Networks-on-Chip: Modeling, System-Level Abstraction, and Application-Specific Architecture Customization

by

Ahmed Abdel Fattah Hassan Morgan

B.Sc. of Electrical Engineering, Benha University, Egypt, 2000
M.Sc. of Electrical Engineering, Benha University, Egypt, 2005

A Dissertation Submitted in Partial Fullfillment of the
Requirements for the Degree of

Doctor of Philosophy

in the Department of Electrical and Computer Engineering

Networks-on-Chip: Modeling, System-Level Abstraction, and Application-Specific Architecture Customization

by

Ahmed Abdel Fattah Hassan Morgan

B.Sc. of Electrical Engineering, Benha University, Egypt, 2000
M.Sc. of Electrical Engineering, Benha University, Egypt, 2005

Supervisory Committee

Dr. Fayez Gebali, Supervisor
(Department of Electrical and Computer Engineering)

Dr. M. Watheq El-Kharashi, Department Member
(Department of Electrical and Computer Engineering)

Dr. Issa Traoré, Department Member
(Department of Electrical and Computer Engineering)

Dr. Jianping Pan, Outside Member
(Department of Computer Science)

Supervisory Committee

Dr. Fayez Gebali, Supervisor
(Department of Electrical and Computer Engineering)

Dr. M. Watheq El-Kharashi, Department Member
(Department of Electrical and Computer Engineering)

Dr. Issa Traoré, Department Member
(Department of Electrical and Computer Engineering)

Dr. Jianping Pan, Outside Member
(Department of Computer Science)

Abstract

This dissertation proposes different methodologies, with their associated models, to customize the architectural design of Application-Specific Networks-on-Chip (ASNoC). Specifically, system-level evaluation models are presented and architecture generation methodologies are built on them to allow the designer to generate the most efficient architecture for a given NoC-based application. Our system-level methodologies enable the designer to discover any flaws early during the design process and to quickly investigate the effect of various design choices on the resultant NoC cost and performance. In this dissertation, we have four main contributions.

In our first contribution, we propose power and reliability evaluation models. The two models are proposed at the system-level to allow for a quick evaluation of

different design decisions. The power model captures the power consumption in NoC routers and links, whereas the reliability one models the probability of the packets being affected by on-chip noise sources.

In our second contribution, we propose a cost-efficient architecture generation methodology for NoC based on network partitioning techniques. Our methodology partially customizes the on-chip network architecture with respect to two cost metrics: power and area. The partitioning technique is formulated using NoC terminology based on the Fiduccia-Mattheyses graph partitioning algorithm. Our partitioning scheme is compared to other partitioning techniques and is found to be the most efficient one for NoC. We further analyze the effect of using network partitioning on NoC power, area, and delay. From this analysis, the area reduction is proved to be guaranteed using network partitioning. Moreover, power and delay efficiencies of using network partitioning with NoC are formulated mathematically. Experimental results show that the proposed methodology is an efficient way to reduce power and area costs of NoC with respect to both standard and previous custom architecture generation techniques.

In our third contribution, we propose a multi-objective Genetic Algorithm (GA)-based optimization methodology for NoC full-custom architectures. For any application, the designer could control the optimization process through different optimization weight factors. Our methodology is evaluated by applying it to different NoC benchmark applications, as case studies. Results show that the architectures generated by our methodology outperform those generated by other techniques with respect to power, area, delay, reliability, and the combination of the four metrics. Finally, the running time of our methodology is an order of magnitude faster than that of previous architecture optimization techniques.

In our fourth contribution, we propose a multi-objective GA-based methodology to optimize the use of standard architectures, which were previously presented in

computer network, with NoC. Our methodology combines the best selection of NoC standard architecture and the optimum mapping of application cores onto that architecture. The methodology is further used to carry out an application-specific mapping-oriented evaluation of different NoC standard architectures. Experimental results show that the mapping achieved by our methodology outperforms those generated by previous mapping techniques with respect to power, area, delay, reliability, and the combination of the four metrics.

This research work aims at quickly validating various design decisions by proposing system-level power and reliability evaluation models. Moreover, in this dissertation, we present three application-specific methodologies to customize the three main categories of architectures that are currently used in implementing on-chip networks; namely, semi-custom, full-custom, and standard architectures, respectively. Our methodologies consider different NoC metrics: power, area, delay, and reliability, simultaneously. We believe that our proposed methodologies bridge an open gap in NoC research by matching the on-chip network architecture to the characteristics and the rapidly growing requirements of modern NoC applications.

Table of Contents

List of Tables

List of Figures

List of Abbreviations

263DEC	263 DECoder MP3 decoder
ACO	Ant Colony Optimizer
AI	Artificial Intelligence
ANoC	Autonomous Network-on-Chips
APCG	APplication Characterization Graph
ARQ	Automatic Repeat reQuest
ASNoC	Application-Specific Networks-on-Chip
AV	Audio Video
BER	Bit Error Rate
CMP	Chip MuliProcessor
CWG	Communication Weighted Graph
DSM	Deep Sub-Micron
EA	Evolutionary Algorithms
EP	Evolutionary Programming
FAR	Flit Arrival Rate
FM	Fiduccia-Mattheyses
GA	Genetic Algorithms
GP	Genetic Programming
GSO	Group Search Optimizer
IC	Integrated Circuit
IP	Intellectual Property
KL	Kernighan-Lin
MILP	Mixed Integer Linear Programming
MWD	Multi Window Display
NI	Network Interface

NoC	Networks-on-Chip
PPR	Ports Per Router
PSO	Particle Swarm Optimizer
PTM	Predictive Technology Model
QoS	Quality of Service
RST	Rectilinear Steiner Tree
SA	Simulated Annealing
SI	Swarm Intelligence
SNFT	Simple Non-Fault-Tolerant
SoC	System-on-Chip
TC	Traffic Continuity
TLM	Transaction Level Model
VLSI	Very Large Scale Integration
VOPD	Video Object Plane Decoder

List of Symbols

1. Architecture Symbols

A	Full-custom architecture optimized solely for area
BFT22	Standard 2-ary 2-fly butterfly tree architecture
BFT23	Standard 2-ary 3-fly butterfly tree architecture
BNT	Standard binary tree architecture
CLS	Standard 2-ary 2-fly clos architecture
D	Full-custom architecture optimized solely for delay
F	Full-custom architecture optimized for the multi-objective function
LR	Semi-custom architecture generated by long-range link insertion
MSH	Standard mesh architecture
NP	Semi-custom architecture generated by network partitioning
P	Full-custom architecture optimized solely for power
PLG	Standard polygon architecture
R	Full-custom architecture optimized solely for reliability
RNG	Standard ring architecture
TRS	Standard torus architecture

2. Mathematical Symbols

A	Adjacency matrix
A_L	Total link area
A_{NoC}	Total NoC area
A_R	Total router area
A_{Ri}	Area of router i
B	Queue size

BER	Bit error rate
c_i	Core number i
c_{ij}	Number of hops between cores i and j
$c_{l(r)}$	per unit length capacitance of the link (router) wires
C	Connectivity matrix
C_0	Total input capacitance of a minimum sized inverter
C_C	Coupling capacitance of a wire with neighboring wires
C_{g0}	Gate capacitance of a minimum sized inverter
C_S	Self capacitance of a wire
D_{NoC}	Overall NoC average delay
$D_{partitioned}$	overall NoC average delay with partitioning
$D_{unpartitioned}$	overall NoC average delay without partitioning
e_{ij}	Edge in the core graph connecting between cores c_i and c_j
f	Multi-objective optimization function
f_{op}	Operating frequency
F	GA fitness function
$FAR_{i_{max}}$	Maximum allowable flit arrival rate for router i
$I_{bias,wire}$	Current flowing from the wire to the substrate
I_{d0}	Drain current when drain and gate voltages are equal to V_{dd}
I_{leak}	Leakage current from source to ground
I_{short}	Short circuit current between source and ground
k_D	Dynamic port-independent power constant
k_{Dp}	Dynamic port-dependent power constant
k_L	Leakage port-independent power constant
k_{Lp}	Leakage port-dependent power constant
$l_{l(r)}$	Wire length for global interconnects (router crossbar)

$nKids$	Number of chromosomes generated by crossover
N	Number of cores in the application
N_b	Number of bits per packet
N_k	Number of cores within partition P_k
N_w	Number of wires in a link, i.e., channel width
N_L	Number of links in the network
N_P	Number of partitions
N_R	Number of routers in the network
p_{avg}	Average number of ports per router
p_{avg_n}	Average number of ports per router without partitioning
p_{avg_p}	Average number of ports per router with partitioning
p_i	Number of ports of router i
p_{max}	Maximum allowable number of ports per pouter
P_k	Partition number k
P_L	Total link power
$P_{Leakage}$	Link leakage power
P_{NoC}	Total NoC power consumption
$P_{partitioned}$	Total NoC power consumption with partitioning
P_R	Total router power
P_{Ri}	Router power of router i
$P_{Ri-Dynamic}$	Dynamic power of router i
$P_{Ri-Leakage}$	Leakage power of router i
P_{short}	Link short circuit power
$P_{switching}$	Link switching power
$P_{unit\ link}$	Power consumed by a single packet in one unit link
$P_{unpartitioned}$	Total NoC power consumption without partitioning

$\mathcal{P}_{ij}$	Probability of sending λ_{ij} packets successfully over a link (l_{ij})
$r_{l(r)}$	Per unit length resistance of the link (router) wires
R_{d0}	Equivalent output resistance of a minimum sized inverter
R_{NoC}	Overall NoC reliability
s_r	Inter-wires spacing for router internal interconnects
s_w	Inter-wires spacing
t_a	Router arbitration delay
t_l	Link propagation delay
t_r	Router propagation delay
u_{N_k}	Unbalance core factor for partition P_k
u_{λ_k}	Unbalance traffic factor for partition P_k
V_{dd}	Supply voltage
w_r	Wire width for router internal interconnects
w_w	Wire width
α_C	Switching activity from the adjacent wires
α_{fi}	Average flit arrival rate over all the ports of router i
α_S	Switching activity on a wire
β_l	Architecture legality factor
γ_A	Area weight factor
γ_D	Delay weight factor
γ_P	Power weight factor
γ_R	Reliability weight factor
η_D	Partitioning delay factor
η_P	Partitioning power factor
λ_{ij}	Number of packets sent from core c_i to core c_j per time step
λ_{inter}	Total inter-partition traffic

λ_k	Total traffic within any partition P_k
$\bar{\lambda}_k$	Total traffic sent from partition P_k to all other partitions
λ_{total}	Total ASNoC traffic
Λ	Traffic distribution matrix
μ	Average internode distance
μ_n	Average internode distance without partitioning
μ_p	Average internode distance with partitioning
σ	Noise standard deviation
τ	Delay of a minimum sized inverter of the target technology
τ_{sc}	Short circuit period corresponding to the flow of I_{short}

Acknowledgment

All praise be to Allah, the Almighty, who alone guided and aided me to bring this dissertation to light.

All thanks to my parents who are the real heroes behind any success in my life. They leave me speechless, because no words can ever return their favors. I would like also to thank my wife, Lamiaa, my daughters, Sarah and Mariam, for their love, continuous support, and for being friends that I cherished.

Next, I would like to express my sincere feelings and deep gratitude to my supervisor, Dr. Fayez Gebali for his help, encouragement, and support. He has always provided me with valuable advice and useful discussions. His efforts have enabled me to bring this work to its present state. I thank him in believing in me and giving me the opportunity to do my Ph.D. under his supervision.

I would like also to express my deep appreciation to Dr. M. Watheq El-Kharashi for his invaluable scholarly advice, inspirations, help, and guidance that helped me through my Ph.D. dissertation work. I will always be indebted to him for all that he has done for me during my stay and study here in Victoria. Thank you very much for being such a fantastic supervisory member and a good friend over these past four years.

I would like to acknowledge the advice and support from my supervisory committee members: Dr. Issa Traoré and Dr. Jianping Pan, for making my dissertation complete and resourceful. I should also mention that it was a privilege and an honor to have Dr. Hoda Abdel-Aty-Zohdy from Oakland University serving as my external examiner. Despite her extremely busy schedule, she spared no effort or time in evaluating the dissertation.

I would like to thank the Ministry of Higher Education of Egypt, the Egyptian mission department, and the Egyptian educational bureau in Canada for the study

leave and the financial support under scholarship No. 2/1/28/2002.

Special thanks should be sent to my dear friend Haytham Elmiligi. We were working in a joint project and throughout the whole period, his help, valuable discussion, and support make the achievement of this work possible.

There are no research work in isolation. I would like to give special thanks to my friends and colleagues who have always supported me through the time I needed for the successful completion of this dissertation. I would like to thank Abdelsalam Amer, Adel Younis, Ahmad Abdullah, Ahmed Awad, Ahmed Fadeel, Bassam Sayed, Emad Shihab, Khalid Almuzini, Khalid Khayyat, Mohamed Almardy, Mohamed El-Gamal, Mohamed Fayed, Mohamed Marsono, Mohamed Yasein, Omar Hamdy, Sherif Saad, Soltan Alharbi, Yousry Abdel-Hamid, and many others that I am sure that I will regret not to have them included in this section.

Ahmed A. Morgan, Victoria, BC, Canada

Dedication

To:

My parents, to you it all goes

My wife, for being my lover and my best friend

My daughters, thank you

Chapter 1

Introduction

Networks-on-chip (NoC) proposed in the new millennium as an efficient way to handle the communication requirements of modern multicore systems. The underlying architecture of these on-chip networks constitutes the main factor that controls the overall system performance. Customizing this architecture with respect to the target application requirements is a first-class objective to reduce the overall system cost and to enhance its performance. This dissertation presents different methodologies to carry out this application-specific customization process.

1.1 Networks-on-Chip

Recent advances in fabrication technologies enabled design engineers to integrate an impressive number of components like microprocessors, memories, and interfaces on a single chip. Ordinary shared buses were no longer able to handle the communication among this large number of components. Therefore, to facilitate the design procedure and to reduce the time-to-market, researchers advocated that one solution is to separate between computation and communication architectures [1]. NoC was proposed as an effective paradigm to make this separation [2, 3].

Figure 1.1 gives an example of a 3x3 mesh NoC. The figure shows that each NoC contains three fundamental components: links, routers, and Network Interfaces (NIs). Links are the channels that connect between nodes with specific bandwidths. Routers implement the routing scheme and route the traffic according to the employed protocol. NIs provide the suitable interface between the Intellectual Property (IP)[1] cores and the NoC. At the sending end, the NI divides packets into smaller flow control units (flits) that could be transferred in a single clock cycle. These flits are encapsulated into packets again by the NI at the receiving end. NIs are usually embedded into the IPs or the routers.

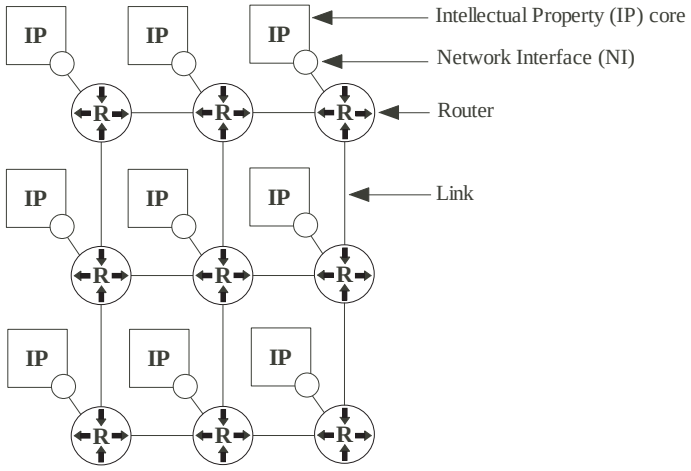


Figure 1.1: An NoC example. (IP cores are not parts of the network.)

1.2 Application-Specific Networks-on-Chip

NoCs are either general-purpose or application-specific. Application-Specific Networks-on-Chips (ASNoCs) differ from general-purpose ones in two main aspects [5]. First, their communication patterns could be statically analyzed and, therefore, the on-chip

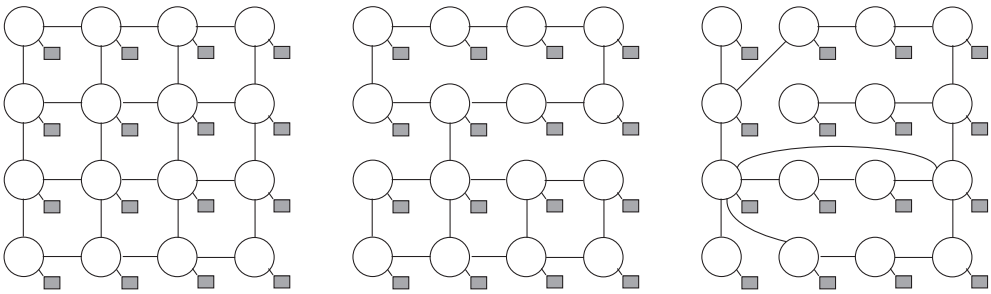
[1] A parameterizable core that can be used in System-on-Chip (SoC) design implementation. It might be soft macro, firm macro, or hard macro [4].

network could be customized according to the application behavior. Second, design objectives for ASNoCs vary from those of general-purpose NoCs. Furthermore, design requirements vary from one application domain to another. For example, multimedia applications require high bandwidth, real-time systems require guaranteed delay, and portable devices require low power consumption. Accordingly, for ASNoCs, the design of the on-chip networks[2] should be customized to comply with application requirements. In this dissertation, we target these ASNoCs. More precisely, we aim at customizing the design of the on-chip network architecture to comply with the application requirements. The work presented in this dissertation tries actually to solve a typical trade-off that exists in any electronic design between performance and cost.

Network architecture is a widely used term that spans from the physical structure through the protocol suite to the services delivered by the network. However, in this dissertation, the term network architecture, as defined in [6], is used to indicate the structural relations between cores and routers that constitute the network and to specify the topology and the physical organization of the network. Consequently, ASNoC architecture are currently implemented as one of three main categories. These are standard, semi-custom, and full-custom architectures. Figure 1.2 gives an example on each of these categories. Standard architectures are those previously defined and used in computer networks and multiprocessor systems, like mesh, torus, ring, etc. For example, Figure 1.2(a) gives a sample standard architecture of a 4×4 mesh. Semi-custom architectures are those slightly customized to enhance certain ASNoC metrics. This partial customization could be done quickly and straightforwardly; however, it does not guarantee the best level of performance. Semi-custom architectures are currently realized by slightly modifying standard architectures or by combining more than one standard architecture. For example, Figure 1.2(b) gives a sample semi-

[2]On-chip networks and NoCs are used interchangeably in this dissertation.

custom architecture of mesh and ring sub-networks connected together. Full-custom architecture are those completely optimized for one or more ASNoC metrics. It guarantees the best possible cost or performance with respect to the metric for which optimization is carried out. For example, Figure 1.2(c) gives a sample full-custom architecture for a 16 node application. Finally, this dissertation targets these three categories of ASNoC architectures by proposing a generation methodology of each one of them.



(a) Standard (b) Semi-custom (c) Full-custom

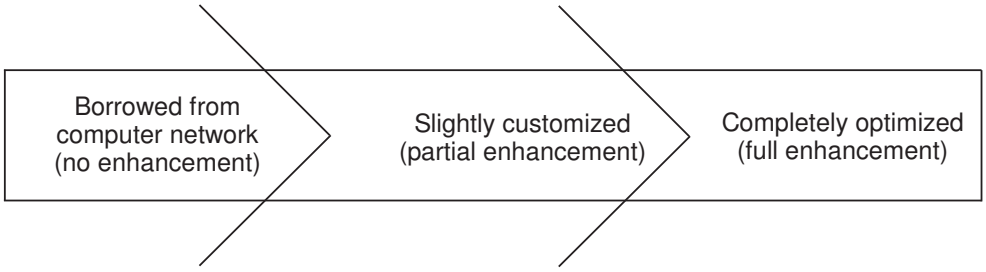


Figure 1.2: Examples on different categories of ASNoC architectures for a 16-core application. (Routers are represented by white circles, whereas cores are represented by dark squares.)

1.3 Problem Statement

NoC is an emerging research area that still has several research problems that need to be addressed [7]. NoCs differ from computer networks in many aspects. NoCs have shorter communication delay and limited silicon area. The power budget constitutes another important constraint of any NoC-based design. Moreover, by increasing the number of cores in modern SoCs, the noise and interference between these cores become significant parameters that pose many new reliability-aware design challenges. All these aspects put more constraints on the design of NoC-based systems and require different approaches to overcome the design challenges [8,9]. Targeting an application-specific design approach poses novel and exciting challenges to researchers. The designers of these ASNoCs have to make several application-specific decisions. These decisions usually trade off many conflicting design choices regarding architecture customization, core mapping, router design, traffic modeling, etc.

Our work targets some of the above mentioned ASNoC issues. In this dissertation, we try to simultaneously reduce the costs and improve the performance of ASNoC-based systems by customizing the design of the underlying architecture to match the target application. Furthermore, we want to define models to evaluate different NoC designs. More precisely, the dissertation aims at addressing the following problem: "Given an ASNoC-based application with both its functional and timing specifications, what is the effect of changing the on-chip network architecture on the overall system cost and performance, and how, based on this analysis, we could model, synthesize, evaluate the performance, and optimally design the underlying on-chip network". To achieve this goal, we have to address the following accompanying problems:

1. **Evaluation models are needed:** ASNoCs still face a problem in evaluating

the cost and the performance of any proposed design. Most of the models that are used in testing and evaluating ASNoC-based designs are originally related to traditional computer networks and multiprocessor systems. Because of the substantially different behavior of ASNoC-based systems, this leads to inaccurate evaluation of tested designs. Therefore, researchers started recently to propose ASNoC-oriented cost and performance evaluation models. However, system-level models that take into account the nature of ASNoC applications and allow for quick generation and evaluation of different ASNoC architectures still need to be addressed. Power consumption and silicon area are the main cost metrics, whereas delay and reliability constitute the most important performance metrics. Consequently, ASNoC-oriented models for each of these metrics are of great importance to ASNoC research community.

2. **Methodologies for application-specific architecture customization are needed:** The hypothesis behind ASNoC is to customize the design of the on-chip network to comply with the application requirements. The cost and the performance of an ASNoC depend primarily on its architecture [5]. Standard, semi-custom, and full-custom architectures are currently in use with ASNoCs. Therefore, it is required to carefully design these three categories of architectures to meet the application requirements and the design constraints. According to the target application, the most desirable requirements for ASNoC-based systems are low power consumption, low silicon area, low or guaranteed delay, and high reliability. As a result, for each category of ASNoC architectures, there is a real need for architecture generation methodologies to allow the on-chip network to conform to one or more of these requirements.

3. **Efficient core mapping techniques are needed:** Mapping application cores onto an architecture states how these cores are physically connected to

each other. Therefore, it constitutes a vital step in any ASNoC-based design. Core mapping is tightly related to the architecture customization and both are usually done simultaneously. The large variation of the inter-core traffic and the heterogeneous nature of computing resources in ASNoC-based systems has posed many challenges in choosing the most suitable mapping for a specific application. Therefore, application-specific methodologies should be presented to efficiently map the application cores onto the customized network architecture in order to meet certain cost and performance requirements.

1.4 Motivation

The application-specific approach proved itself as a promising solution to address the matching problem between the application requirements and the on-chip network design [10]. However, many issues still need to be addressed. Previous work in this area addressed the ASNoC-based designs on a by-case basis. Consequently, there is a demanding need for efficient ASNoC design methodologies with elaborate evaluation models to fulfill the ever-increasing communication demands of modern large-scale SoC-based systems. This dissertation focuses on the development of these efficient, application-specific on-chip networks needed by future multi-billion-transistor SoC designs. In this dissertation, we present evaluation models and novel architecture customization/optimization methodologies that simultaneously help improving the performance and reducing the cost of any ASNoC-based system significantly.

1.5 Contributions

As explained in Section 1.2, ASNoCs could be realized by standard, semi-custom, or full-custom architectures. For each of these three categories, it is required to

customize the underlying on-chip network architecture to comply with the application in-hand requirements. This emphasizes the need for application-specific architecture customization and optimization methodologies. Moreover, it requires proposing ASNoC-oriented cost and performance models to evaluate different architectures. System-level modeling allows for carrying out this evaluation quickly and early during the design process. Accordingly, this dissertation addresses this application-specific architecture generation problem. The main contributions of this dissertation could be summarized as follows:

First, we propose two evaluation models for ASNoC. More precisely, we present system-level power and reliability models. The former is an analytical model that evaluates the power consumption in routers and links of any on-chip network, whereas the latter is a probabilistic one that captures the probability of sending the application packets successfully in the presence of noise. The two models are technology dependent as they are using some parameters from the employed fabrication technology. The power model has been published in brief in [11] and in full in [12]. Whereas, the reliability model has been published in brief in [13] and in full in [14].

Second, we present a cost-efficient generation methodology for ASNoC semi-custom architectures. The methodology is based on network partitioning techniques and considers the two cost metrics: power and area. The partitioning problem itself is formulated using NoC terminology according to the famous Fiduccia-Mattheyses (FM) algorithm [15]. The methodology is then evaluated through different case studies and is found to generate semi-custom ASNoC architectures that outperform those of previous techniques with respect to both power and area. This work has been published in brief in [16–18] and is submitted for publication in full in [19].

Third, we present an optimization methodology for ASNoC full-custom architectures. The methodology uses Genetic Algorithms (GA) with binary chromosome

representation and is a multi-objective one that considers four NoC metrics: power, area, delay, and reliability. Moreover, it combines both architecture generation and application core mapping. The methodology is then evaluated and compared to other architecture generation techniques using four real ASNoC benchmark applications. Results show that the architecture generated by our methodology has lower cost and better performance than those generated by previous generation techniques. This work has been published in brief in [20, 21] and in full in [22].

Fourth, we present an optimization methodology for ASNoC standard architectures. Similar to the full-custom generation methodology, our standard architecture optimization one uses GA, but with integer chromosome representation, and is a multi-objective one that considers the same four metrics: power, area, delay, and reliability. The methodology combines standard architecture selection and optimum mapping of application cores onto this architecture. Based on the optimum mapping of our methodology, we present fair cost and performance evaluations of different ASNoC standard architectures. Our methodology is evaluated and compared with previous standard architecture customization techniques for different ASNoC benchmark applications. Results show that the proposed methodology is more efficient than previous standard architectures customization techniques with respect to all the four metrics mentioned above. This work has been published in brief in [23, 24] and is submitted for publication in full in [25].

1.6 Dissertation Organization

This dissertation is organized as follows:

Chapter 2 reviews the related work. It starts by surveying previous NoC cost and performance evaluation models. The chapter then discusses network partitioning techniques and their use with on-chip networks. It quickly reviews GA as one of the

bio-inspired optimization techniques. Finally, the chapter surveys the work done for ASNoC architecture realization[3]. This covers in details the three main categories of ASNoC architectures: standard, semi-custom, and full-custom.

Chapter 3 proposes NoC power and reliability models. Terms and parameters related to both models are presented and discussed in details through this chapter.

Chapter 4 proposes a cost-efficient semi-custom architecture generation methodology. The formulation of the partitioning problem is presented with an analysis of the effect of partitioning on ASNoC cost and performance. The chapter concludes with an evaluation of the methodology and a comparison with previous architecture generation techniques.

Chapter 5 proposes a full-custom architecture optimization methodology. The GA representation of the problem is discussed in details. Different architectures generated by our methodology are then presented. The chapter concludes by comparing the architectures generated by our methodology with those generated by previous architecture generation techniques.

Chapter 6 proposes a standard architecture optimization methodology. The GA representation of the problem is discussed in details. The methodology is then used to evaluate and compare the cost and the performance of different standard architectures. Thereafter, our mapping technique is evaluated by comparison with previous standard architecture mapping techniques.

Chapter 7 summarizes this dissertation, states our contributions, and suggests directions for future research.

[3]architecture realization and architecture generation are used interchangeably in this dissertation.

Chapter 2

Literature Review

This chapter presents a review of the literature related to the work presented in this dissertation. We aim mainly at providing a state-of-the-art survey for different topics related to our research. Therefore, this chapter summarizes the work done by different NoC research groups not only before our work but in parallel to it as well.

This chapter is organized as follows. Section 2.1 gives an introduction about the evolution of SoC communication from the ordinary shared buses to NoCs. Sections 2.2 reviews different models that are used in evaluating NoC power consumption, area, delay, and reliability, respectively. Section 2.3 discusses the most common network partitioning techniques and the use of these techniques for ASNoC customization. Section 2.4 gives a quick introduction of GA as one of the bio-inspired optimization techniques. Section 2.5 surveys the work done in customizing on-chip network architectures according to application requirements. This includes the three main categories of ASNoC architectures: standard, semi-custom, and full-custom. Finally, Section 2.6 summarizes the chapter.

2.1 Introduction

NoC emerged in the new millennium as an efficient on-chip communication paradigm due to the change in two technological aspects. The first aspect is the evolution of the Integrated Circuit (IC) technology which enables the designer to integrate many computational engines like microprocessors, memories, and interfaces in one chip. This large number of computational resources requires an efficient communication. The efficiency of this communication becomes the key element that determines the overall system performance. The second aspect is the shrinking in the size of these computational resources as a result of the Deep Sub-Micron (DSM) technology. The direct impact of this shrinking is that the interconnection delay overrides the computational delay [26]. Accordingly, the efficiency of the on-chip interconnection becomes more and more the dominant factor in determining the overall system performance [6].

To handle the on-chip communication problems, many researchers believed that the solution is to mimic computer network communication. This approach is not only capable of dealing with complex systems, but also it provides reliable services. Although the first prototype of networked integrated multiprocessor system was proposed in 1986 [27], it was not until the beginning of this century that NoC became a hot research and development area. This is because of the increased complexity of modern applications and the new technological trends in the last few years. Consequently, researchers try to adjust networking techniques to suite the nature of integrated circuits. On-chip networks should be simple, fast, effective and energy-saving. In [28], L. Benini and D. Bertozzi surveyed the evolution of SoC communication from the ordinary shared buses to the fast-growing and widely-accepted NoC approach.

The most famous NoCs that are already implemented were ÆTHEREAL [29],

SONICS [30], MANGO [31], PROTEO [32], NOSTRUM [33], XPIPES [34], SPIN [35], CHAIN [36], and ASOC [37,38]. In [39], T. Bjerregaard and K. mahadevan surveyed the architectures of these NoCs as well as other NoC-related aspects. Finally, NoC problems, emerging challenges, hot research areas, and future directions were discussed in [7, 9, 40–42].

2.2 NoC Evaluation Models

This section reviews the work done by different research groups in NoC modeling. The metrics considered in this dissertation, power, area, delay, and reliability are surveyed in details. As we employ area and delay models from the literature, they are highlighted in this section. However, they are discussed in more details in Chapter 3.

2.2.1 Power Models

The research on different power-related topics lies on the heart of NoC research. Most of NoC applications are power-hungry. Therefore, power modeling [43], power reduction techniques [44, 45], and power-aware design methodologies [46] took much of the efforts of NoC researchers. On-chip networks consume power in their routers and links. Therefore, a power analysis of different types of NoC routers was carried out in [47]. Similar analysis of different NoC wiring styles was presented in [48]. Moreover, for NoC links, circuit-level power models were proposed in [43, 45, 49] for both local IP-to-router and global router-to-router interconnects. Finally, different power models for NoC routers were also presented in [43, 50, 51].

Before starting our research work, power models for the whole on-chip network could be classified into two main categories: regression [43, 50, 52–54] and bit-energy models [48, 55–58]. The former modeled the power consumption using fitting factors that were obtained by linear or other regression methods, whereas the latter assumed

that the energy required to transfer a single bit through the network is known, a priori. The main problem with these models was how to find the fitting factors or the single bit energy. The first technique used to find them was through estimation, which was proved to be inaccurate [50]. The second technique required either simulating or implementing the application onto different architectures. The long time associated with either simulation or implementation made this method practically less attractive. Therefore, what was really needed is a system-level power model for the whole on-chip network that could be used in a high abstraction level to quickly evaluate different NoC designs. In this dissertation, we propose a system-level power model that allows for an early design space exploration. Our model is discussed in details in Section 3.3.

In the last few years, reducing the power consumption of on-chip networks proved itself as the most important design objective for NoC-based systems. Therefore, in parallel to our research work, power-related topics attracted more attention from the NoC research and design communities. For example, the effects of buffer allocation schemes, source and load impedances, and packet blocking time on the NoC power consumption were analyzed in [59], [60], and [61], respectively. Moreover, a Markovian power models for 2D torus NoC was presented in [61]. For system-level power modeling, although we were from the early research groups to present a system-level power model for NoC in [11,12], many other models appeared during the last two years. This emphasizes the importance of system-level power modeling for NoC research. For example, an analytical system-level power model was presented in [62], enhanced in [63], and further enhanced in [64]. A tool, ORION 2.0, for an early system-level power estimation was built on this model. Moreover, another tool, McPAT, based on a similar system-level power model was presented in [65]. Finally, different power models, including our model, were compared in [66].

2.2.2 Area Models

NoC area consists of the area of its links and routers. On one hand, link area evaluation of different on-chip communication infrastructures was presented in [67]. More precisely, analytical models for the link area of shared-bas, segmented-bus, 2D mesh NoC, and point-to-point architectures were proposed. These models quantified the area advantages of NoC over other communication infrastructures. However, the presented NoC area model did not consider the router area and was only restricted to 2D mesh architecture. On the other hand, the effect of the buffer size on NoC router area for different router configurations was analyzed in [68]. That analysis showed a linear dependency of the router area on the buffer size. Consequently, router area model was presented in [50], which abstracted the router area as the summation of the areas of the output buffers, the input buffers, the arbitration logic, and the crossbar. The model was based on fitting factors that took too long to calculate: 24 hours according to the authors. Moreover, the evaluation of the model showed an area inaccuracy of up to 22%, which is a very significant ratio that prevented the model from being used for accurate area evaluation. Finally, similar models that abstracted the router area for Chip MuliProcessors (CMPs) as the summation of the input module, output module, and the corssobar without the need for any fitting factors was presented in [69].

