OCIN_TSIM – A DVFS AWARE SIMULATOR FOR NOC DESIGN SPACE EXPLORATION AND OPTIMIZATION

A Thesis

by

SUBODH PRABHU

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

May 2010

Major Subject: Computer Engineering

OCIN_TSIM – A DVFS AWARE SIMULATOR FOR NOC DESIGN SPACE EXPLORATION AND OPTIMIZATION

A Thesis

by

SUBODH PRABHU

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

May 2010

Major Subject: Computer Engineering

ABSTRACT

Ocin_tsim – A DVFS Aware Simulator for NoC Design Space Exploration and Optimization. (May 2010)

Subodh Prabhu, B.E., Delhi University

Co-Chairs of Advisory Committee: Dr. Jiang Hu
Dr. Paul V. Gratz

Networks-on-Chip (NoCs) are a general purpose, scalable replacement for shared medium wired interconnects offering many practical applications in industry. Dynamic Voltage Frequency Scaling (DVFS) is a technique whereby a chip's voltage-frequency levels are varied at run time, often used to conserve dynamic power. Various DVFS-based NoC optimization techniques have been proposed. However, due to the resources required to validate architectural decisions through prototyping, few are implemented. As a result, designers are faced with a lack of insight into potential power savings or performance gains at early architecture stages.

This thesis proposes a DVFS aware NoC simulator with support for per node power-frequency modeling to allow fine-tuning of such optimization techniques early on in the design cycle. The proposed simulator also provides a framework for benchmarking various candidate strategies to allow selective prototyping and optimization.

As part of the research, DVFS extensions were built for an existing NoC performance simulator and released for public use. This thesis presents some of the preliminary results from our simulator that show the average power consumed per node for all the benchmarks in SPLASH 2 benchmark suite [74] to be quite similar to each other. This thesis also serves as a technical manual for the simulator extensions. Important links for downloading and using the simulator are provided at the end of this document in Appendix C.

DEDICATION

This thesis is dedicated to my father, who taught me the most important lesson of how to live this life with discipline and principles. It is also dedicated to my mother, who gave me my sense of perseverance, as well as encouraged me to pursue education. Without them, I would not have been me.

This thesis is also dedicated to my lovely wife-to-be, my siblings and all other family members for their unstinted support. It is also dedicated to all my friends and support group for being there when I needed them, and for nurturing in me the ideas that extend well beyond this thesis.

ACKNOWLEDGEMENTS

I would like to thank my co-advisors and committee co-chairs, Dr. Jiang Hu and Dr. Paul V. Gratz, for their constant support and guidance in my research. Dr. Hu was instrumental in getting me started on an initial study, and with his expert guidance, paved the way for my thesis topic. Dr. Gratz, with his expertise in NoC, helped me in quickly ramping up on this topic and always kept me focused on the right step to carry my work forward. In Dr. Hu and Dr. Gratz, I found my ideal advisors. Without their expert advice and numerous insightful comments over the past year, it would have been virtually impossible to finish this body of work. As I proceed in my career, I hope to take with me, teachings and experiences accumulated from working with them.

I would also like to acknowledge my other committee member, Dr. Vivek Sarin, for his support throughout the course of this research. Special thanks to Dr. Boris Grot, who along with Dr. Gratz, had built the initial Ocin_tsim simulator and who gave several insightful comments as the co-author of the initial paper on Ocin_tsim.

Thanks are also due to Dr. Li-Shiuan Peh and her student Ms. Bin Li from Princeton University for providing the Orion modeling framework for development of our simulator.

Thanks also go to all my friends, colleagues and CAMSIN group members for their encouragement and intellectual discussions which helped me shape this work.

Finally, thanks to my department faculty and staff, external faculty and campus employers for making my time at Texas A&M University a great experience.

NOMENCLATURE

CAMSIN	Computer Architecture, Memory Systems and Interconnect Networks
CMP	Chip Multi-processor
DFS	Dynamic Frequency Scaling
DVFS	Dynamic Voltage Frequency Scaling
FIFO	First In First Out
FPGA	Field Programmable Gate Array
GD	GD Graphics Library
HW	Hardware
I/O	Input/Output
IP	Intellectual Property
NoC	Network on Chip
Ocin_tsim	On Chip Interconnect Network Timing Simulator
OCR	On Chip Regulator
SoC	System on Chip
SVN	Subversion
SW	Software
VF	Voltage Frequency
VFI	Voltage Frequency Island

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER I

INTRODUCTION

Today's dominant approach to scaling compute performance while mitigating wire delays and power consumption is through the use of single chip multi-processors (CMPs). Current commercial products in this family range from Intel's 8-core superscalar CMP [1] to a 64-core network and media processor from Tilera [2] to a 256-element reconfigurable processor array from Rapport, Inc. [3]. With technology scaling, CMPs with hundreds to thousands of general and special-purpose cores are likely to appear in a multitude of application domains in the near future. Similarly, system-on-chip (SoC) designs have recently emerged in the embedded market as a means to reduce power consumption and costs by integrating a diverse set of many IP blocks onto a single chip. In both cases, with rising core and IP block counts, communication between cores has become a major challenge as bus-based and ad-hoc interconnects do not scale well [4, 5]. In response, researchers have proposed packet-based networks-on-chip (NoCs) as a structured and scalable alternative [5, 6]. For instance, Intel recently announced Tera-flop performance at 5 GHz using an 80-core substrate with a mesh interconnect [7, 8]. As NoCs are still an emerging field, it has enjoyed a considerable amount of recent research activity.

This thesis follows the style of *IEEE Transactions on Computers*.

1.1 Network on Chip (NoC)

One of the major problems in future SOC designs arises from non-scalable delays in global wires [9]. Global wires carry signals across a chip and typically do not scale in length with technology scaling [10]. Though gate delays scale down with technology, global wire delays typically increase exponentially or linearly by inserting repeaters. Even after repeater insertion [11], the delay may exceed the limit of one or multiple clock cycles. In ultra-deep submicron processes, it is claimed that 80 percent or more of the delay of critical paths will be due to interconnects [12]. As a result, many large designs use as hoc FIFO buffers to synchronously propagate data over large distances to overcome this problem. According to the ITRS report, "Global synchronization becomes prohibitively costly due to process variability and power dissipation, and cross-chip signaling can no longer be achieved in a single clock cycle." [13]. Thus, system design must work on networking and distributed computation paradigms with functional blocks integrated into the communication backbone.

The most frequently used on-chip interconnect architecture is an arbitrated bus, where all communication devices share the same transmission medium. Advantages of such shared-bus architectures are simple topology, low area and extensibility. However, for a long bus line, the intrinsic parasitic resistance and capacitance can be quite high. Moreover, every additional IP block added to the bus adds to this parasitic capacitance, in turn increasing propagation delay. As the bus length and/or the number of IP blocks increases, the associated bit transfer delay over the bus becomes large and will eventually exceeds the targeted clock period. This places a limit on the number of IP blocks that can be connected to a bus and thereby limits the system scalability [14].

One solution to deal with this problem is to split the bus into multiple segments and employ a hierarchical architecture [15]; however, this is ad hoc in nature and has the inherent limitations of the bus-based architecture. In SoCs consisting of several IP blocks, bus-based interconnects will face serious bandwidth problems as all attached devices must share the same medium [14]. To overcome the above-mentioned problems, use of a communication-centric approach to integrate IPs in complex SoCs is advocated. This new model separates the resource elements (i.e., the IPs) from the communication infrastructure (i.e., the network). The need for global synchronization thus disappears. This new approach is explicitly parallel, exhibits modularity to minimize global wires and utilizes locality in power minimization [16].

In a network-centric approach, communication between IPs happens in the form of packets. A common characteristic of such architectures is that the IP blocks communicate with each other using intelligent switches or routers. As such, these switches dubbed infrastructure IPs (I2Ps) [16] provide a robust data transfer medium for the functional IP modules.

There is another manner of explaining the relevance of Network on Chips [17]. Reliable communication between circuit components requires a protocol definition that provides some rules describing how the interaction shall take place. These rules ensure that the overall system performance requirements are met, while physical resources like area or energy are minimized. Traditional on-chip communication designs use ad-hoc approaches that often fail to meet some strict scalability requirement of next-generation SOC designs. Bottlenecks can arise in performance, throughput, power, energy, reliability, synchronization, predictability and concurrency Designers traditionally stuck to point-to-point connections and bus-based techniques. This approach is acceptable for a small number of blocks when the performance/ latency trade-off is relatively simple.

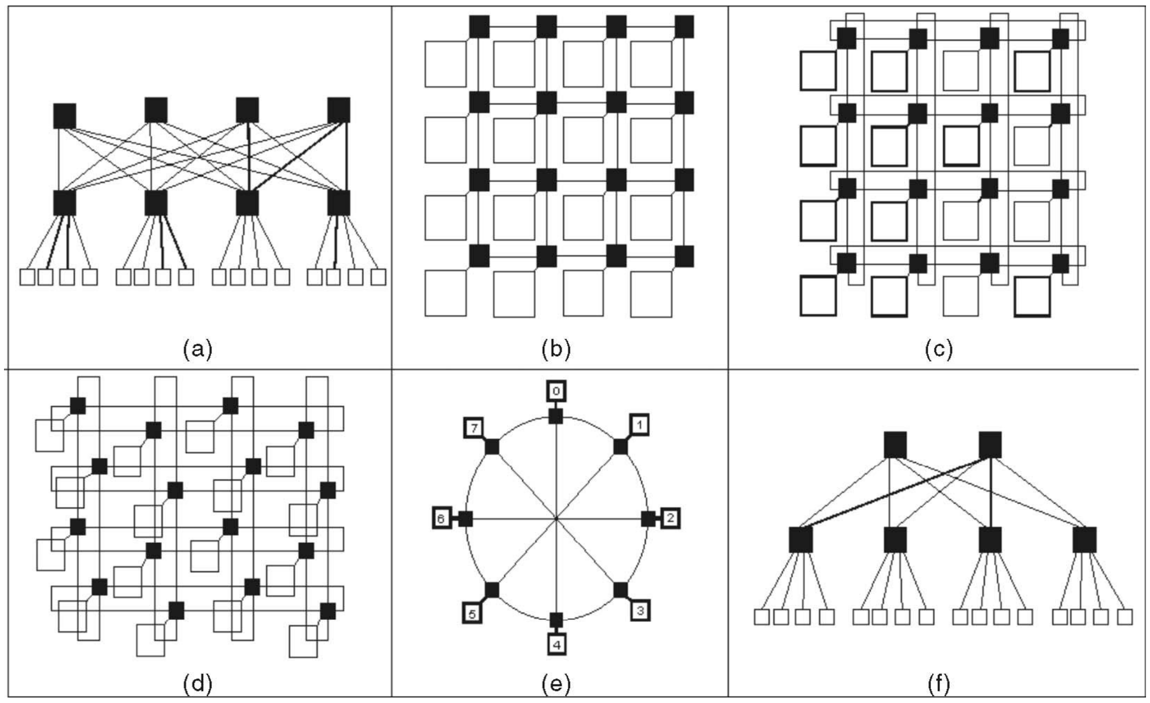


Fig. 1: Popular NoC topologies [9] – (a) SPIN, (b) Mesh 4x4, (c) Torus 4x4, (d) Folded Torus 4x4, (e) Octagon and (f) Butterfly Fat Tree (BFT)

To address these challenges, it was critical to take a global view of the communication problem. Communication design was introduced at higher levels of abstraction than the architecture and RTL level. Communication protocol functions were separated into layers that interact through via well-defined interfaces to simplify synthesis and validation tasks. Such an approach also maximizes re-use like for example in 802.11 wireless network standards where a single media-access layer supports different physical implementations through unified interface. This layered-stack approach to the design of the on-chip inter-core communications is called the Network-on-Chip (NOC) methodology [18]. Fig. 1 shows some of the popular topologies employed in an NoC architecture [9]. Designing NOCs is not an easy task, and may result in protocol implementations that are incorrect (e.g. due to deadlocks and race conditions), or sub-optimal (e.g. are power hungry or introduce unacceptable latency).

