Stephanie Friederich

Automated Hardware Prototyping for 3D Network on Chips

Automated Hardware Prototyping for 3D Network on Chips

Zur Erlangung des akademischen Grades eines

DOKTOR-INGENIEURS (Dr.-Ing.)

an der Fakultät für
Elektrotechnik und Informationstechnik
am Karlsruher Institut für Technologie (KIT)
genehmigte

DISSERTATION

von

Dipl.-Ing. Stephanie Friederich
geboren in Bad Mergentheim

Tag der mündlichen Prüfung:
23. Mai 2017

Hauptreferent: Prof. Dr.-Ing. Jürgen Becker
Korreferent: Prof. Dr.-Ing. Norbert Wehn

Karlsruhe, 01. Dezember 2017

FIRST EDITION

Abstract

More than 50 years ago, in 1965, Intel® co-founder Gordon Moore forecast the development process of transistor technology. He predicted that the number of transistors in integrated circuits will approximately double every two years. His statement is still valid, however an end of Moore's law is on the horizon. Chip performance improvement no longer merely depends on increasing transistor counts. With Moore's law at an end new key elements to increase the processing performance need to be investigated. Two possible approaches for "More than Moore" are 3D integration methods and heterogeneous systems. Likewise, a trend towards many-core processor architectures based on *networks on chips (NoCs)* have emerged in processor development over the last decade.

Besides the end of Moore's law, with shrinking technology sizes below 60 nm, new challenges are arising. Heat dissipation becomes a major issue in large scaled integrated chips. In order to face this problem in modern multi-core architectures, the power dissipation of network resources needs to be reduced without severely affecting the communication performance. The combination of frequency scaling and power gating controlled by hardware for 3D networks on chips, including a *field programmable gate array (FPGA)* prototype are the major topics of this work. Therefore, an existing clock synchronous 2D network has been extended to a three-dimensional asynchronous network with multiple frequency regions. Also, a scalable and low resource approach for online power management has been developed.

Verification of new hardware is the most time consuming task of the development process. To speed up this task and enable early software development, an automated and user-friendly project generator tool has been developed in this work. A graphical user interface to compile the entire tool flow from architecture and parameter definition to simulation, synthesis, and test is part of this tool. In addition, the size of the architecture poses a particular challenge for prototyping. Previous work has failed to address a fast and straightforward prototyping, especially of architectures with more than 50 processor cores. This work includes design space explorations and FPGA based prototypes of different 3D NoCs implementations with more than 80 CPUs. Using the simulation and prototype, it could be shown that the static and dynamic power dissipation of the network can be reduced with a negligible performance loss.

Zusammenfassung

Vor mehr als 50 Jahren stellte Intel® Mitbegründer Gordon Moore eine Prognose zum Entwicklungsprozess der Transistortechnologie auf. Er prognostizierte, dass sich die Zahl der Transistoren in integrierten Schaltungen alle zwei Jahre verdoppeln wird. Seine Aussage ist immer noch gültig, aber ein Ende von Moores Gesetz ist in Sicht. Mit dem Ende von Moore's Gesetz müssen neue Aspekte untersucht werden, um weiterhin die Leistung von integrierten Schaltungen zu steigern. Zwei mögliche Ansätze für "More than Moore" sind 3D-Integrationsverfahren und heterogene Systeme. Gleichzeitig entwickelt sich ein Trend hin zu Multi-Core Prozessoren, basierend auf *Networks on chips (NoCs)*.
Neben dem Ende des Mooreschen Gesetzes ergeben sich bei immer kleiner werdenden Technologiegrößen, vor allem jenseits der 60 nm, neue Herausforderungen. Eine Schwierigkeit ist die Wärmeableitung in großskalierten integrierten Schaltkreisen und die daraus resultierende Überhitzung des Chips. Um diesem Problem in modernen Multi-Core Architekturen zu begegnen, muss auch die Verlustleistung der Netzwerkressourcen stark reduziert werden. Diese Arbeit umfasst eine durch Hardware gesteuerte Kombination aus Frequenzskalierung und Power Gating für 3D On-Chip Netzwerke, einschließlich eines FPGA Prototypen. Dafür wurde ein Takt-synchrones 2D Netzwerk auf ein dreidimensionales asynchrones Netzwerk mit mehreren Frequenzbereichen erweitert. Zusätzlich wurde ein skalierbares Online-Power-Management System mit geringem Ressourcenaufwand entwickelt.
Die Verifikation neuer Hardwarekomponenten ist einer der zeitaufwendigsten Schritte im Entwicklungsprozess hochintegrierter digitaler Schaltkreise. Um diese Aufgabe zu beschleunigen und um eine parallele Softwareentwicklung zu ermöglichen, wurde im Rahmen dieser Arbeit ein automatisiertes und benutzerfreundliches Tool für den Entwurf neuer Hardware Projekte entwickelt. Eine grafische Benutzeroberfläche zum Erstellen des gesamten Designablaufs, vom Erstellen der Architektur, Parameter Deklaration, Simulation, Synthese und Test ist Teil dieses Werkzeugs. Zudem stellt die Größe der Architektur für die Erstellung eines Prototypen eine besondere Herausforderung dar. Frühere Arbeiten haben es versäumt, eine schnelles und unkompliziertes Prototyping, insbesondere von Architekturen mit mehr als 50 Prozessorkernen, zu realisieren. Diese Arbeit umfasst eine Design Space Exploration und FPGA-basierte Prototypen von verschiedenen 3D-NoC Implementierungen mit mehr als 80 Prozessoren.

Contents

1 Introduction

The fundamental goal in processor development is the maximization of processing performance. This can be achieved for example by increasing transistor counts and by higher frequencies. Subsequent of Moore's law [74] transistor count of integrated circuits approximately doubles every two years.[1] Despite the fact that this statement is more than 50 years old, it is still valid. The expansion of transistors integrated into one die can be demonstrated by the development of Intel® processor progression, see figure 1.1. Frequency increase however is limited due to physical conditions and following new opportunities have been explored. The most important one in recent years has been the introduction of multi-core processors with two or more independent processing units integrated into a single chip.
To explore further opportunities in processor development, the *International Technology Roadmap for Semiconductors (ITRS)* presents the industry-wide consensus on the estimate of the industry's research and development needs up to a 15-year horizon. It provides a guide to the efforts of companies and research institutions focusing of semiconductors. The report forecasts an end of Moore's law within the next years [105], since technology cannot further be scaled down. Following the statement that performance can be increased by increasing the transistor count is not valid anymore. However, the report also already presents a solution for further performance increase besides Moore's law, called "More than Moore". In modern processor chips, performance depends on other factors besides transistor count. The ITRS focus on five long-term goals to face problems beyond Moore's law. (1) Implementation of advanced muti-gate structures also beyond *complementary metal-oxide-semiconductor (CMOS)*; (2) implementation of new memory structures, with on chip memory sizes in the terabyte range; (3) reliability; (4) power scaling; and (5) integration for functional diversity which is also referred to as 3D integration. For more information see the latest ITRS report from 2013 [53].

[1] Depending on the reference, the time varies between 18 month and two years.

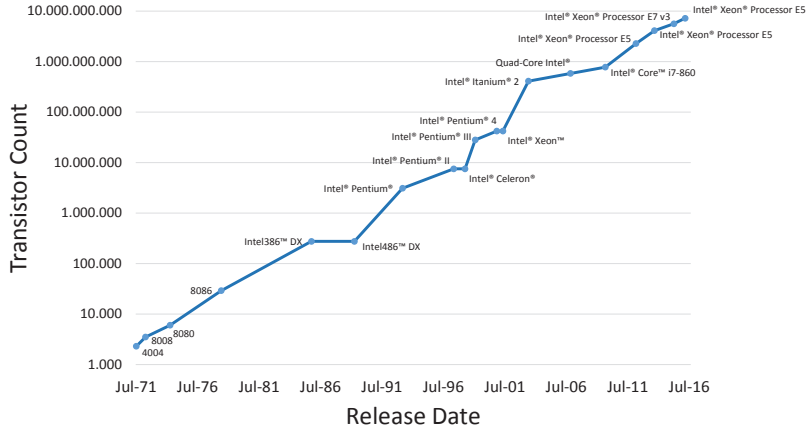


Figure 1.1: Transistor scaling on the basis of Intel® processor progression [1]. According to Moore's law the transistor count is doubled approximately every two years.