Accurate area calculation methodology for 2D NoC was presented in [70] by summing up the silicon areas from the floorplanner after implementing the design. The same methodology was used with 3D NoC in [71]. Despite the accuracy of this methodology, it required implementing all the architectures to be evaluated. Therefore, it could not be used early in the design process for a quick system-level design space exploration. Accordingly, a system-level model was presented in [72] for buffered interconnect. Although that link model was an accurate one and it allowed

for early design space exploration, it lacked the modeling of NoC routers. Finally, an accurate analytical NoC area model was presented and evaluated in [62]. The model was further enhanced in [63]. A tool, ORION 2.0 [62], was built on this model that calculate the NoC area by summing up the silicon area starting from the gate level. It is worth mentioning that a similar router area model as the one presented in [62] was proposed and a tool, McPAT, was built on it by another research group in [65]. This model is a system-level one and allows for an early design space exploration. Therefore, we employ it for our area evaluation throughout this dissertation. More details about the model is presented in Section 3.4.

2.2.3 Delay Models

Packets suffer three kinds of delays in their ways from source nodes to destination nodes. These are the arbitration and propagation delays through routers, the propagation delay through links, and the serialization, or packetization, delay through NIs. Accordingly, NoC delay models are either end-to-end models or NI-to-NI ones. The former considers the above mentioned three kinds of delays, whereas the latter includes only the delays through routers and links. First, for router delay modeling, a Markovian-based model was presented in [73] and a queuing theory-based model with Quality of Service (QoS) assurance was presented in [74]. Second, for link delay modeling, a circuit-level analysis of the effect of source and load impedances on link delay was presented in [60]. An analytical link model was presented in [72] for buffered interconnect. Third, for NI-to-NI delay modeling, many analytical models were presented in [10, 75] and an iterative methodology to calculate the delay of 3D NoCs was presented in [76]. Fourth, for end-to-end delay modeling, a network calculus-based model was presented in [77], an analytical model with arbitrary buffer allocation schemes was presented in [59], a systemC Transaction Level Model (TLM) was

presented in [78,79], and a transaction-based model for the master/slave handshaking process was presented in [80].

Most of the above mentioned delay models were proposed for 2D mesh architectures [10, 58, 59, 73–75, 77–79]. Nonetheless, some of them could be extended to different NoC architectures. However, the main problem for most of these models was the assumption that router and link delays for a unit flit were known, a priori. Consequently, a general end-to-end delay model for both 2D and 3D architectures was presented in [81]. Furthermore, this model presented analytical equations to calculate the router and link delays for a unit flit based on the targeted fabrication technology. It is also a system-level model that could be used for early design space exploration. Accordingly, we employ it for our delay evaluation throughout this dissertation. More details about the model is presented in Section 3.5.

2.2.4 Reliability Models

In recent years, NoC reliability became a hot research area because of the low voltage swing and the many on-chip noise sources of the modern DSM technology. In the literature, reliability was usually achieved by some form of fault-tolerance, which enables the network to overcome any fault, or error, in its components without a disruption of the network operation [82]. Therefore, the use of different error control schemes with NoC were presented and analyzed in [56, 83]. Moreover, different on-chip faults associated with new fabrication technologies were classified and analyzed in [56, 84]. In general, on-chip faults were categorized into two main groups; namely, hard and soft faults. Hard faults are permanent, like stuck at fault, whereas, soft faults are transient, like crosstalk. Consequently, a high-level permanent fault model for NoC switches was presented in [85]. Furthermore, an analytical reliability model for 2D mesh, which considered only permanent faults, was presented in [86].

Soft faults caused by the increased internal noises in modern DSM technology were shown to be the dominant source of on-chip faults in [6]. In the context of on-chip networks, these soft faults were represented so far by the Bit Error Rate (BER) model [87]. Moreover, for this BER model, the Gaussian distribution was used to represent the on-chip noise sources associated with these soft faults [88]. Accordingly, for different error control schemes, Gaussian BER reliability models for a single on-chip link was presented in [89]. However, to the best of our knowledge, there is no research work presented so far, with respect to soft faults, that modeled the reliability of the whole network on the system-level. Therefore, in this dissertation, we aim at filling this open gap in NoC research by presenting a system-level model that could be used in early design stages to evaluate the overall on-chip network reliability.

In parallel to our research work, many reliability-related work was presented by different research groups. For example, many adaptive fault-tolerant routing schemes were presented in [90,91], a router designing methodology to handle different hard and soft faults was proposed in [92], a self-corrected coding scheme for reliable interconnection NoC was presented in [45], an HSPICE-based simulation was carried out in [93] to evaluate the reliability of different wave pipelined interconnects with constant BERs, a NoC simulator that employed the BER model to evaluate the impact of different error control schemes on NoC power and performance was presented in [94], and an implementation of fault-tolerant vertical links for 3D NoCs was presented in [95]. To the best of our knowledge, there is no research work that has been published yet that models the overall NoC reliability mathematically in the system-level to allow for an early design space exploration. Therefore, the reliability model presented in this dissertation should be of great importance to the NoC research community. Our reliability model is discussed in details in Section 3.6.

2.2.5 Overall Summary of NoC Evaluation Models

This subsection concludes our review of the work done by different research groups with respect to NoC evaluation models. We summarize the major models presented till now to evaluate the four NoC metrics considered in this dissertation. In a chronic order, Table 2.1 represents this summary in a more concise form. Furthermore, the network architectures or components targeted by these models are also included. The table first emphasizes that power consumption modeling had most of the efforts exerted by different research groups. It also shows that NoC reliability modeling is still a virgin area that requires lots of research efforts.

2.3 Employment of Network Partitioning Techniques with NoCs

Network partitioning techniques are widely used in many fields, such as parallel computing, power system analysis, and Very Large Scale Integration (VLSI) design to divide a large network into smaller partitions [96, 97]. These techniques could divide the network into exactly two subnets, two-way partitioning [98], or more than two subnets, multiple-way partitioning [99,100]. In the literature, the network partitioning problem was usually presented as a graph partitioning one. Such a graph consisted of a set of vertices, representing the nodes, and a set of weighted edges, representing the communications between these nodes. However, this graph partitioning problem was known to be an NP hard one [101]. Therefore, many heuristics and optimization-based techniques were proposed to solve it. For example, an Artificial Intelligence (AI)-based method was used in [102], stochastic-based method was used in [103], the geometric information of the graph was used in the inertial method in [104], and the eigenvectors of the Laplacian matrix was used in the spectral method in [105].

Table 2.1: Summary of the work done by different research groups for NoC modeling. (P: Power, A: Area, D: Delay, and R: Reliability.)

Year, [Ref.]	**Network or component**	**Metric**				**Main research objective**
		P	**A**	**D**	**R**	
2002, [55]	Routers only	X				Analysis of the power consumption of different switch fabrics
2003, [57]	2D mesh	X				Power-aware mapping technique
2004, [52]	STBus	X				Regression power modeling
2004, [67]	QNoC	X	X			Power and area analysis of shared-bus, segmented-bus, point-to-point, and NoC
2005, [87]	Links only				X	Scheme for designing self calibrating NoCs
2005, [58]	2D mesh	X				Power and timing-aware mapping technique
2005, [56]	2D mesh	X		X	X	Novel router architecture for low latency
2005, [54]	2D mesh	X				Power evaluation of NoC and bus-based architectures
2005, [43]	Arbitrary	X				Power modeling of links, FIFOs, and routers
2006, [49]	Links only	X		X		Power and delay evaluation of local IP-to-router and global router-to-router links
2006, [69]	Arbitrary	X	X			Power and area modeling of tiled CMP NoCs
2006, [75]	2D mesh	X		X		NoC architecture customization methodology
2006, [73]	2D mesh			X		Delay-aware mapping technique
2007, [50]	2D mesh	X	X			Regression power and area modeling
2007, [89]	Arbitrary	X			X	Power and reliability analysis of different error-control schemes
2007, [84]	2D mesh				X	Reliability modeling and classification of soft faults in NoCs
2007, [81]	2D, 3D grids	X		X		Power and delay evaluation of 2D and 3D architectures
2008, [51]	2D torus	X		X		Power and delay evaluation of using virtual channels with NoCs
2008, [45]	Arbitrary	X				Power reduction technique using a green coding scheme
2008, [86]	2D mesh				X	Reliability evaluation of 2D mesh
2009, [78]	2D mesh			X		SystemC TLM2 delay modeling for wormhole NoCs
2009, [80]	Custom			X		NoC architecture customization methodology
2009, [74]	2D mesh			X		Delay modeling with QoS assurance
2009, [62]	Arbitrary	X	X			Tool for system-level power and area estimation
2010, [72]	Links only	X	X	X		Power, area, and delay modeling of buffered links
2010, [64]	Routers only	X	X			Enhancing power and area estimation using a machine learning technique
2010, [59]	Arbitrary	X		X		Power and delay evaluation of different buffer allocation schemes

Simple heuristics, like linear, scattered, and random algorithms [106] were also used to quickly carry out the partitioning without considering the weights of the vertices nor the edges. However, the most popular and widely used partitioning heuristic is the Kernighan-Lin (KL) [107] and its derivatives, like Fiduccia-Mattheyses (FM) [15] and Min-Cut [108]. These heuristics aimed at minimizing the total weights of the edges cut due to partitioning. In a nutshell, in [101], Chamberlain surveyed and compared different partitioning techniques focusing on their application onto parallel computing. Finally, many public partitioning software packages, like Chaco [106] and PARMETIS [109] could be employed to carry out network partitioning.

The possibility of using network partitioning techniques for ASNoC architecture customization was highlighted in [110]. In that study, a tool, OIDIPUS, was proposed to map application cores onto a restricted architecture of two rings connected together, as two partitions. Although the work presented in [110] did not formulate nor evaluate the use of network partitioning techniques with on-chip networks, it paved the road of using such techniques for ASNoC architecture generation. Therefore, in parallel to our research work, different research groups started using network partitioning techniques with ASNoCs. First, for voltage and frequency islands NoC-based systems, network partitioning techniques were used to divide the whole systems into partitions that were implemented as separate islands [111, 112]. The use of partitioning with voltage and frequency island NoC-based systems was evaluated in [113]. Second, for multicast 2D NoCs routing, network partitioning techniques were used to enhance the bandwidth efficiency and the overall performance of NoC-based systems in [114, 115]. Similar adaptive unicast/multicast routing techniques were used to reduce the packet latency for 3D NoCs based on hamiltonian path partitioning in [116].

For on-chip architecture realization, in parallel to our work, some research groups also advocated using network partitioning to customize the underlying ASNoC architecture. First, a latency-oriented greedy algorithm was presented in [117]

to divide any ASNoC into subnets that were implemented as ordinary bus-based systems. These buses were then connected together as a mesh architecture to construct the whole network. Second, a circuit-level customization methodology was presented in [118] to reduce the power consumption of 2D ASNoC using network partitioning. Third, for 3D ASNoCs, an iterative algorithm, similar to the one formulated mathematically in this dissertation, was presented in [119] to divide a large network into smaller partitions. Each partition was then implemented in a separate layer. Layers were then connected together to construct the overall 3D network architecture. A tool, SunFloor 3D, was built on this algorithm and presented in [120] to generate and synthesize 3D architectures for ASNoC-based systems. Different partitioning schemes, which were presented for 3D ASNoC architecture generation, were surveyed and evaluated in [121]. Fourth, for multimedia and other bandwidth constrained applications, network partitioning techniques were used in [122] to build a low energy tree-based architecture that is suitable for these applications. A similar study was presented in [123] that combines network partitioning with a Rectilinear Steiner Tree (RST) algorithm to reduce the power consumption of ASNoC-based systems.

Despite this large amount of work that has been proposed in parallel to our work, ours is still unique in two aspects. First, we mathematically formulate the use of network partitioning as a cost-effective way for ASNoC architecture generation. Second, based on our formulation, we build a system-level cost-efficient methodology for ASNoC architecture generation. Our system-level methodology allows for a quick generation of the underlying architecture and an early evaluation of different design alternatives.

2.4 Genetic Algorithms

Bio-inspired optimization techniques are those which mimic the natural biological systems. These techniques were proposed to solve the real complex problems that could not be solved by conventional optimization methods. Bio-inspired techniques could be classified into two main categories: Evolutionary Algorithms (EAs) and Swarm Intelligence (SI). The former were inspired by genetic evolution, whereas the latter tried to mimic animal behavior. The most famous EA techniques are Genetic Algorithms (GAs) [124], Genetic Programming (GP) [125], and Evolutionary Programming (EP) [126]. The most widely used SI techniques are Particle Swarm Optimizer (PSO) [127], Ant Colony Optimizer (ACO) [128], and Group Search Optimizer (GSO) [129]. As we are using GAs in this dissertation, the following paragraphs give some details about them. For more information about the GAs and the others, the reader is referred to [130, 131].

GAs are inspired by the process of natural evolution. Accordingly, any potential solution is represented in the form of a chromosome. A set of chromosomes constitutes a generation. The algorithm adopts a stochastic global search method to evolve a new generation from the current one. The first phase of the algorithm is the selection, in which the most fitted chromosomes from the current generation are selected. Fitness is evaluated based on an objective function that models the optimization problem. Thereafter, a new generation is produced from the selected chromosomes by applying different genetic operators:

- *Elitism*: This ensures that the best individual will survive to the next generation. Accordingly, at least, the most fitted chromosome is copied without changes from the current generation to the new one.

- *Crossover*: Chromosomes that are previously selected are considered as parents

and are allowed to mate. Accordingly, parts of different chromosomes are exchanged together to produce two new children, or offspring. Crossovering good chromosomes likely results in better offspring. Finally, these newly created offspring are added to the next generation.

- *Mutation*: Selected chromosomes from the current generation are slightly altered to introduce some diversity in the new generation. This diversity prevents the chromosomes from becoming too similar to one another. Consequently, the algorithm is likely protected from being trapped in local minima or maxima.

The algorithm continues going in an iterative manner to evolve good individuals from one generation to the next until the best solution is reached. Finally, a stopping criterion should be employed to stop the algorithm.

2.5 ASNoC Architecture Realization Techniques

The developments of on-chip network architectures reflected the evolution of NoC as an on-chip communication paradigm. When NoCs were first proposed to replace shared buses, they were implemented using architectures that were commonly used in computer networks and multiprocessor systems, like mesh, torus, ring, etc. During that period, researchers aimed mainly at evaluating the use of NoCs rather than enhancing their underlying architectures. In this dissertation, these architectures are referred to as standard architectures.

As NoC proved itself as a promising candidate to carry out the communication requirements of modern SoC applications, researchers started to realize that the requirements and design objectives of NoC-based systems are different from those of computer networks. For example, most of NoC-based systems are power and silicon area-limited. Moreover, the level of uncertainty in NoC-based systems is not

as high as that of computer networks. Therefore, researchers started to customize on-chip network architectures slightly to meet the actual requirements and design objectives of NoC-based systems. In this dissertation, the architectures resulted from this partial customization are referred to as semi-custom architectures.

The partial customization of NoC architectures slightly reduced the cost and enhanced the performance of NoC-based systems. However, it did not guarantee the minimum cost nor the maximum performance. Therefore, as NoC became a well established on-chip communications paradigm, researchers optimized the NoC architectures completely to guarantee the lowest possible cost with the highest possible performance. In this dissertation, the architectures resulted from this optimization process are referred to as full-custom architectures.

The above mentioned developments of NoC architectures were not exactly chronic such that new architectures completely replaced previous ones. Nevertheless, the three categories of NoC architectures are still currently in use by the research community. Different research groups even defend certain architectures over others. Table 2.2 gives a general comparison between the three architectures. Moreover, in the following subsections, we survey the work done by different research groups regarding these three categories of NoC architectures in more details.

Table 2.2: Comparison between different categories of NoC architectures.

Comparison criteria	**Standard**	**Semi-custom**	**Full-custom**
Design and analysis techniques	Many	Moderate	Few
Routing strategies	Many	Many/moderate	In research
Deadlock and livelock freedom strategies	Many	Moderate	In research
Tools for automatic architecture generation	Many	Few	Few
Regularity of underlying network	High	Moderate	Low
Ease of implementation and floorplanning	High	Moderate	Low
Architecture generation time	Low	Low	High/very high
Performance	Moderate/low	Moderate/low	High
Cost	High	High	Low

2.5.1 ASNoC Realization using Standard Architectures

The research work done by different research groups with respect to NoC standard architectures could be classified into three main research directions:

1. Application cores mapping onto standard architectures.
2. Employment of known standard architectures for on-chip network realization.
3. Analysis and evaluation of different NoC standard architectures.

For application cores mapping onto standard architectures, the mapping problem was shown to be a type of the constrained quadratic assignment problems, which are NP-hard ones [57]. Therefore, different algorithms and heuristics were proposed to carry out this mapping process. The most famous of these algorithms are PBB [57], GMAP [57], PMAP [132], NMAP [133], and BMAP [134]. Moreover, different optimization techniques, like EAs and Simulated Annealing (SA), were used in [73, 135–137] to obtain the optimum cores mapping onto 2D mesh architecture. However, the problem with these mapping algorithms and techniques is that they were either single-objective or were proposed specifically for 2D mesh. Therefore, there is a demanding need for a mapping technique that is multi-objective and could be used with any NoC architecture. In this dissertation, we target this open research problem by presenting this general multi-objective mapping technique. More precisely, in Chapter 6, we present a GA-based standard architecture optimization methodology that integrates both best architecture selection and optimum core mapping. Our methodology considers power, area, delay, and reliability, simultaneously and is not limited to any specific standard architecture.

In parallel to our research work, more mapping techniques were presented to reduce the power consumption of NoC standard architectures. For example, an

algorithm was presented in [138] to map application cores specifically onto the WK-recursive architecture, a heuristic based on priority lists was proposed in [139] to map application cores specifically onto the 2D mesh architecture, and a GA-based technique was presented in [140] to acquire the optimum mapping specifically onto the 2D mesh architecture as well. Despite this large amount of work that has been proposed in parallel to our work, ours is still unique in being a multi-objective one that is suitable for any NoC standard architecture.

For the employment of known standard architectures with NoC, early on-chip network research used predefined 2D standard architectures to realize NoC-based designs. Accordingly, different NoCs were presented in [29–38] that were built on these standard architectures. Moreover, these NoCs were surveyed and compared in [39]. At a more advanced stage, researchers used 3D architectures to realize higher-performance on-chip networks [81,141,142]. Different 3D standard architectures used with NoCs were surveyed and compared in [42].

The problem of choosing the right standard architecture for any NoC-based system was addressed in [143]. A tool, SUNMAP, was presented to automatically select the best architecture for a given application and map its cores onto that architecture. However, SUNMAP was limited to only five standard architectures (mesh, torus, hypercube, butterfly, and clos). Moreover, SUNMAP was built on a heuristic-based mapping technique, NMAP, rather than an optimization-based one. Therefore, SUNMAP is not guaranteed to result in the optimum standard NoC architecture. Thus, there is a real need for a standard architecture optimization methodology that is not limited to specific standard architectures and guarantees the optimum realization of the underlying on-chip network. Our GA-based standard architecture optimization methodology, presented in Chapter 6, fills this open research gap. More precisely, our methodology is not limited to specific standard architectures and is based on GA optimization. Therefore, for any NoC-based system, it guarantees

the optimum realization of the underlying standard on-chip network architecture.

For the analysis and evaluation of different NoC standard architectures, six of the 2D architectures (SPIN, mesh, torus, folded torus, octagon, and BFT) were compared with respect to power consumption, area, delay, and throughput in [70]. Similarly, different grid and tree-based 3D architectures were compared with respect to the same four metrics in [71]. 2D mesh, ring, and spidergon architectures were compared with respect to delay and throughput in [144]. Additionally, 2D mesh, torus, and fat tree architectures were compared with respect to power consumption and delay in [145]. Finally, 2D mesh is evaluated with respect to power consumption, delay, and reliability in [56]. However, to the best of our knowledge, none of these previous evaluations were based on the optimum cores mapping onto these architectures. This might be misleading, and therefore, results in favoring one architecture over others that might have lower cost or higher performance with different mapping. Therefore, in this dissertation, we target this problem by presenting a fair quantitative evaluation of the cost and the performance of different NoC standard architectures. More precisely, we use our standard architecture optimization methodology, presented in Chapter 6, to carry out power, area, delay, and reliability evaluations of NoC standard architectures based on the optimum cores mapping onto these architectures.

2.5.2 ASNoC Realization using Semi-Custom Architectures

The research work in semi-custom architecture realization partially customized the underlying on-chip network architecture to comply with certain application requirements. Accordingly, semi-custom architectures could be considered intermediate between general-purpose standard architectures and application-specific full-custom ones. To the best of our knowledge, this partial customization was carried out by

two main techniques: single standard architecture modification and multiple standard architectures combination.

For the modification technique, a specific standard architecture was slightly modified to enhance certain cost and performance metrics. The most famous modification-based semi-custom architecture generation methodologies were the long-range link insertion and the link removal ones. On one hand, the long-range link insertion methodology was presented in [75] to enhance the delay of a 2D mesh architecture by connecting distant nodes, which communicate with each other frequently, by direct long-range links. It was also used with the ring architecture in [146]. On the other hand, the link removal methodology was presented in [147] to reduce the area of 2D mesh architecture by removing direct links in between cores, which do not communicate with each other very frequently. It was also used with the spidergon architecture in [146].

For the combination technique, multiple standard architectures were combined together to generate an overall semi-custom architecture that was better than any of the combined architectures with respect to certain cost and performance metrics. The combined architectures might be similar or different. Accordingly, power-efficient semi-custom architectures were obtained by combining two ring architectures in [110] and tree architectures in [148]. Moreover, a delay efficient semi-custom architecture was obtained by combining star with mesh architectures in [149]. As such, these previous combination-based methodologies assumed a fixed semi-custom underlaying network architecture such that the designer could not select the standard architectures to be combined. Moreover, to the best of our knowledge, there was no mathematical formulation proposed of how to partition application cores over the standard architectures used to build the overall semi-custom one. In this dissertation, we target these problems by formulating and presenting an architecture realization methodology, in Chapter 4, based on network partitioning techniques to generate NoC

semi-custom architectures with lower power consumption and area. Furthermore, our methodology combines any standard architectures based on the designer needs.

In parallel to our research work, many semi-custom architecture generation methodologies using network partitioning were presented in [117–119]. We previously discussed and surveyed these methodologies in Section 2.3. Nevertheless, our work is still unique in two aspects. First, we mathematically formulate the use of network partitioning as a cost-effective way for ASNoC architecture generation. Second, our methodology is a system-level one that allows for a quick generation of the underlying architecture and an early evaluation of different design alternatives.

2.5.3 ASNoC Realization using Full-Custom Architectures

Full-custom architectures outperform other categories of architectures with respect to the metric for which customization is carried out. In the literature, full-custom architectures were realized by using either heuristics or optimization-based techniques. For example, heuristics were used for power-oriented full-custom architecture generation in [150, 151]. Heuristics help generating the required architecture in a reasonable time. However, the use of heuristics could not guarantee the best possible cost nor performance.

Optimization-based techniques required long generation time, but they guaranteed the best possible cost and performance. To the best of our knowledge, two optimization-based full-custom architecture generation methodologies were presented in the literature: a Mixed Integer Linear Programming (MILP)-based methodology in [152] and a GA-based one in [153]. The two methodologies used circuit-level information from the floorplanning to minimize the power consumption of the underlying on-chip network architecture. The two methodologies had two main drawbacks. First, both were proposed for a power consumption minimization. Although power is a first-class objective for NoC-based systems, including other

metrics into the optimization process is of a great importance to the NoC research community as well. Second, as the two methodologies carried out an iterative floorplanning to minimize the power consumption on the circuit-level, they took too long to generate the required architectures. For example, the MILP-based methodology failed completely to generate architectures for applications of more than twelve cores within a time out period of 8 hours. Therefore, a system-level multi-objective optimization methodology for NoC full-custom architecture generation constitutes an urgent need to the NoC research community. In this dissertation, we fulfill this need by presenting that methodology. More precisely, in Chapter 5, we present a multi-objective methodology using GA to generate an optimized full-custom NoC architecture with respect to power, area, delay, and reliability, simultaneously. Our methodology is a system-level one to quickly generate the required architecture and to be practically valuable.

In parallel to our research work, many full-custom architecture generation methodologies were presented. For example, a GA-based power-oriented methodology was presented in [154] and a heuristic-based throughput-oriented methodology was presented in [80]. However, to the best of our knowledge, our methodology is still unique in being a multi-objective and a system-level one that allows for an early design space exploration.

2.5.4 Overall Summary of ASNoC Architecture Realization Techniques

This subsection concludes our review of the work done by different research groups with respect to on-chip network architecture realization. We summarize the major work presented till now regarding standard, semi-custom, and full-custom NoC architectures. In a chronic order for each category of these architectures, Table 2.3 represents this summary in a more concise form. The techniques, whether algorithmic,

heuristic, or optimization-based, employed to realize the underlying architecture are also included. Similar to our previous summary of NoC evaluation models, presented in Subsection 2.2.5, the table further emphasizes that customizing NoC architectures with respect to power consumption had most of the efforts exerted by different research groups. Moreover, various issues related to NoC reliability are still virgin research areas that require lots of efforts. Finally, for each category of NoC architectures, the table clearly shows the demanding need of the NoC research community to multi-objective NoC architecture generation methodologies.

2.6 Chapter Summary

This chapter presented a state-of-the-art review of the research work done in different NoC areas that are addressed in this dissertation. Throughout the whole chapter, we surveyed the work done by different research groups before and in parallel to ours. First, we surveyed various models that were used for NoC power, area, delay, and reliability evaluation, respectively. We then explored different network partitioning techniques and how they were employed in the on-chip networking domain. Thereafter, we quickly highlighted the basic principles of GA as a bio-inspired optimization technique. Finally, we reviewed the work presented for on-chip network architecture realization. The research work done with respect to the three main ASNoC architectures; namely, standard, semi-custom, and full-custom were discussed in details.

In the next chapter, we propose system-level power and reliability evaluation models. The two models are used, with area and delay models adopted from the literature, to formulate our multi-objective optimization function. The models are also used throughout this dissertation to evaluate the cost and the performance of different NoC architectures.

Table 2.3: Summary of the work done by different research groups for NoC architecture realization. (P: Power, A: Area, D: Delay, and R: Reliability)

Mapping cores onto standard architectures						
Year, [Ref.]	**Network**	**Realization technique**	**Metric**			
			P	**A**	**D**	**R**
2000, [132]	Arbitrary	Algorithmic			X	
2003, [57]	2D mesh	Algorithmic	X			
2004, [133]	Arbitrary	Algorithmic			X	
2005, [135]	2D mesh	Optimization, EA	X			
2006, [73]	2D mesh	Optimization, GA			X	
2007, [134]	2D mesh	Algorithmic			X	
2008, [136]	2D mesh	Optimization, simulated annealing			X	
2009, [138]	WK-recursive	Algorithmic	X			
2009, [139]	2D mesh	Algorithmic	X			
2010, [140]	2D mesh	Optimization, GA	X			
Evaluation of standard architectures						
Year, [Ref.]	**Networks**		**Metric**			
			P	**A**	**D**	**R**
2005, [145]	2D Mesh, torus, and fat tree		X		X	
2005, [70]	2D Mesh, torus, folded torus, SPIN, octagon, and BFT		X	X	X	
2005, [56]	2D Mesh		X		X	X
2006, [144]	2D Mesh, ring, and spidergon				X	
2007, [81]	2D and 3D grids		X		X	
2008, [142]	2D grids, 3D grids, and 3D trees		X	X	X	
2009, [71]	3D grids and trees		X	X	X	
Semi-custom architecture generation						
Year, [Ref.]	**Network**	**Realization technique**	**Metric**			
			P	**A**	**D**	**R**
2004, [110]	Ring	Algorithmic, combination-based	X			
2006, [75]	2D mesh	Algorithmic, modification-based			X	
2007, [146]	Ring and spidergon	Algorithmic, modification-based	X	X	X	
2007, [141]	Mesh and tree	Algorithmic, combination-based			X	
2008, [148]	Tree	Algorithmic, combination-based	X			
2008, [147]	2D mesh	Algorithmic, modification-based		X		
2008, [149]	Mesh and star	Algorithmic, combination-based			X	
2010, [117]	Mesh and bus	Algorithmic, combination-based			X	
Full-custom architecture generation						
Year, [Ref.]	**Realization technique**		**Metric**			
			P	**A**	**D**	**R**
2005, [153]	Optimization, GA		X			
2006, [150]	Heuristic		X			
2006, [152]	Optimization, MILP		X			
2008, [151]	Heuristic		X			
2009, [154]	Optimization, GA		X			
2009, [80]	Heuristic				X	

Chapter 3

NoC Evaluation Models

This chapter presents the first contribution of this dissertation. We propose two system-level NoC performance evaluation models for power and reliability. Although we are adopting area and delay models from the literature throughout this dissertation, they are also discussed in this chapter for completeness. After explaining the four models, our multi-objective function formulation is presented. Finally, the models are analyzed to explore different design variables behind them that control the power, area, delay, and reliability of different NoC architectures.

This chapter is organized as follows. Section 3.1 gives an introduction about NoC modeling. Section 3.2 presents some information of how to represent NoC applications and architectures in the system-level using graph theory concepts. The power, area, delay, and reliability models are represented in Sections 3.3, 3.4, 3.5, and 3.6, respectively. Section 3.7 discusses the multi-objective function formulation. Evaluation models are analyzed in Section 3.8 to indicate the design variables that govern the NoC architecture customization process. Finally, Section 3.9 summarizes the chapter.

3.1 Introduction

NoC-based systems have a substantially different behavior than traditional computer networks. Accordingly, using models related to the latter in testing the former might lead to inaccurate evaluation of tested designs. Therefore, evaluation models that take into consideration the nature of NoC-based systems need to be addressed. Moreover, it is of great importance to have these models at the system-level to allow the evaluation to be done early during the design process. The advantages of this early evaluation is threefold. First, it permits the designer to check the effect of different design alternatives on the NoC cost and performance from the beginning of the design process. Second, it allows different on-chip network architectures to be quickly compared and evaluated at the system-level. Third, system-level evaluation models enable an early discovery of design errors and flaws, preventing any serious consequences. In this chapter, we target these NoC-oriented system-level performance and cost evaluation models. More precisely, this chapter presents two main contributions:

1. Presenting a system-level analytical model to evaluate the power consumption of NoCs. The model represents the power consumption in both NoC routers and links, and

2. Presenting a system-level probabilistic model to evaluate the reliability of NoCs. The model captures the possibility of the NoC traffic being altered by on-chip noise sources.

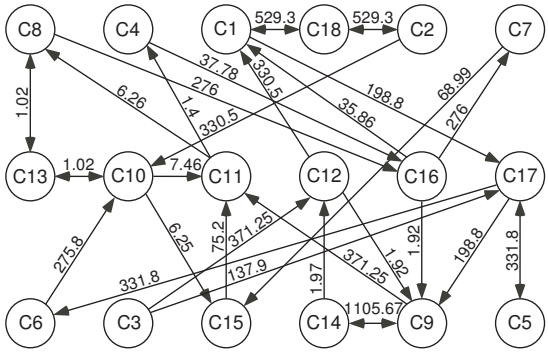
3.2 Graph Theory Representation of NoCs

Graph theory concepts could be used to represent NoC applications at the system-level. Given the application description and traffic characteristics, any application

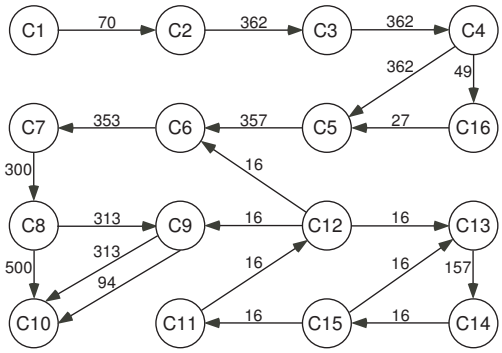
could be described by a core graph[1] [133, 143]. Core graph represents the processing elements in the application and the traffic between them. For example, Figure 3.1 shows sample core graphs of five real NoC benchmark applications. These are an Audio Video (AV) benchmark with 18 cores [10, 80], a Video Object Plane Decoder (VOPD) benchmark with 16 cores [133, 134], a 263 decoder MP3 decoder (263DEC) with 14 cores [152], an MPEG-4 decoder with 12 cores [143], and a Multi Window Display (MWD) benchmark with 9 cores [80]. As shown in the figure, a core graph is a directed graph G($\mathcal{C}$, $\mathcal{E}$), where each vertex $c_i \in \mathcal{C}$ represents a core in the application, and the directed edge $e_{ij} \in \mathcal{E}$ represents the communication between cores c_i and c_j. The weight λ_{ij} of the edge e_{ij} expresses the number of packets/time step sent from core c_i to core c_j. Using NoC terminology, a core graph is usually represented in the form of a traffic distribution matrix (Λ) [11, 155]. For N cores, the dimensions of the matrix is $N \times N$. Any element λ_{ij} in the matrix represents the weight of the edge e_{ij}. That is, the number of packets sent from core c_i and core c_j per time step. For example, the traffic distribution matrix for the MWD benchmark shown in Figure 3.1(e) is represented as:

$$\begin{bmatrix} 0 & 64 & 128 & 0 & 0 & 0 & 0 & 0 & 0 \\ 64 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 128 \\ 128 & 0 & 0 & 192 & 96 & 0 & 0 & 0 & 0 \\ 0 & 0 & 192 & 0 & 96 & 0 & 0 & 0 & 96 \\ 0 & 0 & 0 & 96 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 64 & 0 & 64 \\ 0 & 0 & 0 & 0 & 0 & 64 & 0 & 64 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 64 & 0 & 0 \\ 0 & 128 & 0 & 96 & 192 & 64 & 0 & 0 & 0 \end{bmatrix}$$

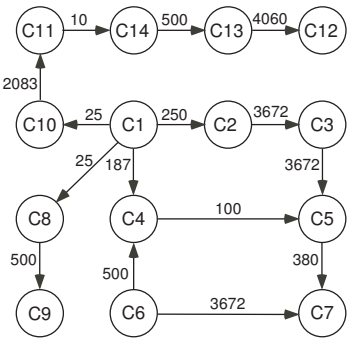
[1]Core graph is also called Application Characterization Graph (APCG) or Communication Weighted Graph (CWG).