1.2 Dynamic Voltage Frequency Scaling (DVFS)

One active area of work on NoC has focused on dynamically varying operating voltage and frequency levels to achieve a balance between power and performance [19]. This technique, referred to as DVFS, is used quite often in SoC designs [20]. Dynamic voltage and frequency scaling (DVFS) was introduced in the 90's [21] to dramatically reduce power consumption in large digital systems by varying both voltage and frequency of the system with respect to changing workloads [22, 23, 24, 25]. Fig. 2 shows the time varying pattern of voltage and frequency in a system exhibiting DVFS [25].

Alternative techniques using voltage/frequency islands (VFIs) for IP blocks are used in achieving fine grain system-level power management [26]. Use of VFIs in the NoC context can provide better power-performance tradeoffs than single voltage, single clock frequency case as it benefits from the natural partitioning and mapping of applications onto the NoC platform. Despite the huge potential for energy savings with

VFIs, the NoC design methodologies considered so far are limited to a single voltage-clock domain [27, 28, 29]. Studies that do consider multiple VFIs assume that each module/core in the design belongs to a different island and different islands are connected by point-to-point (P2P) links [30, 31].

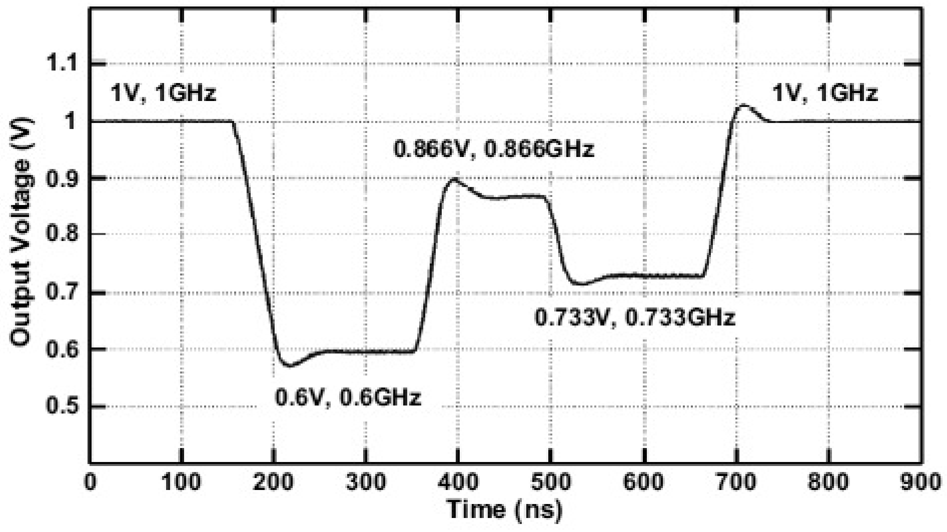


Fig. 2: Graph showing VF variations in a DVFS system [25]

Power-gating is a standby-leakage reduction method developed in [32, 33, 34, 35, 36]. In a power gating design, sleep transistors are used as switches to shut off power supplies to parts of a design in standby mode [37]. Clock gating was also proposed as a power saving technique [38, 39, 40]. Some studies indicate that the clock signals in digital computers consume a large (15–45%) percentage of the system power [41]. Thus, the circuit power can be greatly reduced by reducing the clock power dissipation. Many clock power reduction techniques have focused on reduced voltage swings, buffer insertion, and clock routing [42]. In many cases, switching of the clock causes a huge gate activity. In circuits with controllable clocks, master clock is used to derive all other clocks which, based on certain conditions, can be slowed down or stopped completely with respect to the master clock [43].

W. Kim, et al. have proposed per-core DVFS techniques for use in fine-grain power management [25]. In recent years, a huge interest has developed in building on chip integrated switching voltage regulators [44, 45, 46, 47]. Fig. 3 shows some of the popular configurations in which on-chip regulators may be employed (if at all) in a DVFS system. These regulators allow multiple on-chip power domains in CMP systems. CMP systems running heterogeneous workloads can benefit from per-core DVFS.

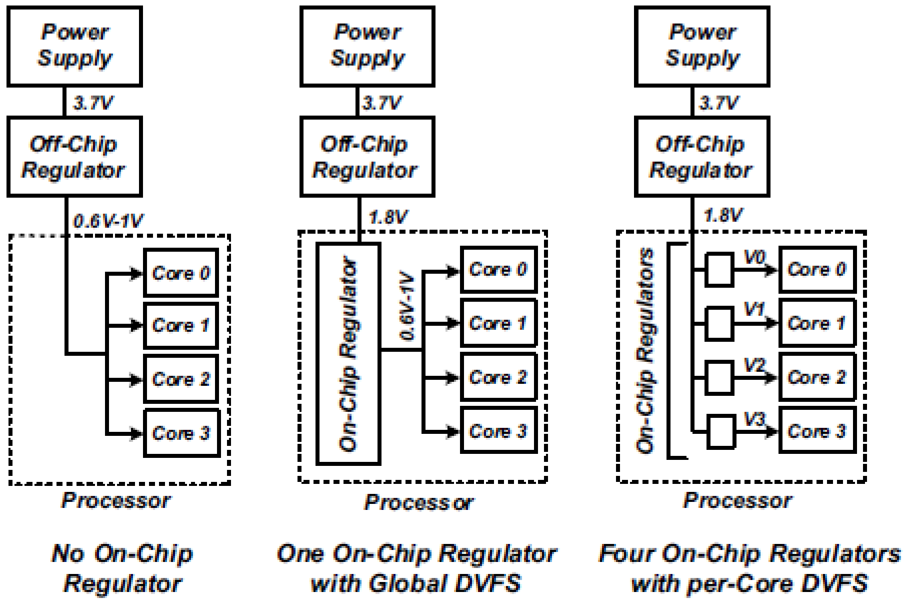


Fig. 3: Popular OCR configurations [25]

1.3 Motivation

In traditional NoC-based designs, power savings estimation and verification is delayed until design prototyping via FPGA or silicon implementation [48]. As indicated in Fig. 4, a large number of design candidates may exist in the planning stages. Once established, power, performance, and area objectives will subsequently reduce the possible design space. At this point, evaluation via simulation can provide a rapid, quantitative basis for refining a limited number of design candidates into a few good prototype choices. It also enables exploring a wider design space in cases where system objectives are relatively flexible. With this motivation behind us, we framed our thesis statement as “To develop a framework that would allow early estimation of power and performance characteristics as well as benchmarking of an NoC based design proposition”.

This document introduces extensions to “Ocin_tsim” (On Chip Interconnection Network Timing Simulator), a cycle-accurate microarchitectural Network-On-Chip (NoC) simulator. These extensions to Ocin_tsim support per-node DVFS modeling for early identification of optimum power-performance savings possible given a proposed DVFS policy.

Existing NoC simulators as we shall see later in this document, either do not model DVFS or assume global DVFS [49, 50, 51, 52, 53]. As a result, many architectural optimizations can be validated only for global performance benefits. To the best of our knowledge, this simulator is the first to accurately model the power and performance of an NoC utilizing per-core DVFS.

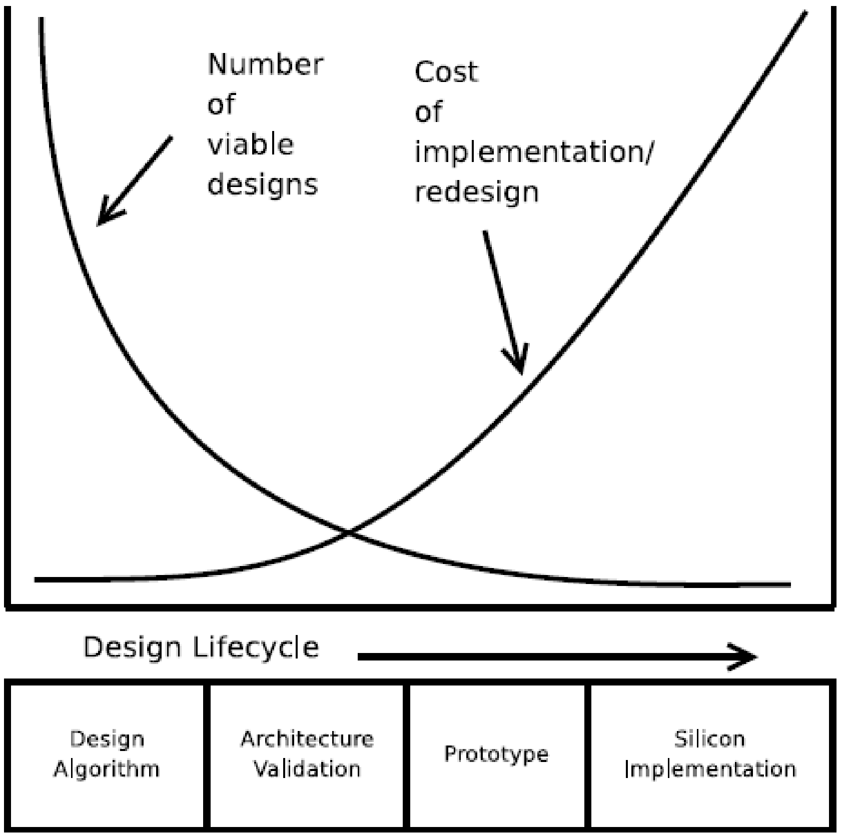


Fig. 4: NoC design trends from architecture to Silicon implementation

This work on the Ocin_tsim simulator makes the following contributions:

1) A novel DVFS-aware NoC simulation environment with power-performance calculations for each node, necessary for modeling multiple voltage-frequency regions dynamically managed at run-time.

2) Support for modular integration of power models allowing users to plug-in their own power models for new process technologies and custom circuit designs.

3) Support for modeling and benchmarking of arbitrary interconnect topologies and router microarchitectures with power-performance calculations.