1.1 Motivation

The fundamental goal is to be able to continue the increase of overall system performance. Since there is an end of higher integration density and higher frequencies in sight, other parameters need to be considered for performance enhancement. Optimized parallel task execution is one possibility described by Intel® [48]. Other eligible parameters are for example new gate structures and 3D integration techniques. These parameters are considered as long term challenges by the ITRS report [53]. However, increasing integration and hence the higher device density also brings limitations with it. Major problems are power and thermal issues where standard cooling techniques are no longer sufficient. CPU power density increased almost exponentially over the last decades , but flattened with the introduction of multi-core processors [83]. Current CPU power density amounts to approximately 100 W/cm^2 (see figure 1.2). The high power density leads to high chip temperatures in direct proportion. With higher temperature values system reliability decreases or results in unrepairable damage. The development of energy-efficient circuits and power saving techniques therefore is indispensable.

With the introduction of tera-scale computers comprising hundreds of cores, simple bus based systems to connect the cores are no longer sufficient. To reduce

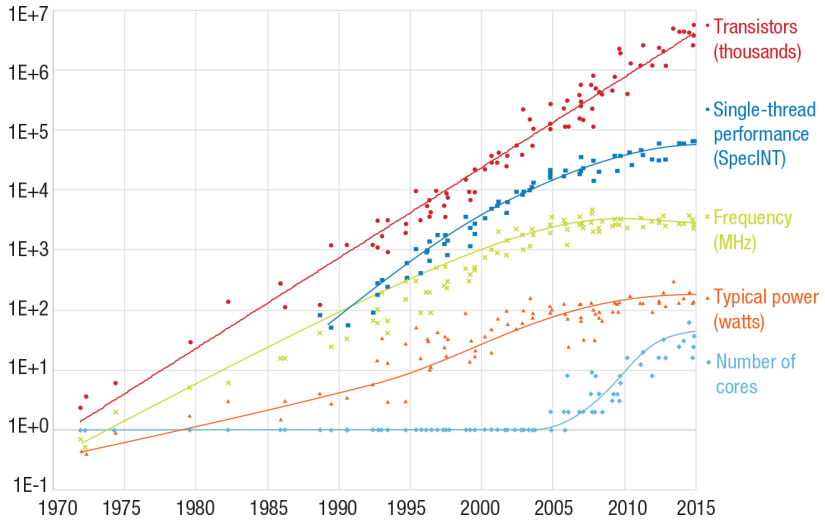


Figure 1.2: Performance enhancement by increasing the frequency ended in 2005. However, Moore's law for transistor count (red line) still continues [23].

communication latency an efficient and high-performance communication infrastructure is mandatory. Networks on chips represent a promising solution for future multi-core systems with hundreds of cores. To meet performance and power constraints of novel multi-core architectures the development of the methods must be extended by the network layer.

1.2 Contribution

Silicon integrated circuits characterized the electronic development of the 21st century. Gordon Moore predicted in 1965 that the transistor count per integrated circuit will double approximately every two years. Concurrent to the growing number of transistors, the lithography process scales linearly, shrinking down to 5 nm node in the latest technologies. However, linear technology scaling will soon become impossible, due to physical restrictions. Three dimensional chip design is the emerging technology to continue the increase of performance of processors, laptops, mobile phones, and other electronics. At the same time, bus

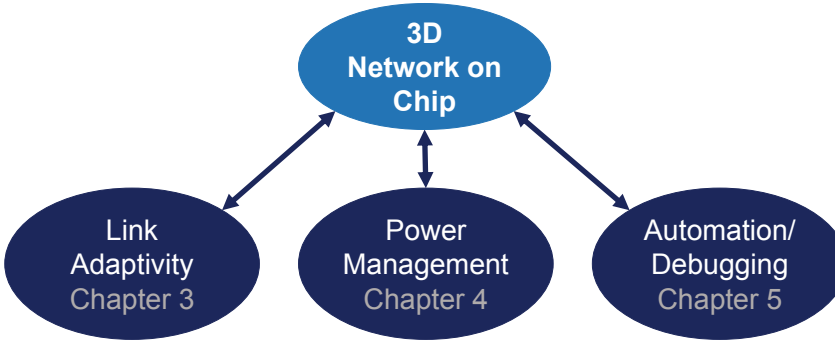


Figure 1.3: Extension and optimization of 3D networks on chip.

based multi-core designs, have been replaced by network on chips.
3D designs face challenges in both manufacturing technology and physical design. This work focuses on the physical design challenges, including vertical interconnects and power management. In addition, special emphasis is placed on testability of these new designs.

The basic router design and other network components, like the network adapter have been designed as part of the work of the *InvasIC* project [7], Heisswolf [45], and Zaib [114]. The design has been developed to be scalable, but with growing architecture sizes new challenges are arising, such as excessive power dissipation and associated heat generation. Also, new architectures have mainly been verified using *hardware description language (HDL)* simulations, without testing the associated software.

This work focuses on the extension of the basic router designs so that it can be applied to the latest technology updates. In order to verify the novel developments a multi-FPGA prototype including a tool for straightforward architecture design and verification is part of this work. Parallel software development and software tests on real hardware become possible.

The extension of chip technology to a third dimension generates new network topology opportunities. This work includes a design space exploration of 3D networks on chips with varying parameter setup. The 3D extension of the network router includes among others pin multiplexing to face problems with restricted cross layer connections, while ensuring that the network performance is not affected. Also, the architecture has been divided into multiple clock regions. The

clock domain crossing is integrated in the router input buffers to keep the resource overhead as small as possible.

Since future high integrated chips will struggle with high power dissipation, a power management unit for the network resources has been developed. The power management includes fine and coarse grained power gating techniques as well as frequency scaling opportunities to optimize power dissipation. One of the major aspects of effective power management is to keep the resource overhead of the controller to a minimum. Otherwise, it would hardly be possible to reduce the overall power consumption of the design if the controller itself consumes much power. To keep the design scalable one power management unit is integrated into each router.

For system evaluation and analysis of the effects of the extensions, an FPGA based prototype has been created. To provide prototypes with over 50 processor cores, a prototyping platform with multiple FPGAs has been used. Thereby, the design partitioning across the FPGAs presents the major difficulty. An automated tool flow enables fast and easy compilation of designs with varying parameters. This provides the opportunity for design space explorations and enhanced debugging and tests. To cope with the limited number of debug interfaces a transactor based debugging interface has been introduced.
The *project generator tool* offers a complete automated tool flow supported by a graphical user interface. Thus, even developers without extensive hardware knowledge can use the tool to build custom designs. The tool automatically builds a simulation and a design for an FPGA based prototype.

1.3 Outline

This work is organized as follows. Chapter 2 reviews the background and gives an overview of the fundamentals discussed in this work. Section 2.1 describes new technologies to cope with the increasing number of components integrated into one chip and the associated increasing transistor count. One of the latest trends in this domain are multidimensional chips. On the one hand they can solve the steadily growing demand of transistors per chip, but on the other hand new challenges are arising. Section 2.2 describes the basics of multidimensional chips and the resulting issues, such as connections between silicon layers and thermal problem caused by power dissipation. The fundamentals of power models are introduced in the following section 2.3.

Basics of the integration of multiple components into a single chip are described in section 2.4. Networks on chips are introduced in section 2.5. At first basic definitions of on chip networks are described, followed by evaluation criteria.