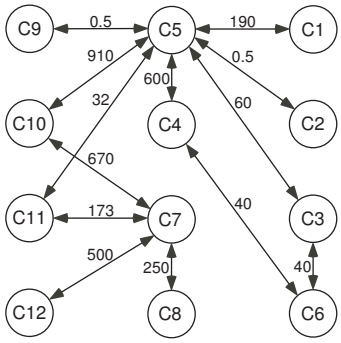
(a) Audio Video (AV) [10, 80]



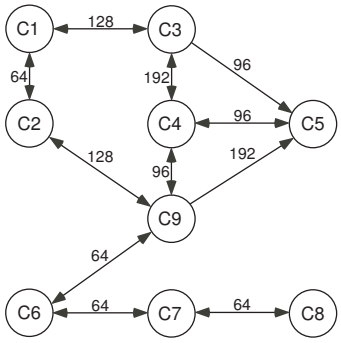
(b) Video Object Plane Decoder (VOPD) [133, 134]



(c) 263 dec mp3 dec (263DEC) [152]



(d) MPEG-4 [143]

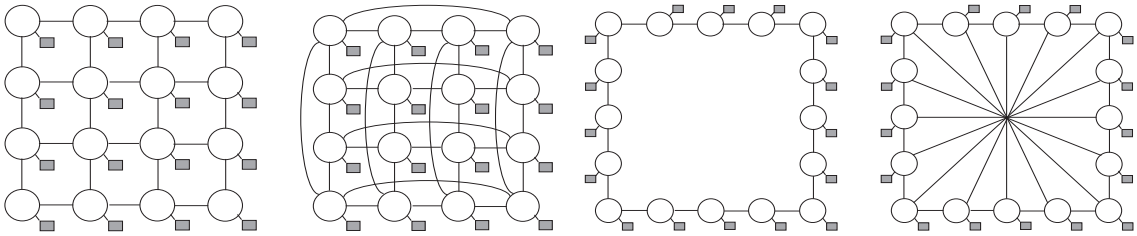


(e) Multi Window Display (MWD) [80]

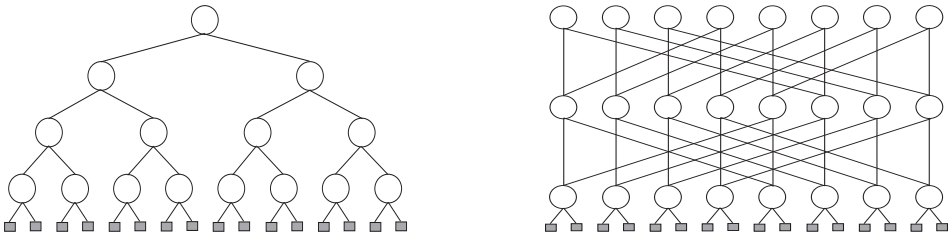
Figure 3.1: Examples of core graphs for different NoC benchmarks.

Similarly, graph-theoretic concepts could be used for a system-level representation of different NoC architectures. Throughout this dissertation, we use single capital letters to represent full-custom architectures, double capital letters to represent semi-custom architectures, and triple capital letters to represent standard architectures. For example, Figure 3.2 shows eight different standard architectures [156]: Mesh (MSH), Torus (TRS), Ring (RNG), Polygon (PLG), Binary Tree (BNT), 2-ary 3-fly butterfly tree (BFT23), 2-ary 2-fly butterfly tree (BFT22), and 2-ary 2-stage clos (CLS). Polygon is only represented for architectures with even number of vertices

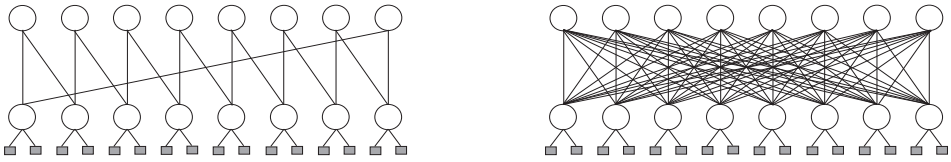
and is actually a ring architecture with diagonal links to connect distant vertices. Each of the architectures in Figure 3.2 could be represented using a graph G($\mathcal{V}$,$\mathcal{L}$), where each vertex $v_i \in \mathcal{V}$ represents a processing unit in the architecture, which might be a core or a router, and $\mathcal{L}$ is a set of edges that represent the physical communication links between these processing units.



(a) Mesh (MSH) (b) Torus (TRS) (c) Ring (RNG) (d) Polygon (PLG)



(e) Binary Tree (BNT) (f) 2-ary 3-fly (BFT23)



(g) 2-ary 2-fly (BFT22) (h) 2-ary 2-stage clos (CLS)

Figure 3.2: Examples of NoC standard architectures [156]. (Routers are represented by white circles, whereas cores are represented by dark squares.)

3.3 Power Model

NoC power requirements are mainly caused by its routers and links. To model the router power in the system-level, we implemented a library of output queuing routers with different number of ports and buffer sizes [157, 158]. As shown in Figure 3.3, ports within the same routers have equal buffer sizes and each of them consists of two channels for packets reception and transmission, respectively. A round robin scheduler serves backlogged queues at the output one after another in a fixed order. Thereafter, for everyone of the routers we implemented, we carried out power simulations at various operating frequencies and target technologies. For example, for 0.18μm technology and with an operating frequency of 500MHz, Table 3.1 shows sample results of the power consumptions of 8-flit routers with different number of ports at various average flit arrival rates, in *flit/cycle*. The last row represents the routers leakage power only.

Table 3.1: Power consumption of NoC output queuing routers with different number of ports at various flit arrival rates. (at 500MHz and implemented in 0.18μm technology.)

Flit arrival	**Total Power** (P_R) (mW)				
rate(flit/cycle)	**4-port**	**5-port**	**6-port**	**7-port**	**8-port**
1.000	64.104	96.885	136.044	137.379	234.287
0.400	32.019	48.440	68.041	86.709	117.173
0.200	12.793	19.380	27.229	34.706	46.901
0.100	6.410	9.705	13.635	17.372	23.481
0.050	3.211	4.862	6.832	8.705	11.762
0.020	1.293	1.963	2.747	3.505	4.726
0.002	0.135	0.203	0.285	0.38	0.487
0.000	0.008	0.013	0.018	0.025	0.032

From the experiments we run, for constant buffer size routers, we noticed linear dependencies of both dynamic and leakage powers on the number of ports and the average flit arrival rate. Therefore, we used linear curve fitting to calculate both

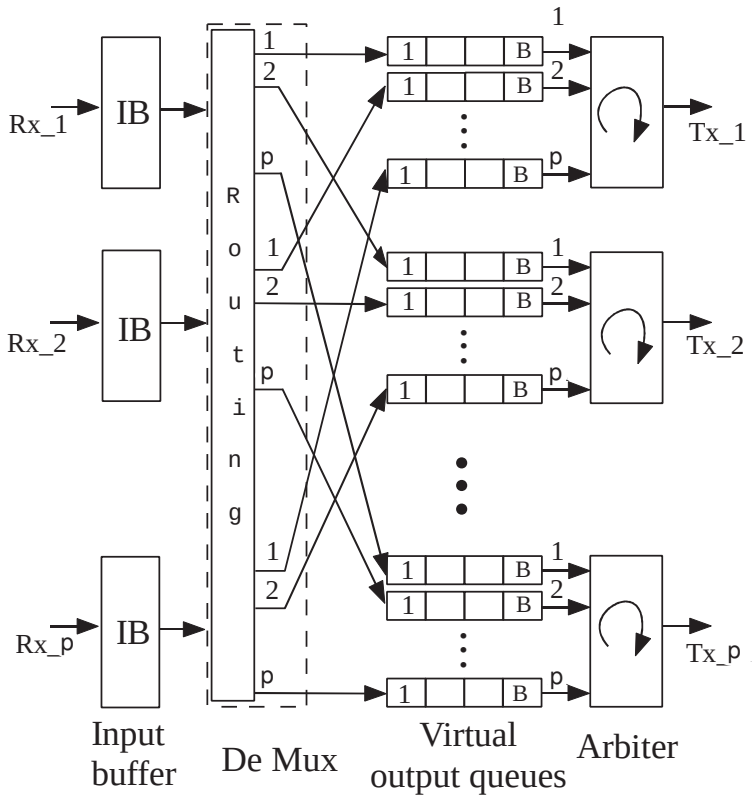


Figure 3.3: Example of a $p \times p$ output queuing router. (Rx represents an input port, Tx represents an output port, p is the number of ports, and B is the maximum queue size.)

powers for routers with any number of ports and flit arrival rate. As a result, the power of router i is modeled in the system-level as

$$P_{Ri} = P_{Ri-Dynamic} + P_{Ri-Leakage} \tag{3.1}$$

$$P_{Ri-Dynamic} = (k_{Dp} \cdot p_i + k_D) \cdot \alpha_{fi} \tag{3.2}$$

$$P_{Ri-Leakage} = (k_{Lp} \cdot p_i + k_L) \cdot \alpha_{fi} \tag{3.3}$$

where $P_{Ri-Dynamic}$, $P_{Ri-Leakage}$, and P_{Ri} are the dynamic, leakage, and total power of router i, respectively. p_i and α_{fi} are the number of ports and the average flit arrival

rate over all the ports of router i. For any architecture, the average flit injection rate could be calculated for each router by the knowledge of the NoC operating frequency, the employed routing strategy, and the traffic distribution matrix (Λ), discussed in Section 3.2. Finally, k_{Dp}, k_D, k_{Lp}, and k_L are the dynamic port-dependent, the dynamic port-independent, the leakage port-dependent, and the leakage port-independent power constants, respectively. These constants are technology-dependent and could be obtained by linear curve fitting of the power simulation results of the employed routers. The power simulation should be done once for each router in the designer library. The resultant power constants will be then used in evaluating and generating different NoC architectures. Consequently, for any architecture with N_R routers, the total router power is represented as

$$P_R = \sum_{i=1}^{N_R} P_{Ri} \tag{3.4}$$

For link power modeling, NoC links are either global or local. Global links connect routers, whereas local links connect cores to routers. Global links are usually long and require driving circuits and repeaters [62]. Therefore, it is shown in [46, 159] that the overall link power, area, and delay are completely dominated by those of global links. Accordingly, throughout this dissertation, the link models proposed and the experimental results presented are to abstract these global links.

To send a certain traffic from core c_i to core c_j, the power consumed in links depends on the amount of traffic transferred, λ_{ij}, the physical characteristics of the links, and the number of hops between the source and the destination cores. Link physical characteristic could be obtained from the corresponding technology library. Moreover, the number of hops between any two cores could be obtained from the architecture connectivity matrix (C) [160]. Any element c_{ij} in the connectivity matrix represents the number of hops between routers R_i and R_j. The connectivity matrix,

in turn, is calculated from the adjacency matrix (A) of the architecture [161]. For an architecture with N_R routers, the adjacency matrix is a binary $N_R \times N_R$ matrix, whereas $a_{ij} = 1$ only if there is a direct link connecting between routers R_i and R_j. Figure 3.4 shows an example for the adjacency and the connectivity matrices of a 6-node ring architecture. In this dissertation, the connectivity matrix is calculated from the adjacency matrix using the Dijkstra shortest path algorithm [162]. Accordingly, given that every core is connected to exactly one router, NoC link power is represented in the system-level as

$$P_L = \sum_{i=1}^{N} \sum_{j=1}^{N} \lambda_{ij} \cdot c_{ij} \cdot P_{unit\ link} \tag{3.5}$$

where N is the number of cores in the application. λ_{ij} is the traffic sent from core c_i to core c_j in packet/time step. Similarly, c_{ij} is the number of hops between the same two cores. $P_{unit\ link}$ is the power consumed by a single packet in one unit link. Previous models were proposed to represent this unit link power based on parameters from the targeted fabrication technology. In this dissertation, we employ the unit link model presented in [43]

$$P_{unit\ link} = P_{switching} + P_{short} + P_{Leakage} \tag{3.6}$$

$$P_{switching} = \frac{1}{2} N_w \cdot V_{dd}^2 \cdot (C_S \cdot \alpha_S + C_C \cdot \alpha_C) \cdot f_{op} \tag{3.7}$$

$$P_{short} = N_w \cdot \tau_{sc} \cdot \alpha_S \cdot V_{dd} \cdot I_{short} \cdot f_{op} \tag{3.8}$$

$$P_{Leakage} = N_w \cdot V_{dd} \cdot (I_{bias,wire} + I_{leak}) \tag{3.9}$$

where $P_{switching}$, P_{short}, and $P_{Leakage}$ are the link switching, short circuit, and leakage power, respectively. The summation of $P_{switching}$ and P_{short} represents the dynamic power consumed in a link. N_w is the number of wires in a link, i.e.,

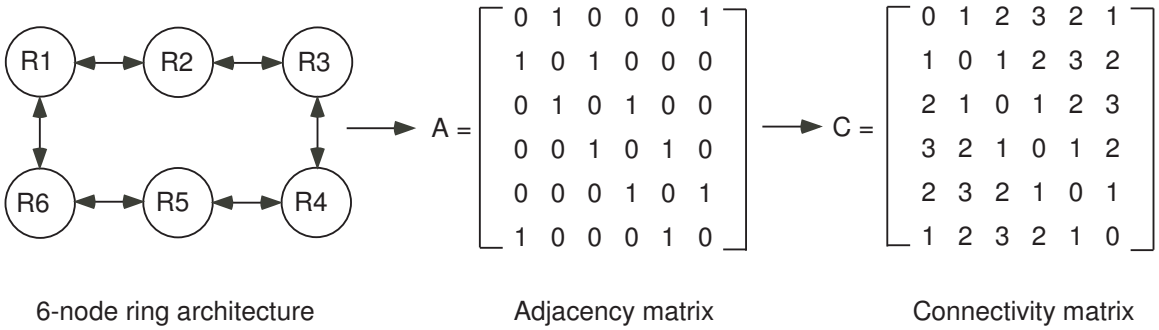


Figure 3.4: An example for adjacency and connectivity matrices of a 6-node ring architecture.

channel width, and V_{dd} is the supply voltage. C_S and C_C represent self and coupling capacitance of a wire and with neighboring wires, respectively. Similarly, α_S is the switching activity on a wire and α_C is the switching activity from the adjacent wires. f_{op} denotes the operating frequency and τ_{sc} is the short circuit period during which I_{short} flows between source and ground. Finally, $I_{bias,wire}$ represents the current flowing from the wire to its substrate and I_{leak} is the leakage current flowing from the source to ground regardless of the gate's state and switching activity. All the parameters for link power calculation could be obtained from the Predictive Technology Model (PTM) [163].

Finally, the total NoC power consumption (P_{NoC}) is represented by

$$P_{NoC} = P_R + P_L \tag{3.10}$$

3.4 Area Model

Similar to NoC power, NoC area consists of the area of its routers and links. As explained in Section 3.3, we are using a library of pre-designed routers with different number of ports. Accordingly, the total router area, in μm^2, is expressed as

$$A_R = \sum_{i=1}^{N_R} A_{Ri} \tag{3.11}$$

where N_R is the number of routers in the network and A_{Ri} is the area of router i.

The link area depends on the number of wires per each link (i.e., the channel width), the wire length and width, and the spacing between wires. In this dissertation, we assume a fixed channel width. Moreover, the wire width and the spacing between wires are technology-dependent and could be obtained from the corresponding technology library [163]. As a result, the link area, in μm^2, is expressed, as in [62], as

$$A_L = N_L \cdot (N_w \cdot (w_w + s_w) + s_w) \cdot l_l \tag{3.12}$$

where A_L is the link area, N_L is the number of links within the network, and N_w is the channel width. w_w, s_w, and l_l are the wire width, the inter-wires spacing, and the wire length for global interconnects, respectively.

Finally, the total NoC area (A_{NoC}), in μm^2, is expressed as

$$A_{NoC} = A_R + A_L \tag{3.13}$$

3.5 Delay Model

The traffic from any source core experiences three types of delays in its way to the destination core. These are the arbitration and propagation delays through routers, the propagation delay through links, and the serialization and de-serialization delays through NIs. In this dissertation, we use the average zero-load delay model presented in [81]. The average zero-load delay model ignores any competition on NoC resources and assumes a contention-free network. Therefore, it is a fast system-level model to check for the effect of different architectures on NoC delay. This delay model was

originally proposed for 3D NoCs. However, it could also be adopted with 2D NoCs that we are targeting in this dissertation. Accordingly, the overall NoC average delay (D_{NoC}), in *seconds*, is represented as

$$D_{NoC} = \mu \cdot (t_a + t_r) + (\mu + 1) \cdot t_l + \frac{N_b}{N_w} \cdot t_l \tag{3.14}$$

where

$$t_a = (\frac{21}{4} \cdot \log_2 \ p_{avg} + \frac{14}{12} + 9) \cdot \tau \tag{3.15}$$

$$\begin{aligned} t_{l(r)} =& 0.377 \cdot r_{l(r)} \cdot c_{l(r)} \cdot l_{l(r)}^2 + 0.693 \cdot (R_{d0} \cdot C_0 \\ &+ R_{d0} \cdot c_{l(r)} \cdot l_{l(r)} + r_{l(r)} \cdot l_{l(r)} \cdot C_{g0}) \end{aligned} \tag{3.16}$$

$$l_r = 2 \cdot (w_r + s_r) \cdot N_w \cdot p_{avg} \tag{3.17}$$

$$R_{d0} = 0.98 \cdot \frac{V_{dd}}{I_{d0}} \tag{3.18}$$

where μ is the average internode distance [75] (i.e., average number of routers between a source node and a destination node). t_a and t_r are the router arbitration and propagation delays, respectively. Similarly, t_l is the link propagation delay. N_b and N_w are the number of bits per packet and the channel width, respectively. p_{avg} is the average number of ports per router and τ is the delay of a minimum sized inverter of the target technology. $r_{l(r)}$ and $c_{l(r)}$ are the per unit length resistance and capacitance of the link (router) wires, respectively. $l_{l(r)}$ is the wire length for the link (router crossbar). R_{d0} and C_{g0} are the equivalent output resistance and the gate capacitance of a minimum sized inverter of the target technology, respectively. C_0 is the total input capacitance of a minimum sized inverter of the target technology, which is the summation of the gate and the drain capacitances. w_r and s_r are the wire width and

the inter-wires spacing for router internal interconnects, respectively. V_{dd} is the supply voltage and I_{d0} is the drain current when both the drain and the gate voltages are equal to the supply voltage. The values of all the technology-dependent parameters could be obtained from the corresponding technology library [163]. Finally, for an N-cores application, the average internode distance (μ), in *hops*, could be calculated using the connectivity matrix (C) of the architecture and the traffic distribution matrix (Λ) of the application as

$$\mu = \frac{\sum_{i=1}^{N}\sum_{j=1}^{N}\lambda_{ij}\cdot c_{ij}}{\sum_{i=1}^{N}\sum_{j=1}^{N}\lambda_{ij}} \tag{3.19}$$

3.6 Reliability Model

NoC reliability calculation becomes currently dominated by the many noise sources existing in the modern DSM technology [6]. Therefore, in this dissertation, we define the reliability of an NoC as the probability of transmitting application packets from source nodes to destination nodes successfully in the presence of noise. Using different representations for the on-chip noise sources will not affect our system-level model. However, in the literature, Gaussian model is widely used to represent these on-chip noise sources [88]. Accordingly, the sum of different on-chip noise sources is modeled as a single noise source with a noise standard deviation σ. As a result, the BER is represented as [164]

$$BER = \frac{1}{\sqrt{2\pi}}\int_{\frac{V_{dd}}{2\sigma}}^{\infty} e^{-\frac{u^2}{2}}du \tag{3.20}$$

where V_{dd} is the supply voltage and σ is the noise standard deviation. Thereafter, we consider the Simple Non-Fault-Tolerant (SNFT) or the Automatic Repeat reQuest (ARQ) communication schemes, in which a single bit in-error causes the whole packet to be a corrupted one [89]. Therefore, for a single link (l_{ij}), the probability ($\mathcal{P}_{ij}$) of

sending λ_{ij} packets, each of size N_b bits, successfully over that link is represented as

$$\mathcal{P}_{ij} = (1 - BER)^{\lambda_{ij} \cdot N_b} \tag{3.21}$$

The probability of transferring a single traffic trace of λ_{ij} packets from a source core c_i to a destination core c_j that have c_{ij} hops in between could be obtained by raising the above mentioned probability to the power of c_{ij}. Accordingly, the probability of transferring all application traffic traces from all sources to all destinations successfully in the presence of noise; i.e., the overall NoC reliability (R_{NoC}) is represented as

$$R_{NoC} = \prod_{i=1}^{N} \prod_{j=1}^{N} \mathcal{P}_{ij}^{c_{ij}} \tag{3.22}$$

where N is the number of cores within the application. $\mathcal{P}_{ij}$ is the probability of transmitting λ_{ij} packets successfully over a single link l_{ij} in the presence of noise and c_{ij} is the number of hops a packet goes through during its transition from the source core c_i to the destination core c_j.

3.7 Multi-Objective Function Formulation

In this section, we formulate our multi-objective function by combining the cost and performance models represented in the previous sections. For any given application, this function is used by our multi-objective GA-based methodologies in Chapters 5 and 6 to generate the best NoC architecture for this application. In general, it is required by this formulation to maximize the reliability while minimize all other metrics. Therefore, our multi-objective function is formulated as the product of the power, the area, and the average delay over the network reliability. Using the product for the formulation ensures fairness between different metrics. This could be

checked by differentiating the objective function to calculate the relative change in its value. This relative change is proportional to the summation of relative changes of all the metrics. Accordingly, our product-based formulation guarantees that non of the metrics will dominate the optimization process. Therefore, our objective function is expressed as

$$f = \frac{(P_{NoC})^{\gamma_P} \cdot (A_{NoC})^{\gamma_A} \cdot (D_{NoC})^{\gamma_D}}{(R_{NoC})^{\gamma_R}} \tag{3.23}$$

where γ_P, γ_A, γ_D, and γ_R are power, area, delay, and reliability weight factors, respectively, which could be used by the designer to control the optimization process. A higher value of any of these factors over the others gives high importance for optimizing with respect to the metric it represents compared to other metrics. Moreover, setting any of these factors to zero, cancels any effect of the corresponding metric on the optimization process. Finally, P_{NoC}, A_{NoC}, D_{NoC}, and R_{NoC} are the NoC power, area, average delay, and reliability as expressed by (3.10), (3.13), (3.14), and (3.22), respectively.

3.8 Exploration of Design Variables

In this section, we explore the design variables behind different NoC metrics. Throughout this dissertation, we use the term, design variables, to express those variables that control the exact value of different NoC metrics. These variables might be explicitly included in the evaluation models presented in this chapter, like the number of routers, or implicitly implied by them, like the routing scheme. The investigation of these design variables helps us understanding how could we enhance different NoC metrics. Consequently, this understanding facilitates proposing different methodologies to customize on-chip network architecture with respect to one or more of these metrics.

The evaluation models and experiments we conducted are further analyzed to investigate the main design variables that govern the value of different NoC metrics. Table 3.2 shows the relevant variables from this analysis. The table classifies these design variables into two categories: independent and intermediate. The former are those variables that are completely independent, whereas the latter are those variables that rely on one or more of the independent ones. Independent design variables are the main targets of architecture customization methodologies as any change on them directly impacts different NoC metrics. We further divide independent variables into symbolic and non-symbolic. Symbolic variables are those explicitly included in different metric models, like the supply voltage (V_{dd}). In contrast, non-symbolic variables are those directly affecting NoC metrics; however, they are not explicitly included in their associated models, like the mapping technique. Finally, intermediate design variables are those relying solely on independent non-symbolic variables. Accordingly, they are halfway between completely independent variables and overall NoC metrics. Throughout this dissertation, we use intermediate variables to numerically quantify the effect of changing non-symbolic independent variables on NoC metrics.

Table 3.2 shows that NoC architecture customization methodologies need to trade off many design variables to realize the best architecture for any given application. As a result, previous work used to assume some of these variables to be fixed to all candidate architectures to facilitate and speed up the customization process. The methodologies presented in this dissertation are not exceptional to this assumptions. Therefore, Table 3.2 also explains how our semi-custom, full-custom, and standard architecture customization methodologies handle these design variables. More precisely, different design variables are dealt with by our methodologies by one of three approaches. First, some variables, like the number of ports per router, are traded off by the methodology itself to enhance, or optimize for, certain NoC

Table 3.2: Summary of different design variables and how our methodologies deal with them.

<table>
<tr><th colspan="5">Independent variables</th></tr>
<tr><th colspan="2" rowspan="2">Design variable</th><th colspan="3">Methodologies presented in this dissertation</th></tr>
<tr><th>Semi-custom</th><th>Full-custom</th><th>Standard</th></tr>
<tr><td rowspan="7">Symbolic</td><td>No. of ports per router (p)</td><td>Traded off</td><td>Traded off</td><td>Traded off</td></tr>
<tr><td>Number of links (N_L)</td><td>Traded off</td><td>Traded off</td><td>Traded off</td></tr>
<tr><td>No. of routers (N_R)</td><td>Input</td><td>Assumed fixed</td><td>Traded off</td></tr>
<tr><td>Supply voltage (V_{dd})</td><td colspan="3">Input</td></tr>
<tr><td>Frequency of operation (f_{op})</td><td colspan="3">Input</td></tr>
<tr><td>Buffer size (B)</td><td colspan="3">Assumed fixed</td></tr>
<tr><td>No. of wires per link (N_w)</td><td colspan="3">Assumed fixed</td></tr>
<tr><td rowspan="4">Non-symbolic</td><td>Traffic</td><td colspan="3">Input</td></tr>
<tr><td>Technology</td><td colspan="3">Input</td></tr>
<tr><td>Mapping</td><td>Input</td><td colspan="2">Traded off</td></tr>
<tr><td>Routing</td><td>Assumed fixed</td><td>Traded off</td><td>Assumed fixed</td></tr>
<tr><th colspan="5">Intermediate variables</th></tr>
<tr><td colspan="5">* Average internode distance (μ) = f(traffic, mapping, routing)
* Flit arrival rate (α_{fi}) = f(traffic, mapping, routing)
* Probability of the traffic being affected by noise ($\mathcal{P}$) = f(traffic, technology, mapping, routing)
* Link length (l_l) = f(technology)</td></tr>
</table>

metrics. Second, some variables, like the frequency of operation, are considered during the customization process; however, they should be provided as inputs to the methodologies. To check for the effect of these variables on different NoC metrics, the designer needs to run our methodologies iteratively with different values of these variables. Finally, the remaining variables, like the number of wires per link, are assumed explicitly fixed to speed up the architecture generation process. The specific assumptions for each methodology are discussed in details in Sections 4.2, 5.2, and 6.2 for the semi-custom, full-custom, and standard architecture generation methodologies, respectively.

3.9 Chapter Summary

In this chapter, we presented the models for different metrics considered in this dissertation. First, we discussed how graph theory concepts could be used to

represent different NoC applications and architectures at the system-level. NoC area and delay models were adopted from the literature, whereas we proposed two new models for NoC power consumption and reliability. Both are system-level models that could be used for quick evaluation and comparison between different NoC architectures. The four models were combined together to establish our multi-objective function that will be used in Chapters 5 and 6 to generate the best NoC full-custom and standard architecture, respectively, with respect to the four metrics, simultaneously. Furthermore, the models were analyzed to investigate which design variables that differentiate between the cost and the performance of various NoC architectures. Finally, these models were used during our research work and they are used throughout this dissertation. Accordingly, they appeared in most of our publications [11–14, 20–22, 24].

In the next chapter, we present a cost-efficient customization methodology for NoC semi-custom architectures. The methodology uses network partitioning techniques to reduce power and area costs of NoC-based systems. The impact of using network partitioning on NoC power, area, and delay is also analyzed.

Chapter 4

A Cost-Efficient Customization Methodology of ASNoC Semi-Custom Architectures using Network Partitioning

This chapter presents the second contribution of this dissertation. For any ASNoC-based application, we propose a customization methodology to generate an on-chip semi-custom architecture with lower power consumption and area. Network partitioning techniques are used to reduce both the routers buffering and the number of links, which are the main sources for power consumption and area in an ASNoC. The impact of using partitioning on different ASNoC metrics; namely, power, area, and delay is analyzed. A methodology for custom architecture generation that employs network partitioning is proposed. The methodology is then applied to different real benchmark applications with different number of cores, as case studies. Results show that the proposed methodology is a promising way to reduce the ASNoC power consumption and area compared to other standard and semi-custom architecture generation techniques.

This chapter is organized as follows. Section 4.1 gives an introduction about

ASNoC architecture customization problem. The assumptions of our methodology are presented in Section 4.2. Formulation of the network partitioning problem are discussed in Section 4.3. The impact of using network partitioning on different NoC metrics is investigated in Section 4.4. Section 4.5 presents the proposed semi-custom architecture generation methodology. Section 4.6 represents some experimental results and verifies the efficiency of our methodology through case studies. Finally, Section 4.7 summarizes the chapter.

4.1 Introduction

Recently, NoC design community started focusing on customizing the design of on-chip networks to comply with different requirements of modern NoC applications. The cost and the performance of any NoC depend primarily on its architecture [5, 75]. Accordingly, one problem, which needs to be addressed is to customize these architectures to meet the target application requirements, constraints, and traffic characteristics, while reducing the associated costs. That is, to minimize both design-time and run-time costs. Run-time cost is dominated by the power consumption, whereas the design area is the key element in determining the design-time cost.

In this chapter, we present an architecture generation methodology to solve the above mentioned cost reduction problem. More precisely, for any application, we employ network partitioning techniques to realize a semi-custom NoC architecture with lower power and area costs. Our methodology achieves this lower costs by reducing the number of ports per router with their associated buffering and the number of links. To this end, this chapter presents four main contributions:

1. Formulating the partitioning problem for any NoC application based on the famous Fiduccia-Mattheyses (FM) algorithm [15],

2. Analyzing the impact of network partitioning on different NoC metrics; namely, power consumption, area, and delay,

3. Presenting a methodology for NoC semi-custom architecture generation using network partitioning, and

4. Evaluating the presented methodology by applying it onto different real benchmark applications, as case studies.

4.2 Assumptions of the Generated Architectures

In this section, we present the assumptions of our methodology. According to our discussion in Section 3.8, these assumptions indicate which design variables are traded off by our methodology. For our semi-custom architecture generation methodology, the designer needs to select a standard architecture and a mapping technique for each partition. However, for the rest of the design variables, we consider the following assumptions:

1. The buffer size for all the ports within routers are the same (8-flit buffers are assumed in this chapter),

2. Each link consists of two channels for packet transmission and reception, respectively. The channel width of all the links in the network is the same and is equal to the flit size (8-bit flits are assumed in this chapter),

3. The application, with its traffic characteristics, is represented at any time by a core graph,

4. Deterministic shortest path routing is used, and

5. The length of all links is the same and allows for a single clock cycle data transfer, i.e., no repeaters are required. This assumption might not hold in the actual circuit-level floorplanning. However, it is still valid for the purpose of system-level evaluation and comparison.

4.3 Partitioning Problem Formulation

The problem of network partitioning is known to be an NP-hard one [101]. Therefore, partitioning is usually formulated as an optimization problem. In this chapter, the FM algorithm [15] is adopted with modification to formulate our partitioning technique. Therefore, it is given the name *ASNoC-FM*. The advantages of using the FM algorithm with ASNoC is discussed in details in Subsection 4.4.3. Nevertheless, the mathematical formulation of the partitioning scheme is presented in this section. Using ASNoC notations, the objective function, to be minimized, is first formulated in Subsection 4.3.1 according to the FM algorithm. Two constraints are then added to the algorithm in Subsection 4.3.2 to match the ASNoC design environment.

