	39,30	32,09

Case 1b	
Number of simulations	11
Number of successful cases	9
Success rate	0,82

ID	RBgt10 ILP	RBgt10 ELAM
3048	37,57	51,18
3055	25,93	42,54
3064	**43,37**	**42,97**
3067	56,62	64,91
3071	**67,47**	**61,93**
3075	**59,97**	**51,83**
3087	**51,20**	**50,89**
3112	**64,20**	**38,81**
3128	**70,73**	**69,96**
3140	46,98	56,14
3152	**69,60**	**60,62**
Average	53,97	53,80

Case 1c	
Number of simulations	12
Number of successful cases	7
Success rate	0,58

ID	RBgt10 ILP	RBgt10 ELAM
4001	**36,56**	**32,20**
4005	**20,49**	**9,70**
4012	**34,17**	**31,44**
4015	**44,93**	**28,74**
4016	**47,33**	**27,33**
4017	**51,33**	**35,03**
4018	**39,52**	**34,84**
4026	**28,90**	**26,26**
4027	**19,83**	**6,16**
4030	21,03	35,84
4035	**35,69**	**19,72**
4040	**12,83**	**11,58**
4041	**21,09**	**16,27**
4044	16,84	23,75
4048	**34,09**	**32,48**
Average	30,98	24,76

Case 2a

Number of simulations	16
Number of successful cases	14
Success rate	0,88

ID	RBgt10 ILP	RBgt10 ELAM
5002	**35,86**	**30,97**
5005	**54,74**	**38,90**
5009	49,26	59,92
5013	**40,63**	**38,01**
5018	**29,65**	**19,66**
5023	**64,89**	**29,84**
5024	20,76	25,21
5026	**44,29**	**30,22**
5031	**32,62**	**30,29**
5034	**39,96**	**16,75**
5035	**36,36**	**34,61**
5038	**18,04**	**15,69**
5043	**22,95**	**12,20**
5047	**31,79**	**19,14**
Average	37,27	28,67

Case 2b

Number of simulations	15
Number of successful cases	13
Success rate	0,87

ID	RBgt10 ILP	RBgt10 ELAM
6018	60,08	77,77
6024	**80,64**	**60,42**
6027	**19,42**	**15,26**
6032	42,34	43,86
6034	**26,78**	**25,23**
6038	**13,20**	**6,61**
6041	31,68	37,06
6044	**43,20**	**34,86**
6062	**54,15**	**49,06**
6073	**50,60**	**36,22**
Average	42,21	38,64

Case 2c	
Number of simulations	11
Number of successful cases	8
Success rate	0,73

ID	RBgt10 ILP	RBgt10 ELAM
7007	**23,73**	**17,03**
7008	17,55	19,39
7012	**21,18**	**17,11**
7015	**49,18**	**41,81**
7016	**59,06**	**49,55**
7017	23,19	25,58
7018	**46,91**	**46,56**
7019	28,53	31,08
7023	**58,76**	**58,29**
7026	**41,02**	**31,29**
7031	26,08	33,72
7033	**39,45**	**39,01**
7036	**28,92**	**26,49**
7044	**9,36**	**5,43**
7048	**29,68**	**27,18**
7049	23,88	26,86
Average	32,90	31,02

Case 3a	
Number of simulations	17
Number of successful cases	12
Success rate	0,71

ID	RBgt10 ILP	RBgt10 ELAM
8001	**31,73**	**25,33**
8006	34,55	50,19
8007	**30,31**	**23,10**
8008	37,03	51,09
8010	**44,23**	**41,73**
8016	**25,13**	**25,10**
8022	**24,08**	**22,75**
8023	**33,18**	**31,97**
8025	52,44	55,48
8026	**27,62**	**23,00**
8060	**67,29**	**43,96**
8072	**56,59**	**29,26**
8074	**40,86**	**40,66**
Average	38,85	35,66