This section concludes with an overview of existing NoC based multi-processor systems. The context of this work, invasive computing, is presented in section 2.6. Enclosed in this section is the description of the hardware layer of the NoC based multi-core architecture as well as a short description of the software layer. The last section in this chapter generally describes hardware development of digital circuits and the hardware description languages used in this work (see section 2.7.1).

The essential part of this work are adaptive 3D on chip networks, power management as well as process automation and debugging (see figure 1.3). The state of the art is subdivided into three sections (3.2, 4.2, and 5.2) according to the content of chapter three to five.

Chapter 3 introduces three-dimensional *networks on chips (NoCs)* comparing different topologies in section 3.1. Section 3.2 summarizes the state of the art of multi-dimensional network technology. The implementation of a heterogeneous bandwidth router design is described in section 3.3. It includes a concept to cope with restricted number of *through-silicon vias (TSV)*. The design includes a clock domain crossing, so that additional synchronization between the chip layers gets redundant. Section 3.4 evaluates the 3D NoC concept in terms of resource consumption and performance gain.

Power optimization methods for on chip networks are presented in chapter 4. Section 4.2 summarizes the state of the art of on chip power optimization. The main part of this chapter illustrates the development of a hardware power management controller, introduced in section 4.3. Furthermore, results are given in section 4.5.

Chapter 5 pictures the hardware development process. The first section in this chapter gives a general overview of system development life cycles and introduces agile methods for hardware development. The succeeding section summarizes the state of the art. The proposed prototyping strategy is illustrated in section 5.3 (simulation) and in section 5.4 (FPGA prototype). Special emphasis is placed on the extension of debug interfaces which is realized by using transactor based interfaces. The system validation and testing concept is described in section 5.5. A user friendly tool for automated design generation, synthesis, and test is introduced in section 5.6.

Finally, this work is concluded in chapter 6. It also gives a brief outlook on future work in section 6.2.

2 Fundamentals

This chapter gives a general introduction to *system on chip (SoC)* designs and *networks on chips (NoCs)* including a listing of recent NoC realizations. The focus is on the invasive multi-processor architecture which builds the basis of the following extensions and optimization technologies. Also, this chapter outlines the challenge one is facing in current *very large scale integrated (VLSI)* designs.

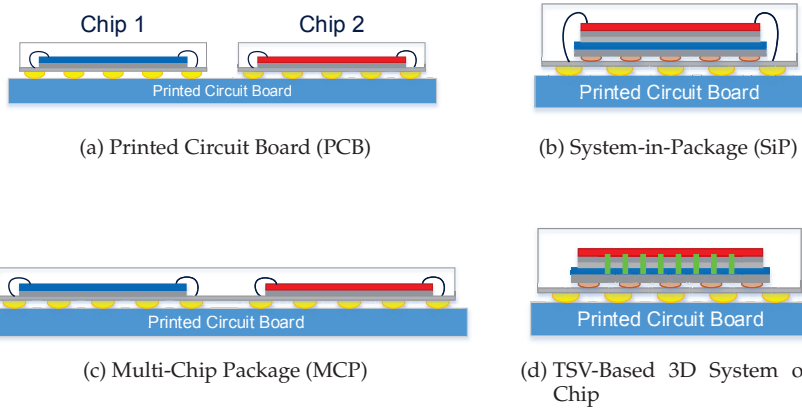
2.1 More than Moore

Moore's law describes the continuing shrinking of physical feature sizes of chips to reduce cost and especially to improve performance. With technology sizes around 2 nm, the distance between structural elements only measures 10 atoms. At that size electron behavior is not stable anymore due to quantum uncertainties. The production of reliable devices is indispensable and consequently comes to an end. However, beyond the shrinking of feature sizes, there are other possibilities to improve performance which are then outlined as More than Moore.
The concept of functional diversification addresses among others heterogeneous integration of separately manufactured components into one die. These *system in package (SiP)* offer enhanced performance beyond the scaling following to Moore's law.

Higher integration levels, functional density benefits, and power management are the main roadmap aspects. The trade-off between performance increase and power reduction thereby plays an important role [12, 105].

2.2 Multi-Layer Chip Technology

Following Moore's law to increase the amount of transistors per unit area, bare or packaged chips are stacked together in vertical direction. The vertical stacking of chips is especially valuable in space constrained environments like cell phones, since it reduces the dimension of the attached *printed circuit board (PCB)*.
Another advantage of multi-layer chips designs, several different technology layers can be combined, while in two-dimensional chips the complete layer has



(a) Printed Circuit Board (PCB)

(b) System-in-Package (SiP)

(c) Multi-Chip Package (MCP)

(d) TSV-Based 3D System on Chip

Figure 2.1: Four versions of multi-chip system integration.

to be processed in one technology. However, there are different technologies optimized for memory, low-power or for high speed processing. Hence, single chip layer designs are specialized for one aspect only. On the other hand, in three-dimensional designs the optimal process technology can be chosen for each design module, separated into multiple layers. Following designs with huge memory combined with low-power processing units are possible.

Figure 2.1 shows four different realizations of multi-chip designs. The most simple design with multiple chips are PCBs where different integrated circuits are soldered onto one board (see figure 2.1a). In this multi-chip design, the integrated circuits are separated from each other building multiple packages.

If multiple chips are integrated into one package format, they are called *system in package (SiP)*. Depending on the composition of the chip layers and the wiring, different versions of SiP can be built. A *Multi-Chip Package (MCP)* contains two or more chips, combined into one package. These chips are either located side by side on one substrate or they can also be stacked on top of each other. Each chip is wire bonded to the substrate which is attached to the PCB. An example implementation is shown in figure 2.1c.

Compared to the MCP design shown in figure 2.1c, the SiP shown in figure 2.1b has a different chip connection type. In figure 2.1b, the bottom chip is connected via flip chip solder bump connections while the upper chip is wire connected.

The fourth version of multi-chip designs is shown in figure 2.1d where the different chips are connected through *through-silicon vias (TSV)* (green lines in the figure). A detailed description of TSV technology is given in the following section.

2.2.1 Through-Silicon Via (TSV)

Stacked chips can be connected vertically through short interconnects, called *through-silicon vias (TSV)*. The number of TSV depends on the chip dimensions and the technology. The pitch size of TSV is in the micro meter range [27], while the pitch size of on chip connections is one dimension smaller [79]. Hence, the amount of TSV in a given chip region is limited and often the bottleneck of 3D chip designs. The amount of TSV per die cannot be directly calculated by dividing the die area through the TSV area, since there must be a gap between two TSV. Figure 2.2 illustrates the dimensional difference between a TSV and transistors. Electrical characteristics of TSV such as delay, resistance, capacitance, and inductance are important for the design of 3D ICs. In order to analyze the electrical behavior of 3D designs, a model of the design is necessary. Compact models for physical TSV constraints are presented in [60, 108]. Especially if the TSV density rises, capacitive coupling to neighboring TSV is possible.

If a multi-processor design, connected through a network, is partitioned across several chip layers, the partitioning is usually done at the router links. However, still there are more connections required than can be built up in TSV connections. The partitioning and handling of restricted vertical connections hence poses a challenge to new implementations of multi-layer designs. There are several solutions to cope with limited inter layer connections described in literature. For example only a portion of routers are connected across the chip layers [103] or the bandwidth of each router link is reduced. However, in both cases it results in a larger average packet latency.

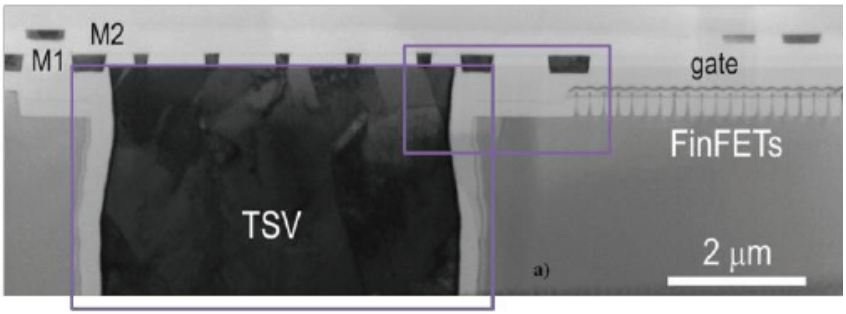


Figure 2.2: Cross sections of a TSV and a *Fin Field Effect Transistor (FinFET)* device at close proximity. The FinFET device has 40 nm height and 20 nm width while the TSV diameter (5 μm) is greater by a factor of 100 [42]. Copyright © 2012, IEEE.