4.3.1 Objective Function Formulation

In order to formulate the partitioning problem mathematically, assume that there are N cores in the application and it is required to divide them into N_P partitions, $P_1 \cdots P_m$. The number of cores within any partition, P_k, is N_k. Accordingly, the total ASNoC traffic (λ_{total}), the total traffic (λ_k) within any partition P_k, and the total traffic sent from this partition to all other partitions ($\bar{\lambda}_k$), in *packets/time step*, could, respectively be expressed as

$$\lambda_{total} = \sum_{i=1}^{N} \sum_{j=1}^{N} \lambda_{ij} \tag{4.1}$$

$$\lambda_k = \sum_i \sum_j \lambda_{ij}, \quad \forall \{i,j\} \text{ where } c_i,\ c_j \in P_k \tag{4.2}$$

$$\bar{\lambda}_k = \sum_i \sum_j \lambda_{ij}, \quad \forall \{i,j\} \text{ where } c_i \in P_k,\ c_j \notin P_k \tag{4.3}$$

The total traffic (λ_{total}) depends on the application itself and will not be affected by partitioning. In contrast, the intra-partition traffic (λ_k) and the inter-partition traffic ($\bar{\lambda}_k$) change according to the employed partitioning scheme. As explained in Section 2.3, the FM algorithm aims at minimizing the total traffic between all partitions. As a result, the total inter-partition traffic (λ_{inter}), in *packets/time step*, to be minimized as an objective function is expressed as

$$\lambda_{inter} = \sum_{k=1}^{N_P} \bar{\lambda}_k \tag{4.4}$$

4.3.2 Constraints Formulation

The FM optimization algorithm is modified in this subsection by adding some constraints to make it more suitable for the on-chip environment. The first constraint to be added is to divide the total ASNoC traffic equally between partitions, i.e., $\lambda_1 = \cdots = \lambda_m = \lambda_{total}/N_P$. This traffic-balance constraint ensures equal activity in each partition and avoids creating hot spots in the design. As the traffic for all partitions cannot exactly be the same, a more realistic unbalance traffic factors, $u_{\lambda_1} \cdots u_{\lambda_{N_P}}$, are used. For any partition (P_k), its traffic factor (u_{λ_k}) is expressed as a percentage of the total traffic (λ_{total}). The choice of these traffic factors gives the designer the flexibility to allow for some traffic variations between partitions. Although these factors are application-dependent and no specific values could be proposed for them, they should be carefully chosen. Very large factors produce

completely unbalanced partitions, whereas no result might be obtained with zero or very small factors. Using these traffic factors, the first constraint is expressed as

$$\lambda_k = \frac{\lambda_{total}}{N_P} \pm u_{\lambda_k} \cdot \lambda_{total}, \quad \forall \ 1 \leq k \leq N_P \tag{4.5}$$

The second constraint to be added to the FM optimization algorithm is to divide the cores equally between partitions, i.e., $N_1 = \cdots = N_{N_P} = N/N_P$. This core-balance constraint is to facilitate the floorplanning. Similar to our first constraint, unbalance core factors, $u_{N_1} \cdots u_{N_{N_P}}$, are used to allow for some variation between the number of cores within partitions. For any partition (P_k), its core factor (u_{N_k}) represents the number of cores above or below the average number of cores for all partitions. Using these core factors, the second constraint is expressed as

$$N_k = \left\lfloor \frac{N}{N_P} \right\rfloor \pm u_{N_k}, \quad \forall \ 1 \leq k \leq N_P \tag{4.6}$$

To this end, our ASNoC-FM optimization problem is formulated as

Minimize :

$$\lambda_{inter} = \sum_{k=1}^{N_P} \bar{\lambda}_k \tag{4.7}$$

Subject to :

$$\lambda_k - (\frac{1}{N_P} \pm u_{\lambda_k}) \cdot \lambda_{total} = 0, \qquad \forall \ 1 \leq k \leq N_P \tag{4.8}$$

and

$$N_k - \left\lfloor \frac{N}{N_P} \right\rfloor \pm u_{N_k} = 0, \qquad \forall \ 1 \leq k \leq N_P \tag{4.9}$$

where N is the total number of cores in the application, N_k is the number of cores in a specific partition (P_k), and N_P is the number of partitions. u_{λ_k} and u_{N_k} are the traffic and core unbalance factors, respectively. λ_k and $\bar{\lambda}_k$ are the intra-partition and the

inter-partition traffic for partition (P_k), as expressed in (4.2) and (4.3), respectively. Finally, λ_{total} and λ_{inter} are the total and the total inter-partition traffic, as expressed in (4.1) and (4.4), respectively.

Our ASNoC-FM algorithm starts with an arbitrary division of cores between partitions. Cores within any partition are then ranked according to the amount of communication with other partitions. The core with the highest rank is then moved to the partition with which it communicates the most. The move is only confirmed and carried out if it reduces the inter-partition traffic according to (4.7) and meets the constraints in (4.8) and (4.9). The ranking process is then repeated and the algorithm continues with all other cores. ASNoC-FM finally produces the cores within each partition corresponding to the minimum inter-partition traffic.

4.4 ASNoC Partitioning: Power, Area, and Delay Analysis

Applying network partitioning techniques on ASNoC affects different design variables, like the number of ports per router, the number of links, and the average internode distance. As a result, using network partitioning changes the values of ASNoC cost and performance metrics. In this section, we analyze the impact of applying network partitioning techniques on different ASNoC metrics. In general, network partitioning always reduces the area, often reduces the power, sometimes reduces the delay, and, unfortunately, always reduces the reliability[1]. The analysis in this section considers only those metrics that are enhanced, or could be enhanced, by applying network partitioning. Therefore, Subsections 4.4.1, 4.4.2, and 4.4.3 discuss the enhancement, or the deterioration, that results from using network partitioning techniques on ASNoC power consumption, area, and delay, respectively.

[1]Addressing the reliability degradation resulted from using network partitioning is left for the future work.

4.4.1 Power Analysis

The number of ports within a router constitutes the major factor of the router's power consumption. As previously shown in Table 3.1, the power of our output queuing router increases significantly as the number of ports increases. For any flit arrival rate, doubling the number of ports results in quadrupling the total router power consumption. Moreover, routers with different configurations were studied in [150, 165] and their power consumptions were proved to be dominated by the number of ports as well. Therefore, power-wise, it is of great importance for any NoC architecture to be generated with routers that have low number of ports.

Network partitioning results in reducing the number of ports for the routers employed to generate the on-chip network, which in turn reduces the power consumption of the resultant NoC architecture. Furthermore, network partitioning reduces the link power by reducing the total number of links within the generated architecture. However, network partitioning increases the average internode distance taking some traffic through longer routes. For any traffic trace, taking more hops results in increasing the power consumption. Therefore, the partitioning outcomes with respect to power consumption are indeed conflicting. Therefore, it is dependent on the application whether partitioning results in reducing the overall power consumption or not. By experimentation with many ASNoC benchmarks, as will be explained in Subsection 4.6.1, the advantages of reducing the number of ports and links often outperform the deterioration happened by increasing the average internode distance. However, to avoid doing time-consuming routers power simulation and application traffic routing and to mathematically decide on the system-level whether partitioning will be power beneficial or not, we present a partitioning power factor (η_P). Based on the power equations in Chapter 3, the power factor is an approximation of the ratio of the post-partitioning power consumption to the pre-partitioning one. The proposed

power factor is proved by experimentation, as will be explained in Subsection 4.6.5, to be a good approximation whether using partitioning results in reducing the overall NoC power or not. The proposed partitioning power factor is represented as

$$\eta_P = \frac{P_{partitioned}}{P_{unpartitioned}} \approx \frac{p_{avg_p} \cdot \mu_p + p_{avg_p}}{p_{avg_n} \cdot \mu_n + p_{avg_n}} \tag{4.10}$$

where $P_{partitioned}$ and $P_{unpartitioned}$ are the total ASNoC power consumptions with and without partitioning, respectively, according to (3.10). p_{avg_p} and μ_p are the average number of ports per router and the average internode distance when partitioning is carried out. Similarly, p_{avg_n} and μ_n are the average number of ports per router and the average internode distance when partitioning is not carried out and all the nodes in the application are realized with one large non-partitioned network. Consequently, if the value of the partitioning power factor is less than 1, using network partitioning is useful with respect to the power metric. In other words, the power saving by removing ports and links using partitioning outperform the power rise by increasing the average internode distance. In contrast, if η_P is greater than 1, the average internode distance is more dominant for this application and partitioning will not result in any power saving. Finally, the power factor could be used to approximate the amount of power saving. Accordingly, the percentage power saving by using partitioning is represented as

$$\%\ Power\ Savings = 100 \cdot (1 - \eta_P) \tag{4.11}$$

4.4.2 Area Analysis

The ever-increasing on-chip communication requirements for modern applications force ASNoCs to replace ordinary shared buses. Nevertheless, ASNoC area is not supposed to exceed 10% of the whole design area [3]. The area of on-chip networks is

mainly determined by routers area [134]. Moreover, from the implementation results of our output queuing routers, we found that ports with their buffers constitute the major percentage of any router area. Similarly, the areas of routers with different configurations were shown in [134] to be directly proportional to the number of ports as well. Therefore, the less the number of ports for routers used in generating the architecture, the lower the ASNoC area.

Network partitioning divides a large network into smaller partitions connected together. Smaller partitions require routers with less number of ports than those used in a single unpartitioned network. As a result, partitioning reduces ASNoC router area. Moreover, using network partitioning also removes some of the links and reduces ASNoC link area. Therefore, using network partitioning techniques in generating ASNoC architecture guarantees a certain ratio of area reduction.

As an example on the area reduction resulting from using network partitioning, Figure 4.1 shows a mesh implementation of a 16-node application before and after partitioning. The two partitions are separated by the dotted line in Figure 4.1(b). (The partitioning strategy is discussed in details in Section 4.3.) On one hand, partitioning results in removing three links in this example. On the other hand, Table 4.1 summarizes the total number of routers needed with different number of ports before and after partitioning and the total ASNoC area. As a conclusion, using network partitioning reduces the ASNoC area by 9.16% in this example.

Table 4.1: Number of routers and total area before and after partitioning of a 16-node application, mapped onto mesh architecture, as shown in Figure 4.1.

	5-port Router	**4-port Router**	**3-port Router**	**Total area (μm^2)**
Before partitioning	4	8	4	1,634,400
After partitioning	2	6	8	1,497,300

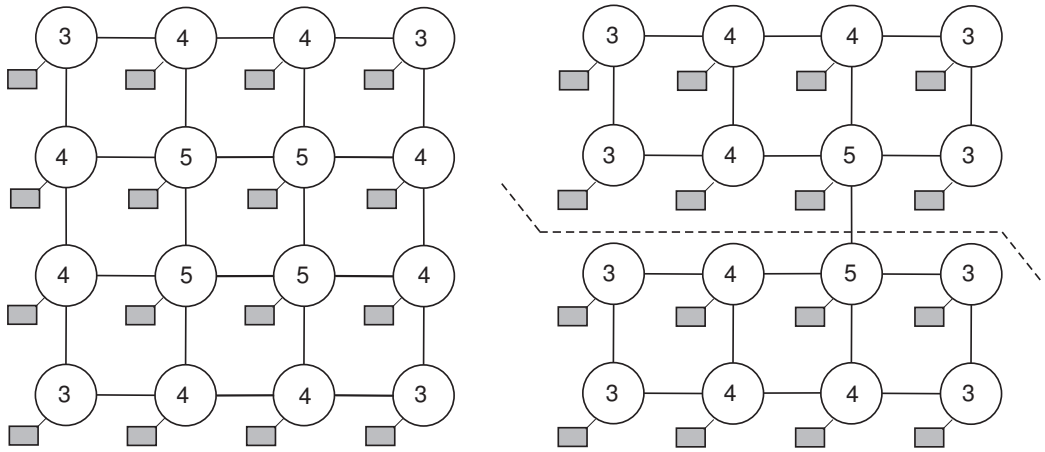


Figure 4.1: Mesh implementation of a 16-core application before and after partitioning.

As the number of nodes becomes larger, the number of links removed by partitioning increases. By removing a link, the number of ports of the two routers that were connected by this link decreases by one. Consequently, the router area decreases in a nearly linear proportion with the number of ports removed. Therefore, the more cores existing in the application, the more links removed by partitioning, and the more routers area saved. As a result, partitioning becomes more useful in reducing the area cost as the number of cores in the application increases.

4.4.3 Delay Analysis

The effect of using network partitioning on ASNoC delay represents a similar trade-off like that of the power consumption. On one hand, partitioning reduces the routers arbitration and propagation delays by reducing the number of ports. On the other hand, the traffic from some nodes, other than those on the boundary, in a specific

partition needs more hops to reach the destination nodes in other partitions. This rise in the average internode distance increases the ASNoC delay. Accordingly, it is first required to formulate the partitioning problem to minimize the traffic exchanged between different partitions. This clarifies the reason of our selection of the FM algorithm to build our partitioning methodology on it. This partitioning technique is mainly used to minimize the inter-partition traffic. The advantages of using the FM algorithm with ASNoC are threefold. First, it mitigates the effect of partitioning on the average internode distance. Second, it avoids creating bottlenecks in the ASNoC through inter-partition links. Third, it reduces the buffering required at those nodes located at the boundary of partitions.

The trade-off between conflicting design variables makes the use of partitioning an application-dependent with respect to the delay metric. Therefore, we propose our partitioning delay factor (η_D) to approximately quantify the advantage, if any, of using partitioning on ASNoC delay. Similar to the power factor, the partitioning delay factor is based on the delay equations in Chapter 3 and decides quickly at the system-level whether the use of partitioning with a specific application will be useful with respect to the delay or not. It is proved by experimentation, as will be represented in Subsection 4.6.5, to be a good approximation of the ratio between post-partitioning and pre-partitioning delays. Our partitioning delay factor is represented as

$$\eta_D = \frac{D_{partitioned}}{D_{unpartitioned}} \approx \frac{\mu_p \cdot (p_{avg_p} + log_2 p_{avg_p})}{\mu_n \cdot (p_{avg_n} + log_2 p_{avg_n})} \tag{4.12}$$

where $D_{partitioned}$ and $D_{unpartitioned}$ are the overall ASNoC delay with and without partitioning according to (3.14). p_{avg_p} and μ_p are the average number of ports per router and the average internode distance when partitioning is carried out. Similarly, p_{avg_n} and μ_n are the average number of ports per router and the average internode distance when partitioning is not carried out and all the nodes in the application

are realized with one large non-partitioned network. If the value of the partitioning delay factor is less than 1, using network partitioning is delay useful. In other words, the delay reduction by lowering the arbitration and propagation delays resulted from reducing the number of ports per router outperforms the delay rise by increasing the average internode distance. In contrast, if η_D is greater than 1, the average internode distance is more dominant for this application and partitioning will not result in any delay enhancement. Finally, the delay factor could be used to approximate the amount of delay reduction. The percentage delay reduction by using partitioning is represented as

$$\%\ Delay\ Reduction = 100 \cdot (1 - \eta_D) \tag{4.13}$$

4.5 Methodology for Semi-Custom Architecture Generation

The proposed cost-efficient methodology for ASNoC semi-custom architecture generation using network partitioning is shown in Figure 4.2. The methodology consists of seven main steps. In the first step, the supply voltage (V_{dd}), frequency of operation (f_{op}), the application core graph, and the targeted fabrication technology are entered to the methodology. Moreover, the required number of partitions, the standard architecture for each partition, and the mapping technique should also be provided in the first step[2]. In the second step, the core graph is divided into smaller partitions according to our ASNoC-FM algorithm. Other partitioning schemes could also be used without loss of generality. Public partitioning software packages, like Chaco [106] and PARMETIS [109] could be employed in this step. Simultaneously, in the second step, an unpartitioned standard architecture is constructed. In the third step, all application cores are mapped onto that unpartitioned architecture. In

[2]The optimum selection of the number of partitions is left for the future work.

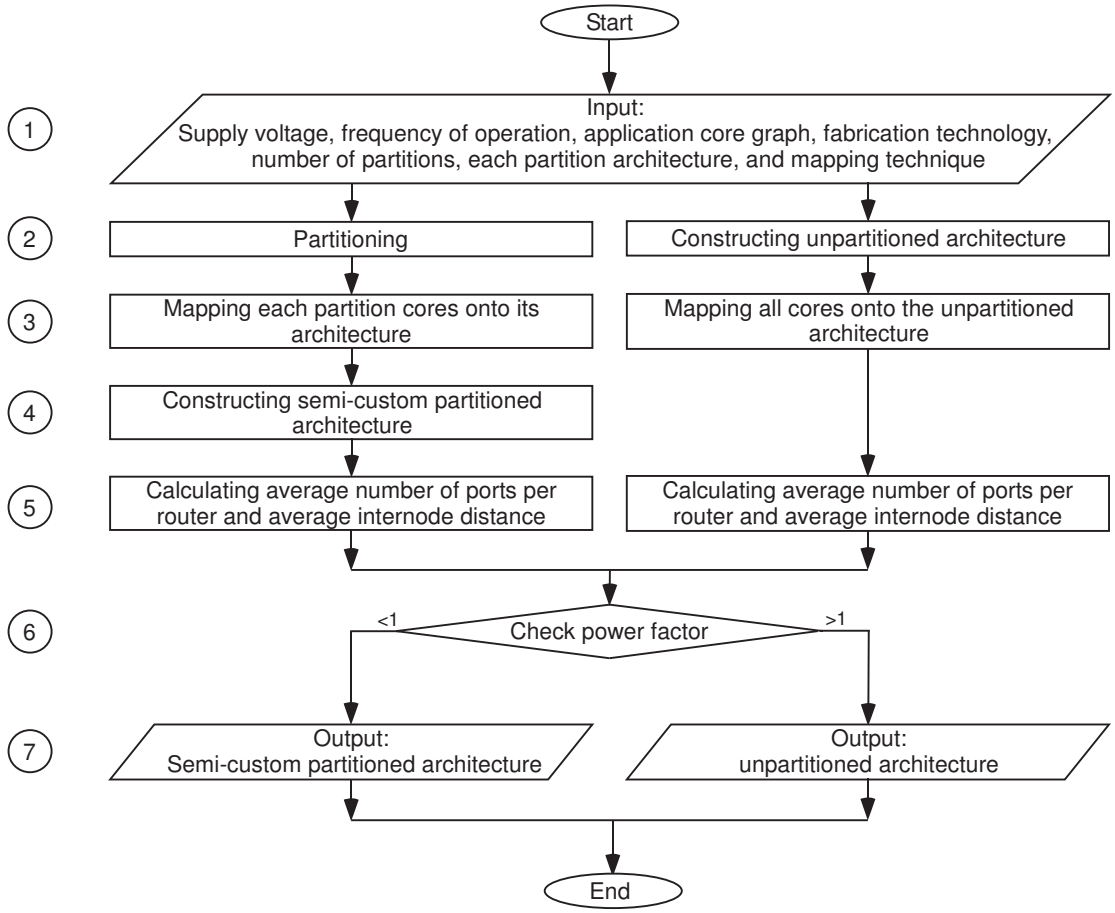


Figure 4.2: Proposed cost-efficient methodology for semi-custom architecture generation using network partitioning.

parallel, cores in each partition are also mapped onto the architecture corresponding to that partition. Similar to the partitioning step, any mapping technique could be used with our methodology. Nevertheless, NMAP is proved to outperform PMAP, GMAP, and PBB in [133]. Moreover, NMAP is a general technique that could map cores onto any architecture. In the fourth step, partitions are connected in a one-by-one fashion to construct the whole semi-custom architecture for the application. To establish the connection between any two partitions, cores in each partition are ranked according to the amount of traffic exchanged with cores in the other partition. The

router connected to the core with the highest rank is selected as the connecting node. Similar to the previous two steps, other connection schemes maybe used to construct a network of partitions. In the fifth step, the average number of ports per routers and the average internode distance are calculated for the partitioned and unpartitioned architectures. In the sixth step, the partitioning power factor is calculated according to (4.10) to check for the usefulness of using network partitioning with respect to the power of the application in-hand. As our methodology is proposed as a cost-efficient one, we only consider the power factor. However, the delay factor could also be included according to the design requirements and application needs. If the power factor is less than 1, the partitioning is useful for the application and the partitioned architecture is outputted in the seventh step. Nevertheless, if the factor is greater than 1, partitioning only enhances the area not the power of that application and the single unpartitioned architecture is outputted.

4.6 Experimental Results

The proposed methodology is applied to four of the real benchmark applications shown in Figure 3.1: VOPD benchmark with 16 cores, 263DEC benchmark with 14 cores, MPEG-4 benchmark with 12 cores, and MWD benchmark with 9 cores. As the number of cores in these benchmarks is not large, the number of partitions is chosen to be two. NMAP is employed for the mapping and mesh architecture is used for all partitions. The reason behind mesh selection is twofold. First, grid-based architectures, like mesh, could be easily implemented inside chips. Second, most of practical NoC implementations are achieved with the mesh architecture [6].

4.6.1 Power Evaluation

In this subsection, we evaluate the efficiency of our methodology with respect to the power metric. Therefore, the total power of the architecture resulting from our partitioning-based methodology is compared to that of a standard unpartitioned mesh architecture, mapped also by NMAP. Moreover, the power of the semi-custom architecture generated by the long-range link insertion methodology, proposed in [75] and discussed in Subsection 2.5.2, is also considered for the comparison. The three architectures are represented throughout the comparison by NP, MSH, and LR, respectively. Table 4.2 shows the results for the three architectures in absolute and normalized values with respect to those of our methodology. Using our methodology, ASNoC power is reduced by 9.38%, 1.86%, 6.34%, and 1.9% with respect to standard architecture for the VOPD, 263DEC, MPEG-4, and MWD benchmarks, respectively. Similarly, our architectures are more power efficient than those of the semi-custom long-range methodology by 11.5%, 4.1%, and 6.46% for the VOPD, MPEG-4, and MWD benchmarks, respectively. For the 263DEC benchmark, our methodology results in the same power consumption like the semi-custom long-range one. Although the two customization methodologies end up with the same power consumption, the power reduction mechanism for both are different. Our methodology mainly reduces the router power with a slight increase on the link power, whereas, the semi-custom long-range methodology mainly reduces the link power with a slight increase on the router power. The total power for both is the same, which is 1.86% less than that of a standard mesh architeture. Finally, on average over the four benchmarks, the power of the architecture resulted from our methodology is less than that of the standard mesh architecture by 4.87% and that of the semi-custom long-range architecture by 5.52%. These results show that our partitioning-based methodology often outperforms other standard and custom architecture realization techniques with respect to the power

metric. Moreover, as network partitioning certainly reduces the area cost of any application, it is an efficient way to reduce the two ASNoC cost metrics: power and area.

Table 4.2: Power comparison between different architectures for different benchmark applications. (% is the percentage of the power with respect to that of the NP architecture.)

Benchmark	**Architecture**					
	NP		**MSH**		**LR**	
	Watts	**%**	**Watts**	**%**	**Watts**	**%**
VOPD	0.2600	100.00	0.2844	109.38	0.2899	111.50
263DEC	1.2320	100.00	1.2549	101.86	1.2319	100.00
MPEG-4	0.5219	100.00	0.5550	106.34	0.5433	104.10
MWD	0.1733	100.00	0.1766	101.90	0.1845	106.46
Average		100.00		104.87		105.52

4.6.2 Area Evaluation

In this subsection, we evaluate the area reduction efficiency of our methodology. Therefore, for each benchmark, we calculate the area of the architecture generated by it and compare it to these of the standard mesh architecture and the semi-custom long-range one. These results are shown in Table 4.3. Results are also normalized with respect to those of our methodology. Using our methodology, ASNoC area is reduced by 9.16%, 7.33%, 8.4%, and 12.04% with respect to standard architecture and by 15.26%, 10.99%, 16.79%, and 24.07% with respect to semi-custom long-range architecutre for the VOPD, 263DEC, MPEG-4, and MWD benchmarks, respectively. On average over the four benchmarks, the area resulted from our methodology is less than that of the standard mesh architecture by 9.23% and that of the semi-custom long-range architecture by 16.78%. These results prove that our partitioning-based methodology always outperforms other standard and semi-custom architecture generation techniques with respect to the area metric.

Table 4.3: Area comparison between different architectures for different benchmark applications. (% is the percentage of the area with respect to that of the NP architecture.)

Benchmark	**Architecture**					
	NP		**MSH**		**LR**	
	μm^2	%	μm^2	%	μm^2	%
VOPD	1,497,300	100.00	1,634,400	109.16	1,725,800	115.26
263DEC	1,247,300	100.00	1,338,700	107.33	1,384,400	110.99
MPEG-4	1,088,700	100.00	1,180,100	108.40	1,271,500	116.79
MWD	759,400	100.00	850,800	112.04	942,200	124.07
Average		100.00		109.23		116.78

4.6.3 Delay Evaluation

In this subsection, we evaluate the delay efficiency of our methodology. Therefore, for each benchmark, the delays of our partitioning-based, standard mesh, and semi-custom long-range architectures are calculated and included in Table 4.4. The delay of the architectures generated by our methodology is found to be slightly better than that of standard mesh architectures by 1.88%, 1.97%, and 1.79% for the VOPD, 263DEC, and MPEG-4 benchmarks, respectively. However, for the MWD benchmark, the standard mesh architecture is more delay efficient than that generated by our methodology by 4.53%. As previously explained in Subsection 4.4.3, the delay enhancement resulted from our methodology is an application-dependent. For our methodology to achieve some delay enhancement, the inter-partition traffic should be low with respect to the total application traffic. Keeping the inter-partition traffic low allows the gains in reducing the router arbitration and propagation delays to overcome the deterioration happened by taking the inter-partition traffic through long routes. In the literature, the traffic locality factor is used to express the ratio of the intra-partition traffic to the total application traffic [166]. As the value of this traffic locality factor goes close to 1, the traffic becomes highly localized and the effect

of partitioning on the average internode distance becomes more insignificant. For the MWD benchmark, unlike other benchmarks, this traffic locality factor is found to be 0.784, which means that the inter-partition traffic is 21.6% of the total traffic. This high inter-partition traffic explains why our partitioning-based methodology could not achieve any delay enhancement for the MWD benchmark. In a nutshell, our methodology is expected to enhance the delay only for those applications that exhibit some tarffic locality. Finally, on average over the four benchmarks, the delay resulted from the architecture generated by our methodology is 0.28% less than that of the standard mesh architecture.

Table 4.4: Delay comparison between different architectures for different benchmark applications. (% is the percentage of the delay with respect to that of the NP architecture.)

Benchmark	**Architecture**					
	NP		**MSH**		**LR**	
	μs	%	μs	%	μs	%
VOPD	2.4400	100.00	2.4859	101.88	2.4113	98.82
263DEC	12.1600	100.00	12.3990	101.97	12.2350	100.62
MPEG-4	5.0140	100.00	5.1039	101.79	4.5934	91.61
MWD	1.6830	100.00	1.6067	95.47	1.5347	91.19
Average		100.00		100.28		95.56

The architectures generated by our methodology achieve some delay enhancement over standard mesh architectures for three of the benchmarks. However, they could not outperform those of the semi-custom long-range methodology, except for the 263DEC benchmark and by only 0.62%. The latter methodology is mainly proposed to customize ASNoC architecture with respect to the delay. Therefore, it outperforms our methodology by 1.18%, 8.39%, and 8.81% for the VOPD, MPEG-4, and the MWD benchmarks, respectively. An analysis of the inter-partition traffic and the traffic locality factor of the 263DEC benchmark explains why our methodology outperforms the semi-custom long-range one. For this benchmark, the traffic locality factor is

found to be 0.998, and hence, the inter-partition traffic constitutes only 0.25% of the total traffic. Therefore, the advantages of reducing the arbitration and propagation delays by our methodology dominate the overall delay and allow it to achieve a better delay performance for this benchmark. Finally, on average over the four benchmarks, the semi-custom long-range methodology outperforms ours by 4.44%.

The above delay discussion expresses a typical trade-off between performance and cost. On one hand, our methodology is proposed as a cost-efficient one, and hence, it enhances area and power costs more than the delay performance. On the other hand, the semi-custom long-range methodology is a performance-efficient one, and hence, it enhances the delay performance on the expense of more power and area costs of the on-chip network.

4.6.4 Partitioning Scheme Evaluation

In this subsection, we verify the efficiency of using our partitioning technique, ASNoC-FM, with on-chip networks. Therefore, Tables 4.5 and 4.6 compare the power and the delay resulted from using it, respectively, with those of other partitioning schemes. As all schemes result in approximately the same area reduction, area comparison is not included in this subsection. From on hand, the table shows that ASNoC-FM completely outperforms random and scattered partitioning schemes with respect to both power and delay. On average over the four benchmarks, our technique is 75.36% and 37.09% more power-efficient than the scattered and random techniques, respectively. It is also 68.35% and 34.44% more delay-efficient than the same two techniques, respectively. On the other hand, ASNoC-FM is better than the spectral scheme by 3.56% and 3.79% from the power and the delay prespectives, repsectively. These results emphasize the efficiency of using ASNoC-FM with respect to on-chip network power and delay.

Table 4.5: Power comparison between different partitioning schemes for different benchmark applications. (% is the percentage of the power with respect to that of ASNoC-FM.)

Benchmark	**Partitioning Scheme**							
	ASNoC-FM		**Scattered**		**Random**		**Spectral**	
	Watts	%	**Watts**	%	**Watts**	%	**Watts**	%
VOPD	0.2600	100.00	0.5675	218.27	0.3270	125.77	0.2600	100.00
263DEC	1.2320	100.00	2.5656	208.25	2.1050	170.86	1.3384	108.64
MPEG-4	0.5219	100.00	0.7930	151.94	0.5705	109.31	0.5226	100.13
MWD	0.1733	100.00	0.2131	122.97	0.2468	142.41	0.1828	105.48
Average		100.00		175.36		137.09		103.56

Table 4.6: Delay comparison between different partitioning schemes for different benchmark applications. (% is the percentage of the delay with respect to that of ASNoC-FM.)

Benchmark	**Partitioning Scheme**							
	ASNoC-FM		**Scattered**		**Random**		**Spectral**	
	μs	%	μs	%	μs	%	μs	%
VOPD	2.440	100.00	5.210	213.52	3.154	129.27	2.440	100.00
263DEC	12.160	100.00	24.262	199.52	19.790	162.75	13.011	107.00
MPEG-4	5.014	100.00	6.933	138.27	5.151	102.73	5.073	101.17
MWD	1.683	100.00	2.055	122.09	2.407	143.01	1.800	106.98
Average		100.00		168.35		134.44		103.79

Tables 4.5 and 4.6 also show that the power and the delay resulted from partitioning techniques other than ASNoC-FM are worse than those of the standard unpartitioned mesh architecture, shown in Tables 4.2 and 4.4, respectively. This clarifies that the advantages of using network partitioning with ASNoC is completely dominated by the best selection of the employed partitioning technique. It further emphasizes the efficiency of using ASNoC-FM in enhancing ASNoC power and delay.

4.6.5 Power and Delay Factors Evaluation

In this subsection, we evaluate the accuracy of our power and delay factors represented in (4.10) and (4.12), respectively. These factors are used to approximate the ratios of the partitioned power and delay to the unpartitioned ones, respectively. Therefore, in this subsection, we compare the values of these factors to the exact power and delay ratios. For the four benchmarks, Table 4.7 shows the percentage error of the two factors. The table clarifies that the error resulted from using our approximated factors never exceeds 4%. On average over the four benchmarks, the percentage error of our power factor is 2.13% and of our delay factor is 2.88%. However, the use of these factors saves a large amount of the architecture generation time. This is the time corresponding to simulate the router library, derive the power constants, and route all packets from sources to destinations. Accordingly, as the architecture generation time is a crucial factor for ASNoC, the use of these factors is indeed very helpfull in quickly evaluating the efficiency of using partitioning for ASNoC architecture generation.

Table 4.7: Percentage error of power and delay factors for different benchmark applications.

Factor	**Percentage error (%)**				
	Benchmark				**Average error**
	VOPD	263DEC	MPEG-4	MWD	
Power factor	0.077	3.56	2.08	2.8	2.13
Delay factor	3.93	3.47	3.79	0.32	2.88

4.7 Chapter Summary

This chapter proposed reducing power and area costs of ASNoC by using network partitioning techniques. Smaller partitions require less number of ports and buffering, which constitute the main sources of the on-chip power and area. The FM

partitioning algorithm was adopted with modification to formulate the partitioning problem. The effect of using network partitioning techniques on ASNoC power, area, and delay was analyzed. Moreover, a methodology was presented, based on our partitioning technique, for cost-efficient semi-custom ASNoC architecture generation. The area reduction is guaranteed using our methodology. However, power and delay enhancements are application-dependent. Therefore, we mathematically formulated power and delay factors that allow the designer to quickly decide at the system-level whether to use network partitioning or not. These factors were proved by experimentation to be a good approximation of the exact power and delay ratios between partitioned and unpartitioned architectures. Finally, our methodology was evaluated through different benchmark applications with different number of cores. Results proved that our partitioning-based methodology is an effective way to reduce ASNoC power and area, especially for large applications that exhibit some sort of traffic locality. This work has been published in brief in [16–18] and is submitted for publication in full in [19].

In the next chapter, we present an optimization methodology for NoC full-custom architectures. The methodology uses GA to completely optimize the on-chip network architecture according to the target application requirements. The methodology is also a multi-objective one that considers power, area, delay, and reliability, simultaneously.

Chapter 5

A Multi-objective Optimization Methodology of ASNoC Full-Custom Architectures using GA

This chapter presents the third contribution of this dissertation. We propose a GA-based optimization methodology to generate the best on-chip full-custom architecture for any NoC application. The optimization could be carried out for single or multiple metrics based on weight factors provided by the designer. Unlike previous full-custom architecture generation techniques, the assumptions used for our methodology allow it to generate the architecture in a reasonable time. Our methodology is evaluated by applying it to different NoC benchmark applications, as case studies. Results show that the architectures generated by our methodology outperform those generated by previous techniques with respect to power, area, delay, reliability, and the combination of the four metrics. Moreover, the methodology generates the required full-custom architecture in multiple orders of magnitude faster than previous generation techniques.

This chapter is organized as follows. Section 5.1 gives an introduction about

the multi-objective architecture generation problem addressed in this chapter. The assumptions of our methodology are presented in Section 5.2. Section 5.3 presents our multi-objective GA-based solution for the full-custom architecture generation problem. The section discusses in details our binary chromosome representation of ASNoC full-custom architectures, the legality criteria of any generated architecture, and the GA-based generation methodology. Section 5.4 presents some experimental results for different NoC benchmarks to validate our work. Finally, Section 5.5 summarizes the chapter.