1.4 Organization

The rest of the thesis is organized as follows. Chapter II discusses prior relevant work, Chapter III discusses Ocin_tsim as it existed before and various decisions made in its design process. In Chapter IV we present the implementation of DFS and power modeling extensions to Ocin_tsim, while Chapter V covers potential use cases of our simulator for some standard designs. Chapter VI presents summary of conclusions and future work. Finally we present some detailed simulator information in Appendices A-D.

CHAPTER II

RELATED WORK

2.1 Performance Modeling

In this chapter we present some of the existing NoC simulation tools for performance and power calculations. We start off by identifying a basis for classifying these simulators. This also serves the purpose in showing how our simulator is different from existing simulators.

2.1.1 Classification

Existing literature landscape in NoC architectural simulators at architectural stage can be classified into the following broad classes on the basis of the parameter that is being simulated:

2.1.1.1 Class A: Micro-architectural Simulators

These simulators compute actual physical quantities of an NoC such as network latencies [49, 50, 51], power consumption [50, 51, 52], and noise immunity [54]. While being exhaustive, they are computationally intense and limited in precision by models employed.

2.1.1.2 Class B: Abstract Simulators

Class B simulators model only a specific section/layer of network such as task scheduling, packet arbitration, packet management and/or packet routing and generalize the results for complete interconnect network. [53]. Abstract simulators provide high simulation performance and can be effective at proving the underlying optimization without the complexities of an actual NoC implementation. These simulators rely upon generalized assumptions about the characteristics of network traffic and often prove inaccurate at estimating the performance of realistic workloads.

2.1.1.3 Placement of Ocin_tsim

The previous division is by no means exhaustive. Our simulator lies squarely in Class A and hence is different from all Class B simulators in that it directly models network latencies and dynamically estimates the power consumption for a given NoC configuration. Our simulator is different from other simulators in Class A in three respects:

1) It allows each node to operate in independent voltage and frequency domains as in a per-core DVFS setup,

2) It uses Orion system of power modeling [55] to compute both static and dynamic power consumption and

3) It provides a textual as well as graphic representation of dynamic network traffic to allow an intuitive logging and visualization.

2.1.2 State of Art in NoC Simulation

The following presents a list of state of the art in NoC simulation presented in no particular order.

2.1.2.1 Transaction level Simulator

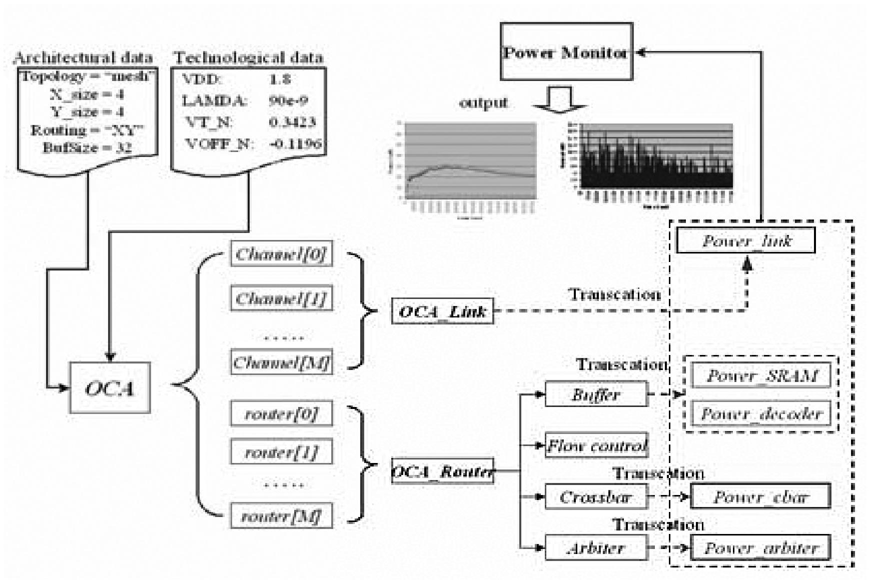


Fig. 5: System flow of transaction-level NoC simulator [52]

A transaction-level model written in SystemC is used for fast simulation speed [52]. Fig. 5 shows the system level view of transaction-level NoC simulator. An architectural energy model estimates communication energy for both dynamic and leakage, dissipating on routers and links through the transaction-level simulation. It supports temporal power profiling for each NoC component and spatial power snapshots for the whole NoC, making it easy to inspect the power implications under application workloads.

2.1.2.2 NoCSim

NoCSim is an extensible and popular simulator for Networks on Chip [50]. It is written in Haskell with the ForSyDe library. It is extensible because it allows the user to define new switches, resources and resource-switch interfaces. The simulator is controlled by scripts based on Haskell commands and functions which are then compiled with the rest of the simulator. A variety of simulation commands produce simulation output data files. Plot commands are used to visualize simulation data with gnuplot.

2.1.2.3 Nostrum NoC Simulation Environment (NNSE)

NNSE allows analyzing performance impact of NoC configuration parameters [51, 56]. It also allows one to configure topology, flow control and routing algorithm etc. and various regular and application specific traffic patterns to evaluate the network in terms of latency and throughput. The simulator is built in SystemC and is reconfigurable so that it is possible to try different NoC platforms with different workload mappings.

2.1.2.4 FPGA Based Emulator

A flexible emulation environment suitable for exploring, evaluating and comparing a wide range of NoC solutions with a very limited effort was implemented on an FPGA [57]. A HW-SW NoC emulation framework was proposed for achieving this (See Fig. 6).

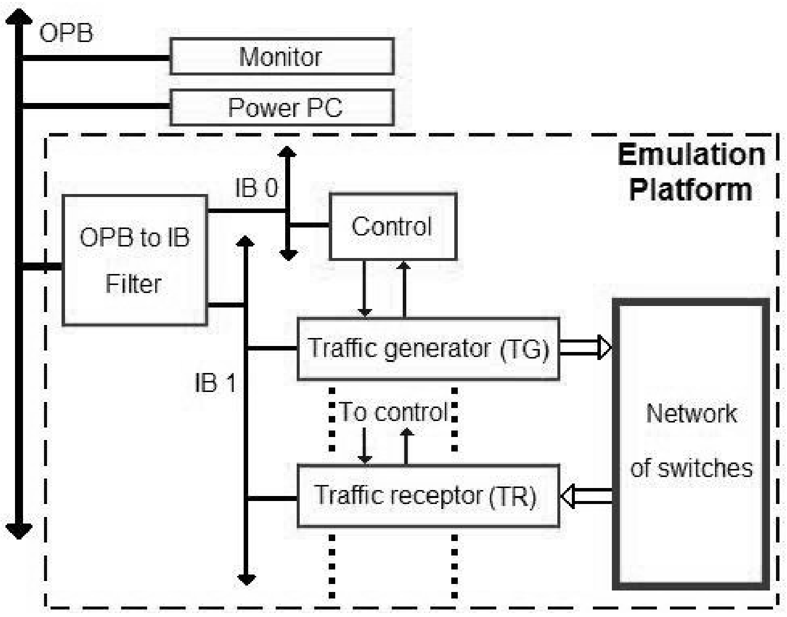


Fig. 6: FPGA based NoC emulation framework [57]

2.1.2.5 GARNET

GARNET (See Fig. 7) is a detailed network model [58, 59] incorporated inside a full-system simulator GEMS framework [60] which allows system-level performance and power modeling of network-level techniques. GARNET also can evaluate techniques that get maximum benefit out of memory hierarchy and the interconnection network. Using GEMS framework, interconnection network evaluations can be done in a full-system mode. GARNET can also interface with power models such as Orion or Wattch but has no support for per-node DVFS. It provides an evaluation of techniques that use the interconnection network as well as other system-level components simultaneously.

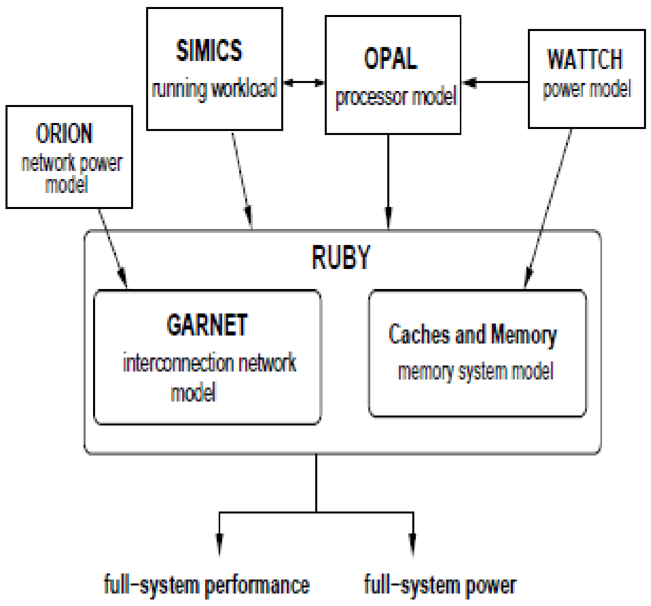


Fig. 7: Overview of GEMS (with GARNET) [58]

2.1.2.6 Network Simulator (NS)

NS is a public domain network simulator which can be used to build a proto-model that can evaluate design options for a specific NOC architecture with 2D mesh of switches [49]. NS is an event driven simulator.

2.1.2.7 Proteo NoC Simulator

A VHDL-based simulation environment for PROTEO NoC was proposed [53]. VHDL was utilized to evaluate several features of virtual channels in mesh-based and hierarchical NoC topologies. The accuracy of VHDL model is high, but it suffers from low simulation speed.

2.1.2.8 VHDL Model

N. Banerjee et al. present a VHDL based cycle accurate register transfer level model for evaluating latency, throughput, dynamic, and leakage power consumption of NoC based architectures [61]. An RTL design is parameterized on (i) size of packets, (ii) length and width of physical links, (iii) number, and depth of virtual channels, and (iv) switching technique.