2.2.2 Heat Distribution

Besides restricted TSVs, the heat distribution of modern high integrated designs poses a challenge. The electric power consumed by the chip is converted into thermal energy. To prevent overheating, the generated heat needs to be spread. In single layer designs, the cooling system is usually directly connected to the chip substrate and as a result the thermal energy can be easily dissipated. In three-dimensional chip designs, layers in the middle of the chip are no longer directly connected to the cooling system since the space between two layers is usually not sufficient to insert cooling mechanisms in between. Hence, stacking multiple dies on top of each other results in a higher power generated per unit surface area. Thus, thermal management is strongly needed in 3D designs.
The maximum junction temperature of a stacked chip varies with the number of chips stacked on top of each other in the package. Lau et. al. [65] analyzes the junction temperature and thermal resistance of 3D stacking. With a maximum temperature of 85 $^{\circ}C$, the maximum number of stacked chips is seven if the power is dissipated uniformly over the whole chip. The gap between a pair of stacked chips and the chip thickness play an important role for the thermal performance.

2.3 Power model fundamentals

This section shortly presents the fundamentals of power consumption of integrated circuits.

2.3.1 Definitions

The power consumption of CMOS circuits can be divided into two parts, static and dynamic power. First, static power is dissipated in the absence of cell switching activity. In some references the static power is refereed to as leakage power. The second part of overall power consumption, dynamic power, is caused mainly by switching activities.

$$\begin{aligned} P_{\text{system}} &= P_{\text{static}} + P_{\text{dynamic}} \\ &= P_{\text{static}} + P_{\text{internal}} + P_{\text{switch}} \end{aligned} \tag{2.1}$$

Figure 2.3 shows a CMOS inverter circuit, including the different currents which flow in this elementary circuit.

2.3.1.1 Static power

The leakage power of a chip is calculated by adding the leakage power dissipation of all cells. Hence, the static power depends on the number of cells per chip as well as the library which is used to design the chip. The static power is independent of the circuit's activity. The static power dissipation is mainly caused by source-to-drain subthreshold leakage (I_{leak}).
A smaller part is contributed by current leaking between the diffusion layers and the substrate. For this reason static power dissipation is often called leakage power. Equation 2.2 presents the calculation of the static power. The cell leakage power $P_{\text{Cell leakage i}}$ is specified in the data-sheet of the particular standard cell library. The leakage power increases as thickness of gates reduces. The static (or leakage) power is expressed in equation 2.2.

$$\begin{aligned} P_{\text{Cell leakage i}} &= V_{dd} * I_{leak} \\ P_{\text{Leakage total}} &= \sum_{\text{Number of cells}} P_{\text{Cell leakage i}} \end{aligned} \tag{2.2}$$

On the occasion of shrinking transistor sizes, different types of problems can be identified, all causing increasing static power dissipation. The leakage current I_{leak} depends exponentially on the difference between the gate-source voltage V_{GS} and the threshold voltage V_T of a logic cell. Hence, if system power supply voltage is scaled down to limit the dynamic power dissipation, the leakage power grows exponentially. Accordingly, with shrinking technology sizes and hence shrinking threshold voltage V_T, the static power gains an increasing share of the total system power P_{system}. Also, the amount of gate leakage increases dramatically with register sizes below 65 nm.

The leakage power of a design can be calculated using tools such as Synopsys® *PowerCompiler*. These tools can also calculate the dynamic power consumption for given runtime scenarios. Section 4.1.1 describes the calculation of dynamic power which was applied in this work.

2.3.1.2 Dynamic power

The dynamic power is composed of the sum of switching power and internal power and is dissipated when the circuit is active.
Every logic transition dissipates energy when switching between zero and one states. The following equation describes the energy dissipated per transition with

C_L as the load capacitance. The load capacitance corresponds to the loads driven by the outputs of the logic gates and depends on the type and length of the wires.

$$E_C = \frac{1}{2} C V_{dd}^2 \tag{2.3}$$

The internal power dissipation is caused by short circuit current when both the NMOS and PMOS are gating and there is a direct path between V_{dd} and GND. This occurs during switching from 1 to 0 and from 0 to 1. The internal power is defined in equation 2.4. As the system clock frequency increases consequently, short circuit power dissipation increases in equal measure.

$$P_{internal} = V_{dd}^2 * I_{SC} * f \tag{2.4}$$

The dynamic power of a gate or a connection line which is dissipated during charging or discharging is defined in equation 2.5. Often this is also called the transient power consumption.

$$P_{switch} = \alpha * C_{pd} * V_{dd}^2 * f \tag{2.5}$$

With:

α	Average number of 0 to 1 transitions in one clock cycle (referred to as switching activity).
C_{pd}	Dynamic power dissipation capacitance
V_{dd}	Power supply voltage
f	Clock frequency
I_{SC}	Short circuit current

The internal energy E_{int} and the load capacitance C_L of each cell are given in the library data-sheet. The switching activity can be determined from design simulations. If no simulation is available, the switching activity is set to one signal switch every ten clock cycles. Simulation tools such as ModelSim® are used to generate the switching activity information of the design, in the format of *Switching Activity Information Format (SAIF)* files. The simulation testbench thereby specifies the design behavior.
In addition to *register-transfer level (RTL)* simulation, gate-level simulation can be used to generate power consumption information. Supplemental, technology library information is integrated into the simulation process.
The *PowerCompiler* by Synopsys® tool flow can be used for a design space explo-

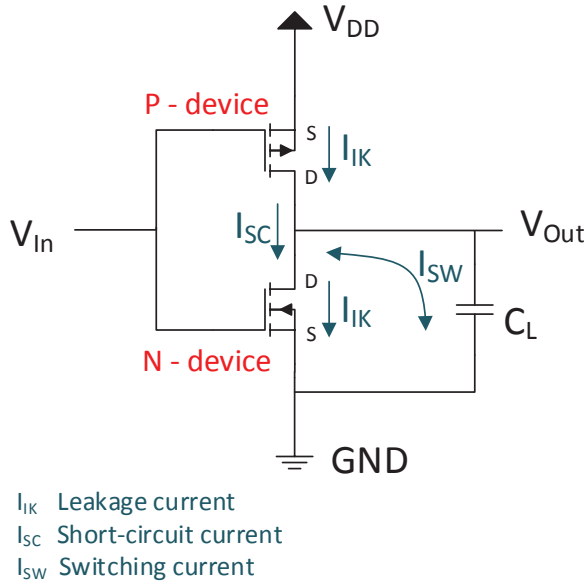


Figure 2.3: Static CMOS inverter.

ration of different architecture settings. For easy reuse the tool flow is written down in a script file.

The goal in low power designs is to reduce the consumed power by limiting the switching energy. This can be achieved by one of the following four options.

1. Reduce the number of clock transitions (clock gating) –> Only possible if components are not used.
2. Reduce V_{dd} –> This will also limit the clock speed and hence reduce the performance.
3. Reduce the number of circuits –> However, less transistors lower the performance.
4. Reduce node capacitance –> One reason why processes are scaled down.

2.3.1.3 CMOS inverter

Figure 2.3 shows the circuit diagram of a CMOS inverter. With the help of this example, the power dissipation of integrated circuits can be exemplary stated, since the CMOS inverter is a nucleus part of all digital designs. The behavior of all other gates can be deducted from the results given by the inverter.
A detailed description of integrated circuit designs and its power consumption can be found in [86, 92].