5.1 Introduction

For any NoC-based system, the on-chip network architecture significantly impacts its cost and performance [70]. On the cost side, it is required to minimize the NoC area and power consumption. On the performance side, it is required to minimize the delay and maximize the reliability. However, performance requirements and cost constraints are usually conflicting. For example, increasing the NoC reliability often requires using more resources, which increases the NoC area and power consumption. Accordingly, modern NoC research proposes different methodologies to fully optimize on-chip network architectures according to the application requirements. However, to the best of our knowledge, the work done so far customizes the NoC architecture for a single metric, which is the most important for the application, assuming other metrics as constraints [75, 80, 152, 154]. Unfortunately, optimizing for a single metric usually affects the performance with respect to other metrics. Therefore, one challenging problem is to optimize on-chip networks for more than one metric. That is to decide which NoC architecture results in the maximum possible performance and the minimum possible cost, simultaneously.

In this chapter, we present an architecture generation methodology to solve the

above mentioned multi-objective optimization problem. More precisely, we employ GA optimization to achieve the best full-custom architecture for any application with respect to power, area, delay, and reliability, simultaneously. To achieve this architecture, our methodology trades off many conflicting NoC design variables. The most important of these variables are the number of ports per router, the number of links, the mapping, the traffic traces routing, the average internode distance, the flit arrival rate, and the probability of the traffic being affected by on-chip noise sources. To this end, this chapter presents three main contributions:

1. Proposing a technique of representing any ASNoC full-custom architecture as a GA binary chromosome,

2. Presenting a methodology for ASNoC full-custom architecture generation based on the proposed GA chromosome representation, and

3. Evaluating the presented methodology by applying it onto different real benchmark applications, as case studies.

5.2 Assumptions of the Generated Architectures

In this section, we present the assumptions of our methodology. The running time is a crucial parameter for any optimization technique. Therefore, as previously mentioned in Section 3.8, our methodology does not consider the trade-off between all the design variables. Nevertheless, some of these variables are considered fixed to allow the required ASNoC architecture to be generated as quickly as possible. Moreover, these assumptions enable us to use more time-efficient data structures in our optimization methodology, like matrices. Consequently, to ensure a low running time of our methodology, we consider the following assumptions:

1. The generated architecture is an indirect network [6] for which each node is connected to only one router and each router has a maximum of one node connected to it, i.e., each core has its own router ($N = N_R$),

2. The buffer size for all the ports within routers are the same (8-flit buffers are assumed in this chapter),

3. Each link consists of two channels for packet transmission and reception, respectively. The channel width of all the links in the network is the same and is equal to the flit size (8-bit flits are assumed in this chapter),

4. The application, with its traffic characteristics, is represented at any time by a core graph, and

5. The lengths of all links are the same and allow for a single clock cycle data transfer, i.e., no repeaters are required. This assumption might not hold in the actual circuit-level floorplanning. However, it is still valid for the purpose of system-level evaluation and comparison.

5.3 Full-Custom Architecture Generation using GA

This section discusses in details our multi-objective GA-based solution for the full-custom architecture generation problem. Accordingly, Subsection 5.3.1 explains our GA chromosome representation for any ASNoC full-custom architecture. Thereafter, Subsection 5.3.2 discusses the legality criteria of the generated architectures by the GA engine. Finally, Subsection 5.3.3 presents our GA-based methodology for ASNoC full-custom architecture generation. The inputs to our methodology are the supply voltage (V_{dd}), frequency of operation (f_{op}), the application core graph, the targeted fabrication technology, the optimization weight factors (γ_P, γ_A, γ_D, and γ_R), and

the data resulted from analyzing the employed router library. Before running the GA optimization, every router in the library is analyzed to find its area (A_{Ri}), its power constants (k_{Dp}, k_D, k_{Lp}, and k_L), and its maximum allowable Flit Arrival Rate (FAR_{max}). FAR_{max} is used to avoid any flit loss and to ensure that the generated architecture does not violate the maximum allowable bandwidth for routers. Finally, all routers in the library are checked to constitute the maximum allowable number of ports per router (p_{max}). p_{max} is used to avoid generating an architecture with routers that are not available in the designer library.

5.3.1 Binary Chromosome Representation of ASNoC Architectures

As explained in Section 2.4, applying GA requires representing different NoC architectures in the form of chromosomes. In this chapter, our GA-based solution represents these architectures as binary vectors. As shown in Figure 5.1, a vector is a direct representation of the adjacency matrix (A) of that architecture. As we assume bidirectional links (i.e., $A_{ij} = A_{ji}$), it is actually sufficient to use either the upper or the lower triangle of the matrix. Therefore, the size of each vector is $N \cdot (N-1)/2$, where N is the number of cores within the application.

The Matlab® GA toolbox [167] is used in solving the optimization problem in this chapter. Different selection, crossover, and mutation functions could be used with our methodology. However, for the results presented in this chapter, the toolbox is set to the following configurations:

- Creation function (Uniform): The initial population is created randomly with a uniform distribution.
- Selection function (Stochastic uniform): Every individual has the same chance of being selected. Although this selection function ignores the individual fitness, it has been chosen because of its suitability for small population size [168].

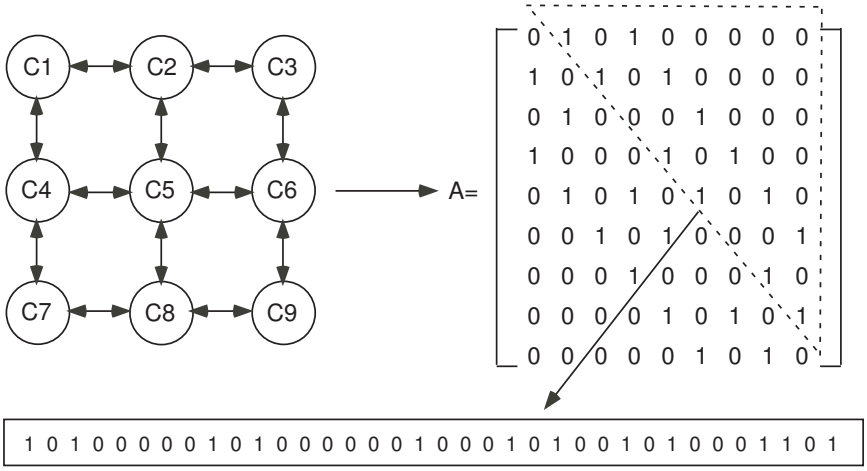


Figure 5.1: Example of a binary chromosome representation for 3×3 mesh architecture.

- Crossover function (Scattered): Offspring genes are chosen randomly from the two parents. Scattered crossover mixes the parents genes completely and allows for evolving good individuals [169].

- Mutation function (Uniform): This is the only built-in function for binary chromosomes in Matlab®. Parents from the current generation are selected randomly and some of their genes are flipped. The number of genes to be flipped and their locations are chosen randomly as well.

5.3.2 Legality Criteria for Generated Architectures

Architectures are generated randomly by the GA engine. Therefore, in this subsection, we present the legality criteria for any generated architecture. Any architecture that fails to meet certain legality constraints is considered invalid. In this chapter, a penalization technique is employed to prevent the survival of invalid architectures from one generation to the next. Accordingly, the optimization objective function (f), as represented in (3.23), is multiplied by an architecture legality factor

(β_l) to represent the overall fitness function of our GA-based solution. This factor is 1 for any valid architecture, whereas a high legality factor is used for invalid architectures to increase the overall values of their fitness functions. The exact value of this legality factor should not be a problem as long as it is high enough to ensure that the overall value of the fitness function of an invalid architecture is greater than that of any valid one. This requires the designer to estimate the worst case objective function value of valid architectures. If this value could not be estimated, a legality factor of more than 100 is proved, by experimentation, to be sufficient enough. Finally, the fitness function (F) of our GA-based methodology is expressed as

$$F = f \cdot \beta_l \tag{5.1}$$

This fitness function (F) is used to evaluate different architectures. Any architecture is considered valid ($\beta_l = 1$) only if it meets the following constraints:

1. **Number of Ports Per Router (PPR) constraint**: This ensures that the generated architecture does not include a router with a number of ports greater than p_{max}, obtained from the router library analysis. Every row i in the adjacency matrix (A) expresses the connection of the corresponding router i to other routers in the architecture. Therefore, this constraint is formulated mathematically as

$$(\sum_{j=1}^{N} A_{ij}) + 1 \leq p_{max}, \quad \forall(i = 1 \; to \; N) \tag{5.2}$$

2. **Traffic Continuity (TC) constraint**: This ensures that there is a path in the generated architecture between any two cores communicating with each other in the core graph. Accordingly, for any traffic (λ_{ij}), in the traffic distribution

matrix, that is greater than zero, there should be a corresponding finite value (c_{ij}) in the connectivity matrix, i.e.,

$$\forall(\lambda_{ij} > 0) \;\Rightarrow\; c_{ij} = finite \tag{5.3}$$

3. **Flit Arrival Rate (FAR) constraint**: This ensures that the flit arrival rate for any router in the generated architecture is less than its corresponding FAR_{max}, which is obtained from the router library analysis. Mathematically, this constraint is formulated as

$$\alpha_{fi_{max}} \leq FAR_{i_{max}}, \quad \forall(i = 1 \; to \; N) \tag{5.4}$$

where $\alpha_{fi_{max}}$ is the maximum flit arrival rate over all the ports of router i. Different flit arrival rates are calculated in our GA-based methodology by routing all the traffic through the generated architecture. As such, our methodology considers possible shortest path routes to check the architecture with respect to this FAR constraint.

5.3.3 Methodology for Full-Custom Architecture Generation

Figure 5.2 shows our methodology for full-custom architecture generation using GA. Before running the GA engine, area and power analysis is carried out on the employed router library to generate the required data for the GA optimization engine. This analysis results in different routers area, power constants, and maximum possible flit arrival rates. It also indicates the maximum usable number of ports per router. The GA optimization starts, in the first step, by reading the inputs required by the engine. Thereafter, the GA engine runs iteratively to find the best architecture with respect to power, area, delay, and reliability. Each generation consists of 10

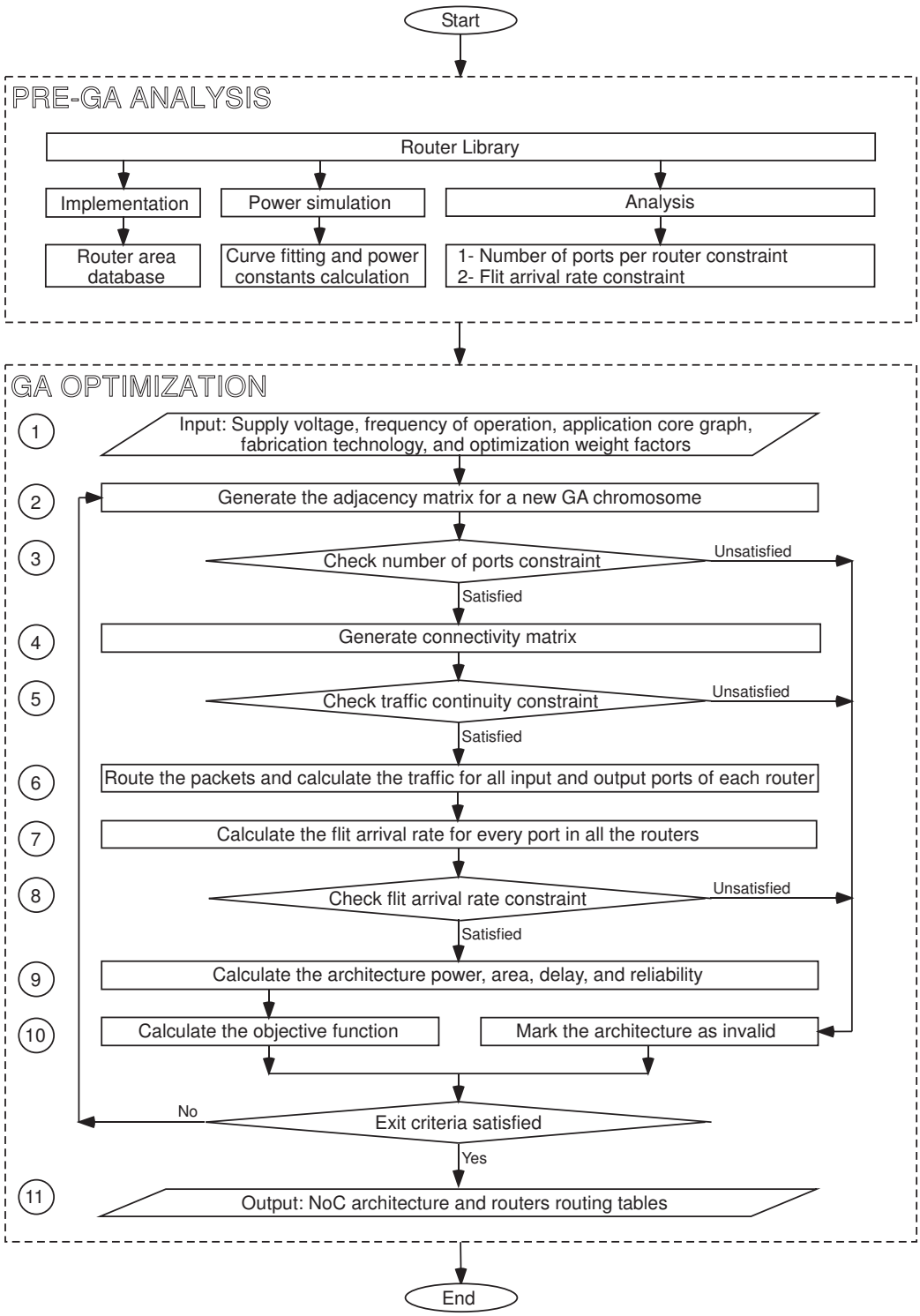


Figure 5.2: Proposed methodology for full-custom architecture generation using GA.

different chromosomes representing 10 different architectures. Every chromosome in any generation is evaluated by steps 2-10 in the flowchart. In the second step, the corresponding adjacency matrix of the chromosome is generated. The PPR constraint is then checked in the third step according to (5.2). The connectivity matrix for any architecture that passes the PPR constraint is generated in the fourth step by the Dijkstra's algorithm [162]. In the fifth step, the TC constraint is checked according to (5.3). Consequently, packets are routed from all sources to all destinations in the sixth step. Moreover, all input and output ports traffic is calculated for every router in the architectures. Based on this traffic, the flit arrival rate for all the ports is calculated in the seventh step. Consequently, the FAR constraint is checked in the eighth step according to (5.4). In the ninth step, the power, area, delay, and reliability of the architecture are calculated according to (3.10), (3.13), (3.14), and (3.22), respectively. These values are then used in the tenth step to calculate the GA fitness function according to (5.1). Any architecture that fails any of the constraints is marked invalid to prevent it from surviving to the next generation. In the eleventh step, once the exit criterion is satisfied, the GA engine generates the best architecture. As this architecture is not guaranteed to be free of loops[1], the routing tables for all the routers in the architecture are generated to avoid any deadlock. Finally, if the exist criterion is not satisfied and a new generation is to be produced, the two architectures with the best fitness function are allowed to survive. Additionally, four new architectures are generated by crossover. Parents are selected randomly from the current generation and crossovered according to the employed Matlab® configurations, as mentioned in Subsection 5.3.1. Finally, the four remaining architectures in each generation are produced by random mutation using the mutation function mentioned in Subsection 5.3.1.

Different values for population, elitism, crossover, and mutation could be used

[1]Proposing a routing algorithm and adding deadlock-free constraints are left as future work.

with our methodology. However, it is not recommended to use a high population size. The convergence time is a key parameter for any optimization technique. High population sizes were found to increase the required running time of the optimizer significantly. For example, we tried using a population size of 100 chromosomes, which, in average, results in triple the running time. This increase in the running time is caused by two main reasons. First, the running time for the GA depends on the population size and the calculations required for every architecture within a generation. On one hand, using more architectures in each generation allows for reaching the optimum quickly. On the other hand, every architecture requires time-consuming calculations of its power, area, delay, reliability, and overall objective function. The GA running time is found to be more proportional to these calculations than to the number of architectures within a generation. In other words, the speedup in the convergence time by using large population size is shadowed by the calculation overhead for the architectures within this population. Second, for an N-core application, it allows for up to $2^{N\cdot(N-1)/2}$ different architectures. Unfortunately, most of these architectures are disconnected, and hence, they violate the TC constraint and are invalid. Consequently, increasing the population size causes more invalid architectures to be introduced every generation. This, in turn, causes the GA to lose a significant amount of time calculating for invalid architectures rather than converging to the optimum one.

5.4 Experimental Results

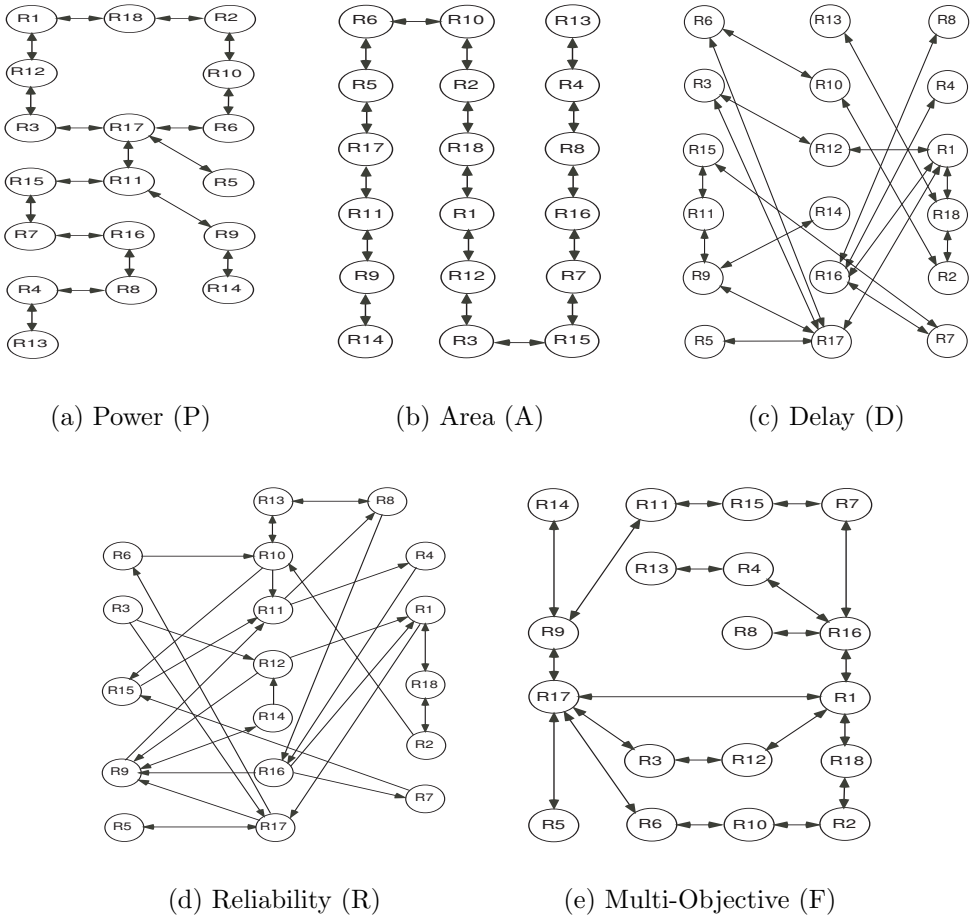
We apply our GA-based methodology on four of the real benchmark applications shown in Figure 3.1: AV benchmark with 18 cores, VOPD benchmark with 16 cores, MPEG-4 benchmark with 12 cores, and MWD benchmark with 9 cores. For every benchmark, we use our GA-based methodology to generate five different architectures.

These are the P architecture that is optimized solely for power ($\gamma_P = 1, \gamma_A = \gamma_D = \gamma_R = 0$), the A architecture that is optimized solely for area ($\gamma_A = 1, \gamma_P = \gamma_D = \gamma_R = 0$), the D architecture that is optimized solely for delay ($\gamma_D = 1, \gamma_A = \gamma_P = \gamma_R = 0$), the R architecture that is optimized solely for reliability ($\gamma_R = 1, \gamma_A = \gamma_P = \gamma_D = 0$), and the F architecture that is optimized for the overall objective function ($\gamma_P = \gamma_A = \gamma_D = \gamma_R = 1$). To better explain our findings, the following subsection discusses how our methodology compromises between different design variables to generate the best full-custom architecture. Thereafter, Subsection 5.4.2 presents a quantitative analysis of the generated architectures with respect to different metrics considered in this study. Finally, Subsection 5.4.3 evaluates the generation time of our GA-based optimization methodology.

5.4.1 Generated Architectures Evaluation: Design Variables Perspective

In this subsection, we discuss the efficiency of our methodology in compromising between different conflicting design variables. Therefore, for each benchmark, we analyze its corresponding five generated architectures with respect to different design variables discussed in Section 3.8. Consequently, this analysis shows how our methodology manages to efficiently trade off different design variables related to each of the five metrics: power, area, delay, reliability, and multi-objective function. For example, Figure 5.3 shows the five generated architectures for the AV benchmark. The core graph of the benchmark is shown in Figure 3.1(a). By analyzing the trade-off in our experiments for the four benchmarks, we notice

1. For the P architecture, router power is proportional to the number of ports and the flit arrival rate. Link power is proportional to the number of links and the number of hops corresponding to each traffic trace. Accordingly, the GA needs to keep the number of ports for each router as minimum as possible. Moreover,



(a) Power (P) (b) Area (A) (c) Delay (D)

(d) Reliability (R) (e) Multi-Objective (F)

Figure 5.3: Optimized architectures for (a) Power, (b) Area, (c) Delay, (d) Reliability, and (e) Multi-objective for the AV benchmark.

routing the traffic traces through the generated architecture should minimize the flit arrival rate. Finally, traffic paths should be kept as short as possible to minimize the link power. These three goals are conflicting. For example, minimizing the number of ports results in higher flit arrival rate and longer traffic paths. However, our GA methodology achieves the best trade-off between these goals and generates the architecture with the minimum possible power. This is realized by optimally minimizing the number of ports, the number of

links, and by routing low traffic traces over longer paths of routers with low traffic.

2. For the A architecture, we tried with both 90nm and 180nm technologies. For both of them, routers area dominates links area. Moreover, router area is directly proportional to the number of ports. Accordingly, the GA tries to minimize the number of ports per each router. Many architectures result in the same minimum area. The one in Figure 5.3(b) represents the minimum-delay mapping of the cores onto what we might call a "line" architecture.

3. For the D architecture, there is a similar compromise to that for the P architecture. On one side, it is required to minimize the number of ports to minimize the arbitration delay. On the other side, traffic traces need more ports to go through the shortest paths to minimize the links and routers propagation delays. GA again achieves the best compromise between these conflicting goals. Interestingly, traces with low traffic are optimally routed over longer paths and through low traffic routers to minimize the overall NoC delay.

4. For the R architecture, maximizing the reliability requires that all traffic traces are routed through the shortest paths to minimize the possibility of being affected by the on-chip noise. Consequently, the generated architecture is just a replica of the application core graph.

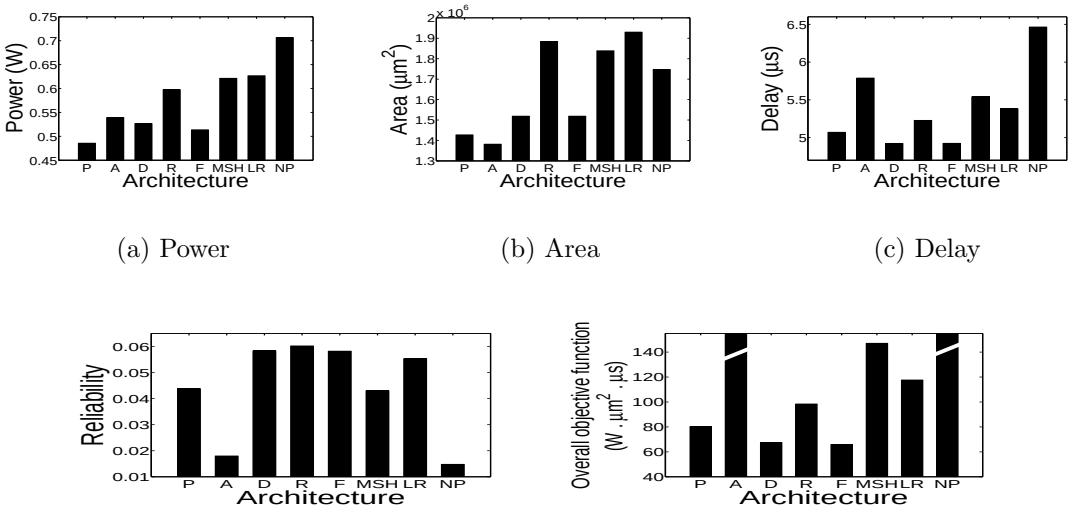
5. For the F architecture, our GA methodology trades off all cost and performance metrics with their associated design variables to generate a multi-objective optimized architecture. As a result, the generated architecture achieves the best compromise between the number of ports per router, the number of links, the mapping, the traffic traces routing, the average internode distance, the flit

arrival rate, and the probability of the traffic being affected by on-chip noise sources.

5.4.2 Generated Architectures Evaluation: Quantitative Perspective

In this subsection, we quantitatively evaluate the efficiency of our methodology. This subsection also explains numerically the trade-off between different ASNoC metrics for different NoC applications. Accordingly, Figures 5.4 to 5.7 show the power, the area, the delay, the reliability, and the overall objective function for different architectures resulted from different architecture generation techniques. P, A, D, R, and F, represent the architectures optimized for power, area, delay, reliability, and the four metrics together using our GA-based methodology, respectively. MSH represents a standard mesh architecture with application cores mapped by the NMAP algorithm [133]. Mesh is selected to represent standard architectures in the comparison as it is shown in [70] to provide the best trade-off between different NoC metrics. Similarly, NMAP is used for mapping because of its superior performance over other mapping techniques, as shown in [133]. LR and NP are two architectures produced by semi-custom generation techniques. More precisely, LR represents the architecture generated by the long-range link insertion methodology presented in [75] and discussed in Subsection 2.5.2. NP represents the architecture generated by our network partitioning based methodology as presented in Chapter 4. The former methodology customizes the architecture for the performance, whereas the latter aims at reducing the NoC costs. We summarize our findings from these figures as follows:

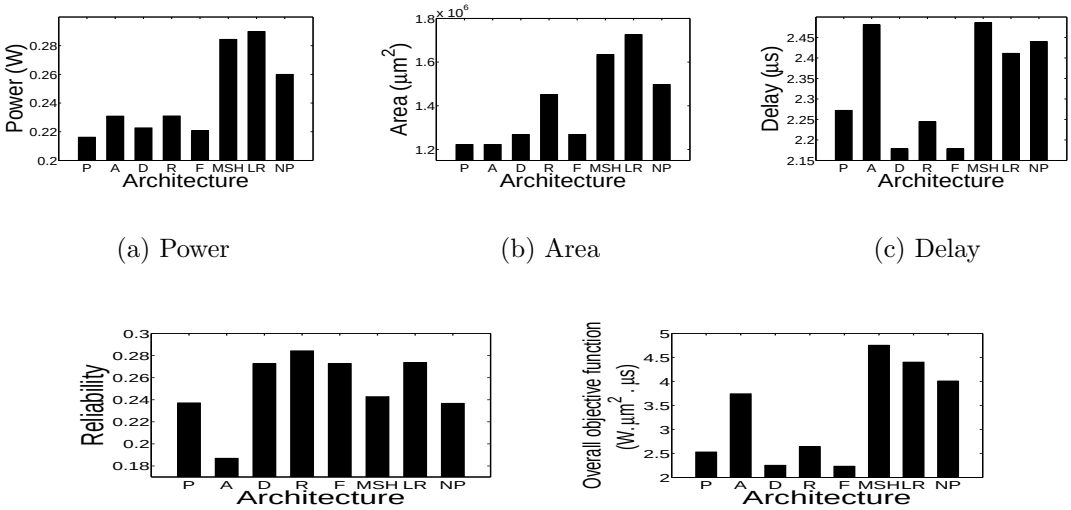
1. All the architectures generated by our optimization-based methodology, except for the A architecture, outperform standard and semi-custom architectures for all the benchmarks,



(a) Power (b) Area (c) Delay

(d) Reliability (e) Overall objective function

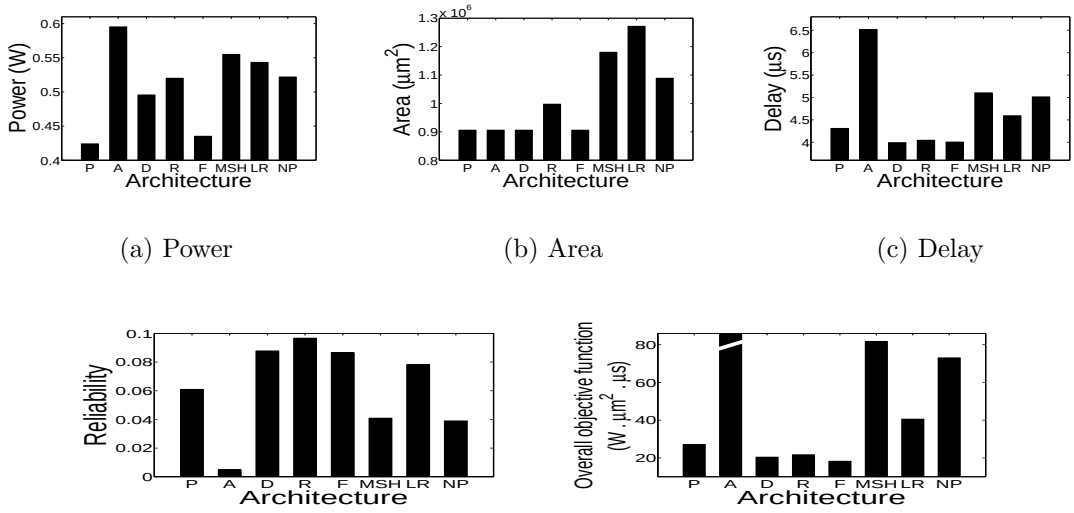
Figure 5.4: Power, area, delay, reliability, and overall objective function comparisons of different architectures for the AV benchmark. (In sub-figure (e), values for A and NP architectures are 238.94 and 540.28 $W \cdot \mu m^2 \cdot \mu s$, respectively.)



(a) Power (b) Area (c) Delay

(d) Reliability (e) Overall objective function

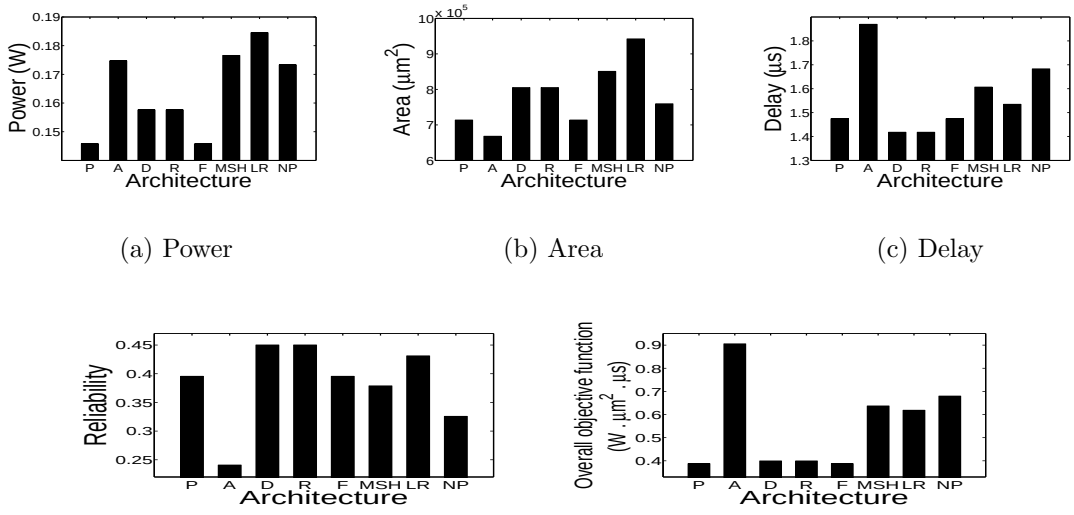
Figure 5.5: Power, area, delay, reliability, and overall objective function comparisons of different architectures for the VOPD benchmark.



(a) Power (b) Area (c) Delay

(d) Reliability (e) Overall objective function

Figure 5.6: Power, area, delay, reliability, and overall objective function comparisons of different architectures for the MPEG-4 benchmark. (In sub-figure (e), the value for the A architecture is 722.38 $W \cdot \mu m^2 \cdot \mu s$.)



(a) Power (b) Area (c) Delay

(d) Reliability (e) Overall objective function

Figure 5.7: Power, area, delay, reliability, and overall objective function comparisons of different architectures for the MWD benchmark.

2. Any single-metric optimized architecture outperforms other architectures for that metric. However, this single-objective optimization causes a clear deterioration in other metrics,

3. The A architecture specially gives bad results for any metric other than the area. Certainly, the use of the minimal possible resources results in a bad NoC performance,

4. The figures show a typical trade-off between performance metrics (delay and reliability) and cost metrics (power and area). For example, the R architecture always results in high power and area. Similarly, the P and A architectures result in low reliability. As a matter of fact, increasing the reliability requires using more resources, which affects the power and the area badly,

5. From the multi-objective perspective, the F architecture results in the best trade-off between all metrics. This is followed by the D and the P architectures. As our GA methodology compromises between more design variables to generate these two architectures, they achieve better overall objective function than the R and the A architecture. The A architecture is affected by the great deterioration happened to its power, delay, and reliability. As a result, it sometimes scores even worse than the standard and the semi-custom architectures.