2.1.2.9 Mapping Tool

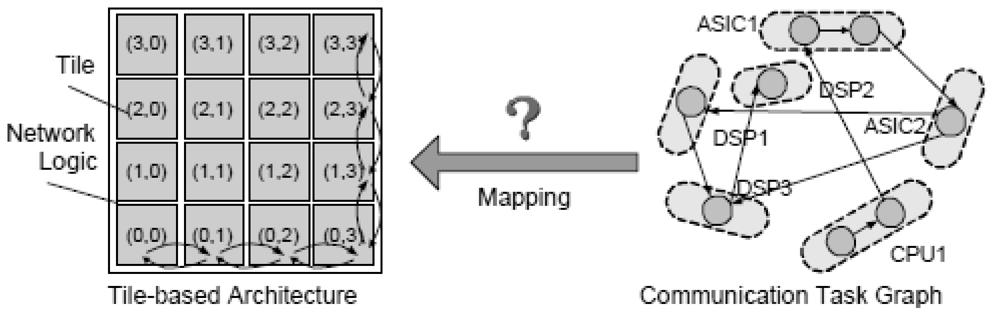


Fig. 8: Mapping problem in a tile based architecture [62]

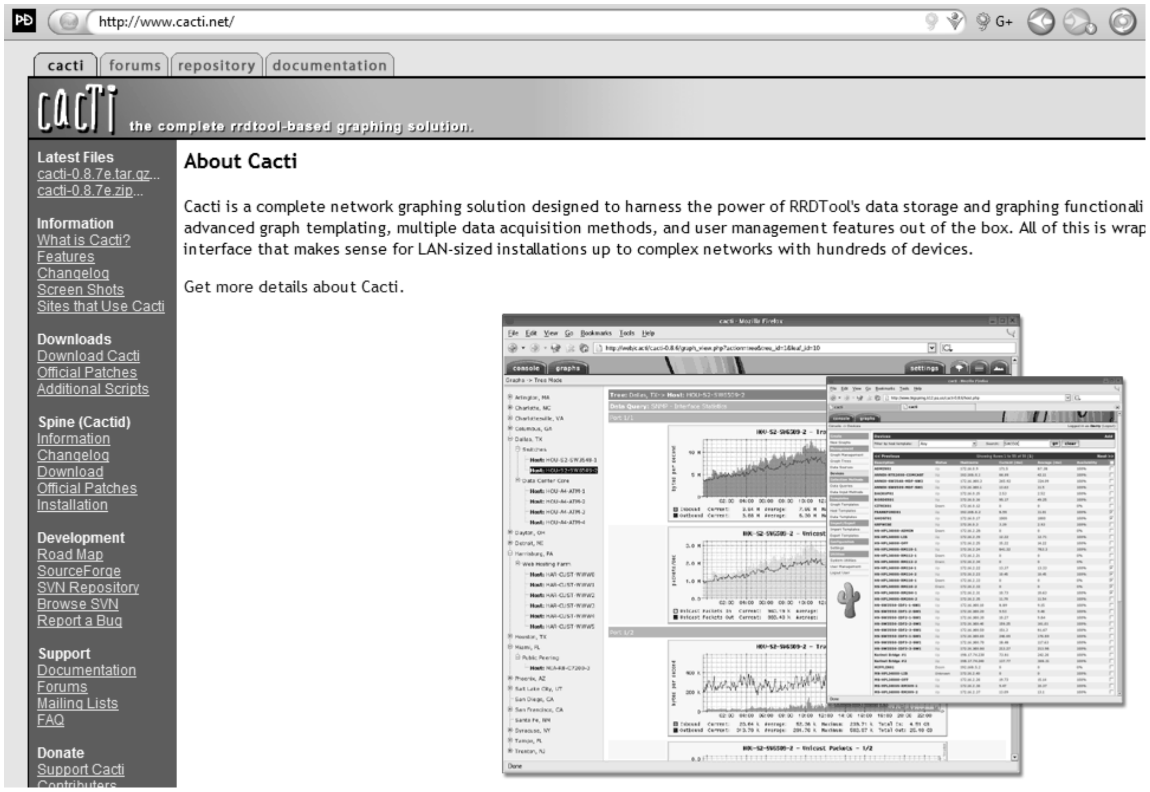
While slightly orthogonal in this list of NoC performance simulators, for context we also cite an algorithm which automatically maps the IPs/cores onto a generic regular Network on Chip (NoC) architecture such that the total communication energy is minimized (See Fig. 8) [62]. The performance of the mapped system is guaranteed to satisfy the specified constraints through bandwidth reservation. Authors have formulated the problem of energy-aware mapping, in a topological sense, and then propose an efficient branch-and-bound algorithm to solve it.

2.2 Power Modeling

Power dissipation has become a significant constraint in modern microprocessor design besides performance, clock frequency and die size [63]. In addition to extra heat created, high power consumption in embedded processors also reduces the battery lifetime and can cause thermal issues such as device degradation, higher packaging cost and reduced chip lifetime. In some sense, power has been elevated to a ‘first class design constraint’ and hence estimating power at the same time as performance studies in a design flow is becoming increasingly important. Just as performance analysis for a design candidate is performed during the design exploration phase with a cycle simulation, similarly augmented cycle simulators can be used to provide power estimates. The only caveat is that architectural power modeling should have sufficient details to have some meaning since there is a sizeable performance penalty for including any power model in the simulator.

2.2.1 Cacti

Cacti is an integrated cache access time, cycle time, area, aspect ratio, and power model [64, 65]. Cacti is used by computer architects to understand the performance tradeoffs inherent in different cache sizes and organizations. The original Cacti tool was released in 1994 as a fast tool to model SRAM caches. Recent versions have added area and active power (including leakage power) modeling to Cacti while also updating the basic circuit structure and device parameters to reflect the advances in scaling semiconductors. A graphical web interface to Cacti exists as shown in Fig. 9.



About Cacti

Cacti is a complete network graphing solution designed to harness the power of RRDTool's data storage and graphing functionali
advanced graph templating, multiple data acquisition methods, and user management features out of the box. All of this is wrap
interface that makes sense for LAN-sized installations up to complex networks with hundreds of devices.

Get more details about Cacti.

Fig. 9: Web interface for CACTI [64, 65]

2.2.2 Wattch

Wattch is an architectural framework for analyzing and optimizing microprocessor power dissipation [66]. Wattch was one of the first tools to complement existing lower-level tools by allowing architects to explore design space early on. Many subsequent frameworks are either based or inspired from this work. Wattch was provided as a power evaluation methodology within the SimpleScalar framework [67].

2.2.3 Orion

ORION [68] was amongst the first power models released with NoCs in mind. It has since been fairly widely used for early-stage power estimation and is interfaced with few NoC performance simulators. ORION 2.0 [55] was an enhancement over original ORION models with new subcomponent power models, area models, and updated technology models.

2.2.4 Nostrum High-Level Power Model (Nos-HPM)

Nos-HPM is a model allowing a fast power analysis to the accuracy of within 5% [69]. The empirical power model of links and switches was formulated and validated with the Synopsys Power Compiler. System simulations with Nos-HPM were shown to run up to 500 times faster than with Power Compiler for a 4 x 4 network.

2.2.5 PIRATE

PIRATE is a design framework which supports a methodology to generate and simulate a configurable NoC–IP core for the power-performance exploration of the interconnect network [70]. The NoC–IP core itself is composed of a set of parameterized modules like interconnection elements and switches to form different on-chip micro-network topologies.

CHAPTER III

OCIN_TSIM

In this chapter, we discuss the implementation details of Ocin_tsim as it existed before. This chapter presents an eagle's eye view as well specifics about the simulator.

3.1 Ideal Requirements

Ocin_tsim was born out of the need for a tool that could quickly and efficiently validate or reject NoC architectural propositions employing DVFS. Existing simulators, as listed in Chapter II, were found lacking for such purposes. As previously discussed, NoC power consumption and packet latency are highly dependent upon routing function, DVFS policy and network traffic. Therefore, detailed microarchitectural simulation of the proposed NoC under a given workload is required to provide an accurate measurement of the system's performance and power characteristics.

The goals of Ocin_tsim's design are as follows:

3.1.1 Flexible Design

The simulator was designed for use in early stage, NoC architectural research where numerous designs must be evaluated. By supporting run-time configurability, Ocin_tsim enables rapid evaluation of diverse workloads and network parameters [71]. Such efficiency is not possible with compile-time configurable simulators.

3.1.2 Extensibility

The simulator has a modular architecture that facilitates extensibility via user-defined pluggable extensions. Sample components for visualization and power modeling are included in the standard distribution of the simulator and may easily be replaced with custom models.

3.1.3 Fast Performance

As with all architectural simulators, an important challenge lies in simulating more cycles in less wallclock time. One of the benefits of having modular simulator architecture is that it enables suppression of unused components to improve simulator performance. For example, in timing-only mode, Ocin_tsim achieves faster simulation speeds by disabling power modeling and visualization modules.

3.2 Organization

This section presents the organization of our simulator. We describe its modular structure, cover a top level overview and present some of the key components. We also discuss its various design decisions and tradeoffs involved therein.

3.2.1 Modular structure

To support our aims, we chose a modular, object-oriented design style to implement the simulator. Each logical component was implemented as a separate C++

class. Such a structure not only supports our extensibility goal by being amenable to modular modifications, but also supports our flexibility goal by allowing use of inheritance and polymorphism to implement runtime configuration.

As an example of how this modular structure can be exploited, consider how per-node frequency support was added to the simulator. Initially Ocin_tsim supported a single, universal clock frequency across all the nodes. Simulation context was provided to each node by a simulator object every simulation step thereby evaluating and thus advancing the simulation. Therefore to implement dynamic, per-node frequency assignment, we simply had to constrain the number of simulation steps when a given simulator object would be evaluated. By including this count of cycles as a runtime configurable parameter, we were able to configure each node to be evaluated at different intervals thereby implementing multiple clock frequencies.

3.2.2 Top-level Structure

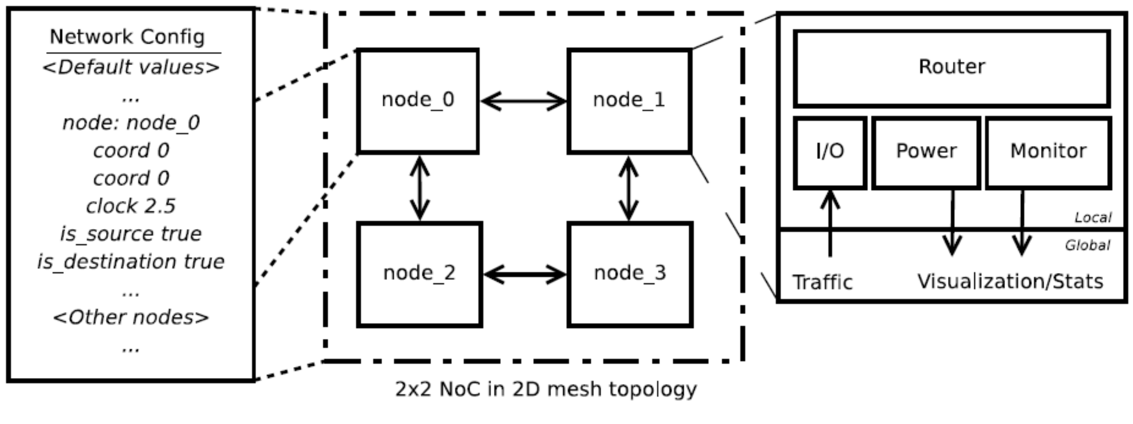


Fig. 10: 2x2 NoC in 2D mesh topology

Fig. 10 shows a structural organization as well as syntactical description of nodes in a simple 2x2 mesh-based NoC. As the diagram shows, each node of an NoC is modeled in Ocin_tsim as an object containing four different module instances. We briefly discuss the role of each of these nodes below:

3.2.2.1 Router Model

The router module models the flow of packets through the given router’s user-defined microarchitecture. This module contains a simulator timing object which synchronizes the packet traversal at an interval set by its clock frequency.