Case 3b

Number of simulations	14
Number of successful cases	11
Success rate	0,79

ID	RBgt10 ILP	RBgt10 ELAM
9004	42,23	56,14
9007	**29,08**	**20,77**
9008	13,98	27,18
9012	23,48	26,48
9016	**24,94**	**24,19**
9023	**39,46**	**29,32**
9027	**37,04**	**33,12**
9039	**58,26**	**56,24**
9050	59,82	64,30
9052	**42,69**	**35,27**
9060	**38,01**	**36,16**
9066	**26,34**	**18,30**
9069	28,20	30,31
9074	**64,31**	**32,22**
9079	**27,10**	**24,28**
Average	37,00	34,28

Case 3c

Number of simulations	16
Number of successful cases	11
Success rate	0,69

ID	RBgt10 ILP	RBgt10 ELAM
10007	**37,13**	**16,93**
10018	**19,84**	**13,06**
10021	**30,74**	**29,33**
10025	**23,12**	**17,58**
10027	**35,23**	**22,78**
10028	**23,73**	**22,13**
10032	26,09	27,27
10033	37,12	52,48
10044	**35,51**	**9,81**
10050	37,55	39,73
10056	31,90	34,59
Average	30,72	25,97

Case 4a

Number of simulations	12
Number of successful cases	8
Success rate	0,67

ID	RBgt10 ILP	RBgt10 ELAM
11002	**13,37**	**13,23**
11009	29,05	29,41
11022	26,12	28,98
11029	**51,08**	**47,67**
11030	**36,10**	**29,01**
11034	**46,05**	**44,52**
11042	**55,33**	**52,68**
11045	27,59	32,09
11059	51,63	53,82
11064	42,28	46,22
11080	**46,36**	**24,77**
Average	38,63	36,58

Case 4b

Number of simulations	12
Number of successful cases	7
Success rate	0,58

ID	RBgt10 ILP	RBgt10 ELAM
12014	**57,10**	**34,73**
12015	32,39	39,46
12022	30,70	31,65
12026	**41,71**	**34,53**
12027	53,94	58,07
12033	**68,70**	**48,36**
12038	29,91	35,28
12043	**59,62**	**44,64**
12055	**46,49**	**24,99**
12056	**44,56**	**32,71**
12061	**40,25**	**37,17**
12062	56,65	61,21
12067	33,51	49,23
Average	45,81	40,93

Case 4c	
Number of simulations	14
Number of successful cases	8
Success rate	0,57

A.2 Results of End-to-End Delay Analysis

In this section, simulation results on end-to-end delay are presented. Following tables contain average end-to-end delay values of both mapping methods for each simulation instances. The values of instances in which the proposed method improves the performance are presented in bold.

ID	Avg. Delay (ns) ILP	Avg. Delay (ns) ELAM
1000	**109,50**	**72,56**
1003	27,80	35,07
1004	**88,48**	**52,67**
1005	**94,70**	**51,64**
1006	105,59	107,26
1009	85,23	115,23
1010	38,42	39,36
1011	**65,50**	**56,43**
1012	**14,88**	**8,86**
1013	**68,41**	**53,50**
1015	**186,04**	**98,28**
1016	**129,70**	**116,56**
1020	41,39	48,62
1027	**17,37**	**10,32**
1028	**163,77**	**55,16**
1029	**52,26**	**32,68**
Average	**80,56**	**59,64**

Case 1a	
Number of simulations	16
Number of successful cases	11
Success rate	0,69

ID	Avg. Delay (ns) ILP	Avg. Delay (ns) ELAM
2001	163,23	221,68
2006	112,07	117,56
2011	**67,41**	**49,16**
2015	**81,98**	**76,20**
2016	**755,91**	**190,78**
2018	119,63	232,76
2022	107,29	108,54
2030	**120,80**	**69,41**
2036	**230,58**	**159,91**
2038	**280,50**	**122,10**
2047	**101,18**	**73,44**
Average	**194,60**	**129,23**