To further quantify our results, Table 5.1 shows the percentage enhancement of the F architecture over all other architectures for different metrics. The values in the table are the average over the four benchmarks. Negative values indicate that the corresponding architecture is better than the F architecture for the selected metric. This is usually corresponding to the architecture that is solely optimized for this metric. For example, power-wise, the F architecture is worse than the P architecture by 2.5%, whereas it outperforms the A, D, R, MSH, LR, and NP architetures by

16.5%, 6.4%, 12.2%, 24.6%, 26.2%, and 23.5%, respectively. Accordingly, from multi-objective perspective, the last row shows that the F architecture is at least 89.3% better than previous architectures generation techniques. Furthermore, the last column in the table represents the mean of these average values. For any metric, this represents the percentage enhancement of the F architeture over all other architectures over the four benchmarks. Accordingly, on average, the F architecture is better than the other architectures by 13.4%, 10.6%, 10.1%, 16%, and 209.1% for the power, the area, the delay, the reliability, and the overall objective function, respectively.

Table 5.1: Average percentage enhancement of the F architecture over other architectures for the four benchmarks.

Metric	**Average enhancement (%)**							
	Architecture							**Mean**
	P	A	D	R	MSH	LR	NP	
Power	-2.5	16.5	6.4	12.2	24.6	26.2	23.5	13.4
Area	-2.4	-4.8	3.2	15.3	24.8	33.9	14.9	10.6
Delay	3.7	30.2	-1.0	1.6	15.8	9.7	20.7	10.1
Reliability	16.8	58.5	-3.9	-8.3	23.5	1.3	40.1	16.0
Overall objective function	20.9	1080.3	4.6	22.4	162.0	89.3	293.5	209.1

Table 5.1 also shows a strong correlation between the two performance metrics: delay and reliability. Architectures that are customized to lower the delay result in high reliability, and vice versa. Furthermore, the D architecture outperforms the F architecture for the reliability metric by 3.9%. Although the R architecture could not outperform the F architecture for the delay metric, it is only 1.6% worse than it. This could be explained by looking into the design variables that are included in both delay and reliability optimizations. On one hand, all design variables, i.e., mapping, traffic routing, and number of hops, which dominate the reliability optimization are included in the delay optimization. Moreover, the F architecture is generated by trading off more design variables than the D architecture. These are the variables

corresponding to the power and the area metrics. Accordingly, this enables the D architecture to outperform the F architecture with respect to the reliability. On the other hand, some design variables that dominate the delay optimization, like the number of ports per router, are not included in the reliability optimization. Therefore, the R architecture could not outperform the F architecture with respect to the delay. Finally, a similar correlation could by noticed from the table between the cost metrics: power and area. The power optimization includes all the design variables required for the area optimization. Consequently, the P architecture is more area-efficient than the F architecture.

5.4.3 Generation Time Evaluation

In this subsection, we evaluate the efficiency of our methodology with respect to the architecture generation time. For any optimization technique to be practically valuable, the required architecture should be generated quickly and in a reasonable time. Accordingly, Table 5.2 shows the average number of generations of our GA-based methodology. Consequently, this indicates how many iterations the GA engine goes through to converge to the optimum architecture. The number of generations of any GA-based technique might differ from one trial to another based on the random generation of the chromosomes. Therefore, for every architecture, we ran the GA seven times. We further excluded the two trials with the maximum and the minimum number of generations. We then averaged the remaining five trials. The table shows that the number of generations required to produce any of the five architectures is proportional to the number of cores within the benchmark. Furthermore, for any benchmark application, the number of generations depends on the number of design variables to compromise in between during the architecture generation process. Therefore, for single-objective optimization, the generation of

the P and the D architectures takes more longer than the A and the R architectures because the former trades off more design variables than the latter.

Table 5.2: Average number of generations for our GA-based full-custom architecture generation methodology.

Benchmark	**Average number of generations**				
	P	A	D	R	F
AV (18 cores)	125	58	77	40	396
VOPD (16 cores)	108	46	58	11	300
MPEG-4 (12 cores)	49	21	25	12	201
MWD (9 cores)	47	7	16	8	114

The number of iteration is a platform-independent scheme to evaluate the running time efficiency of different optimization techniques. However, previous full-custom architecture generation techniques presented their running time in seconds. Therefore, Table 5.3 shows the average running time of our GA-based methodology corresponding to the average number of generations in Table 5.2. For our experiments, we used a desktop computer with 3.6 GHz Intel Xeon CPU and 2.75 GB RAM. Furthermore, the running time of two previous full-custom architecture optimization techniques, discussed in Subsection 2.5.3, are also included in the table for applications with the same number of cores. MILP is the mixed integer linear programming technique presented in [152], whereas ISIS is the GA-based technique presented in [153]. The table shows that the maximum execution time for our methodology, for the F architecture of the AV benchmark, is 72.35 seconds. Accordingly, our methodology outperforms previous optimization techniques with respect to the running time complexity by multiple orders of magnitude. For example, the MILP technique failed to converge to a solution within a time out period of 8 hours for three of the benchmarks. Moreover, our methodology is at least 14 times faster than the ISIS technique. This low running time complexity of our methodology makes it very attractive for ASNoC full-custom architecture generation. We note here that a large

amount of the execution time of our methodology is spent to generate the routing tables. As a result, the timing complexity could be further enhanced by proposing more elaborate adaptive routing strategies.

Table 5.3: Average execution time for different full-custom architecture optimization techniques. (t.o.: time out of 8 hours.)

Benchmark	**Average execution time (seconds)**						
	P	A	D	R	F	MILP	ISIS
AV (18 cores)	24.42	11.45	15.28	7.99	72.35	t.o.	$\succ$1567
VOPD (16 cores)	20.03	8.63	10.79	2.43	58.23	t.o.	1564
MPEG-4 (12 cores)	9.23	4.09	4.77	2.63	30.27	t.o.	901
MWD (9 cores)	8.10	1.56	3.09	1.81	21.92	118	324

5.5 Chapter Summary

In this chapter, we presented a GA-based multi-objective methodology for ASNoC full-custom architecture optimization. Our methodology considered four NoC metrics: power, area, delay, and reliability, simultaneously. First, the assumptions behind our methodology was presented. The GA-solution of the full-custom architecture generation problem was then explained. The optimization could be carried out for power, area, delay, reliability, or the four of them according to weight factors supplied by the designer. Results showed that the proposed methodology is an efficient way to compromise between different NoC metrics. The multi-objective optimized architecture resulted from our methodology was found to be at least 89.3% better than previous generation techniques. Finally, the execution time of our methodology is at least 14 times faster than that of previous architecture optimization techniques. This work has been published in brief in [20, 21] and in full in [22].

In the next chapter, we present a GA-based multi-objective optimization methodology for NoC standard architectures. The methodology selects the best

standard architecture for any given application and optimally maps its cores onto that architecture. The methodology is also used to carry out an application-specific cost and performance evaluation of different NoC standard architectures.

Chapter 6

A Multi-objective Optimization Methodology of ASNoC Standard Architectures using GA

This chapter presents the fourth contribution of this dissertation. We propose a GA-based optimization methodology to generate the best on-chip standard architecture for any NoC application. Similar to the full-custom architecture generation methodology presented in Chapter 5, the standard architecture optimization could be carried out for single or multiple metrics based on weight factors provided by the designer. The methodology presented in this chapter combines the best selection of NoC standard architecture with the optimum mapping of application cores onto that architecture. Our methodology is applied onto different NoC benchmark applications, as case studies. For every benchmark, we use the optimum mapping of our methodology to carry out fair cost and performance evaluations of different NoC standard architectures. These evaluations results emphasize the application-specific nature of NoC-based systems, and hence, the importance of our optimization methodology. Finally, for all evaluation metrics, the results also show that the

mapping achieved by our methodology outperforms those generated by previous mapping techniques for all standard NoC architectures considered in this study.

This chapter is organized as follows. Section 6.1 gives an introduction about the multi-objective standard architecture generation problem addressed in this chapter. The assumptions of our methodology are presented in Section 6.2. Section 6.3 presents our GA-based solution for the standard architecture generation problem. This section discusses in details out integer chromosome representation of NoC standard architectures, the legality criteria of any generated architecture, and the GA-based generation methodology. Section 6.4 presents some experimental results for different NoC benchmarks to validate our work. Finally, Section 6.5 summarizes the chapter.

6.1 Introduction

Standard architectures are still attractive to many NoC-based applications due to three main advantages. First, different analysis and design techniques are already presented for standard architectures that could be easily employed with NoC. Second, many predefined routing algorithms could also be used with these standard architectures. Third, the use of standard architectures ensures the regularity of the underlying network, and hence, facilitates the implementation and the floorplanning. To the best of our knowledge, the work presented so far for application-specific customization of standard architectures could be categorized into two main approaches. The first approach deals with the architectures selection and the core mapping as two separate problems. As the two problems are tightly related, this approach might be misleading. The second approach combines between architecture selection and core mapping. However, the methodologies presented for this approach were either single-objective, which results in significant deterioration with respect to other metrics, or were built on heuristics, which could not guarantee the optimum

on-chip network for the given application. Therefore, one challenge problem is to customize the standard NoC architecture for any given application using multi-objective optimization-based technique that combines between architecture selection and application cores mapping onto that architecture.

In this chapter, we present an architecture generation methodology to solve the above mentioned multi-objective optimization problem. More precisely, we employ GA optimization to realize the best standard architecture for any application with respect to power, area, delay, and reliability, simultaneously. Our methodology combines both standard architecture selection and optimum core mapping onto that architecture. To realize this optimum standard architecture, our methodology trades off many conflicting NoC design variables. The most important of these variables are the number of ports per router, the number of links, the number of routers, the mapping, the average internode distance, the flit arrival rate, and the probability of the traffic being affected by on-chip noise sources. To this end, this chapter presents four main contributions:

1. Proposing a technique of representing any ASNoC standard architecture as a GA integer chromosome,

2. Presenting a methodology for ASNoC standard architecture generation based on the proposed GA chromosome representation,

3. Evaluating the presented methodology by applying it onto different real benchmark applications, as case studies, and

4. Using the optimum mapping of the presented methodology to carry out a fair application-specific cost and performance evaluations of different NoC standard architectures.

6.2 Assumptions of the Generated Architectures

In this section, we present the assumptions of our methodology. In this chapter, we consider only the eight architectures shown in Figure 3.2. However, the same used methodology could be applied for any other standard or custom architectures. Moreover, as explained in Section 5.2, these assumptions consider some design variables to be fixed for all architectures in order to allow for the generation of the required architecture as quickly as possible. Consequently, to ensure this low running time of our methodology, we consider the following assumptions:

1. The buffer size for all the ports within routers are the same (8-flit buffers are assumed in this chapter),

2. Each link consists of two channels for packet transmission and reception, respectively. The channel width of all the links in the network is the same and is equal to the flit size (8-bit flits are assumed in this chapter),

3. The application, with its traffic characteristics, is represented at any time by a core graph,

4. Deterministic shortest path routing is used, and

5. The length of all links is the same and allows for a single clock cycle data transfer, i.e., no repeaters are required. This assumption might not hold in the actual circuit-level floorplanning. However, it is still valid for the purpose of system-level evaluation and comparison.

6.3 Standard Architecture Generation using GA

This section discusses in details our multi-objective GA-based solution for the standard architecture optimization problem. Accordingly, Subsection 6.3.1 explains our GA chromosome representation for any ASNoC standard architecture. Thereafter, Subsection 6.3.2 discusses the legality criteria of any generated architecture by the GA engine. Finally, Subsection 6.3.3 presents our GA-based methodology for ASNoC standard architecture generation. The inputs to our methodology are the supply voltage (V_{dd}), frequency of operation (f_{op}), the application core graph, the targeted fabrication technology, the optimization weight factors (γ_P, γ_A, γ_D, and γ_R), and the data resulted from analyzing the employed router library. For every router in the library, these are its area (A_{Ri}), its power constants (k_{Dp}, k_D, k_{Lp}, and k_L), and its maximum allowable flit arrival rate (FAR_{max}).

6.3.1 Integer Chromosome Representation of ASNoC Architectures

As explained in Section 2.4, applying GA requires representing different NoC architectures in the form of chromosomes. In this chapter, our GA-based standard optimization methodology represents these architectures as integer vectors. As shown in Figure 6.1, the size of each vector is $N + 1$, where N is the number of cores in the application. The first integer number is the architecture ID followed by N numbers representing the position of every core in the architecture (i.e., the one-by-one mapping of application cores onto different nodes of the architecture).

The Matlab® GA toolbox [167] is used in solving the optimization problem in this chapter. As previously explained in Subsection 5.3.1, we used the built-in uniform creation function and the built-in stochastic uniform selection function. However, using any of the built-in mutation or crossover functions in Matlab® often results in having repeated numbers through the mapping part of the newly generated

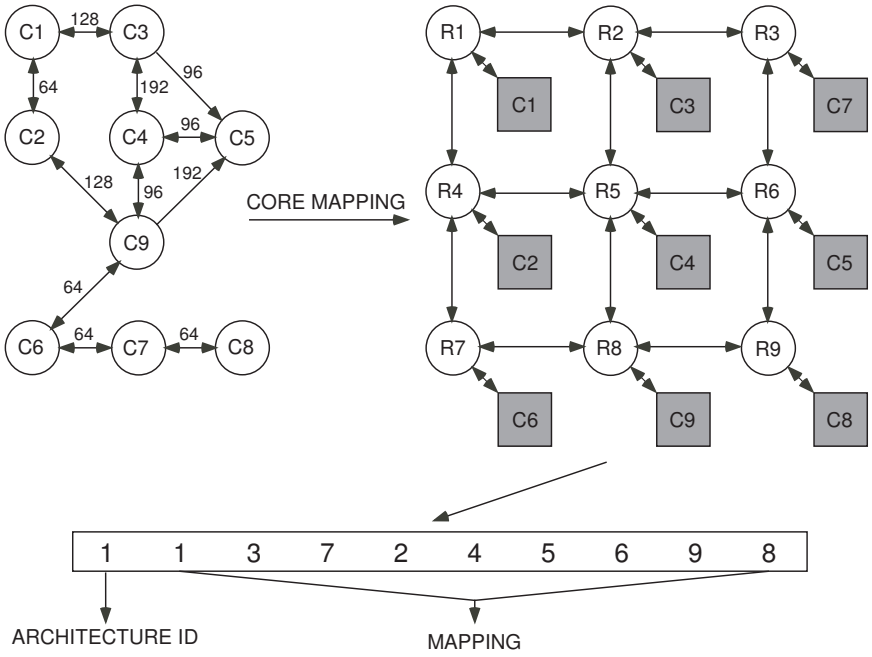


Figure 6.1: Example of an integer chromosome representation for the MWD benchmark with a random mapping onto 3×3 mesh architecture.

chromosome. This means that a core in the application is mapped onto two or more nodes in the architecture. According to our chromosome representation, this should not be allowed. One solution that we tried with this problem was to mark these chromosomes as invalid. However, as the possibility of having valid chromosomes using this solution is too low, it prevented the GA engine from converging into the optimum architecture within a reasonable time. Therefore, in order to ensure the convergence of our optimization methodology, we built our own special mutation and crossover functions. First, for the chromosomes generated by mutation, half of them is generated completely random. This is done to ensure introducing new architectures to the optimizer. The second half is generated by swapping any two cores that are selected randomly whether they are communicating together or not. Second, Algorithm 1 represents our crossover function in details. The main idea behind this

function is to bring communicating cores closer to each other, while maintaining the build-in randomness of GA. The inputs to the function is the best chromosome from the previous generation, the required number of chromosomes to be generated by crossover, the number of cores within the application, and the traffic distribution matrix. The function then goes in an iterative manner to generate a new chromosome from the one generated in the previous iteration. The first chromosome is generated from the best chromosome that is given as an input. Every iteration starts, in line 4, by selecting a node in the architecture randomly, *SelectedNode*. Thereafter, in line 5, the corresponding core that is mapped onto this node is identified, *SelectedCore*. In lines 6-7, a core that is mapped onto one of the nodes adjacent to *SelectedNode* is chosen randomly. This represents the first core to be swapped. Similarly, in line 8, the second core to be swapped is chosen randomly from the cores that are communicating with *SelectedCore*. Finally, the swapping of the two cores is carried out in line 9.

Algorithm 1 Crossover Function

Require: V is a vector representing the best chromosome of the previous generation
Require: $nKids$ is the number of chromosomes generated by crossover
Require: N is the number of cores within the application
Require: Δ is the application traffic distribution matrix

1: $xoverKids(0) = V$
2: $Architecture = V(1)$
3: **for** $i = 1$ **to** $nkids$ **do**
4: $\quad SelectedNode = \text{SelectRandomNode}(N)$
5: $\quad SelectedCore = xoverKids[i-1, SourceNode + 1]$
6: $\quad SwapNode1 = \text{GetRandomAdjacentNode}(N, Architecture, SelectedNode)$
7: $\quad SwapCore1 = xoverKids[i-1, SwapNode1 + 1]$
8: $\quad SwapCore2 = \text{GetRandomCommunicatingCore}(SelectedCore, \Delta)$
9: $\quad xoverKids(i) = \text{Swap}(xoverKids(i-1), SwapCore1, SwapCore2)$
10: **end for**

6.3.2 Legality Criteria for Generated Architectures

Architectures are generated randomly by the GA engine. Therefore, in this subsection, we present the legality criterion for any generated architecture. Any architecture that fails to meet this legality constraint is considered invalid. Similar to our custom architecture optimization methodology presented in chapter 5, the optimization objective function (f) is multiplied by an architecture legality factor (β_l) to prevent the survival of invalid architectures from one generation to the next. This factor is 1 for any valid architecture, whereas high legality factors are used for invalid architectures to increase the overall values of their fitness functions. Therefore, the fitness function (F) of our GA-based methodology is expressed as

$$F = f \cdot \beta_l \tag{6.1}$$

The fitness function (F) is used to evaluate different architectures. Any architecture is considered valid ($\beta_l = 1$) only if it meets the flit arrival rate (FAR) constraint. This constraint ensures that the flit arrival rate for any router in the generated architecture is less than its corresponding FAR_{max}, which is obtained from the router library analysis. Mathematically, this constraint is formulated as

$$\alpha_{fi_{max}} \leq FAR_{i_{max}}, \quad \forall(i = 1 \; to \; N_R) \tag{6.2}$$

where $\alpha_{fi_{max}}$ is the maximum flit arrival rate over all the ports of router i. Different flit arrival rates are calculated in our GA-based methodology by routing all the traffic through the generated architecture according to the employed routing protocol.

6.3.3 Methodology for Standard Architecture Generation

Figure 6.2 shows our methodology for standard architecture optimization using GA. Before running our GA engine, area and power analysis is carried out on the employed router library to generate the required data for the GA optimization engine. This analysis results in different routers area, power constants, and maximum possible flit arrival rates. The GA optimization starts, in the first step, by reading the inputs required by the engine. Thereafter, the GA engine runs iteratively to find the best architecture with respect to power, area, delay, and reliability. Each generation consists of 20 different chromosomes representing 20 different architectures. Every chromosome in any generation is evaluated by steps 2-7 in the flowchart. In the second step, the connectivity matrix corresponding to the architecture of the chromosome is generated. In the third step, packets are routed according to the routing scheme of that architecture from all source cores to all destination cores. Moreover, all input and output ports traffic is calculated for every router in the architectures. Based on this traffic, the flit arrival rate for all the ports is calculated in the fourth step. Consequently, the FAR constraint is checked in the fifth step according to (6.2). In the sixth step, the power, area, delay, and reliability of the architecture are calculated according to (3.10), (3.13), (3.14), and (3.22), respectively. These values are then used in the seventh step to calculate the GA fitness function according to (6.1). Any architecture that fails the FAR constraint is marked invalid to prevent it from surviving to the next generation. In the eighth step, once the exit criterion is satisfied, the GA engine generates the best architecture with the optimum mapping onto that architecture. Finally, if the exist criterion is not satisfied and a new generation is to be produced, the two architectures with the best fitness function are allowed to survive. Moreover, nine new architectures are generated by crossover and the nine remaining architectures in each generation are produced by random mutation. Different values

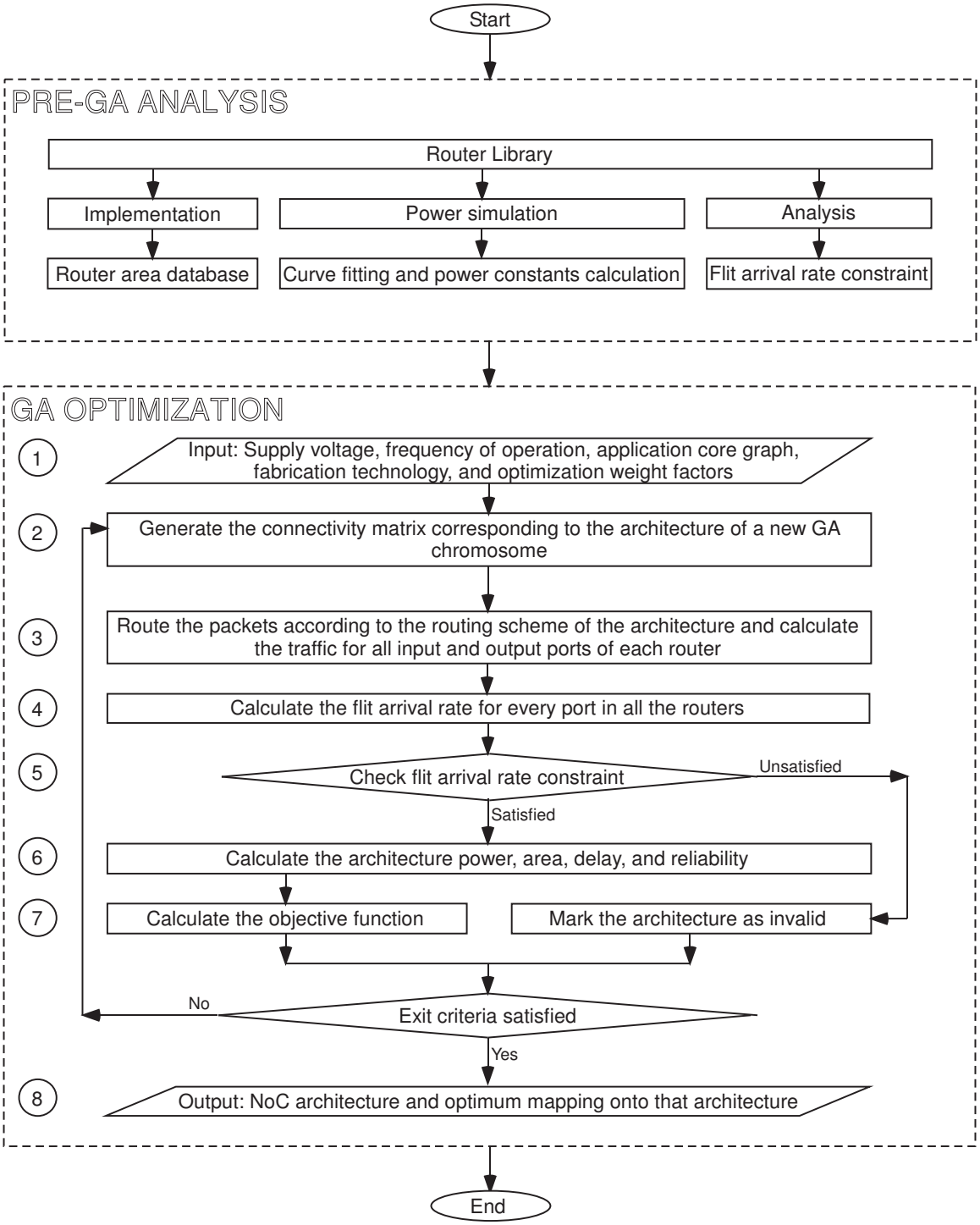


Figure 6.2: Proposed methodology for standard architecture optimization using GA.

for population, elitism, crossover, and mutation could be used with our methodology as well.

6.4 Experimental Results

We apply our GA-based methodology on four of the real benchmark applications as shown in Figure 3.1: AV benchmark with 18 cores, VOPD benchmark with 16 cores, MPEG-4 benchmark with 12 cores, and MWD benchmark with 9 cores. For every benchmark, we use our GA-based methodology to generate five different architectures. These are the architecture that is optimized solely for power ($\gamma_P = 1, \gamma_A = \gamma_D = \gamma_R = 0$), the architecture that is optimized solely for area ($\gamma_A = 1, \gamma_P = \gamma_D = \gamma_R = 0$), the architecture that is optimized solely for delay ($\gamma_D = 1, \gamma_A = \gamma_P = \gamma_R = 0$), the architecture that is optimized solely for reliability ($\gamma_R = 1, \gamma_A = \gamma_P = \gamma_D = 0$), and the architecture that is optimized for the overall objective function ($\gamma_P = \gamma_A = \gamma_D = \gamma_R = 1$). To better explain our findings, the following subsection discusses how our methodology selects the best application-specific standard architecture out of those considered in this study. Moreover, it also presents quantitative evaluations of different NoC standard architectures based on the optimum mapping onto these architectures. Finally, Subsection 6.4.2 evaluates the optimum mapping achieved by our methodology.

6.4.1 Selection and Evaluation of NoC Standard Architectures

The first objective of our methodology is to select the best standard architecture for any given application. In this subsection, we emphasize the importance of this goal from an application-specific perspective by comparing the architecture selected by our GA-based methodology to all other architectures considered in this study. Previous work in NoC standard architecture evaluations was mapping-

independent and used synthetic traffics rather than real benchmark ones. Nonetheless, comparisons presented in this subsection are application-specific and based on the optimum mapping onto each architecture. It is worth mentioning that the area of any architecture could be straightforwardly calculated without the need for running the optimization methodology. However, the area affects the overall objective function evaluation. Consequently, we include the area comparison in this subsection as well. For the five evaluation metrics, Tables 6.1 to 6.5 summarize the results for the eight standard architectures considered in this study[1]. In the following paragraphs, we will discuss each of these tables in more details. In general, the architecture on the first row of any table is the one selected by our methodology. For each metric, the percentage of other architectures with respect to this best architecture is also included in the tables. The tables first show that the best architecture with respect to any metric is an application-dependent. For any metric, the best architecture is the one that achieves the best trade-off between different design variables related to this metric. Some of these variables are functions of the applications traffic. This clarifies the importance of using an application-specific standard architecture realization methodology with ASNoC-based systems. Furthermore, for all benchmarks, the tables show very large variations between the cost and the performance of different standard architectures. This further emphasizes the importance of our application-specific standard architecture optimization methodology. Finally, from the tables, we notice:

1. For the power metric, based on the benchmark, Table 6.1 shows that the 2-ary 2-fly butterfly and the ring architectures achieve the lowest power cost. Similar to our discussion in Subsection 5.4.1, the two architectures achieve this low power by realizing the best trade-off between different conflicting design

[1]No polygon architecture is implemented for the MWD benchmark because it has odd number of cores.

variables. The most important of these variables are the number of port per router, the number of links, the number of routers, the mapping, the average internode distance, and the flit arrival rate. In contrast, the clos architecture has the worst power cost. This is because of the large number of routers, links, and ports per every router, which dominate other design variables and makes the clos as the most power inefficient architecture. Finally, our results with respect to power partially agree with previous evaluations of NoC standard architectures in [70,145]. These evaluations did not consider the ring architecture, and hence, they selected mesh and 2-ary 2-fly butterfly to be the most power efficient architectures.

Table 6.1: Power comparison between different standard architectures for different benchmark applications. (% is the percentage of the power with respect to that of the best architecture.)

Benchmark							
AV		**VOPD**		**MPEG-4**		**MWD**	
Arch.	%	**Arch.**	%	**Arch.**	%	**Arch.**	%
BFT22	100.00	RNG	100.00	RNG	100.00	RNG	100.00
BNT	100.72	BFT22	108.19	BFT22	105.99	MSH	106.92
RNG	101.15	MSH	112.90	MSH	108.95	BFT22	107.63
BFT23	102.93	PLG	119.87	PLG	111.52	BNT	108.47
MSH	109.35	BNT	120.28	BNT	116.51	BFT23	119.73
PLG	114.73	BFT23	136.62	TRS	128.81	TRS	130.34
TRS	133.30	TRS	143.19	BFT23	135.51	CLS	165.33
CLS	225.17	CLS	244.09	CLS	194.31		

2. For the area metric, Table 6.2 shows that the binary tree architecture achieves the lowest area cost. This is because of the low number of routers and number of ports per each router of that architecture. In contrast, similar to the power metric, the clos architecture lies on the bottom of all architectures because of its high router and links areas. Finally, our results with respect to the area metric partially agree with previous area evaluation of NoC standard

architectures in [70]. This evaluation did not consider the binary tree nor the ring architectures, and hence, it selected the 2-ary 2-fly butterfly to be the most area efficient architecture.

Table 6.2: Area comparison between different standard architectures for different benchmark applications. (% is the percentage of the area with respect to that of the best architecture.)

Benchmark							
AV		**VOPD**		**MPEG-4**		**MWD**	
Arch.	**%**	**Arch.**	**%**	**Arch.**	**%**	**Arch.**	**%**
BNT	100.00	BNT	100.00	BNT	100.00	BNT	100.00
BFT22	108.19	BFT22	109.40	BFT22	112.93	RNG	118.02
RNG	108.19	RNG	109.40	RNG	112.93	BFT22	131.04
MSH	139.42	MSH	140.86	MSH	139.98	MSH	140.66
PLG	139.42	PLG	140.86	PLG	145.43	TRS	185.95
TRS	170.58	TRS	172.41	TRS	177.94	BFT23	220.66
BFT23	182.18	BFT23	184.14	BFT23	190.04	CLS	244.30
CLS	326.46	CLS	298.45	CLS	242.94		

3. For the delay metric, based on the benchmark, Table 6.3 shows that the 2-ary 3-fly butterfly , the 2-ary 2-fly butterfly, and the mesh architectures has the best delay. Compared to previous evaluations, this result emphasizes the importance of evaluating different architectures according to the real benchmark traffic and based on the optimum mapping. Previous mapping-independent evaluations with synthetic traffic in [70, 144] selected the polygon to be the most delay efficient architecture. However, our real application-specific delay evaluation shows that the polygon is worse than the best architecture of different benchmarks by 3.4% to 17.75%. In contrast, the clos and binary tree have the worst delay over all architectures. On one hand, clos architecture allows the traffic between any two cores to be exchanges in at most two hops. However, the large number of ports for each router makes the arbitration delay very high. This significantly increases the overall delay of the clos leaving it at the bottom

of all architectures. On the other hand, the binary tree has a low arbitration delay. However, the large average internode distance increases the overall delay making the binary tree from the worst architectures with respect to the delay metric. In a nutshell, the performance of any architecture with respect to any metric depends on how much compromise does this architecture achieve between all the design variables related to this metric.

Table 6.3: Delay comparison between different standard architectures for different benchmark applications. (% is the percentage of the delay with respect to that of the best architecture.)

Benchmark							
AV		**VOPD**		**MPEG-4**		**MWD**	
Arch.	**%**	**Arch.**	**%**	**Arch.**	**%**	**Arch.**	**%**
BFT23	100.00	BFT22	100.00	MSH	100.00	MSH	100.00
BFT22	103.41	RNG	103.40	BFT22	100.91	BFT22	102.68
BNT	114.02	MSH	108.56	PLG	103.30	RNG	103.90
MSH	116.79	PLG	108.56	RNG	105.88	TRS	104.85
RNG	117.35	TRS	115.23	TRS	106.04	BFT23	106.07
PLG	117.75	BFT23	121.76	BFT23	124.17	CLS	112.96
TRS	121.31	BNT	125.38	CLS	124.31	BNT	113.41
CLS	126.37	CLS	130.45	BNT	124.65		

4. For the reliability metric, maximizing the reliability requires that all traffic traces are routed through the shortest possible paths to minimize the possibility of being affected by on-chip noise sources. Consequently, Table 6.4 shows that clos and torus architectures result in the best reliability performance because of their low average internode distances. In contrast, binary tree and ring architectures take the traffic through long paths, and hence, they have the worst reliability performance over all architectures.

5. For the overall multi-objective function, based on the benchmark, Table 6.5 shows that 2-ary 2-fly butterfly, mesh, and ring architectures achieve the best

Table 6.4: Reliability comparison between different standard architectures for different benchmark applications. (% is the percentage of the reliability with respect to that of the best architecture.)

Benchmark							
AV		**VOPD**		**MPEG-4**		**MWD**	
Arch.	**%**	**Arch.**	**%**	**Arch.**	**%**	**Arch.**	**%**
CLS	100.00	CLS	100.00	TRS	100.00	TRS	100.00
BFT23	76.24	TRS	94.34	MSH	93.11	CLS	95.78
BFT22	53.85	PLG	93.83	PLG	84.75	MSH	93.73
TRS	52.58	MSH	93.64	CLS	71.15	BFT22	80.60
MSH	46.68	BFT22	92.98	BFT22	62.68	BFT23	80.60
PLG	44.41	RNG	85.21	RNG	49.03	RNG	78.88
BNT	28.19	BFT23	58.88	BFT23	24.10	BNT	64.96
RNG	25.79	BNT	47.58	BNT	18.28		

compromise between power, area, delay, and reliability. Consequently, they realize the best trade-off between all design variables related to these metrics, as discussed in Section 3.8. Furthermore, a specific architecture might perform well for certain metrics; however, its multi-objective function might be worse than others that perform moderately for all metrics. For example, the clos architecture achieves the best reliability performance. Nevertheless, its high power, area, and delay make it the worst over all architectures with respect to the combination of the four metrics.