2.3.1.4 Wake-up Latency

After electric devices are power gated and switched on again, they need some time after wake-up until they are fully functional again. This time is defined as wake-up latency. The wake-up latency depends on the power gating technology and on the depth of the sleep state of the gates. One possibility to power gate CMOS circuits is the insertion of sleep transistors. Tovinakere et al. [99, 100] describe in their work a model to estimate the wake-up latency for circuits power gated with sleep transistors. It that case a PMOS sleep transistor is inserted between the supply voltage (V_{dd}) and the node of the circuit where the supply voltage would usually be connected. Hence, the wake-up time is equivalent to the switching time of a PMOS transistor. Depending on the technology and transistor size, the wake-up time varies between 40 ns and 0.26 ns. However, the sleep transistors are larger than the rest of the circuit since they need to handle the required amount of switching current. If the power gating is part of the power distribution network instead of being part of the standard cell, the switching time will be higher.

2.4 Systems on Chip

Systems on chips (SoCs) are devices which comprise several parts on one chip, instead of distributing the parts across a PCB. Components which are integrated into a single die are for example CPUs including embedded software, memory, and peripherals. SoCs are designed for specific purposes, same as *application-specific integrated circuit (ASIC)* designs. Since ASIC designs do not necessarily include a CPU, SoCs can be seen as a subset to ASICs. SoC designs are widely used and part of every day's life. They are implemented in consumer electronics, mobile devices, as well as in automotive industry products.

The growing demand of computing resources encourages the integration process according to Moore's law. With increasing design complexity and components

integrated into one chip, the communication amount grows and a more advanced communication infrastructure than simple bus based systems is required. On chip networks present one possible solution to this problem, where heterogeneous nodes are connected through routing interfaces.

2.5 Network on Chip

The growing size of multi-core systems and especially the increasing number of cores integrated into one chip leads to a high amount of communication. In the past, all cores in multi-core architectures have been connected by bus systems to communicate with each other and to get access to memory and peripherals. However, bus based systems do not scale well and with an increasing number of cores other connection types need to be considered. Tiled multi-core systems connected by an on chip network close this performance gap since network structures scale very well [18, 29, 37].

There are several design opportunities to create on chip networks. All have in common that the heart of the communication infrastructure are routers connected with each other. In addition, each router is connected to one or multiple cores. Section 2.5.4 gives a brief introduction to available implementations of multi-core systems based on network architectures. Beforehand, fundamentals about communication infrastructure and network topologies and characteristics are given.

2.5.1 Communication Infrastructure and Amdahl's law

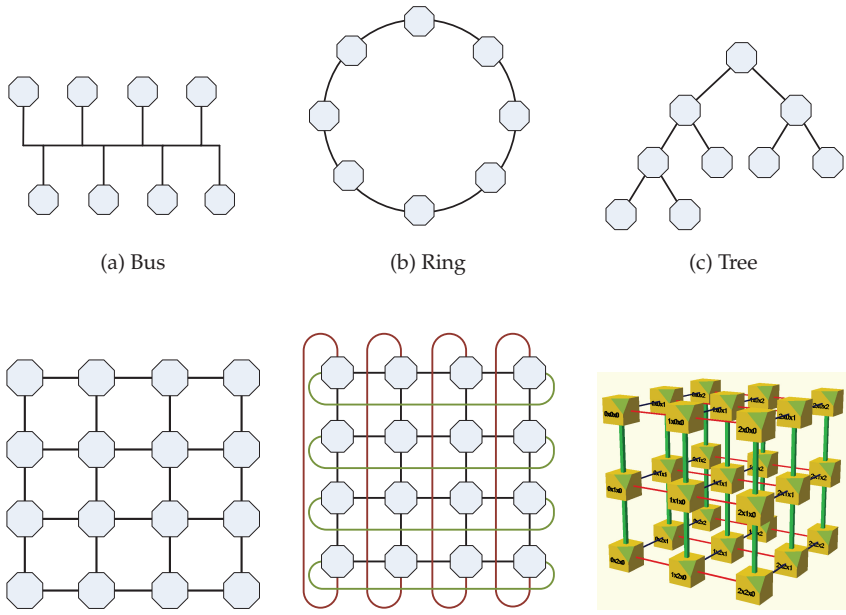
More than half a century ago Gene Amdahl investigated the execution time speed-up of parallelization in multi-core systems [11]. At that time, single processing elements have reached the limits of speed-up and first implementations of multi-core systems have been explored. Amdahl's law is a model for the relationship between the expected speed-up of parallelized implementations of an algorithm, relative to the serial algorithm, under the assumption that the problem size remains the same when parallelized. Equation 2.6 describes the expected speed-up defined by Amdahl's law.

$$\begin{aligned} &Speedup(r_p, n) = \frac{1}{r_s + \frac{r_p}{n}} \\ &With \quad r_s + r_p = 1 \end{aligned} \tag{2.6}$$

Where r_s represents the ratio of the sequential portion in one program and r_p represents the ratio of the parallel portion in one program. n represents the number of processors, without any restrictions in this equation. However, software cannot be parallelized infinitely.

2

Amdahl's law claims that even the problem size is fixed, independent of the number of processors. Gustafson [43] presented in his work, that the problem sizes scales with the number of processors used for parallelization. However, the main problem of Amdahl's law is that the communication is not considered, which means that the speed-up does not increase as stated in equation 2.6. With an increasing number of cores attached to one interconnection, the interconnect becomes a bottleneck and following the speed-up curves need to be adapted. Also, modern heterogeneous chip architectures are not considered in Amdahl's law [50]. Besides all the drawbacks, Amdahl's law is still valid in a sense.



(a) Bus (b) Ring (c) Tree

(d) 2D mesh (e) 2D torus (f) 3D mesh

Figure 2.4: Two and three-dimensional network topologies.

2.5.2 Network Topologies

This section describes some basic network topologies. The presented network topologies follow some basic rules. The topology must be simple, otherwise comprehensive routing strategies are required. Also, they need to be scalable, so that the size is not restricted. The choice of an appropriate network topology is important with regards to resource consumption and communication latency. Network topologies can be divided into direct and indirect networks. In direct networks, the routing element is directly connected to a limited number of neighboring routing elements as well as a limited number of *processing elements (PEs)*. In indirect networks the connections to neighboring routing elements are not fixed. With both networks, the communication topology can change due to the application demands. The most popular implementations of an indirect network is a bus based structure. Other examples are multistage networks and crossbar switches. Since indirect networks do not scale well, they will not be considered in the following.

To compare the network topologies with each other, specific network parameters will be examined. The distance of communication is defined as the minimum number of routers that a packet has to pass from the source to the destination node. The network diameter is defined as the maximum shortest path length between any pair of nodes in the topology. It indicates the worst case message latency. Depending on the routing algorithm it is possible that the packets take an even longer route through the network, than the network diameter.

Another indicator for communication latency is the average distance between nodes. The average distance of all packets in the network rely on the traffic pattern. For example, equation 2.8 defines the average distance in a 2D mesh network applying XY routing.

Mesh based NoCs are becoming more and more popular due to their simple routing strategies (XY routing) and their scalability. To reduce the network diameter in 2D networks, three-dimensional network implementations will be introduced as well. The following sections describe the network topologies implemented in this work. Existing realizations of network topologies are described in chapter 2.5.4.

2.5.2.1 Mesh Networks

The basic definition of mesh networks states that each node is connected to its neighbors. In this work only rectangular $(dim\_x * dim\_y)$ mesh networks are considered. An example implementation is shown in figure 2.4d. The mesh network diameter and average distance is given in the following equation.

$$Diameter_{2Dmesh} = (dim\_x - 1) + (dim\_y - 1) \tag{2.7}$$

$$\Delta_{2Dmesh} = \frac{dim\_x + dim\_y}{3} \tag{2.8}$$

Due to the simple structure of mesh based networks, simple routing strategies such as XY routing can be applied. Beyond, the architecture scales very well. Most of the current available multi-core chips including a communication network are based on mesh networks.