Case 1b

Number of simulations	11
Number of successful cases	7
Success rate	0,64

ID	Avg. Delay (ns) ILP	Avg. Delay (ns) ELAM
3001	**276,84**	**189,45**
3032	304,54	416,62
3055	82,43	173,07
3064	145,91	147,94
3067	236,65	309,19
3071	**218,52**	**174,79**
3075	**244,26**	**198,37**
3087	**227,43**	**214,03**
3112	**221,49**	**151,04**
3128	380,48	380,48
3140	194,73	195,80
3152	**305,58**	**300,88**
Average	**236,57**	**237,64**

Case 1c

Number of simulations	12
Number of successful cases	6
Success rate	0,50

ID	Avg. Delay (ns) ILP	Avg. Delay (ns) ELAM
4000	**95,59**	**67,46**
4001	**175,04**	**125,13**
4005	**87,96**	**31,25**
4008	**145,96**	**100,80**
4011	**105,02**	**85,09**
4012	**125,20**	**116,15**
4016	**149,51**	**103,29**
4017	**266,27**	**142,09**
4027	**52,83**	**26,69**
4036	**110,69**	**56,04**
4037	**94,34**	**87,69**
4040	28,84	29,22
4041	61,68	67,86
4042	**117,58**	**78,51**
4044	54,61	74,38
4048	**82,99**	**74,77**
Average	**109,63**	**79,15**

Case 2a

Number of simulations	16
Number of successful cases	13
Success rate	0,81

ID	Avg. Delay (ns) ILP	Avg. Delay (ns) ELAM
5001	**199,69**	**134,51**
5002	153,94	160,53
5005	**301,98**	**113,34**
5018	**115,45**	**88,83**
5023	**295,09**	**94,82**
5024	75,86	146,44
5026	60,31	70,36
5028	69,55	79,68
5031	87,06	144,30
5034	**228,31**	**55,28**
5035	**131,12**	**127,04**
5038	65,64	72,22
5043	**92,23**	**62,71**
5047	**63,94**	**51,14**
5048	107,45	115,26
Average	**136,51**	**101,10**

Case 2b

Number of simulations	15
Number of successful cases	8
Success rate	0,53

ID	Avg. Delay (ns) ILP	Avg. Delay (ns) ELAM
6004	**191,20**	**98,27**
6018	237,27	315,96
6024	**270,93**	**215,88**
6027	33,10	36,61
6032	174,52	198,27
6034	104,01	119,54
6038	**24,39**	**20,53**
6041	**202,18**	**157,15**
6044	**194,43**	**161,84**
6062	**268,08**	**215,91**
6073	**206,16**	**111,95**
Average	**173,30**	**150,17**

Case 2c	
Number of simulations	11
Number of successful cases	7
Success rate	0,64

ID	Avg. Delay (ns) ILP	Avg. Delay (ns) ELAM
7004	63,09	99,60
7007	**78,50**	**57,23**
7008	46,48	51,86
7012	39,53	70,36
7015	**245,98**	**189,91**
7016	**318,26**	**235,48**
7017	68,70	82,87
7018	**238,04**	**209,88**
7019	**117,73**	**110,57**
7023	**201,56**	**126,86**
7026	**163,64**	**83,89**
7031	80,59	86,34
7033	**181,32**	**170,52**
7036	**86,37**	**65,59**
7044	**34,19**	**21,63**
7048	**104,02**	**95,94**
7049	**104,53**	**89,93**
Average	**127,80**	**108,73**

Case 3a	
Number of simulations	17
Number of successful cases	12
Success rate	0,71

ID	Avg. Delay (ns) ILP	Avg. Delay (ns) ELAM
8000	**148,73**	**62,39**
8001	85,88	90,36
8003	**42,19**	**4,53**
8006	90,96	127,68
8007	100,73	116,89
8008	132,58	157,88
8016	**82,64**	**79,32**
8022	**116,32**	**106,00**
8023	**114,03**	**108,24**
8025	**243,84**	**198,75**
8026	**54,71**	**51,33**
8060	**115,47**	**97,57**
8072	**183,59**	**178,00**
8074	114,39	150,25
Average	**116,15**	**109,23**