6.4.2 Core Mapping Evaluation

In this subsection, we quantify the mapping efficiency of the GA-based standard architecture optimization methodology. For each of the eight architectures, we compare the results generated by it to those of previous mapping techniques. Accordingly, we use both NMAP and BMAP[2] techniques to map benchmark cores

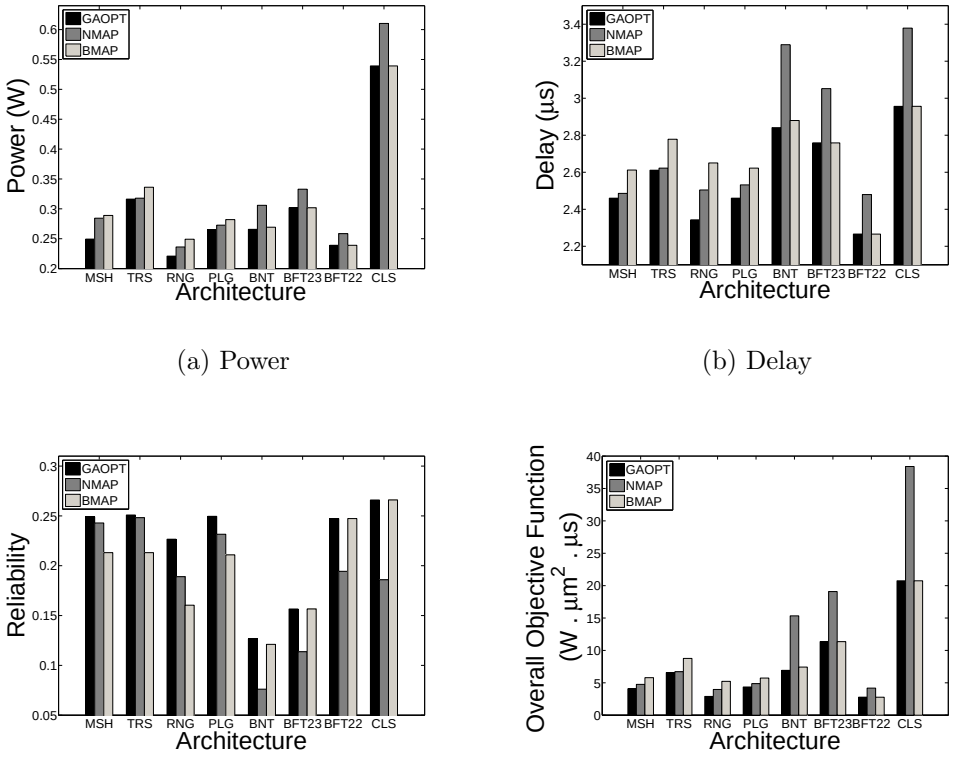
[2]BMAP could only be used with applications with N core, given that $\log_2 N$ = integer.

Table 6.5: Overall objective function comparison between different standard architectures for different benchmark applications. (% is the percentage with respect to the best architecture.)

Benchmark							
AV		**VOPD**		**MPEG-4**		**MWD**	
Arch.	**%**	**Arch.**	**%**	**Arch.**	**%**	**Arch.**	**%**
BFT22	100.00	BFT22	100.00	MSH	100.00	RNG	100.00
BFT23	118.14	RNG	104.24	BFT22	110.20	MSH	103.26
MSH	193.22	MSH	146.73	PLG	112.83	BFT22	115.71
PLG	202.37	PLG	156.55	TRS	138.82	BNT	121.92
RNG	238.01	TRS	236.83	RNG	139.14	TRS	163.56
TRS	263.56	BNT	248.96	BNT	453.95	BFT23	223.81
BNT	322.55	BFT23	408.63	CLS	471.38	CLS	306.69
CLS	446.85	CLS	746.40	BFT23	757.89		

onto these architectures. It is worth mentioning that NMAP was proved to outperform PBB, PMAP, and GMAP in [133].

Figure 6.3 shows samples of our comparison results for the VOPD benchmark. As area is not a function of the mapping, we do not include it into the figure. For all the metrics, the figure shows that the GA-based methodology outperforms other mapping techniques for all architectures. On average over all architectures, the GA-based mapping is better than NMAP by 8.9%, 7.7%, 25.7%, and 49.1% with respect to power, delay, reliability, and the overall multi-objective function, respectively. Similarly, it outperforms BMAP by 5.4%, 4.2%, 12.3%, and 24.3% with respect to the same four metrics, respectively. The figure also clarifies that NMAP is more efficient than BMAP for grid-based (mesh and torus) and ring-based (ring and polygon) architectures. Nevertheless, BMAP outperforms NMAP for all the metrics for tree-based architectures.



(a) Power

(b) Delay

(c) Reliability

(d) Overall objective function

Figure 6.3: Comaprison between different mapping techniques for the VOPD benchmark.

6.5 Chapter Summary

In this chapter, we presented a GA-based multi-objective methodology for NoC standard architecture optimization. Our methodology considered four NoC metrics: power, area, delay, and reliability, simultaneously. First, the assumptions behind our methodology was presented. The GA-solution of the standard architecture generation problem was then explained. The optimization could be carried out for power, area, delay, reliability, or the four of them according to weight factors supplied by the designer. Our methodology combined standard architecture selection and optimum

core mapping onto that architecture. The mapping efficiency of the methodology was evaluated by comparing the results from it to those of previous mapping techniques. From multi-objective perspective, the mapping resulted from our methodology was found to be at least 24.3% better than that of previous techniques. Moreover, for different benchmarks, we used our methodology to carry out real application-specific evaluations of different NoC standard architectures. Results showed that the proposed methodology efficiently compromized between different NoC metrics. This work has been published in brief in [23, 24] and is submitted for publication in full in [25].

Chapter 7

Conclusions and Future Work

The customization of on-chip network architectures lies at the heart of reducing the cost and enhancing the performance of ASNoC-based systems. In this dissertation, we proposed evaluation models and presented three methodologies to customize the architecture of standard, semi-custom, and full-custom ASNoCs. The models were proposed at the system-level to allow for an early design space exploration. Based on these models, our methodologies automatically generate the best architecture that is tailor-made for a specific application and satisfies the designer constraints. Comparative analysis were carried out as a proof of concept to explain the significance of our contributions. We believe that our contributions could help reducing the cost and improving the performance of the underlying architectures for modern communication-intensive SoC applications.

7.1 Summary

This section summarizes the research work presented in this dissertation. Currently in the ASNoC research community, on-chip architectures are realized using one of three generation techniques. These are through using pre-defined standard architectures,

partially customizing the network architecture to enhance certain metrics, or fully optimizing the network architecture to get the maximum possible performance with the minimum possible cost. Each of these three techniques has its own advantages and disadvantages with respect to the availability of design methodologies, the simplicity of routing, the freedom of deadlock and livelock, the availability of generation tools, the regularity of the underlying network, the simplicity of implementation and floorpanning, the architecture generation time, the total fabrication and running-time costs, and the overall performance. Throughout our dissertation, we did not advocate any of these generation techniques. However, for each of the three techniques, we presented a novel methodology to customize the underlying network architecture. The three methodologies were proposed at the system-level to allow for a quick generation of the required architecture and an early evaluation of different design alternatives. Figure 7.1 gives an overall picture of this dissertation and illustrates the different methodologies that were presented throughout this research work.

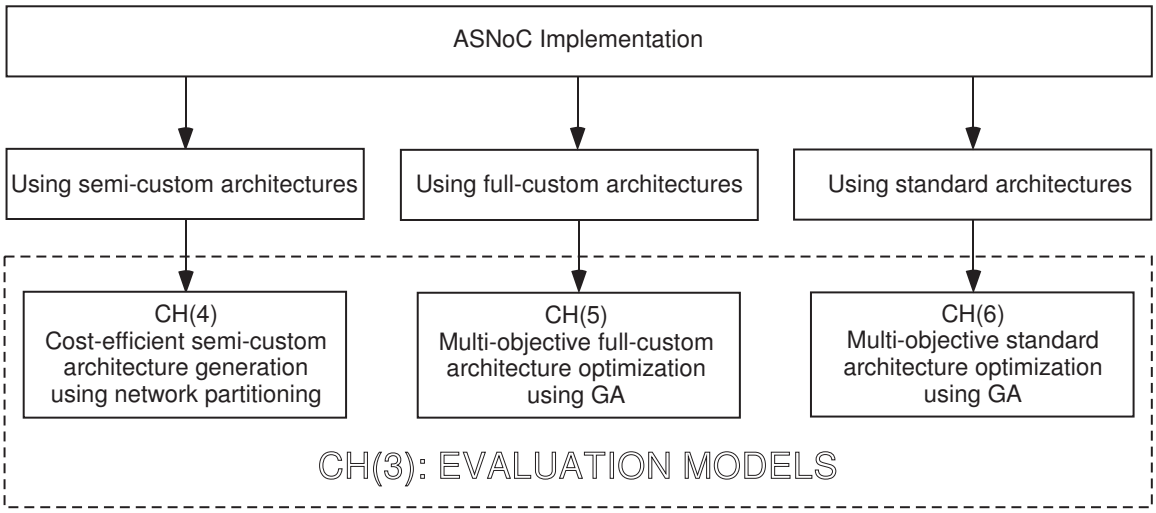


Figure 7.1: Overall picture of the dissertation.

In the first part of the research work, presented in Chapter 3, we proposed two system-level power and reliability evaluation models. The two models, with area and

delay models employed from the literature, were then used throughout the remaining chapters to evaluate the cost and performance of different ASNoC architectures.

In the second part of the research work, presented in Chapter 4, we proposed a cost-efficient customization methodology for ASNoC semi-custom architectures. The methodology is based on network partitioning techniques and proved useful in reducing ASNoC area and power consumption for different ASNoC benchmarks.

In the third part of the research work, presented in Chapter 5, we proposed a multi-objective optimization methodology for ASNoC full-custom architectures. The methodology is based on GAs and proved useful in generating ASNoC full-custom architectures with the lowest possible cost and the maximum possible performance, simultaneously.

In the fourth part of the research work, presented in Chapter 6, we proposed a multi-objective optimization methodology for ASNoC standard architectures. The methodology is based on GAs and proved useful in selecting the best ASNoC standard architecture and optimally mapping application cores onto that architecture.

7.2 Contributions

The contribution of this research work can be summarized as follows:

7.2.1 NoC Power and Reliability Models

For our first contribution, we proposed two models to evaluate NoC power consumption and reliability, respectively. The power model is an analytical one that represents the total power consumption in NoC routers and links. The reliability model is a probabilistic one that measures how reliable the network is to transfer packets successfully in the presence of noise. The two models were formulated at the system-level to allow for an early design space exploration and a quick generation of

the required NoC architecture. The power model has been published in brief in [11] and in full in [12]. Moreover, the reliability model has been published in brief in [13] and in full in [14].

7.2.2 Cost-Efficient Semi-Custom Architectures Generation using Network Partitioning

For our second contribution, we proposed a novel power and area-aware generation methodology for ASNoC semi-custom architectures using network partitioning techniques. The partitioning problem was formulated and evaluated with respect to other partitioning schemes. The effect of using network partitioning on ASNoC power, area, and delay was then analyzed. Moreover, the efficiency of our methodology for any application was formulated using power and delay factors. Finally, our methodology was evaluated through different case studies of real ASNoC benchmarks. On average over all the benchmarks, our methodology was found to be at least 4.87% more power efficient and 9.23% more area efficient than previous architecture generation techniques. This work has been published in brief in [16–18] and was submitted for publication in full in [19].

7.2.3 Multi-objective Full-Custom Architectures Optimization using GA

For our third contribution, we proposed a novel multi-objective optimization methodology for ASNoC full-custom architectures using GA. The methodology considered four ASNoC metrics: power, area, delay, and reliability. The importance of every metrics for the optimization process could be controlled by the designer through weight factors. Moreover, binary chromosome representation was used to describe any ASNoC architecture. Finally, our methodology was evaluated through different case studies of real ASNoC benchmarks. From multi-objective perspective, on average

over all the benchmarks, our methodology was found to be at least 89.3% better than previous generation techniques. Moreover, our methodology generated the required ASNoC architecture at least 14 times faster than previous full-custom architecture optimization techniques. This work has been published in brief in [20, 21] and in full in [22].

7.2.4 Multi-objective Standard Architectures Optimization using GA

For our fourth contribution, we proposed a novel multi-objective optimization methodology for ASNoC standard architectures using GA. Integer chromosome representation was used to describe any ASNoC standard architecture. The methodology combined standard architecture selection and optimum core mapping of application IPs onto that architecture. Our methodology was further used to carry out application-specific evaluations of NoC standard architectures based on the optimum mapping. The methodology was then evaluated through different case studies of real ASNoC benchmarks. Results showed that our methodology selected the best standard architecture out of all architectures considered in this study. Moreover, from multi-objective perspective, on average over all architectures, the mapping generated by our methodology was found to be at least 24.3% better than previous mapping techniques. This work has been published in brief in [23, 24] and was submitted for publication in full in [25].

7.3 Directions for Future Work

Although the results of our methodologies are very promising, we cannot claim that the works in this dissertation will completely solve the ASNoC architecture customization and optimization problem. Accordingly, we believe that our research work can be extended along the following research directions.

7.3.1 Enhancement of NoC Evaluation Models

The models presented in this dissertation proved useful in evaluating different ASNoC metrics. However, they could still be enhanced by considering the following points:

- **Power Model**: Using our power model requires simulating the routers available in the designer library once before running any of the generation methodologies to find their areas and different power constants. This might be still a time consuming step. Therefore, modeling router power without the need for a pre-simulation step will help speed up the architecture generation process.

- **Reliability Model**: Our reliability model considered only the probability of the packets being affected by on-chip noise, which is currently the most important reliability degradation factor. However, on one side, including other sources of faults in ASNoC, like stuck-at-faults and process variation, will further enhance the model. On the other hand, we represented different sources of on-chip noise using Gaussian distribution. Nevertheless, back annotation of physical synthesis information into our system-level models could result in more accurate reliability calculation.

7.3.2 Integration of More Design Variables

The work presented in this dissertation considered most of the important design variables that dominate the ASNoC cost and performance. However, there are still more variables that could be included onto our methodologies. Example of these variables are the buffer size and the channel width. We believe that integrating these variables onto the work presented in this dissertation would help improving the ASNoC architecture generation process.

7.3.3 Dynamic Reconfiguration of ASNoC Architectures

In recent years, Autonomous Networks-on-Chip (ANoC) started to become a hot research area [170, 171]. The basic idea behind ANoCs is to dynamically reconfigure the underlying network according to the running time conditions. One of the main advantages of our methodologies is their low architecture generation time. Accordingly, they are very promising methodologies for ANoCs. Nonetheless, enhancing their execution time further would potentially make them more attractive for ANoCs. One approach to enhance the execution time of our methodologies is to try restoration techniques to ensure the legality of the generated chromosomes instead of the penalization ones. Restoration techniques try to reconstruct any architecture that fails the legality constraints instead of marking it as invalid. Therefore, they are expected to further speed up our methodologies.

7.3.4 A Tool for Unified Architecture Generation

Currently, each of our methodologies could separately be employed to generate the required ASNoC architecture. Moreover, for the semi-custom architecture generation methodology, various standalone tools are used to carry out the partitioning and the mapping steps. Therefore, we believe that combining all the methodologies in an integrated ASNoC architecture generation suite will be of great importance for ASNoC design and research communities.

7.3.5 Integration of System and Circuit-Level Methodologies

The increased complexity of modern SoC designs and the shorter time-to-market mandated the use of system-level design methodologies. System-level abstraction enabled designers to manage and to realize these complex systems within a reasonable time frame. However, circuit-level methodologies are indeed more accurate.

Therefore, for on-chip networks, integrating the system-level methodologies presented in this dissertation with circuit-level ones is expected to achieve the best compromise between productivity and accuracy for ASNoC-based systems.

Bibliography

[1] M. Sgroi, M. Sheets, A. Mihal, K. Keutzer, S. Malik, J. Rabaey, and A. Sangiovanni-Vincentelli, "Addressing the system-on-a-chip interconnect woes through communication-based design," in *Proceedings of the 38th Annual IEEE/ACM Design Automation Conference (DAC'01)*, Las Vegas, Nevada, USA, June 18–22 2001, pp. 667–672.

[2] L. Benini and G. D. Micheli, "Networks on chips: A new SoC paradigm," *IEEE Computer Magazine*, vol. 35, no. 1, pp. 70–78, Jan. 2002.

[3] A. Jantsch, "NoCs: A new contract between hardware and software," in *Proceedings of the 2003 Euromicro Symposium on Digital System Design (DSD'03)*, Belek-Antalya, Turkey, Sept. 1–6, 2003, pp. 10–16.

[4] H. Kalte, D. Langen, E. Vonnahme, A. Brinkmann, and U. Ruckert, "Dynamically reconfigurable system-on-programmable-chip," in *Proceedings of the tenth IEEE Euromicro Workshop on Parallel, Distributed and Network-based Processing (EUROMICRO-PDP'02)*, Canary Islands, Spain, Jan. 9–11, 2002, pp. 235–242.

[5] L. Benini, "Application specific NoC design," in *Proceedings of the 2006 IEEE/ACM Conference on Design, Automation and Test in Europe (DATE'06)*, vol. 1, Munich, Germany, Mar. 6–10, 2006, pp. 1–5.

[6] L. Benini and G. D. Micheli, *Networks on chips: Technology and tools*. San Francisco, CA, USA: Morgan Kaufmann, July 2006.

[7] U. Ogras, J. Hu, and R. Marculescu, "Key research problems in NoC design: A holistic perspective," in *Proceedings of the Third IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS'05)*, Jersey City, NJ, USA, Sept. 19–21, 2005, pp. 69–74.

[8] A. Ivanov and G. De Micheli, "Guest editors' introduction: The network-on-chip paradigm in practice and research," *IEEE Design and Test of Computers*, vol. 22, no. 5, pp. 399–403, Sept. 2005.

[9] J. Owens, W. Dally, R. Ho, D. Jayasimha, S. Keckler, and L.-S. Peh, "Research challenges for on-chip interconnection networks," *IEEE Micro*, vol. 27, no. 5, pp. 96–108, Sept.-Oct. 2007.

[10] J. Hu, U. Ogras, and R. Marculescu, "System-level buffer allocation for application-specific networks-on-chip router design," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 25, no. 12, pp. 2919–2933, Dec. 2006.

[11] H. Elmiligi, A. A. Morgan, M. W. El-Kharashi, and F. Gebali, "Power-aware topology optimization for networks-on-chips," in *Proceedings of the 2008 IEEE International Symposium on Circuits and Systems (ISCAS'08)*, Seattle, WA, USA, May 18–21, 2008, pp. 360–363.

[12] H. Elmiligi, A. A. Morgan, M. W. El-Kharashi, and F. Gebali, "Power optimization for application-specific networks-on-chips: A topology-based approach," *Journal of Microprocessors and Microsystems*, vol. 33, no. 5-6, pp. 343–355, Aug. 2009.

[13] H. Elmiligi, A. A. Morgan, M. W. El-Kharashi, and F. Gebali, "A reliability-aware design methodology for networks-on-chip applications," in *Proceedings of the fourth IEEE International Conference on Design and Technology of Integrated Systems in Nanoscale Era (DTIS'09)*, Cairo, Egypt, Apr. 6–9, 2009, pp. 107–112.

[14] H. Elmiligi, A. A. Morgan, M. W. El-Kharashi, and F. Gebali, "Improving networks-on-chip performability: A topology-based approach," *International Journal of Circuit Theory and Applications*, to appear.

[15] C. Fiduccia and R. Mattheyses, "A linear-time heuristic for improving network partitions," in *Proceedings of the 19th Annual IEEE/ACM Design Automation Conference (DAC'82)*, Las Vegas, NV, USA, June 14–18, 1982, pp. 175–181.

[16] A. A. Morgan, H. Elmiligi, M. W. El-Kharashi, and F. Gebali, "Application-specific networks-on-chip topology customization using network partitioning," in *Proceedings of the First International Forum on Next-Generation Multi-core/Manycore Technologies (IFMT'08)*, Cairo, Egypt, Nov. 24–25, 2008, pp. 1–6.

[17] A. A. Morgan, H. Elmiligi, M. W. El-Kharashi, and F. Gebali, "Networks-on-chip topology generation techniques: Area and delay evaluation," in *Proceedings of the third IEEE International Design and Test Workshop (IDT'08)*, Monastir, Tunisia, Dec. 20–22, 2008, pp. 33–38.

[18] A. A. Morgan, H. Elmiligi, M. W. El-Kharashi, and F. Gebali, "Area-aware topology generation for application-specific networks-on-chip using network partitioning," in *Proceedings of the 2009 IEEE Pacific Rim Conference on Communications, Computers and Signal Processing (PACRIM'09)*, Victoria, BC, Canada, Aug. 23–26, 2009, pp. 979–984.

[19] A. A. Morgan, H. Elmiligi, M. W. El-Kharashi, and F. Gebali, "Networks-on-chip architecture customization using network partitioning: A system-level performance evaluation," *Journal of Research and Practice in Information Technology*, submitted.

[20] A. A. Morgan, H. Elmiligi, M. W. El-Kharashi, and F. Gebali, "Area and delay optimization for networks-on-chip architectures using genetic algorithms," in *Proceedings of the fourth International Design and Test Workshop (IDT'09)*, Riyadh, Saudi Arabia, Nov. 15–17, 2009, pp. 1–6.

[21] A. A. Morgan, H. Elmiligi, M. W. El-Kharashi, and F. Gebali, "Multi-objective optimization for networks-on-chip architectures using genetic algorithms," in *Proceedings of the 2010 IEEE International Symposium on Circuits and Systems (ISCAS'10)*, Paris, France, May 30–June 2, 2010, pp. 3725–3728.

[22] A. A. Morgan, H. Elmiligi, M. W. El-Kharashi, and F. Gebali, "Bio-inspired NoC architecture optimization," in *Autonomic Networking-on-Chip: Bio-inspired Specification, Development, and Verification*, P. Cong-Vinh, Ed. CRC Press, 2011, to appear.

[23] A. A. Morgan, H. Elmiligi, M. W. El-Kharashi, and F. Gebali, "Multi-objective optimization of NoC standard architectures using genetic algorithms," in *Proceedings of the tenth IEEE International Symposium on Signal Processing and Information Technology (ISSPIT'10)*, Luxor, Egypt, Dec. 15–18, 2010, pp. 85–90.

[24] H. Elmiligi, A. A. Morgan, M. W. El-Kharashi, and F. Gebali, "Networks-on-chip topology optimization subject to power, delay, and reliability constraints," in *Proceedings of the 2010 IEEE International Symposium on Circuits and Systems (ISCAS'10)*, Paris, France, May 30 – June 2, 2010, pp. 2354–2357.

[25] A. A. Morgan, H. Elmiligi, M. W. El-Kharashi, and F. Gebali, "A unified multi-objective mapping and architecture customization of networks-on-chip," *IET Computers and Digital Techniques*, submitted.

[26] R. Ho, K. Mai, and M. Horowitz, "The future of wires," *Proceedings of the IEEE*, vol. 89, no. 4, pp. 490–504, Jan. 2001.

[27] W. Dally and C. Seitz, "The torus routing chip," *Distributed Computing*, vol. 1, no. 4, pp. 187–196, Dec. 1986.

[28] L. Benini and D. Bertozzi, "Network-on-chip architectures and design methods," *IEE Proceedings on Computers and Digital Techniques*, vol. 152, no. 2, pp. 261–272, Mar. 2005.

[29] K. Goossens, J. Dielissen, and A. Radulescu, "Æthereal network on chip: Concepts, architectures, and implementations," *IEEE Design and Test of Computers*, vol. 22, no. 5, pp. 21–31, Sept. 2005.

[30] W.-D. Weber, "A quality-of-service mechanism for interconnection networks in system-on-chips," in *Proceedings of the 2005 IEEE/ACM Conference on Design, Automation and Test in Europe (DATE'05)*, vol. 2, Munich, Germany, Mar. 7–11, 2005, pp. 1232–1237.

[31] T. Bjerregaard, "The MANGO clockless network-on-chip: Concepts and implementation," Ph. D. Thesis, Informatics and Mathematical Modelling, Technical University of Denmark (DTU), Feb. 2006, [Online], Available: http://www2.imm.dtu.dk/pubdb/p.php?4025, Last accessed: Dec. 2010.

[32] D. Siguenza-Tortosa, T. Ahonen, and J. Nurmi, "Issues in the development of a practical NoC: The proteo concept," *Journal of Integration, the VLSI, Elsevier*, vol. 38, no. 1, pp. 95–105, Oct. 2004.

[33] M. Millberg, E. Nilsson, R. Thid, and A. Jantsch, "Guaranteed bandwidth using looped containers in temporally disjoint networks within the nostrum network on chip," in *Proceedings of the 2004 IEEE/ACM Conference on Design, Automation and Test in Europe (DATE'04)*, vol. 2, Paris, France, Feb. 16–20, 2004, pp. 890–895.

[34] M. Dall'Osso, G. Biccari, L. Giovannini, D. Bertozzi, and L. Benini, "Xpipes: A latency insensitive parameterized network-on-chip architecture for multi-processor SoCs," in *Proceedings of 21st IEEE/ACM International Conference on Computer Design (ICCD'03)*, San Jose, CA, USA, Oct. 13–15, 2003, pp. 536–539.

[35] A. Andriahantenaina and A. Greiner, "Micro-network for SoC: Implementation of a 32-port spin network," in *Proceedings of the 2003 IEEE/ACM Conference on Design, Automation and Test in Europe (DATE'03)*, Munich, Germany, Mar. 3–7, 2003, pp. 1128–1129.

[36] J. Bainbridge and S. Furber, "Chain: A delay-insensitive chip area interconnect," *IEEE Micro*, vol. 22, no. 5, pp. 16–23, Sept. 2002.

[37] U. Ogras and R. Marculescu, "An architecture and compiler for scalable on-chip communication," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 12, no. 7, pp. 711–726, July 2004.

[38] J. Liang, S. Swaminathan, and R. Tessier, "aSOC: A scalable, single-chip communications architecture," in *Proceedings of the 2000 International Conference on Parallel Architectures and Compilation Techniques (PACT'00)*, Philadelphia, PA, USA, Oct. 15–19, 2000, pp. 37–46.

[39] T. Bjerregaard and K. Mahadevan, "A survey of research and practices of network-on-chip," *ACM Computing Surveys*, vol. 38, no. 1, pp. 1–51, Mar. 2006.

[40] D. Bertozzi, S. Kumar, and M. Palesi, "Networks-on-chip: Emerging research topics and novel ideas," *Hindawi Publishing Corporation, VLSI Design*, vol. 2007, no. 26454, pp. 1–3, 2007.

[41] R. Marculescu, U. Ogras, L.-S. Peh, N. Jerger, and Y. Hoskote, "Outstanding research problems in NoC design: System, microarchitecture, and circuit perspectives," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 28, no. 1, pp. 3–21, Jan. 2009.

[42] A.-M. Rahmani, K. Latif, P. Liljeberg, J. Plosila, and H. Tenhunen, "Research and practices on 3D networks-on-chip architectures," in *Proceedings of the 28th Nordic Microelectronics conference (NORCHIP'10)*, Tampere, Finland, Nov. 15–16, 2010, pp. 1–6.

[43] S. Bhat, "Energy models for networks-on-chip components," Master Thesis, Department of Mathematics and Computer Science, Technische Universiteit Eindhoven, Dec. 2005, [Online], Available: http://www.es.ele.tue.nl/epicurus/files/report_sbhat.pdf, Last accessed: Dec. 2010.

[44] J. Palma, L. Indrusiak, F. Moraes, R. Reis, and M. Glesner, "Reducing the power consumption in networks-on-chip through data coding schemes," in *Proceedings of the 14th IEEE International Conference on Electronics, Circuits and Systems (ICECS '07)*, Marrakech, Morocco, Dec. 11–14, 2007, pp. 1007–1010.

[45] P.-T. Huang, W.-L. Fang, Y.-L. Wang, and W. Hwang, "Low power and reliable interconnection with self-corrected green coding scheme for network-on-chip," in *Proceedings of the second IEEE/ACM International Symposium on Networks-on-Chip (NOCS'08)*, Newcastle, UK, Apr. 7–10, 2008, pp. 77–83.

[46] K. Lee, S.-J. Lee, and H.-J. Yoo, "Low-power network-on-chip for high-performance SoC design," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 14, no. 2, pp. 148–160, Feb. 2006.

[47] A. Banerjee, R. Mullins, and S. Moore, "A power and energy exploration of network-on-chip architectures," in *Proceedings of the First IEEE International Symposium on Networks-on-Chip (NOCS'07)*, Princeton, NJ, USA, May 7–9, 2007, pp. 163–172.

[48] Y. Hu, H. Chen, Y. Zhu, A. Chien, and C.-K. Cheng, "Physical synthesis of energy-efficient networks-on-chip through topology exploration and wire style optimization," in *Proceedings of the 23rd IEEE/ACM International Conference on Computer Design: VLSI in Computers and Processors (ICCD'05)*, San Jose, CA, USA, Oct. 2–5, 2005, pp. 111–118.

[49] M. Kim, D. Kim, and G. Sobelman, "Network-on-chip link analysis under power and performance constraints," in *Proceedings of the 2006 IEEE International Symposium on Circuits and Systems (ISCAS'06)*, Island of Kos, Greece, May 21–24, 2006, pp. 4163–4166.

[50] P. Meloni, I. Loi, F. Angiolini, S. Carta, M. Barbaro, L. Raffo, and L. Benini, "Area and power modeling for networks-on-chip with layout awareness," *Hindawi Publishing Corporation, VLSI Design*, vol. 2007, no. 50285, pp. 1–12, 2007.

[51] A. Kiasari, D. Rahmati, H. Sarbazi-Azad, and S. Hessabi, "A markovian performance model for networks-on-chip," in *Proceedings of the 16th IEEE Euromicro Conference on Parallel, Distributed and Network-Based Processing (PDP'08)*, Toulouse, France, Feb. 13–15, 2008, pp. 157–164.

[52] A. Bona, V. Zaccaria, and R. Zafalon, "System level power modeling and simulation of high-end industrial network-on-chip," in *Proceedings of the 2004 IEEE/ACM Conference on Design, Automation and Test in Europe (DATE'04)*, vol. 3, Paris, France, Feb. 16–20, 2004, pp. 318–323.

[53] J. Xu, W. Wolf, J. Henkel, and S. Chakradhar, "A methodology for design, modeling, and analysis of networks-on-chip," in *Proceedings of the 2005 IEEE International Symposium on Circuits and Systems (ISCAS'05)*, vol. 2, Kobe, Japan, May 23–26, 2005, pp. 1778–1781.

[54] P. Wolkotte, G. Smit, N. Kavaldjiev, J. Becker, and J. Becker, "Energy model of networks-on-chip and a bus," in *Proceedings of the 2005 International Symposium on System-on-Chip (SoC'05)*, Tampere, Finland, Nov. 17, 2005, pp. 82–85.

[55] T. T. Ye, G. D. Micheli, and L. Benini, "Analysis of power consumption on switch fabrics in network routers," in *Proceedings of the 39th Annual IEEE/ACM Design Automation Conference (DAC'02)*, New Orleans, LA, USA, June 10–14, 2002, pp. 524–529.

[56] J. Kim, D. Park, C. Nicopoulos, N. Vijaykrishnan, and C. Das, "Design and analysis of an NoC architecture from performance, reliability and energy perspective," in *Proceedings of the 2005 ACM/IEEE Symposium on Architectures for Networking and Communications Systems (ANCS'05)*, Princeton, NJ, USA, Oct. 26–28, 2005, pp. 173–182.

[57] J. Hu and R. Marculescu, "Energy-aware mapping for tile-based NoC architectures under performance constraints," in *Proceedings of the eighth IEEE Asia and South Pacific Design Automation Conference (ASP-DAC'03)*, Kitakyushu, Japan, Jan. 21–24, 2003, pp. 233–239.

[58] C. Marcon, N. Calazans, F. Moraes, A. Susin, I. Reis, and F. Hessel, "Exploring NoC mapping strategies: An energy and timing aware technique," in *Proceedings of the 2005 IEEE/ACM Conference on Design, Automation and Test in Europe (DATE'05)*, vol. 1, Munich, Germany, Mar. 7–11, 2005, pp. 502–507.

[59] M. Arjomand and H. Sarbazi-Azad, "Power-performance analysis of networks-on-chip with arbitrary buffer allocation schemes," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 29, no. 10, pp. 1558–1571, Oct. 2010.

[60] E. Mensink, D. Schinkel, E. Klumperink, E. Tuijl, and B. Nauta, "Power efficient gigabit communication over capacitively driven RC-limited on-chip interconnects," *IEEE Journal of Solid-State Circuits*, vol. 45, no. 2, pp. 447–457, Feb. 2010.

[61] A. Asad, A. Zonouz, M. Seyrafi, M. Soryani, and M. Fathy, "Modeling and analyzing of blocking time effects on power consumption in network-on-chips," in *Proceedings of the 2009 International Conference on Reconfigurable Computing and FPGAs (ReConFig'08)*, Quintana Roo, Mexico, Dec. 9–11, 2009, pp. 290–295.