2.5.2.2 Torus Networks

Torus networks have a similar structure as mesh networks, but in addition to mesh networks, torus networks are symmetric. Hence, every router has the same number of neighbors it is connected to. Due to higher connectivity, torus networks have lower latencies compared to mesh networks. However, the higher connectivity requires more connections which need to be realized on the chip. The diameter of a torus network with even x and y dimensions, is given in equation 2.9. If the x dimension is an odd number, dim_x must be replaced by $(dim\_x - 1)$ in this equation. The same applies respectively for dim_y.

$$Diameter_{2Dtorus} = \frac{dim\_x}{2} + \frac{dim\_y}{2} \tag{2.9}$$

At a first sight it looks like there are quite long connections in a torus network, connecting one edge router with a router at the opposite edge (see figure2.4e). However, with an intelligent placement of the routers, a maximum length of a two router hop can be realized. An example placement of a 4x1 torus network is shown in figure 2.5.

2.5.2.3 Multidimensional Networks

Multidimensional networks are built up of multiple network layers. Compared to 2D network designs, the average distance between nodes is smaller in 3D networks. Mesh based networks have been a popular topology for 2D NoC implementations and they can easily be extended to a third dimension. An example 3D mesh network is shown in figure 2.4f.

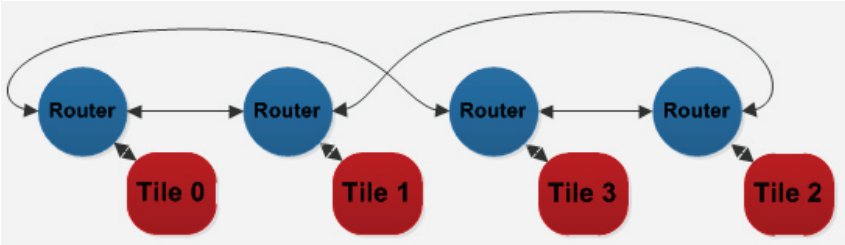


Figure 2.5: Router placement of a 2D Torus design to prevent long connections.

2.5.3 Network Evaluation

Besides the network diameter and the average distance between nodes, the following section describes further characteristics of a network. Pande et. al. [78] discuss network performance evaluation and design trade-offs with regard to latency, throughput, energy dissipation, and silicon area requirements. The same parameter are used later in this work to evaluate the novel network extension developments.

2.5.3.1 Communication Latency

Network message latency is defined as the time elapsed from the moment a packet header flit is injected into the network until the moment the packet tail flit leaves the network again. In the remainder of this work this is referred to as packet latency. Depending on the source and destination node as well as the routing algorithm the packet flits passes several router nodes on its way through the network. Each router which needs to be passed by a packet adds additional latency. Hence, the goal is to minimize the distance between source and destination node to minimize the packet latency. In contrast to the throughput measurements, the latency examines entire packets instead of single flits, since virtual channel allocation and flow control are performed on the granularity of packets.
For network evaluation the average packet latency (L_{avg}) is calculated according to the following equation:

$$L_{avg} = \frac{\sum_1^P L_i}{P} \tag{2.10}$$

In this equation P describes the total number of packets transmitted into the network, and L_i is defined as the latency of packet *i*. L_i is composed of the sum of the latency added by each router the packet passes. The latency L_{avg} and also L_i depend on the allocation of source and destination of a packet. Since the focus is not on application allocation, the packet latency of a single router is used to assess the network performance.

Definition. *The packet transmission latency used for performance evaluation in this work is defined as the number of cycles elapsed between the arrival of a packet header at a router and the time the packet tail leaves the same router.*

2.5.3.2 Throughput

Instead of characterizing the network by its bandwidth (bits/s), the network throughput can also be used for performance evaluation. The throughput *TP* is defined as the maximum amount of information delivered per clock cycle [32]. Since the size of a message packet can vary but the channel width is equal for all designs, the throughput is measured in flits per clock cycle. In case of this work a normalized throughput is used by dividing it by the size of the network (number of network nodes or routers). As a result, the unit of the normalized throughput is flits per clock cycle per node (see equation 2.11). Accordingly, a throughput (TP) equal one corresponds to scenario where each router in the network transmit one flit per clock cycle.
The total time for equation 2.11 is measured starting when the first flit enters the network till the last flit reaches its destination.

$$TP = \frac{number\,of\,transmitted\,flits}{(number\,of\,nodes) * (total\,time)} \tag{2.11}$$

To explore the network characteristics the throughput is usually measured with varying injection rates. The injection rate is defined as the rate at which flits are inserted into the network from a local processing node. For simulations and stress tests the flits are generated synthetically. In a normal design the data is generated by the processing node and the network adapter transfers the data into network flits. An injection rate of 0.25 hence indicates that at the local port of a router a new flit is inserted in one out of four clock cycles. Depending on the traffic pattern which should be simulated, the injection rate can be a constant value or following a process. The following section describes some traffic patterns which have been used in this work for evaluations.

2.5.3.3 Traffic Patterns

For performance evaluation of the network, different traffic patterns have been applied. Depending on the source and destination distribution of messages, they can be grouped into synthetic and realistic patterns. Synthetic traffic patterns refer to abstract network communication models. They are used to generate the worst case scenarios for the network, while realistic traffic emulates the communication behavior of real applications.
The purpose of using different traffic patterns is to test different network characteristics. Depending on the message length, injection rate and message destination distribution, the performance of the network can vary heavily.
The traffic patterns which have been used in this work are described as follows.

Synthetic Traffic Synthetic traffic does not depend on any real application, but is rather used to emulate the worst case scenarios for the network. However, synthetic traffic can also be used to mimic real traffic patterns without any computational effort. For simulations, it is then possible to simulate and evaluate the network without CPUs, caches and applications. In the following, three popular synthetic traffic patterns are described.

Uniform Random Traffic: Each node sends data to random source nodes with equal probability. The amount of flits sent per cycle and node is defined as injection rate. The injection rate is a variable value which can be set in the testbench. With increasing injection rates, the network can be tested for congestion.

Hot Spot Traffic: This traffic pattern is based on uniform random traffic but some specific nodes (called Hot Spots) receive packages with a higher probability than other nodes. The Hot Spot node represents a very busy node, for example nodes connected to the main memory.

Transpose Traffic: There are two implementations of transpose traffic. First, a node (i,j) only sends messages to node (n-i,n-j) where n is defined as the network diameter. Second, each node sends messages to a node with an address of the reversed dimension index. For example in a 2D mesh network, node (i,j) only sends messages to node (j, i). Since the source and destination need to be two different nodes, node (i,i) send messages to node ($n - i + 1$,$n - i + 1$), where n presents the network diameter.

Realistic Traffic The more accurate way to evaluate the network performance is the usage of real applications. Simulation models usually do not comprise the processing elements, hence the traffic of real applications is only modeled. Depending on the distribution of the application across different nodes, the communication traffic strongly varies. Consequently, nonuniform traffic distributions are more likely.

2.5.4 Reference NoC Architectures

There are already several embedded NoCs on the market. A subset is explained in detail in the following. For additional information see [45].

2.5.4.1 Tilera®

The Tile processor is a multi-core design manufactured by Mellanox Technologies, Ltd. Currently two realizations are available, the TILE-Gx36™ including 36 cores and the TILE-Gx72™ comprising 72 cores. Both implementations are optimized for networking, video, and data center applications.
In literature far more often the preceding model is mentioned [17]. Figure 2.6 shows a schematic diagram of the TILE64 processor architecture. The iMesh [109] network is a packet switched, wormhole routed with point-to-point connections. It consists of five independent networks which are listed in the following:

- IDN - System and I/O
- MDN - Cache misses, DMA, other memory
- TDN - Tile to tile memory access
- UDN and STN - User-level streaming and scalar transfer

2.5.4.2 KALRAY

The KALRAY massively parallel processor array (MPPA®) architecture has been designed for power efficient, computation intensive, and embedded applications. Up to one thousand processing cores are integrated on a single die. Clusters of processing cores are connected through an on chip network [44]. To enable predictable transfer times in the NoC and to provide quality of service, the network makes use of virtual channel buffers.
Currently available on the market is the MPPA2® high-speed I/O processor which integrates 288 cores and 128 crypto co-processors on one chip.