3.2.2.2 Power Model

Each node also carries an Orion [55] instance that computes the per-node power consumption based on actual run-time activity. The original Orion release was modified to allow per-node voltage-frequency selection. These values may be changed at runtime for fully dynamic voltage and frequency modeling. Other power models can also be easily adapted for use with our simulator.

3.2.2.3 Monitor and Visualization

Ocin_tsim contains monitoring modules for data collection and subsequent post processing using built-in statistics and/or visualization blocks. Monitor modules can be used to report statistics on various network elements such as packet routing, arbitration logic, link bandwidth, FIFO usage and other router statistics. The visualization module uses the "GD" standard graphics library to capture the selected monitor statistic into a series of images [72]. By processing the data gathered by various monitor modules, the simulator is able to generate graphic representations of network buffer occupancy, link utilization, and other network statistics at the end of the simulation or during the course of one, producing a time-varying series of images. When combined with a network power model, this tool can elucidate interesting trends into per-node differences in power consumption, thermals, and traffic congestion. For a complete list of supported visualization modes, please refer to Appendix B. In complex, time varying environments such as those found in NoCs, graphical representations can often present a more efficient way to gain an intuitive grasp of network performance bottlenecks.

3.2.2.4 Input / Output Model

Ocin_tsim contains injector and ejector modules which mimic traffic generation and reception by the local processor or IP block attached to each router. While injector and ejector modules mimic various possible ways in which traffic may be induced across the NoC, an instance of I/O module in each node controls their behavior on a per-node basis.

Fig. 10 shows a sample node declaration in the network configuration file. Runtime configuration via simple text files allows the description of complex NoC topologies via simple declarative statements. The configuration files are discussed in detail in Appendix B.

3.2.3 Traffic Generation and Routing

Ocin_tsim supports several traffic generation, routing, and port selection functions. As with other simulator fields, all of these are runtime configurable. Packets can be generated either in random or bimodal mode (alternate between two different packet sizes).

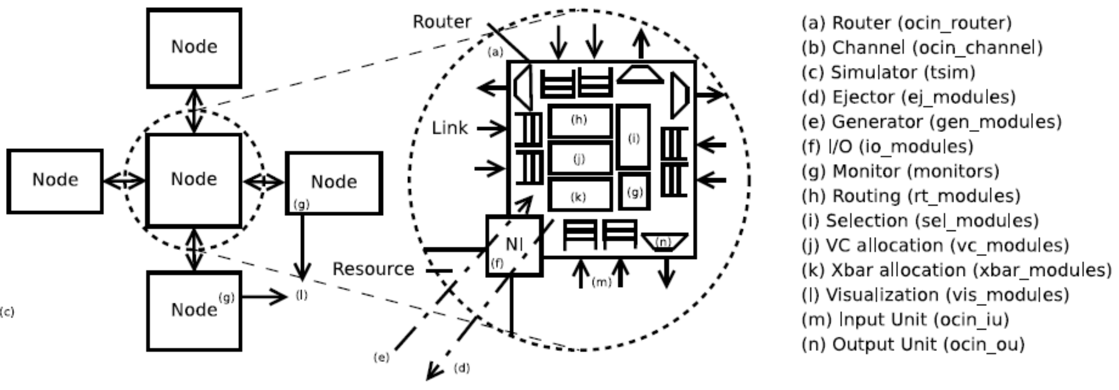


Fig. 11: Simulator-microarchitecture mapping

The simulator supports a variety of synthetic traffic patterns including bit complement, transpose, bit reverse and hotspot. In addition, Ocin_tsim has a trace-driven packet generation mode to support network-level replay of simulated applications. Similarly, routing functions, port selection functions, and resource allocation functions can be chosen at run-time. For a comprehensive list of run-time configuration options, please refer to Appendix B.

3.3 Simulator Internals

This section presents the internals of our simulator. We describe its functionality, internal source structure and how it maps to the microarchitecture of an NoC router. Various command line arguments and configuration fields used in our simulator are presented in Appendix A and Appendix B.

3.3.1 Topology and Router Microarchitecture

Fig. 11 shows a simple five-node NoC connected in a star topology as implemented in our simulator. Each router connects a local resource to the rest of the network via a node interface. Packets are routed over network links and stored in Virtual Channel (VC) buffers. Virtual Channels and switch bandwidth are assigned by allocation modules. Data injection (from resource into the router) and ejection (from router into the resource) is handled by generator and ejector modules. Each router instance also has a monitor module for statistics and visualization.

3.3.2 Simulator Code Descriptions

As shown in Fig. 11, code for Ocin_tsim is completely modular; each file maps to a component C++ class. Top level files integrate the microarchitectural components of a router into a single router object instance. Multiple instances of this router object are replicated and connected together to simulate the behavior of an NoC. The following list describes the functionality of the code files in *src/* directory.

– *ocin/tsim_ocin.cpp*: top level file. Performs top level argument parsing, binds simulation agent to an instance of ocin_top.

– *ocin/ocin_top.**: top level integration files, perform parameter instantiation from configuration files, creates and initializes tiles and wires structure and invoke all required modules such as visualization or ejector (See Fig. 11).

– *ocin/ocin_router.**: router class. Includes routines for assigning numbered VCs and logical ports to each router, also include the evaluate subroutine (See Fig. 11(a)).

– *ocin/ocin_channel.**: include routines to transmit flits and calculate corresponding credit/cost (See Fig. 11(b)).

– *ocin/ocin_cost_msg.**: define the cost message structure to be used later on.

– *ocin/ocin_defs.**: top level define files containing various shared macros and custom data types.

– *ocin/ocin_helper.**: contains various utility functions used elsewhere in simulator.

– *ocin/ej_modules/**: default, non-blocking top-level ejector module (See Fig. 11(d)).

– *ocin/gen_modules/**: various traffic generation modes including random, complement, transpose, reverse, selfsimilar, file trace and hotspot (See Fig. 11(e)).

– *ocin/io_modules/**: router I/O component modules (See Fig. 11(f)), input units (See Fig. 11(m)) and output units (See Fig. 11(n)).

– *ocin/monitors/**: monitor modules. Maintains copies of various network parameters such as flits traversed through a node, injection/acceptance ratio, stalls encountered, and requests/grants made (See Fig. 11(g)).

– *ocin/rt_modules/**: routing functions such as adaptive, XY dimension-order and o1turn (See Fig. 11(h)).

– *ocin/sel_modules/**: selection function. Arbitration modules (See Fig. 11(i)).

– *ocin/vc_modules/**: virtual channel allocation modules (See Fig. 11(j)).

– *ocin/xbar_modules/**: crossbar allocation including routines for simple, 2-level and speculative allocation (See Fig. 11(k)).

– *ocin/vis_modules/**: visualization modules. Includes routines for interacting with PNG/JPEG/GD libraries (See Fig. 11(l)).

– *tsim/**: the simulation engine and modular agent tsim module. Defines the simulation interface (See Fig. 11(c)).

CHAPTER IV

IMPLEMENTATION

In this chapter, we present the novel features that were introduced in Ocin_tsim as part of this thesis and also show results from new experiments that could be run as a direct result of these changes.

4.1 Novel Features

Following novel features were introduced in Ocin_tsim as part of the work for this thesis.

4.1.1 DFS Modeling

Ocin_tsim models per-node frequency in a manner analogous to sim clock in Verilog. Each clock period used in simulator is specified as a multiple of root clock time period where root clock is the time step at which simulation is advanced. Thus, a node with a clock period multiplier of two will be evaluated every two simulation steps. Signals crossing clock domains pay a synchronization latency of twice the destination clock period.

It is important to note here that root frequency sets the relative updating interval of software execution cycle and hence determines the simulation time. As a result, for optimum simulation time, root clock should normally be set to Highest Common Factor (HCF) of all the clock periods in the network. In addition to its applications in DVFS, dynamic frequency scaling (DFS) is also applied in globally asynchronous, locally synchronous (GALS) designs containing mesochronous clock boundaries. Our simulator does not impose any restrictions on the use of dynamic frequency tuning and can be used for DVFS as well as DFS designs.

4.1.2 Power Modeling

Another novel feature of our simulator is support for power modeling at architectural level in a DVFS setup. Each node in Ocin_tsim can compute per node power consumption based on actual, traffic-induced activity. Our simulator can be used with any off-the-shelf or custom power model to generate power estimates with desired accuracy.

4.2 Simulator Usage

This section describes installation procedure and presents typical use cases for NoC based design. The software is made available under MIT license [73].

4.2.1 Required Libraries

The following libraries are required to successfully build Ocin_tsim:

– GD Graphics library, required by visualization module can be found at http://www.boutell.com/gd/.

– PNG Library, required by visualization module in drawing to PNG files can be found at http://www.libpng.org/pub/png/.

– JPEG Library, required by visualization module in drawing to jpeg files can be found at http://www.ijg.org/.

– Orion library, if you wish to perform power estimations in your simulation. We have tested our simulator to work with Orion version 2.0. Orion 2.0 library can be downloaded from http://www.princeton.edu/~peh/orion.html. You may also use the version of Orion 2.0 library made available at http://www.ece.tamu.edu/~ocintsim/ which supports programmatic control of voltage and frequency.

4.2.2 Installation

Ocin_tsim may be obtained online at http://www.ece.tamu.edu/~ocintsim/. The following commands may be used to extract and build the simulator:

```
tar -zxvf ocin_tsim.tar.gz
cd tmax
make ocin
```

4.3 Some Standard Test Cases

4.3.1 Timing Simulation

A simple Ocin_tsim use case is timing simulation for a small 2x2 NoC design. Ocin_tsim comes with a preconfigured 2x2 NoC network out-of-the-box. For this particular configuration, we have chosen a 10% injection bandwidth, 1 hop per cycle, 128 baseline channel width and run all the nodes at same clock frequency. Each node has 8 virtual channels and 4 element deep queues. Per-node configuration for this NoC is specified in the included simple test net.cfg file.

```
cd simple test
../bin/tsim_ocin -config simple test.cfg -n 1000
```

4.3.2 Timing Simulation with Power

To demonstrate the use of Ocin_tsim for DVFS modeling, we will use the same 4-node NoC from the example above. Node configuration file should be modified to assign different startup voltage/frequency, if so desired. By altering the voltage/frequency level of a node based upon a required parameter (say, the incoming traffic), end-user can implement a policy for DVFS level selection. It is important to note here that in the absence of a power model, any DVFS level may be provided and the simulator would still successfully finish the simulation. However, for the simulation to have any physical relevance it is important to stick to the granularity and/or valid DVFS levels/range to switch between, as defined in your power model.

Next we provide the simulator a 'hook' to a power model. This hook is a simple API call for the power computing function (in power model) made for each flit activity. For this exercise we use a modified version of the Orion 2.0 library made available at http://www.ece.tamu.edu/~ocintsim/. This library is derived from Orion 2.0 and modified to support programmatic control of voltage and frequency. By including the power calculating APIs in the simulator, any simulation now will perform both timing measurements as well as power estimation.