Case 3b	
Number of simulations	14
Number of successful cases	9
Success rate	0,64

ID	Avg. Delay (ns) ILP	Avg. Delay (ns) ELAM
9002	**124,77**	**49,81**
9004	**185,77**	**167,41**
9007	**135,88**	**88,60**
9008	64,18	150,16
9016	**96,49**	**94,12**
9023	**150,08**	**95,01**
9027	**121,30**	**109,92**
9039	233,94	446,71
9050	**250,27**	**147,28**
9052	**188,10**	**101,11**
9060	**107,48**	**94,10**
9066	**79,03**	**50,08**
9069	95,59	96,50
9074	**1431,62**	**118,62**
9079	74,14	82,98
Average	**222,58**	**126,16**

Case 3c	
Number of simulations	15
Number of successful cases	11
Success rate	0,73

ID	Avg. Delay (ns) ILP	Avg. Delay (ns) ELAM
10004	**120,22**	**111,02**
10007	**199,54**	**64,22**
10017	**166,12**	**110,19**
10018	**75,34**	**52,70**
10021	**73,28**	**64,61**
10025	**78,69**	**64,33**
10028	62,06	84,62
10031	140,28	140,28
10033	104,47	138,82
10044	**111,31**	**41,05**
10050	92,54	141,40
10056	**165,29**	**116,94**
Average	**115,76**	**94,18**

Case 4a

Number of simulations	12
Number of successful cases	8
Success rate	0,67

ID	Avg. Delay (ns) ILP	Avg. Delay (ns) ELAM
11001	230,35	420,83
11002	46,79	67,87
11009	**251,23**	**147,42**
11029	**169,89**	**135,95**
11030	**142,08**	**123,12**
11034	140,82	143,09
11041	35,01	47,22
11042	**1144,24**	**167,33**
11045	62,96	85,13
11059	**278,29**	**198,28**
11064	95,73	108,61
11080	**214,35**	**116,42**
Average	**234,31**	**146,77**

Case 4b

Number of simulations	12
Number of successful cases	6
Success rate	0,50

ID	Avg. Delay (ns) ILP	Avg. Delay (ns) ELAM
12005	167,15	184,71
12014	**261,15**	**124,35**
12015	124,94	142,70
12022	**132,78**	**101,91**
12026	**185,81**	**128,67**
12027	182,74	187,17
12033	**249,55**	**161,55**
12043	**151,94**	**132,87**
12044	133,70	174,03
12055	**120,14**	**92,95**
12056	**209,75**	**179,19**
12061	**106,98**	**91,68**
12062	207,20	235,62
12067	126,80	223,91
Average	**168,62**	**154,38**

Case 4c	
Number of simulations	14
Number of successful cases	8
Success rate	0,57

CURRICULUM VITAE

PERSONAL INFORMATION

Surname, Name: Çelik, Coşkun
Nationality: Turkish (TC)
Date and Place of Birth: April 13, 1979, Ankara
Marital Status: Married
Phone: +90 312 582 33 11
Email: ccelik@gazi.edu.tr

EDUCATION

Degree	Institution	Year of Graduation
M.S.	METU Electrical and Electronics Engineering,	2004
B.S.	Gazi University Electrical and Electronics Eng.,	2001
High School	Çankaya High School, Ankara,	1997

PROFESSIONAL EXPERIENCE

Year	Place	Enrollment
2001 - Present	Gazi University Electrical and Electronics Eng.	Research Assistant

FOREIGN LANGUAGES

Advanced English

PUBLICATIONS

1. Çelik, C. and C.F. Bazlamaçcı, "Energy and Loss Aware Application Mapping for Networks-on-Chip with Self Similar Traffic", Journal of Systems Architecture (2013), in review.

2. Çelik, C. and C.F. Bazlamaçcı, "Effect of application mapping on network-on-chip performance", Proc. of the 20th Euromicro Int. Conf. on Parallel, Distributed and Network-Based Computing (PDP 2012), Munich (Germany) 465-472, IEEE (2012).

3. Çelik, C. and C.F. Bazlamaçcı, "Performance analysis of reliable multicast protocols", Proc. of the 20th Int. Symp. on Computer and Information Sciences (ISCIS'05), İstanbul (Turkey), (2005).