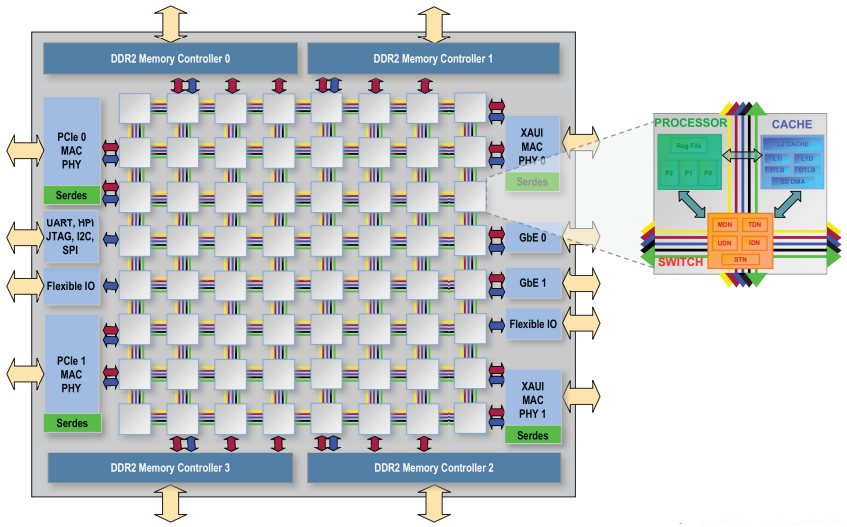


Figure 2.6: Architecture overview of the Tilera® Tile64 processor [17].

2.5.4.3 Intel® Research

Intel® is the global market leader in the semiconductor industry with more than 15% market share [6]. In 2009 Intel® introduced the *Single-chip Cloud Computer (SCC)* [70], a research microprocessor with the highest number of integrated cores at this time. The 48 cores are connected via an on chip network, including advanced power management technologies and support for message-passing [39]. The mesh based network comprises 24 routers, each connected to two cores. Figure 2.7 shows an overview of the SCC architecture. The cores can run individual instances of operating systems. Each core has its own level 1 and level 2 cache. Among the cores, there is no hardware cache coherency implemented. Due to a distributed memory structure either message-passing or software cache coherency can be used.

The performance of the SCC chip has been evaluated besides others in regards of communication performance, power and energy consumption and memory throughput [41]. Since the SCC is only a research chip, the evaluation results cannot be directly compared to measurements of standard high-edge processors. However, the chip provides a great variety of benchmarking possibilities.

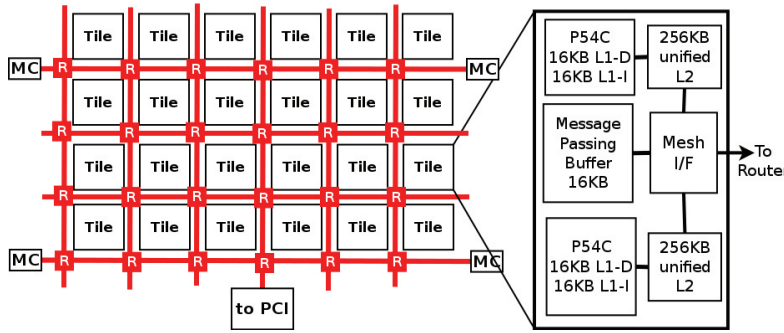


Figure 2.7: Architecture overview of the Intel® SCC. Showing the routers (R) connected as a mesh architecture and a schematic of one tile. Four memory controllers (MC) are connected at the border [98].

For power management, the chip is divided into three groups, where each group provides its own clocks and power sources. The three groups are tiles, mesh network, and memory controllers. The voltage and the frequency of the tiles can be dynamically configured during runtime by software, resulting in a power consumption from 25 W to 125 W [26, 98]. Each tile, consisting of two cores, can be separately power gated. Additionally, the performance level of the cores can be controlled by software. The developer can either directly write to a hardware register to access the power management or can use a special message passing library for controlling architectural characteristics.
The SCC thermal model is described in [88], but with focus on the cores. The heat generation of the routers of the SCC chip is not described in any work.

The SCC is only a research project to propose possible improvements for future chips and figure out the weaknesses. However, this chip already proved that the future of many-core processor chips will be based on *network on chip (NoC)* technologies.

Besides the SCC, Intel® introduced other *network on chip (NoC)* based chips. One example is the 80-node Intels® Teraflops research chip (see [102]). It was the first tera-scale programmable silicon with an on-die mesh network. Also, current processor chips with up to 24 cores, like the Xeon processor as shown in figure 2.8, are available on the market.

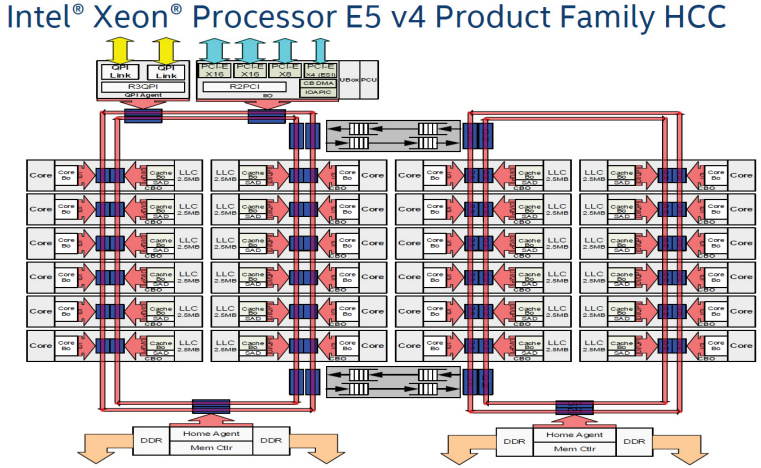


Figure 2.8: Architecture of Intels® latest Xeon® processor, with 24 cores [1].

2.5.4.4 Further NoC Architectures

There is a vast amount of NoC implementations described in literature, most of them developed within research projects where no real chips have been produced. In the following, four of these networks are described briefly: (1) *Nostrum*; (2) *Æthereal*; (3) *MANGO*; and (4) *Argo*. They all provide latency and throughput guarantees to obtain time predictability and hence are the most similar networks compared to the NoC described in this work. All four use virtual channels in equal fashion to share communication resources for multiple connections.

The *Nostrum* network is also a regular packet switched network but without any buffers within the routers [72]. The absence of the buffers requires a more sophisticated routing algorithm, in this case a deflective routing algorithm is applied. The main achievement of the *Nostrum* network are special packet types, called containers, which are used for guaranteed bandwidth communication. However, if there are multiple guaranteed bandwidth connections established, other communication can be blocked. There is a power model available for the *Nostrum* network [82].

Philips's *Æthereal* network also combines guaranteed service and best effort connections [38]. *Æthereal* is using contention free routing to reserve wires and buffers and release the resources after use.

MANGO is a message passing asynchronous NoC providing hard real-time guaranteed services through *Open Core Protocol (OCP)* interfaces [19]. It supports connectionless *best effort (BE)* communication as well as connection oriented *guaranteed service (GS)* communication to provide guarantees.

The last network architecture presented here is called *Argo* [59]. It consists of an router design supporting message passing among processors to support hard real-time applications. The network is implemented as a locally synchronous but globally asynchronous network. All routers in the architecture are within one clock domain while each processor core runs its own clock domain. The *Argo* has no input or output buffers for the virtual channels in order to reduce the hardware complexity.