Fig. 12 and TABLE 1 show the results from a simulation done on an 8x8 NoC with the SPLASH (Stanford ParalleL Applications for SHared memory) 2 benchmark suite [74]. SPLASH 2 is a suite of parallel programs popularly used as a workload for quantitative evaluation of ideas in shared address-space multiprocessing. Based on our trace-driven simulations, we observe that the average power consumed per node for all the benchmarks in SPLASH 2 benchmark suite is quite similar to each other.

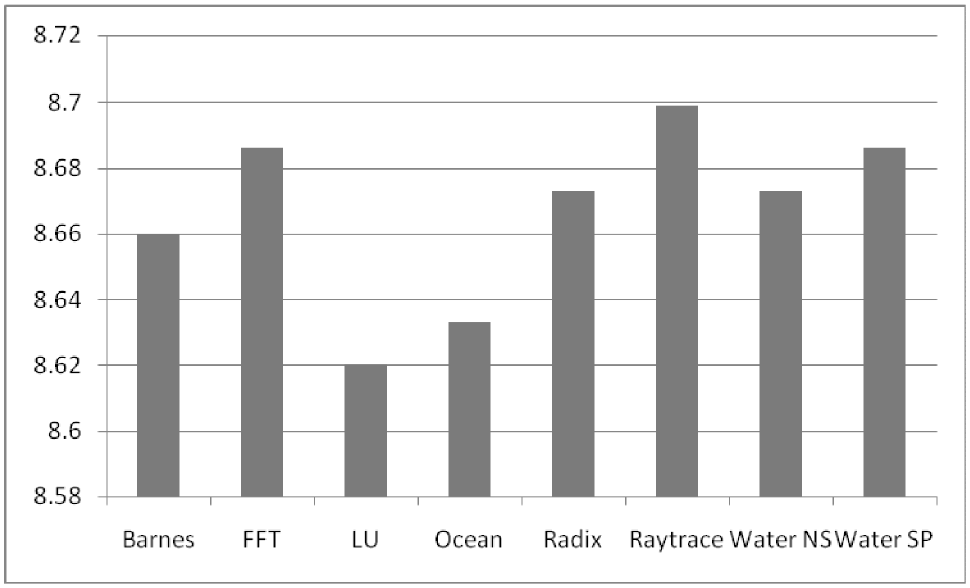


Fig. 12: Per-node average power in mW for Splash 2 benchmark suite.

TABLE 1
Average power dissipation per-node for Splash 2 benchmarks

Benchmark	Barnes	FFT	LU	Ocean	Radix	Raytrace	Water NS	Water SP
Power (mW)	8.66	8.686	8.62	8.633	8.673	8.699	8.673	8.686

Similarly, power figures over the complete simulation time can be plotted across regular windows to observe a dynamic power profile for any such NoC design. Such a simulation would be quite useful in identifying any performance bottlenecks or thermal hotspots and resolve them by modifying the design proposition. It can be seen here that having an architectural tool that can estimate power as well as performance can be used to attain an early understanding of the expected savings from a new design proposition. By extending Ocin_tsim for DVFS modeling, we believe that we have taken a key step in attaining this goal.

CHAPTER V

CONCLUSIONS AND FUTURE WORK

5.1 Thesis Summary

An accurate estimation of power and timing of interconnection networks in early phases of the design process can prove to be effective in deeper exploration of NoC design space. Existing NoC simulators are inadequate for power performance simulation of future technologies employing per-node DVFS with the goal of minimizing power consumption. In this work, we introduced a simulator that combines power and performance models, thereby enabling rapid and accurate evaluation of future NoC architectures. We have also presented an extensible framework for integrating power models from external sources into our timing simulator.

Extensions to Ocin_tsim are both underway and planned. These are covered briefly in the Future Work section. At present, this tool provides researchers with a fast and efficient simulation infrastructure for screening architectural propositions before deploying them for FPGA testing or production. We hope that you find these tools useful, and encourage you to contact us with suggestions on improving the release, documentation and the tool itself. Relevant accessibility and support information are provided in Appendix C.

5.2 Future Work

We believe that Ocin_tsim is a great tool for performing architectural simulations of NoC and gives researchers a tool to evaluate their NoC architectural propositions early on in the design cycle. An extensible project such as this leaves one with many possible future development paths to pick from. We cover some of those in this section.

5.2.1 DVFS Extensions

Work is underway on using our simulator for performing a limit study on DVFS switching times. We believe that such a study could provide insights into emphasis required on approaches utilizing faster DVFS switching for performance benefits at the cost of reduced regulator efficiency. Work is also planned on implementing certain basic DVFS policies and to expose them to the end user by means of a configuration field. Currently the users need to implement their own DVFS policies by tinkering with the simulator.

5.2.2 Plugin-like Interface

An extension planned for future is to provide an API layer for integration of power models. Such an API layer would provide a standard interface enabling researchers to import their custom power models into Ocin_tsim framework easily.

5.2.3 3D Visualization

Another thought for a possible future extension is using a 3D visualization engine to provide real-time 3D rendering of various network parameters using time as the third dimension. Such a visualization technique could be more intuitive for some users in faster debugging of performance degradation issues rooted in real-time bottlenecks. Current visualization strategy is computationally prohibitive to be performed in real-time.

REFERENCES

[1] S. Rusu, S. Tam, H. Muljono, J. Stinson, D. Ayers, J. Chang, R. Varada, M. Ratta, and S. Kottapalli, “A 45nm 8-Core Enterprise Xeon Processor,” *Int. Solid-State Circuits Conf.*, pp. 98–99, Feb. 2009.

[2] D. Wentzlaff, P. Griffin, H. Hoffmann, L. Bao, B. Edwards, C. Ramey, M. Mattina, C.-C. Miao, J. F. B. III, and A. Agarwal, “On-Chip Interconnection Architecture of the Tile Processor,” *IEEE Micro*, vol. 27, no. 5, pp. 15-31, Sep.-Oct. 2007.

[3] Anonymous editors. (2010, Mar 15). *KiloCore KC256* [Online]. Available: http://en.wikipedia.org/wiki/kilocore, 2010.

[4] L. Benini and G. D. Micheli, “Networks On Chips: A New SoC Paradigm,” *Computer*, vol. 35, no. 1, pp. 70–78, Jan. 2002.

[5] W. J. Dally and B. Towles, “Route Packets, Not Wires: On-Chip Interconnection Networks,” *Int. Conf. Design Automation (DAC)*, pp. 684–689, Jun. 2001.

[6] A. Hemani, A. Jantsch, S. Kumar, A. Postula, J. Berg, M. Millberg, and D. Lindqvist, “Network on Chip: An Architecture for Billion Transistor Era,” *IEEE NorChip Conf.*, Nov. 2000.

[7] S. Vangal, J. Howard, G. Ruhl, S. Dighe, H. Wilson, J. Tschanz, D. Finan, P. Iyer, A. Singh, T. Jacob, S. Jain, S. Venkataraman, Y. Hoskote, and N. Borkar, “An 80-Tile 1.28 TFLOPS Network-on-Chip in 65nm CMOS,” *IEEE Int. Solid-State Circuits Conf. Digest of Technical Papers*, pp. 98-589, Feb. 2007.

[8] Y. Hoskote, S. Vangal, A. Singh, N. Borkar, and S. Borkar, “A 5-GHz Mesh Interconnect for a Teraflops Processor,” *IEEE Micro*, vol. 27, no. 5, pp. 51–61, Sep.-Oct. 2007.

[9] P.P. Pande, C. Grecu, M. Jones, A. Ivanov, and R. Saleh, “Performance Evaluation and Design Trade-Offs for Network-on-Chip Interconnect Architectures,” *IEEE Tran. Computers*, vol. 54, no. 8, pp. 1025-1040, Aug. 2005.

[10] R. Ho, K.W. Mai, and M.A. Horowitz, “The Future of Wires,” *Proc. IEEE*, vol. 89, no. 4, pp. 490-504, Apr. 2001.

[11] P. Kapur, J.P. Mc Vittie, and K.C. Saraswat, “Technology and Reliability Constrained Future Copper Interconnects—Part II: Performance Implications,” *IEEE Tran. Electron Devices*, vol. 49, no. 4, pp. 598-604, Apr. 2002.

[12] D. Sylvester and K. Keutzer, “Impact of Small Process Geometries on Microarchitectures in Systems on a Chip,” *Proc. IEEE*, vol. 89, no. 4, pp. 467-489, Apr. 2001.

[13] Semiconductor Industry Association (SIA). (2003). *International Roadmap for Semiconductors*, 2003 edition, Austin, TX. International SEMATECH, 2003. [Online]. Available: http://www.itrs.net/links/2003itrs/home2003.htm

[14] C. Grecu, P.P. Pande, A. Ivanov, and R Saleh, “Structured Interconnect Architecture: A Solution for the Non-Scalability of Bus-Based SoCs,” *Proc. Great Lakes Symp. VLSI*, pp. 192-195, Apr. 2004.

[15] C. Hsieh and M. Pedram, “Architectural Energy Optimization by Bus Splitting,” *IEEE Tran. Computer-Aided Design*, vol. 21, no. 4, pp. 408-414, Apr. 2002.

[16] M. Horowitz and B. Dally, “How Scaling Will Change Processor Architecture,” *Proc. Int. Solid-State Circuits Conf.*, pp. 132-133, Feb. 2004.

[17] J. Nurmi, *Interconnect-Centric Design for Advanced SoC and NoC*. Springer Science + Business Media Inc., Germany, 2005.

[18] 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,” *Proc. 38th Design Automation Conf.*, Las Vegas, pp. 667-72, Jun. 2001.

[19] P. Macken, M. Degrauwe, M. V. Paemel, and H. Oguey, “A Voltage Reduction Technique for Digital Systems,” *IEEE Int. Solid-State Circuits Conf.*, pp. 238–239, Feb. 1990.

[20] C. Lai, J. H. Lin, and Y. F. Wang, “DVFS SoC Architecture and Implementation,” *SoC Technology Journal*, vol. 3, pp. 84–91, Nov. 2005.

[21] C. Isci, A. Buyuktosunoglu, C.-Y. Cher, P. Bose, and M. Martonosi, “An Analysis of Efficient Multi-core Global Power Management Policies: Maximizing Performance for a Given Power Budget,” *Proc. 39th Annu. IEEE/ACM Int. Symp. Microarchitecture*, vol. 26, no. 1, pp. 119-129, Feb. 2006.

[22] G. Semeraro, G. Magklis, R. Balasubramonian, D. H. Albonesi, S. Dwarkadas, and M. L. Scott, “Energy-efficient Processor Design Using Multiple Clock Domains with Dynamic Voltage and Frequency Scaling,” *Int. Symp. High-Performance Computer Architecture*, pp. 29-40, Feb. 2002.

[23] T. Simunic, L. Benini, A. Acquaviva, P. Glynn, and G. D. Micheli, "Dynamic Voltage Scaling and Power Management for Portable Systems," *Design Automation Conf.*, pp. 524-529, Jun. 2001.