[62] A. Kahng, B. Li, L.-S. Peh, and K. Samadi, "ORION 2.0: A fast and accurate NoC power and area model for early-stage design space exploration,"

in *Proceedings of the 2009 IEEE/ACM Conference on Design, Automation and Test in Europe (DATE'09)*, Nice, France, Apr. 20–24, 2009, pp. 423–428.

[63] A. Kahng, B. Lin, and K. Samadi, "Improved on-chip router analytical power and area modeling," in *Proceedings of the 15th IEEE Asia and South Pacific Design Automation Conference (ASP-DAC'10)*, Taipei, Taiwan, Jan. 18–21, 2010, pp. 241–246.

[64] K. Jeong, A. Kahng, B. Lin, and K. Samadi, "Accurate machine-learning-based on-chip router modeling," *IEEE Embedded Systems Letters*, vol. 2, no. 3, pp. 62–66, Sept. 2010.

[65] S. Li, J. Ahn, R. Strong, J. Brockman, D. Tullsen, and N. Jouppi, "McPAT: An integrated power, area, and timing modeling framework for multicore and manycore architectures," in *Proceedings of the 42nd annual ACM/IEEE international symposium on Microarchitecture (MICRO'09)*, New York, NY,USA, Dec. 12–16, 2009, pp. 469–480.

[66] L. Ost, G. Guindani, L. Indrusiak, S. Maatta, and F. Moraes, "Using abstract power estimation models for design space exploration in NoC-based MPSoC," *IEEE Design and Test of Computers*, to appear.

[67] E. Bolotin, I. Cidon, R. Ginosar, and A. Kolodny, "Cost considerations in network on chip," *Integration, the VLSI Journal, Special issue: Networks on chip and reconfigurable fabrics*, vol. 38, no. 1, pp. 19–42, Oct. 2004.

[68] F. Moraes, N. Calazans, A. deMello, L. Mller, and L. Ost, "HERMES: An infrastructure for low area overhead packet-switching networks on chip," *Integration, the VLSI Journal*, vol. 38, no. 1, pp. 69–93, Oct. 2004.

[69] J. Balfour and W. Dally, "Design trade-offs for tiled CMP on-chip networks," in *Proceedings of the 20th IEEE/ACM Annual International Conference on Supercomputing (ICS'06)*, Queensland, Australia, Jun. 28 - Jul. 1, 2006, pp. 187–198.

[70] P. Pande, C. Grecu, M. Jones, A. Ivanov, and R. Saleh, "Performance evaluation and design trade-offs for network-on-chip interconnect architectures," *IEEE Transactions on Computers*, vol. 54, no. 8, pp. 1025–1040, Aug. 2005.

[71] B. Feero and P. Pande, "Networks-on-chip in a three-dimensional environment: A performance evaluation," *IEEE Transactions on Computers*, vol. 58, no. 1, pp. 32–45, Jan. 2009.

[72] L. Carloni, A. Kahng, S. Muddu, A. Pinto, K. Samadi, and P. Sharma, "Accurate predictive interconnect modeling for system-level design," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 18, no. 4, pp. 679–684, Apr. 2010.

[73] W. Zhou, Y. Zhang, and Z. Mao, "An application specific NoC mapping for optimized delay," in *Proceedings of the first IEEE International Conference on Design and Test of Integrated Systems in Nanoscale Technology (DTIS'06)*, Tunis, Tunisia, Sept. 5–7, 2006, pp. 184–188.

[74] N. Nikitin and J. Cortadella, "A performance analytical model for network-on-chip with constant service time routers," in *Proceedings of the 2009 IEEE/ACM International Conference on Computer-Aided Design (ICCAD'09) - Digest of Technical Papers*, San Jose, CA, USA, Nov. 2–5, 2009, pp. 571–578.

[75] U. Ogras and R. Marculescu, "It's a small world after all: NoC performance optimization via long-range link insertion," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 14, no. 7, pp. 693–706, July 2006.

[76] S. Yan and B. Lin, "Design of application-specific 3D networks-on-chip architectures," in *Proceedings of the 26th IEEE/ACM International Conference on Computer Design (ICCD'08)*, Lake Tahoe, CA, USA, Oct. 12–15, 2008, pp. 142–149.

[77] M. Bakhouya, S. Suboh, J. Gaber, and T. El-Ghazawi, "Analytical modeling and evaluation of on-chip interconnects using network calculus," in *Proceedings of the Third IEEE International Symposium on Networks-on-Chip (NOCS'09)*, San Diego, CA, USA, May 10–13, 2009, pp. 74–79.

[78] A. Kohler and M. Radetzki, "A SystemC TLM2 model of communication in wormhole switched networks-on-chip," in *Proceedings of the 2009 IEEE Forum on Specification and Design Languages (FDL'09)*, Sophia Antipolis, France, Sept. 22–24, 2009, pp. 1–4.

[79] Y. Lu, S. Sezer, and J. McCanny, "TLM2.0 based timing accurate modeling method for complex NoC systems," in *Proceedings of the 2010 IEEE International Symposium on Circuits and Systems (ISCAS'10)*, Paris, France, May 30 – June 2, 2010, pp. 2900–2903.

[80] V. Dumitriu and G. Khan, "Throughput-oriented NoC topology generation and analysis for high performance SoCs," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 17, no. 10, pp. 1433–1446, Oct. 2009.

[81] V. Pavlidis and E. Friedman, "3-D topologies for networks-on-chip," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 15, no. 10, pp. 1081–1090, Oct. 2007.

[82] A. Varma and C. Eaghavendra, *Interconnection Networks for Multiprocessors and Multicomputers Teory and Practice*. IEEE Computer Society Press, 1991.

[83] S. Murali, L. Benini, and G. D. Micheli, "Mapping and physical planning of networks-on-chip architectures with quality-of-service guarantees," in *Proceedings of the tenth IEEE Asia and South Pacific Design Automation Conference (ASP-DAC'05)*, vol. 1, Shanghai, China, Jan. 18–21, 2005, pp. 27–32.

[84] A. Dalirsani, M. Hosseinabady, and Z. Navabi, "An analytical model for reliability evaluation of NoC architectures," in *Proceedings of the 13th IEEE International On-Line Testing Symposium (IOLTS'07)*, Crete, Greece, July 8–11, 2007, pp. 49–56.

[85] A. Alaghi, N. Karimi, M. Sedghi, and Z. Navabi, "Online NoC switch fault detection and diagnosis using a high level fault model," in *Proceedings of the 22nd IEEE International Symposium on Defect and Fault-Tolerance in VLSI Systems (DFT '07)*, Rome, Italy, Sept. 26–28, 2007, pp. 21–29.

[86] M. Valinataj, S. Mohammadi, and S. Safari, "Inherent reliability evaluation of networks-on-chip based on analytical models," in *Proceedings of the 2008 International Symposium on System-on-Chip (SoC'08)*, Tampere, Finland, Nov. 4–6, 2008, pp. 1–4.

[87] F. Worm, P. Ienne, P. Thiran, and G. D. Micheli, "A robust self-calibrating transmission scheme for on-chip networks," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 13, no. 1, pp. 126–139, Jan. 2005.

[88] D. Bertozzi, L. Benini, and G. D. Micheli, "Error control schemes for on-chip communication links: The energy-reliability tradeoff," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 24, no. 6, pp. 818–831, June 2005.

[89] A. Ejlali, B. Al-Hashimi, P. Rosinger, and S. Miremadi, "Joint consideration of fault-tolerance, energy-efficiency and performance in on-chip networks," in *Proceedings of the 2007 IEEE/ACM Conference on Design, Automation and Test in Europe (DATE'07)*, Nice, France, Apr. 16–20, 2007, pp. 1647–1652.

[90] T. Lehtonen, P. Liljeberg, and J. Plosila, "Fault tolerant distributed routing algorithms for mesh networks-on-chip," in *Proceedings of the 2009 International Symposium on Signals, Circuits, and Systems (ISSCS'09)*, Iasi, Romania, 9–10, 2009, pp. 1–4.

[91] Y. Zou and S. Pasricha, "NARCO: Neighbor aware turn model-based fault tolerant routing for NoCs," *IEEE Embedded Systems Letters*, vol. 2, no. 3, pp. 85–89, Sept. 2010.

[92] A. Kohler, G. Schley, and M. Radetzki, "Fault tolerant network on chip switching with graceful performance degradation," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 29, no. 6, pp. 883–896, June 2010.

[93] P. Teehan, G. Lemieux, and M. Greenstreet, "Estimating reliability and throughput of source-synchronous wave-pipelined interconnect," in *Proceedings of the Third IEEE International Symposium on Networks-on-Chip (NOCS'09)*, San Diego, CA, USA, May 10–13, 2009, pp. 234–243.

[94] Q. Yu and P. Ampadu, "A flexible parallel simulator for networks-on-chip with error control," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 29, no. 1, pp. 103–116, Jan. 2010.

[95] I. Loi, F. Angiolini, S. Fujita, S. Mitra, and L. Benini, "Characterization and implementation of fault-tolerant vertical links for 3-D networks-on-chip," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 30, no. 1, pp. 124–134, Jan. 2011.

[96] F. Li, Y. Xiao, and Q. Zhang, "Graph partition based traffic balance in industrial network," in *Proceedings of the second IEEE International Conference on Communications and Networking in China (CHINACOM'07)*, Shanghai, China, Aug. 22–24, 2007, pp. 438–442.

[97] G. Karypis and V. Kumar, "Multilevel algorithms for multi-constraint graph partitioning," in *Proceedings of the 1998 IEEE/ACM Conference on Supercomputing (SC'98)*, San Jose, CA , USA, Nov. 7–13 1998, pp. 1–17.

[98] C. Cheng and Y. Wei, "An improved two-way partitioning algorithm with stable performance," *IEEE Transactions on Computer-Aided Design*, vol. 10, no. 12, pp. 1502–1511, Dec. 1991.

[99] L. Sanchis, "Multiple-way network partitioning," *IEEE Transactions on Computers*, vol. 38, no. 1, pp. 62–81, Jan. 1989.

[100] L. Sanchis, "Multiple-way network partitioning with different cost functions," *IEEE Transactions on Computers*, vol. 42, no. 12, pp. 1500–1504, Dec. 1993.

[101] B. Chamberlain, "Graph partitioning algorithms for distributing workloads of parallel computations," University of Washington, WA, USA, Tech. Rep. UW-CSE-98-10-03, Oct. 1998.

[102] Y. Saab and V. Rao, "An evolution-based approach to partitioning ASIC systems," in *Proceedings of the 26th Annual IEEE/ACM Design Automation Conference (DAC'89)*, Las Vegas, NV, USA, June 25–29, 1989, pp. 767–770.

[103] Y. Wan, S. Roy, A. Saberi, and B. Lesieutre, "A stochastic automaton-based algorithm for flexible and distributed network partitioning," in *Proceedings of the 2005 IEEE Swarm Intelligence Symposium (SIS'05)*, California, USA, June 8–10, 2005, pp. 273–280.

[104] H. Simon, "Partitioning of unstructured problems for parallel processing," *Computing Systems in Engineering*, vol. 2, no. 2/3, pp. 135–148, 1991.

[105] P. Chan, M. Schlag, and J. Zien, "Spectral k-way ratio-cut partitioning and clustering," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 13, no. 9, pp. 1088–1095, Sept. 1994.

[106] B. Hendrickson and R. Leland, "The Chaco user's guide: Version 2," Sandia National Laboratories, Albuquerque, NM, USA, Tech. Rep. SAND95-2344, July 1995.

[107] B. Kernighan and S. Lin, "An efficient heuristic procedure for partitioning graphs," *Bell System Technical Journal*, vol. 49, no. 2, pp. 291–307, Feb. 1970.

[108] B. Krishnamurthy, "An improved min-cut algorithm for partitioning VLSI networks," *IEEE Transactions on Computers*, vol. 33, no. 5, pp. 438–446, May 1984.

[109] G. Karypis, K. Schloegel, and V. Kumar, "PARMETIS: Parallel gragh partitioning and sparse matrix ordering library, Version 3.1," University of Minnesota, Minneapolis, MN 55455, USA, Tech. Rep., Aug. 2003.

[110] T. Ahonen, D. Siguenza-Tortosa, H. Bin, and J. Nurmi, "Topology optimization for application-specific networks-on-chip," in *Proceedings of the 2004 ACM international workshop on System level interconnect prediction (SLIP'04)*, Paris, France, Feb. 14–15, 2004, pp. 53–60.

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

[112] A.-M. Rahmani, P. Liljeberg, J. Plosila, and H. Tenhunen, "Developing reconfigurable FIFOs to optimize power/performance of voltage/frequency island-based networks-on-chip," in *Proceedings of the 13th IEEE International Symposium on Design and Diagnostics of Electronic Circuits and Systems (DDECS'10)*, Vienna, Austria, Apr. 14–16, 2010, pp. 105–110.

[113] C. Seiculescu, S. Murali, L. Benini, and G. D. Micheli, "Comparative analysis of NoCs for two-dimensional versus three-dimensional SoCs supporting multiple voltage and frequency islands," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 57, no. 5, pp. 364–368, May 2010.

[114] L. Wang, Y. Jin, H. Kim, and E. Kim, "Recursive partitioning multicast: A bandwidth-efficient routing for networks-on-chip," in *Proceedings of the Third IEEE International Symposium on Networks-on-Chip (NOCS'09)*, San Diego, CA, USA, May 10–13, 2009, pp. 64–73.

[115] L. Wang, P. Kumar, R. Boyapati, K. Yum, and E. Kim, "Efficient lookahead routing and header compression for multicasting in networks-on-chip," in *Proceedings of the 2010 ACM/IEEE Symposium on Architectures for Networking*

and Communications Systems (ANCS'10), La Jolla, CA, USA, Oct. 25–26, 2010, pp. 1–10.

[116] M. Ebrahimi, M. Daneshtalab, P. Liljeberg, and H. Tenhunen, "Partitioning methods for unicast/multicast traffic in 3D NoC architecture," in *Proceedings of the 13th IEEE International Symposium on Design and Diagnostics of Electronic Circuits and Systems (DDECS'10)*, Vienna, Austria, Apr. 14–16, 2010, pp. 127–132.

[117] K.-L. Tsai, F. Lai, D.-S. X. C.-Y. Pan and, H.-J. Tan, and H.-C. Lee, "Design of low latency on-chip communication based on hybrid NoC architecture," in *Proceedings of the 8th IEEE International Northeast Workshop on Circuits and Systems (NEWCAS'10)*, Montreal, QC, Canada, June 20–23, 2010, pp. 257–260.

[118] B. Yu, S. Dong, S. Chen, and S. Goto, "Floorplanning and topology generation for application-specific network-on-chip," in *Proceedings of the 15th IEEE Asia and South Pacific Design Automation Conference (ASP-DAC'10)*, Taipei, Taiwan, Jan. 18–21, 2010, pp. 535–540.

[119] S. Murali, C. Seiculescu, L. Benini, and G. D. Micheli, "Synthesis of networks on chips for 3D systems on chips," in *Proceedings of the 14th IEEE Asia and South Pacific Design Automation Conference (ASP-DAC'09)*, Yokohama, Japan, Jan. 19–22, 2009, pp. 242–247.

[120] C. Seiculescu, S. Murali, L. Benini, and G. D. Micheli, "SunFloor 3D: A tool for networks on chip topology synthesis for 3D systems on chips," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 29, no. 12, pp. 1987–2000, Dec. 2010.

[121] M. Ebrahimi, M. Daneshtalab, P. Liljeberg, and H. Tenhunen, "Performance analysis of 3D NoCs partitioning methods," in *Proceedings of the 2010 IEEE Computer Society Annual Symposium on VLSI (ISVLSI'10)*, Lixouri, Kefalonia, Greece, July 5–7, 2010, pp. 479–480.

[122] D. Majeti, A. Pasalapudi, and K. Yalamanchili, "Low energy tree based network on chip architectures using homogeneous routers for bandwidth and latency constrained multimedia applications," in *Proceedings of the Second International Conference on Emerging Trends in Engineering and Technology (ICETET'09)*, Nagpur, India, Dec. 16–18, 2009, pp. 358–363.

[123] P. Ezhumali and C. Arun, "Low power and area consumption custom networks-on-chip architectures using RST algorithms," *International Journal of Computer Science and Information Security (IJCSIS)*, vol. 8, no. 6, pp. 107–115, Sept. 2010.

[124] D. E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley Professional, Jan. 1989.

[125] J. R. Koza, *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. MIT Press, Dec. 1992.

[126] Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs*. Springer, Mar. 1996.

[127] J. Kennedy and R. C. Eberhart, "Particle swarm optimization," in *Proceeding of the 1995 IEEE International Conference on Neural Networks (ICNN95)*, Perth, Australia, Dec. 1995, pp. 12–13.

[128] E. Bonabeau, M. Dorigo, and G. Theraulaz, *Swarm Intelligence: From Natural to Artificial Systems*. Oxford University Press, Aug. 1999.

[129] S. He, Q. H. Wu, and J. R. Saunders, "A novel group search optimizer inspired by animal behavioural ecology," in *Proceeding of the 2006 IEEE Congress on Evolutionary Computation (CEC'2006)*, Vancouver, BC, Canada, July 16–21, 2006, pp. 1272–1278.

[130] W. J. Tang and Q. H. Wu, "Biologically inspired optimization: A review," *Transactions of the Institute of Measurement and Control*, vol. 31, no. 6, pp. 495–515, 2009.

[131] A. Younis and Z. Dong, "Trends, features, and tests of common and recently introduced global optimization methods," *Engineering Optimization*, vol. 42, no. 8, pp. 691–718, Aug. 2010.

[132] N. Koziris, M. Romesis, P. Tsanakas, and G. Papakonstantinou, "An efficient algorithm for the physical mapping of clustered taskgraphs onto multiprocessor architectures," in *Proceedings of the Eighth IEEE Euromicro Workshop on Parallel and Distributed Processing (EURO-PDP'2000)*, Rhodos, Greece, Jan. 19–21, 2000, pp. 406–413.

[133] S. Murali and G. D. Micheli, "Bandwidth-constrained mapping of cores onto NoC architectures," in *Proceedings of the 2004 IEEE/ACM Conference on Design, Automation and Test in Europe (DATE'04)*, vol. 2, Paris, France, Feb. 16–20, 2004, pp. 896–901.

[134] W. Shen, C. Chao, Y. Lien, and A. Wu, "A new binomial mapping and optimization algorithm for reduced-complexity mesh-based on-chip network," in *Proceedings of the First IEEE International Symposium on Networks-on-Chip (NOCS'07)*, Princeton, NJ, USA, May 7–9, 2007, pp. 317–322.

[135] G. Ascia, V. Catania, and M. Palesi, "Mapping cores on network-on-chip," *International Journal of Computational Intelligence Research*, vol. 1, no. 2, pp. 109–126, Dec. 2005.

[136] Z. Lu, L. Xia, and A. Jantsch, "Cluster-based simulated annealing for mapping cores onto 2D mesh networks on chip," in *Proceedings of the 11th IEEE International Workshop on Design and Diagnostics of Electronics Circuits and Systems (DDECS'08)*, Bratislava, Slovakia, Apr. 16–18, 2008, pp. 1–6.

[137] W. Zhou, Y. Zhang, and Z. Mao, "Pareto based multi-objective mapping IP cores onto NoC architectures," in *Proceedings of the 2006 IEEE Asia Pacific Conference on Circuits and Systems (APCCAS'06)*, Singapore, Singapore, Dec. 4–7, 2006, pp. 331–334.

[138] W. Hu, C. Du, L. Yan, and C. Tianzhou, "A fast algorithm for energy-aware mapping of cores onto WK-recursive NoC under performance constraints," in *Proceedings of the 16th Annual International Conference on High Performance Computing (HiPC'09)*, Kochi, India, Dec. 16–19, 2009, pp. 359–367.

[139] S. Saeidi, F. Vardi, and A. Khademzadeh, "Application mapping scenarios onto network on chip based priority lists," in *Proceedings of the Second International Conference on Computer and Electrical Engineering (ICCEE'09)*, Dubai, United Arab Emirates, Dec. 28–30, 2009, pp. 545–549.

[140] J. Qi, H. Zhao, J. Wang, and Z. Li, "A new hierarchical genetic algorithm for low-power network on chip design," in *Proceedings of the International Conference on Intelligent Control and Information Processing (ICICIP'10)*, Dalian, China, Aug. 13–15, 2010, pp. 159–162.

[141] Y.-L. Jeang, T.-S. Wey, H.-Y. Wang, and C.-W. Hung, "Mesh-tree architecture for network-on-chip design," in *Proceedings of the Second International Conference on Innovative Computing, Information and Control (ICICIC'07)*, Kumamoto, Japan, Sept. 5–7, 2007, pp. 262–265.

[142] H. Matsutani, M. Koibuchi, D. Hsu, and H. Amano, "Three-dimensional layout of on-chip tree-based networks," in *Proceedings of the International Symposium on Parallel Architectures, Algorithms, and Networks (I-SPAN'08)*, Sydney, Australia, May 7–9, 2008, pp. 281–288.

[143] S. Murali and G. D. Micheli, "SUNMAP: A tool for automatic topology selection and generation for NoCs," in *Proceedings of the 41st IEEE/ACM Design Automation Conference (DAC'04)*, San Diego, CA, USA, June 7–11, 2004, pp. 914–919.

[144] L. Bononi and N. Concer, "Simulation and analysis of network on chip architectures: Ring, spidergon and 2D mesh," in *Proceedings of the 2006 IEEE/ACM Conference on Design, Automation and Test in Europe (DATE'06)*, vol. 2, Munich, Germany, Mar. 6–10, 2006, pp. 154–159.

[145] M. Kreutz, C. Marcon, L. Carro, N. Calazans, and A. Susin, "Energy and latency evaluation of NoC topologies," in *Proceedings of the 2005 IEEE International Symposium on Circuits and Systems (ISCAS'05)*, vol. 6, Kobe, Japan, May 23–26, 2005, pp. 5866–5869.

[146] G. Palermo, G. Mariani, C. Silvano, R. Locatelli, and M. Coppola, "Mapping and topology customization approaches for application-specific STNoC designs," in *Proceedings of the 18th IEEE International Conference on Application-specific Systems, Architectures and Processors (ASAP'07)*, Montreal, QC, Canada, July 9–11, 2007, pp. 61–68.

[147] D. Wang, H. Matsutani, H. Amano, and M. Koibuchi, "A link removal methodology for networks-on-chip on reconfigurable systems," in *Proceedings of International Conference on Field Programmable Logic and Application (FPL'08)*, Heidelberg, Germany, Sept. 8–10, 2008, pp. 269–274.

[148] K.-C. Chang and T.-F. Chen, "Low-power algorithm for automatic topology generation for application-specific networks on chips," *IET Computers and Digital Techniques*, vol. 2, no. 3, pp. 239–249, May 2008.

[149] J. Kim, M. Lee, W. Kim, J. Chang, Y. Bae, and H. Cho, "Performance analysis of NoC structure based on star-mesh topology," in *Proceedings of 2008 International SoC Design Conference (ISOCC'08)*, vol. 2, Busan, Korea, Nov. 24–25, 2008, pp. 162–165.

[150] K. Srinivasan and K. Chatha, "A low complexity heuristic for design of custom network-on-chip architectures," in *Proceedings of the 2006 IEEE/ACM Conference on Design, Automation and Test in Europe (DATE'06)*, vol. 1, Munich, Germany, Mar. 6–10, 2006, pp. 1–6.

[151] S. Lin, L. Su, H. Su, D. Jin, and L. Zeng, "Hierarchical cluster-based irregular topology customization for networks-on-chip," in *Proceedings of the fifth IEEE/IFIP International Conference on Embedded and Ubiquitous Computing (EUC'08)*, vol. 1, Shanghai, China, Dec. 17–20, 2008, pp. 373–377.

[152] K. Srinivasan, K. Chatha, and G. Konjevod, "Linear-programming-based techniques for synthesis of network-on-chip architectures," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 14, no. 4, pp. 407–420, Apr. 2006.

[153] K. Srinivasan and K. Chatha, "ISIS: A genetic algorithm based technique for custom on-chip interconnection network synthesis," in *Proceedings of the 18th IEEE International Conference on VLSI Design (VLSID'05)*, Kolkata, India, Jan. 3–7, 2005, pp. 623–628.

[154] G. Leary, K. Srinivasan, K. Mehta, and K. Chatha, "Design of network-on-chip architectures with a genetic algorithm-based technique," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 17, no. 5, pp. 674–687, May 2009.

[155] H. Elmiligi, A. A. Morgan, M. W. El-Kharashi, and F. Gebali, "A topology-based design methodology for networks-on-chip applications," in *Proceedings of the second International Design and Test Workshop (IDT'07)*, Cairo, Egypt, Dec. 16–18, 2007, pp. 61–65.

[156] W. Dally and B. Towles, *Principles and Practices of Interconnection Networks*. San Francisco, CA, USA: Morgan Kaufmann Publishers, 2004.

[157] H. Elmiligi, A. A. Morgan, M. W. El-Kharashi, and F. Gebali, "Performance analysis of networks-on-chip routers," in *Proceedings of the second International Design and Test Workshop (IDT'07)*, Cairo, Egypt, Dec. 16–18, 2007, pp. 232–263.

[158] H. Elmiligi, A. A. Morgan, M. W. El-Kharashi, and F. Gebali, "A delay-aware topology-based design for networks-on-chip applications," in *Proceedings of the fourth International Design and Test Workshop (IDT'09)*, Riyadh, Saudi Arabia, Nov. 15–17, 2009, pp. 1–5.

[159] K. Lee, S.-J. Lee, D. Kim, K. Kim, G. Kim, J. Kim, and H.-J. Yoo, "Networks-on-chip and networks-in-package for high-performance SoC platforms," in *Pro-*

ceedings of the first IEEE Asian Solid-State Circuits Conference (A-SSCC'05), Hsinchu, Taiwan, Nov. 1–3, 2005, pp. 485–488.

[160] G. Agnarsson and R. Greenlaw, *Graph Theory: Modeling, Applications, and Algorithms*. Old Tappan, NJ, USA: Prentice Hall, 2006.

[161] E. L. Witzke and S. D. Frese, "Some limitations of adjacency matrices in computer network analysis," *ACM SIGCOMM Computer Communication Review*, vol. 18, no. 5, pp. 43–47, Oct. 1988.

[162] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische Mathematik*, vol. 1, no. 1, pp. 269–271, Dec. 1959.

[163] Predictive Technology Model (PTM), Arizona State University, Available: http://ptm.asu.edu/, Last accessed: Dec. 2010.

[164] J. Proakis, *Digital Communications*, 4th ed. McGraw-Hill Inc., Aug. 2000.

[165] M. Coenen, S. Murali, A. Radulescu, K. Goossens, and G. D. Micheli, "A buffer-sizing algorithm for networks on chip using TDMA and credit-based end-to-end flow control," in *Proceedings of the Third IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS'06)*, Seoul, Korea, Oct. 22–25, 2006, pp. 130–135.

[166] H.-J. Yoo, K. Lee, and J. Kim, *Low-power NoC for high-performance SoC design*. San Francisco, CA, USA: CRC Press, July 2008.

[167] MATLAB® R2007a, "Genetic algorithm and direct search toolbox 2.1," The MathWorks Inc., http://www.mathworks.com/, Natick, MA, USA, 2007.

[168] B. G. Liptak, *Instrument Engineers' Handbook: Process control and optimization*, 4th ed. CRC Press, Sept. 2005.

[169] P. Carlone, G. Palazzo, and R. Pasquino, "Pultrusion manufacturing process development: Cure optimization by hybrid computational methods," *International Journal of Computers and Mathematics with Applications, Elsevier*, vol. 53, no. 9, pp. 1464–1471, Apr. 2007.

[170] H. Tenhunen, "Autonomous NoC (aNOC): A co-operative project between KTH and University of Turku," [Online], http://www.mpsoc-forum.org/2006/slides/Tenhunen.pdf, 2006, Last accessed: Dec. 2010.

[171] A. Yin, P. Liljeberg, Z. Lu, and H. Tenhunen, "Monitoring agent based autonomous reconfigurable network-on-chip," in *Proceedings of the Workshop Digest on Diagnostic Services in Network-on-Chips (DSNoC) in the 45th Design Automation Conference (DAC'08)*, Anaheim, CA, USA, June 8-13, 2008, pp. 1–2.

Appendix A

List of Publications

A.1 Book Chapters

1. **A. A. Morgan**, H. Elmiligi, M. W. El-Kharashi, and F. Gebali, "Bio-inspired NoC architecture optimization," in *Autonomic Networking-on-Chip: Bio-inspired Specification, Development, and Verification*, P. Cong-Vinh, Ed. CRC Press, 2011, to appear.

A.2 Journals

1. **A. A. Morgan**, H. Elmiligi, M. W. El-Kharashi, and F. Gebali,"A Unified Multi-Objective Mapping and Architecture Customization of Networks-on-Chip," *IET Computers and Digital Techniques*, submitted.

2. **A. A. Morgan**, H. Elmiligi, M. W. El-Kharashi, and F. Gebali,"Networks-on-Chip Architecture Customization using Network Partitioning: A System-Level Performance Evaluation," *Journal of Research and Practice in Information Technology*, submitted.

3. H. Elmiligi, **A. A. Morgan**, M. W. El-Kharashi, and F. Gebali,"Improving networks-on-chip performability: A topology-based approach," *International Journal of Circuit Theory and Applications*, to appear.

4. H. Elmiligi, **A. A. Morgan**, M. W. El-Kharashi, and F. Gebali, "Power optimization for application-specific networks-on-chips: A topology-based approach," *Journal of Microprocessors and Microsystems*, vol. 33, no. 5–6, pp. 343–355, Aug. 2009.

A.3 Conferences and Workshops

1. **A. A. Morgan**, H. Elmiligi, M. W. El-Kharashi, and F. Gebali, "Multi-objective optimization of NoC standard architectures using genetic algorithms," in *Proceedings of the tenth IEEE International Symposium on Signal Processing and Information Technology (ISSPIT'10)*, Luxor, Egypt, Dec. 15–18, 2010, pp. 85–90.

2. **A. A. Morgan**, H. Elmiligi, M. W. El-Kharashi, and F. Gebali, "Multi-objective optimization for networks-on-chip architectures using genetic algorithms," in *Proceedings of the 2010 IEEE International Symposium on Circuits and Systems (ISCAS'10)*, Paris, France, May 30–June 2, 2010, pp. 3725–3728.

3. **A. A. Morgan**, H. Elmiligi, M. W. El-Kharashi, and F. Gebali, "Area and delay optimization for networks-on-chip architectures using genetic algorithms," in *Proceedings of the fourth International Design and Test Workshop (IDT'09)*, Riyadh, Saudi Arabia, Nov. 15–17, 2009, pp. 1–6.

4. **A. A. Morgan**, H. Elmiligi, M. W. El-Kharashi, and F. Gebali, "Area-aware topology generation for application-specific networks-on-chip using network

partitioning," in *Proceedings of the 2009 IEEE Pacific Rim Conference on Communications, Computers and Signal Processing (PACRIM'09)*, Victoria, BC, Canada, Aug. 23–26, 2009, pp. 979–984.

5. **A. A. Morgan**, H. Elmiligi, M. W. El-Kharashi, and F. Gebali, "Networks-on-chip topology generation techniques: Area and delay evaluation," in *Proceedings of the third IEEE International Design and Test Workshop (IDT'08)*, Monastir, Tunisia, Dec. 20–22, 2008, pp. 33–38.

6. **A. A. Morgan**, H. Elmiligi, M. W. El-Kharashi, and F. Gebali, "Application-specific networks-on-chip topology customization using network partitioning," in *Proceedings of the First International Forum on Next-Generation Multi-core/Manycore Technologies (IFMT'08)*, Cairo, Egypt, Nov. 24–25, 2008, pp. 1–6.

7. H. Elmiligi, **A. A. Morgan**, M.W. El-Kharashi, and F. Gebali, "Networks-on-chip topology optimization subject to power, delay, and reliability constraints," in *Proceedings of the 2010 IEEE International Symposium on Circuits and Systems (ISCAS'10)*, Paris, France, May 30-June 2, 2010, pp. 2354–2357.

8. H. Elmiligi, **A. A. Morgan**, M. W. El-Kharashi, and F. Gebali, "A delay-aware topology-based design for networks-on-chip applications," in *Proceedings of the fourth International Design and Test Workshop (IDT'09)*, Riyadh, Saudi Arabia, Nov. 15–17, 2009, pp. 1–5.

9. H. Elmiligi, **A. A. Morgan**, M. W. El-Kharashi, and F. Gebali, "A reliability-aware design methodology for networks-on-chip applications," in *Proceedings of the fourth IEEE International Conference on Design and Technology of Integrated Systems in Nanoscale Era (DTIS'09)*, Cairo, Egypt, Apr. 6–9, 2009, pp. 107–112.

10. H. Elmiligi, **A. A. Morgan**, M. W. El-Kharashi, and F. Gebali, "Power-aware topology optimization for networks-on-chips," in *Proceedings of the 2008 IEEE International Symposium on Circuits and Systems (ISCAS'08)*, Seattle, WA, USA, May 18–21, 2008, pp. 360–363.

11. H. Elmiligi, **A. A. Morgan**, M. W. El-Kharashi, and F. Gebali, "A topology-based design methodology for networks-on-chip applications," in *Proceedings of the second International Design and Test Workshop (IDT'07)*, Cairo, Egypt, Dec. 16–18, 2007, pp. 61–65.

12. H. Elmiligi, **A. A. Morgan**, M. W. El-Kharashi, and F. Gebali, "Performance analysis of networks-on-chip routers," in *Proceedings of the second International Design and Test Workshop (IDT'07)*, Cairo, Egypt, Dec. 16–18, 2007, pp. 232–263.