[24] Q. Wu, P. Juang, M. Martonosi, and D. W. Clark, "Voltage and Frequency Control with Adaptive Reaction Time in Multiple-Clock-Domain Processors," *11th Int. Symp. High-Performance Computer Architecture*, pp. 178-189, Feb. 2005.

[25] W. Kim, M. Gupta, G. Y. Wei, and D. Brooks, "System Level Analysis of Fast, Per-core DVFS Using On-chip Switching Regulators," *Int. Symp. High Performance Computer Architecture*, pp. 123–134, Feb. 2008.

[26] U.Y. Ogras, R. Marculescu, P. Choudhary, and D. Marculescu, "Voltage-Frequency Island Oartitioning for GALS-Based Networks-on-Chip," *Proc. 44th Annu. Design Automation Conf.,* pp. 110-115, Jun. 2007.

[27] D. Bertozzi, "NoC Synthesis Flow for Customized Domain Specific Multiprocessor Systems-on-Chip*,"* *IEEE Tran. Parallel and Distributed Systems*, vol. 16, no. 2, pp. 113-129, Feb. 2005.

[28] J. Dielissen, A. Radulescu, K. Goossens, and E. Rijpkema, "Concepts and Implementation of the Philips Network-on-Chip," *IP-based SoC Design*, Nov. 2003.

[29] M. Millberg, E. Nilsson, R. Thid, and A. Jantsch, "Guaranteed Bandwidth Using Looped Containers in Temporally Disjoint Networks within the Nostrum Network on Chip," *Proc. Design Automation and Test in Europe (DATE)*, pp. 890-895, Feb. 2004.

[30] Y. S. Dhillon, A. U. Diril, A. Chatterjee, and H. S. Lee, "Algorithm for Achieving Minimum Energy Consumption in CMOS Circuits Using Multiple Supply and Threshold Voltages at the Module Level," *Proceedings of ICCAD*, pp. 693-700, Nov. 2003.

[31] K. Niyogi and D. Marculescu, "Speed and Voltage Selection for GALS Systems Based on Voltage/Frequency Islands," *Proceedings of ASP-DAC*, pp. 292-297, Jan. 2005.

[32] M. Powell, S.-H Yang, B. Falsafi, K. Roy, and T.N. Vijaykumar, "Reducing Leakage in a High-Performance Deep-Submicron Instruction Cache," *IEEE Tran. VLSI Systems*, vol. 9, no. 1, pp. 77-89, Feb. 2001.

[33] S. Shigematsu, S. Mutoh, Y. Matsuya, and J. Yamada, "A 1-V High-Speed MTCMOS Circuit Scheme for Power-Down Application Circuits," *IEEE Journal on Solid-State Circuits*, vol. 32, no. 6, pp. 861-869, Jun. 1997.

[34] B.H. Calhoun, F.A. Honore, and A.P Chandrakasan, "A Leakage Reduction Methodology for Distributed MTCMOS," *IEEE Journal on Solid-State Circuits*, vol. 39, no. 5, pp. 818-826, May 2004.

[35] C. Long and L. He, "Distributed Sleep Transistor Network for Power Reduction," *Proc. IEEE/ACM Design Automation Conf.*, pp. 181-186, Jun. 2003.

[36] A. Ramalingam, B. Zhang, A. Davgan, and D. Pan, "Sleep Transistor Sizing Using Timing Criticality and Temporal Currents," *Proc. ASP-DAC*, pp. 1094-1097, Jan. 2005.

[37] K. Shi and D. Howard, “Challenges in Sleep Transistor Design and Implementation in Low-Power Designs,” *Proc. 43rd Annu. Design Automation Conf.,* pp. 113-116, Jul. 2006.

[38] D.E. Lackey, P.S. Zuchowski, T.R. Bednar, D.W. Stout, S.W. Gould, and J.M. Cohn, “Managing Power and Performance for System-on-Chip Designs Using Voltage Islands,” *IEEE/ACM Int. Conf. Computer-Aided Design*, pp. 195–202, Nov. 2002.

[39] J. Tschanz, S. Narendra, Y. Yibin, B. Bloechel, S. Borkar, and V. De, “Dynamic-Sleep Transistor and Body Bias for Active Leakage Power Control of Microprocessors,” *IEEE Int. Solid-State Circuits Conf.*, vol. 1, pp. 102–481, Feb. 2003.

[40] Q. Wu, M. Pedram, and X. Wu, “Clock-gating and Its Application to Low Power Design of Sequential Circuits,” *IEEE Custom Integrated Circuits Conf.*, pp. 479–482, May 1997.

[41] M. Pedram, “Power Minimization in IC Design: Principles and Applications,” *ACM Tran. Design Automation*, vol. 1, no. 1, pp. 3–56, Jan. 1996.

[42] G. Friedman, “Clock Distribution Design in VLSI Dircuits: An Overview,” *Proc. IEEE ISCAS*, San Jose, CA, pp. 1475–1478, May 1994.

[43] Q. Wu, M. Pedram, and X. Wu, “Clock-Gating and Its Application to Low Power Design of Sequential Circuits,” *Proc. IEEE Custom Integrated Circuits Conf.*, vol 47, pp. 415-420, May 2000.

[44] S. Abedinpour, B. Bakkaloglu, and S. Kiaei, “A Multi-Stage Interleaved Synchronous Buck Converter with Integrated Output Filter in a 0.18um SiGe Process,” *IEEE Int. Solid-State Circuits Conf.*, pp. 1398-1407, Feb. 2006.

[45] P. Hazucha, G. Schrom, H. Jaehong, B. Bloechel, P. Hack, G. Dermer, S. Narendra, D. Gardner, T. Karnik, V. De, and S. Borkar, “A 233-MHz 80%-87% Efficiency Four-Phase DC-DC Converter Utilizing Air-Core Inductors on Package,” *IEEE Journal of Solid-State Circuits*, vol. 40, no. 4, pp. 838-845, Apr. 2005.

[46] G. Schrom, P. Hazucha, J. Hahn, D. Gardner, B. Bloechel, G. Dermer, S. Narendra, T. Karnik, and V. De, “A 480-MHz, Multi-Phase Interleaved Buck DC-DC Converter with Hysteretic Control,” *IEEE Power Electronics Specialist Conf.*, vol. 6, pp. 4702-4707, Jun. 2004.

[47] J.Wibben and R. Harjani, “A High Efficiency DC-DC Converter Using 2nH On-Chip Inductors,” *IEEE Symp. VLSI Circuits*, pp. 22-23, Jun. 2007.

[48] U.Y. Ogras, J. Hu, and R. Marculescu, “Key Research Problems in NoC Design: a Holistic Perspective,” *3rd IEEE/ACM/IFIP Int. Conf. Hardware/Software Codesign and System Synthesis*, pp. 69–74, Feb. 2005.

[49] Y. Sun, S. Kumar, and A. Jantsch, “Simulation and Evaluation of a Network on Chip Architecture Using Ns-2,” *IEEE NorChip Conf.*, Nov. 2002.

[50] A. Jantsch, *Nocsim: A NoC Simulator*. School of Information and Communication Technology, Royal Institute of Technology, Stockholm, version 0.4 alpha edition, Jan. 2006.

[51] R. Thid, M. Millberg, and A. Jantsch, "Evaluating NoC Communication Backbones with Simulation," *IEEE NorChip Conf.*, pp. 27–30, Nov. 2003.

[52] J. Xi and P. Zhong, "A Transaction-Level NoC Simulation Platform with Architecture-Level Dynamic and Leakage Energy Models," *16th ACM Great Lakes Symp. VLSI (GLSVLSI)*, pp. 341–344, Apr.-May 2006.

[53] D. Siguenza-Tortosa and J. Nurmi, "VHDL-based Simulation Environment for Proteo NoC," *7th IEEE Int. High-Level Design Validation and Test Workshop*, pp. 1–6, Oct. 2002.

[54] R. Marculescu, "Networks-on-Chip: The Quest for On-Chip Fault-Tolerant Communication," *IEEE Computer Society Annu. Symp. VLSI*, pp. 8–12, Feb. 2003.

[55] 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," *Design Automation and Test in Europe (DATE)*, pp. 423-428, Apr. 2009.

[56] Z. Lu, R. Thid, M. Millberg, and A. Jantsch, "NNSE: Nostrum Network-on-Chip Simulation Environment," *Proc. SSoCC*, Apr. 2005.

[57] D. Atienza, P. D. Valle, G. Paci, F. Poletti, L. Benini, G. D. Micheli, and J. M. Mendias, "A Fast HW/SW FPGA-based Thermal Emulation Framework for Multi-Processor System-on-Chip," *DAC*, pp. 618-623, Jul. 2006.

[58] A. Kumar, N. Agarwal, L.-S. Peh, and N.K. Jha, "A System-Level Perspective for Efficient NoC Design," *Proc. IEEE Int. Symp. Parallel and Distributed Processing, (IPDPS)*, pp. 1-5, Apr. 2008.

[59] N. Agarwal, L.-S. Peh, and N. Jha, *GARNET: A Detailed Interconnection Network Model inside a Full-system Simulation Framework*. CE-P08-001, Dept. of Electrical Engineering, Princeton University, Feb., 2008.

[60] M.M.K. Martin, D.J. Sorin, B.M. Beckmann, M.R. Marty, M. Xu, A.R. Alameldeen, K.E. Moore, M.D. Hill, and D.A. Wood, "Multifacet's General Execution-Driven Multiprocessor Simulator (GEMS) Toolset," *SIGARCH Computer Architecture News*, vol. 33, no. 4, pp. 92-99, Sep. 2005.

[61] N. Banerjee, P. Vellanki, and K. S. Chatha, "A Power and Performance Model for Networks-on-Chip Architectures," *Proc. Design Automation Test Europe*, vol. 2, pp. 1250-1255, Feb. 2004.

[62] J.Hu and R.Marculescu, "Energy-Aware Mapping for Tile-based NOC Architectures Under Performance Constraints," *ASP-DAC*, pp. 233-239, Jan. 2003.

[63] R. Graybill and R. Melhem, *Power Aware Computing*. Kluwer Academic/Plenum Publishers, ISBN 0-306-46786-0, May 2002.

[64] P. Shivakumar and N.J. Jouppi, *CACTI 3.0: An Integrated Cache Timing, Power, and Area Model*. WRL Research Report, Feb. 2001.

[65] D. Tarjan, S. Thoziyoor, and N. Jouppi, *CACTI 4.0*. HPL-2006-86, HP Laboratories, Jun. 2006.

[66] D. Brooks, V. Tiwari, and M. Martonosi, "Wattch: A Framework for Architectural Level Power Analysis and Optimizations," *Proc. 27th Annu. Int. Symp. Computer Architecture*, IEEE CS Press, Los Alamitos, Calif., pp. 83-94, Jun. 2000.

[67] D.C. Burger and T.M. Austin, “The SimpleScalar Tool Set,” version 2.0. *Computer Architecture News*, pp. 13-25, Jun. 1997.

[68] H. Wang, X. Zhu, L.-S. Peh, and S. Malik, “Orion: A Power-Performance Simulator for Interconnection Networks,” *Proc. MICRO 35*, pp. 294-305, Nov. 2002.

[69] S. Penolazzi and A. Jantsch, “A High Level Power Model for the Nostrum NoC,” in *DSD*, pp. 673–676, Aug. 2006.

[70] G. Palemoro and C. Silvano, “PIRATE: A Framework for Power/Performance Exploration of Network-On-Chip Architectures,” *PATMOS*, vol. 3254, pp. 521-531, Sep. 2004.

[71] S. Prabhu, B. Grot, P.V. Gratz, and J. Hu, “Ocin_tsim - DVFS Aware Simulator for NoCs,” *The 1st Workshop on SoC Architecture, Accelerators and Workloads (SAW-1)*, Jan. 2010.

[72] T. Boutell (2008). *GD Lib, A Graphics Library for Fast Creation of GIF Images* [Online]. Available: http://www.boutell.com/gd/, 2008.

[73] Open Source Initiative (1998). *Open Source Initiative - The MIT License* [Online]. Available: http://www.opensource.org/licenses/mit-license.php, 1998.

[74] S.C. Woo, M. Ohara, E. Torrie, J.P. Singh, and A. Gupta, “The SPLASH-2 Programs: Characterization and Methodological Considerations,” *22nd Annu. Int. Symp. Computer Architecture*, pp. 24-36, Jun. 1995.

[75] P. Gratz, B. Grot, and S. W. Keckler, “Regional Congestion Awareness for Load Balance in Networks-on-Chip,” *IEEE 14th Int. Symp. High Performance Computer Architecture (HPCA)*, pp. 203–214, Feb. 2008.

[76] J.P. Singh, W.-D. Weber, and A. Gupta, “SPLASH: Stanford Parallel Applications for Shared Memory,” *Computer Architecture News*, vol. 20, no. 1, pp. 5-44, Mar. 1992.

APPENDIX A

COMMAND LINE ARGUMENTS

The following command-line arguments are available in the simulator.

-h, --help: displays usage information

-v, --version: displays version information

-n cycles: maximum number of simulation cycles (default = 10000000)

-config filename: input configuration file

-out filename: prefix for output file (and path)

-debug, -d: displays verbose debug info in the log

-debug_start_cycle: verbose debug info display start at given cycle

-debug_stop_cycle: verbose debug info display stops at given cycle

-parms, -p : creates simulator parameters file

-noparms: does not create simulator parameters file

-stats, -s: creates simulator statistics file

-nostats: does not create simulator statistics file

-logfile: enables simulator logging to file

-nologfile: disables simulator logging to file

-logout: enables simulator logging to stdout

-nologout: disables simulator logging to stdout

-logerr: enables simulator logging to stderr

-nologerr: disables simulator logging to stderr

--< name >: initializes simulator parameter < name > with a value of 1

--< name >=< value >: initializes simulator parameter < name >=< value >.

APPENDIX B

CONFIGURATION FIELDS

Configuration parameters are broadly divided into global and local (node based) categories and are stored in two separate files. While global settings affect network-wide parameters, local settings determine per-node configurations.

B.1 Global Configuration

Global configuration file specifies NoC-wide parameters such as packet generation mode, wire latency, switch for visualization module, percent injection bandwidth or number of warm up cycles. Valid values are presented in parentheses, with default parameters de-italicized.

B.1.1 General

Used in controlling generic parameters of NoC and global parameters of routers.

– *inj_vc_count*: injector VC count (1)

– *ej_vc_count*: ejector VC count (1)

– *percent_inj_bw*: percent injection bandwidth (20)

– *hops_per_cycle*: number of hops per cycle (0)

– *wire_delay*: models per-hop link latency (0)

– *router_pipeline_lat*: models any additional router pipeline latency (0)

– *random_req_size*: request size in random mode. When enabled, all packets are sized randomly. (0)

– *bimodal_req_size*: request size in bimodal mode. When enabled, all packets are sized to one of two different possible lengths. (0)

– *bimodal_size1*: packet size 1 in bimodal mode (64)

– *bimodal_size2*: packet size 2 in bimodal mode (512)

– *selfsim_inj*: self similar injection switch. When enabled, random numbers used in packet injection rate calculations are read from the "selfsim file" trace file rather being calculated via the pseudo-random number generator. This allows the off-line generation of self similar random numbers to drive the packet injection.

– *aggressive_vc_alloc*: switch for aggressive VC allocation (can be used only with deterministic routing functions) (0)

– *adaptive_1avail*: switch for ensuring winning out port has at least one VC available (0)

– *baseline_channel_width*: baseline channel width (128)

B.1.2 Regional Congestion Awareness (RCA)

RCA is an efficient method for port selection based on aggregated congestion information communicated by neighboring nodes [75].

Following configuration fields control this behavior.

– *extra_rca_delay*: models Regional Congestion Awareness (RCA) delay (0)

– *low_bw_rca*: switch for modeling serialized updates over multi-cycles for bandwidth limited RCA (0)

– *low_bw_rca_latency*: models serialization latency for low bandwidth RCA field (0)

– *0delay_cost_msg_update*: switch for updating cost message with zero delay (0)

– *same_cycle_local_cost*: switch for computing local congestion cost in same cycle. If local congestion is a function of grant signals which come late in cycle, then congestion cost should be computed in next cycle. On the other hand, if local congestion depends upon VC/crossbar requests, then local cost can be computed in same cycle. (0)

– *cost_multiplier_local*: weightage for local cost. Local costs at each node are scaled by this factor before computing total cost. (1)

– *cost_multiplier_remote*: weightage for remote cost. Remote costs arrived at by aggregating costs upstream are scaled by this factor before computing total cost. (1)

– *use_max_quadrant_cost*: switch for using maximum congestion as representative for a quadrant, by default performs average (0)

– *cost_precision*: cost precision (0)

– *dim_ave*: perform diminishing average with existing value while updating history FIFO (0)

B.1.3 Visualization and Statistics

It configures visualization module and also defines interval as well as verbosity with which statistics are reported.

– *vis_on*: visualization switch (0)

– *vis_start*: time instant to start visualization (MAX VAL)

– *vis_stop*: time instant to stop visualization (MAX VAL)

– *vis_fifo_type*: buffer parameter selected for visualization (*free_buff* | vc_alloc)

– *vis_link_type*: link parameter selected for visualization (*xbar_gnts* | used_buff | vc_used | xbar_reqs | xbar_demand | xb_buff | link_util | pkt_delay)

– *stats_interval*: statistics print interval (1000)

– *node_stats*: switch to display per-node statistics (0)

– *node_bw_stats*: display switch for node by node offered bandwidth statistics (0)

– *chkpt_interval*: checkpoint interval (1)

– *incr_stats*: switch for printing incremental statistics (0)

B.1.4 Simulation and Trace File

It contains various configuration fields for controlling simulator behavior as well as trace file handling.

– *flit_max*: maximum flit count (1)

– *midpoint_file*: fast forward to the middle of generator file (0)

– *seed*: unique seed (random)

– *netcfg_file*: location of node configuration file (“)

– *tracefile_name*: trace file location (“). Traces are accepted in compressed SPLASH file format [76].

– *incr_chkpt*: switch for check pointing cost variables (0)

– *cost_reg_history*: limit on history depth for maintaining cost (1)

– *warmup_cycles*: number of warm up cycles not included in simulation log (0)

– *max_packets*: maximum number of packets allowed in simulation (0)

– *pkt_throttle*: throttle on packet size to avoid simulator hang-ups (5000)

– *selfsim_trace1*: self similar trace file 1 (“)

– *selfsim_trace2*: self similar trace file 2 (“)

B.2 Local Configuration

Local configuration file specifies node-specific parameters such as source/destination info, mode of routing, connected ports, traffic pattern generation and selection as well as startup clock frequency/voltage level. Node-specific parameters follow a line starting with *node:* declaration and are applicable only for the node number specified. Following is a list of various local configuration fields supported. Valid values are presented in parentheses with default value de-italicized.

– *node:* node name declaring a new node with specified name, remaining lines till next node declaration or EOF to be treated in this node’s context (“)

– *coord*: node coordinates (x and y coordinates specified on two separate lines in that order)

– *is source*: source switch (*true* | false)

– *is_destination*: destination switch (*true* | false)

– *src_type*: source traffic pattern (*rand* | bitcomp | transpose | selfsim | bitrev | hotspot | file)

– *dst_type*: destination traffic pattern (*noblock*)

– *router_type*: type of router (*basic*)

– *xbar_type*: type of crossbar design (*fullcon*)

– *vc_alloc*: VC allocator type (*2level*)

– *xb_alloc*: crossbar allocator type (*2level* | spec_2level | bec_2level | bec_power_2level | bec_spec_2level)

– *clock*: clock frequency divider, divides the root clock frequency by specified value to arrive at clock frequency for each node (1)

– *volt*: voltage level (default Vdd of power model used)

– *rt_algo_type*: routing algorithm (*xydor* | adaptive_xy | o1turn_bec | xydor_bec)

– *rt_sel_type*: routing selection function for choosing output port (bec | o1turn | last_match | *first_match* | first_avail | no_turn | random | stat | local)

– *rt_cost_fn*: routing cost function in computing link cost (*local* | free_vc_nohist | buff_nohist | buff_hist)

– *rt_cost_reg*: routing cost function in computing cost of an output port (*free_vc_nohist* | buff_nohist | xb_demand)

– *rt_cost_mgr*: routing cost manager for aggregating and propagating congestion info (none | *local* | 1D | 1domni | fanin | quadrant)

– *vc_count*: VC count per port (4)

– *vc_classes*: number of priority classes in a VC (1)

– *que_depth*: FIFO queue depth per VC (2)

– *port_count*: number of ports (2)

– *port_dest*: named destination port (“)

APPENDIX C

ACCESSIBILITY AND SUPPORT

While an internal SVN based repository was used for the source code under development, a read only mirror access (with option of allowing collaborators if required) is provided through Google code via following link:

http://code.google.com/p/ocintsim/

For more formalized distribution of the simulator as a downloadable package as well as for future updates, a website was setup for the simulator at following link:

http://www.ece.tamu.edu/~ocintsim/

For technical support and bug fixes, following distribution list can be used:

ocintsim-dev@listserv.tamu.edu

First-time users need to register themselves beforehand by sending an email to *listserv@listserv.tamu.edu* with only following text in its body:

SUBSCRIBE ocintsim-dev firstname lastname

VITA

Name:	Subodh Prabhu
Address:	c/o Department of Electrical and Computer Engineering, WERC, Texas A&M University, College Station, TX 77843, USA
Email Address:	subodh.prabhu@gmail.com
Education:	B.E., Electronics & Communications, Delhi University, 06/2005 M.S., Computer Engineering, Texas A&M University, 05/2010
Work Experience:	Design Engineer I, Imaging, STMicroelectronics, 06/2007- 07/2008 Design Engineer II, Imaging, STMicroelectronics, 08/2005- 05/2007
Website:	http://www.sprabs.com
LinkedIn Profile:	http://www.linkedin.com/in/subodhprabhu