2.5.5 Summary

With a growing number of cores integrated into one chip, networks on chips is given a greater emphasis in research. Also, first multi-core chips with a network communication infrastructure are available on the market.

2.6 Invasive Computing

In the Transregional Collaborative Research Center *Invasive Computing*, one of the goals is to design a new scalable multi-core embedded processor design for applications with high computational processing demands. The development includes new programming concepts, languages, compilers and operating systems which support resource-aware programming. Future multi-core architectures with more than 1000 processor cores integrated in one chip still lack appropriate programming paradigms. The holistic approach is challenging and requires early prototypes to prove the concept.
In the following, the hardware architecture of *Invasive Computing* is described, which is built up of a network structure connecting heterogeneous computing clusters. Subsequently, the software layer of *Invasive Computing* is briefly presented.

2.6.1 Invasive Hardware Architecture

The *invasive* hardware architecture is a scalable heterogeneous many-core architecture. The architecture consists of heterogeneous clusters connected by a NoC. The *invasive* NoC is the underlying architecture which has been used for this work. An

example 4x4 mesh architecture is shown in figure 2.9. The design is organized in a mesh 2D-array of tiles to increase scalability. Other topologies are also possible and are discussed in chapter 2.5.2. Each router in this design has bidirectional connections to its neighbor routers and one bidirectional connection to a local tile. A network adapter builds the interface between the router and the bus of the tile.

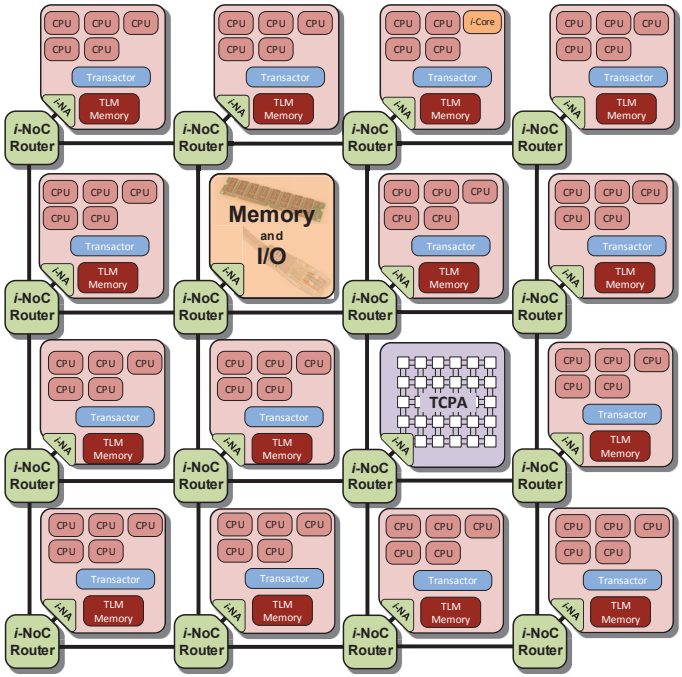


Figure 2.9: *InvasIC* heterogeneous hardware architecture. A mesh based network connects clusters containing different PE, memory or IO.

In contrast to the NoC technologies invented by Intel® and Tilera® , the tiles of the invasive hardware architecture consists of a multi-core system. The cluster architecture integrates a variable number of LEON3 cores, all connected to one local *Advanced High-performance Bus (AHB)* [34]. Furthermore, each cluster includes an L2 cache and a tile local memory. Some nodes within the network contain a global DDR memory in addition. The combination of a distributed and

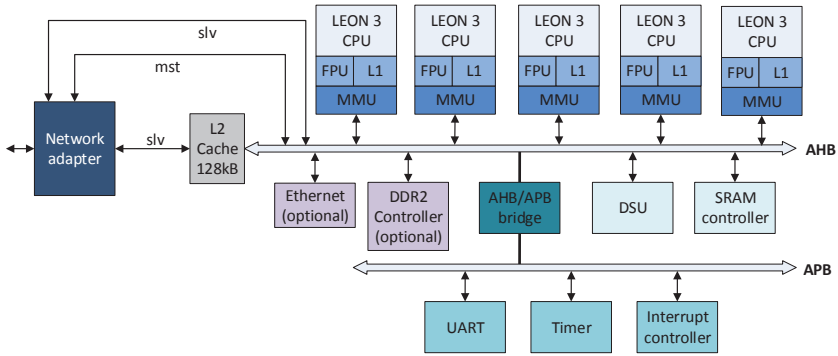


Figure 2.10: *InvasIC* basic computing tile (cluster) architecture. The network adapter on the left side build the connection to the router.

global memory architecture reduces the network accesses, since tile local memory is sufficient for most of the computations. This arrangement has implications for the degree of network capacity utilization, since network accesses can be reduced. However, on the other hand the realization of huge distributed memory is more challenging. Besides different memory setups , it is possible that the clusters contain special hardware, such as massively parallel processing arrays (*tightly-coupled processor array (TCPA)*) or reconfigurable PEs called *i*-Core, with a runtime adaptive instruction set.

2.6.1.1 Cluster Architecture

The cluster architecture integrates a variable number of LEON3 cores, all connected to a local AHB bus [35]. Furthermore, each cluster includes different memory controllers and I/O interfaces. A simple compute tile comprises level one (L1) caches, one L2 cache and a tile local memory to realize a distributed memory structure. The Ethernet and DDR memory controller are optional.
For debugging reasons, each tile contains one debugging unit (DSU) to give access to the memory mapped registers and trace information of the processor cores. Further information about hardware debugging is given in chapter 5.
In the given architecture each router has a bidirectional connection to a local cluster. A network adapter (left side in the figure) builds the interface between the router and the local bus of this cluster. Figure 2.10 shows one possible implementation of the cluster architecture.

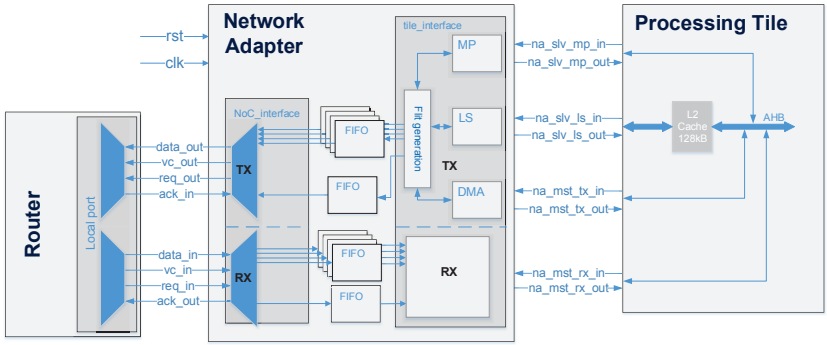


Figure 2.11: InvasIC network adapter block diagram. The *network adapter (NA)* in the middle builds the interface between the router on the left side and the tile on the right side.

2.6.2 Invasive Network on Chip

The basic network design is organized in a 2D array mesh of tiles to increase scalability. Other topologies are also possible and will be further discussed in chapter 3. The *invasive* network is a packet switching network with wormhole flow control. The network layer is separated into routers and network adapters. The latter build the interface between the network and the local compute cluster.

2.6.2.1 Network Adapter

The network adapter builds the interface between the AHB bus inside the tile and the local port of the router [47]. Thus, the network adapter is responsible for translating the bus transfers into network packets and vice versa.
The NA supports different transmission schemes and hence has a bus slave interface as well as a bus master interface. The master interface is necessary to provide hardware accelerated *direct memory access (DMA)* service. Whereas the slave interface is responsible for load/store and message passing communication. Figure 2.11 shows a schematic of the network adapter. Furthermore, the network adapter initiates the request for invading communication resources, by setting up GS connections. The application writes to a memory mapped register inside the network adapted and the necessary header flits are then generated automatically, controlled by the hardware of the network adapter. To retreat the communication resources, the application writes again to this memory mapped register.

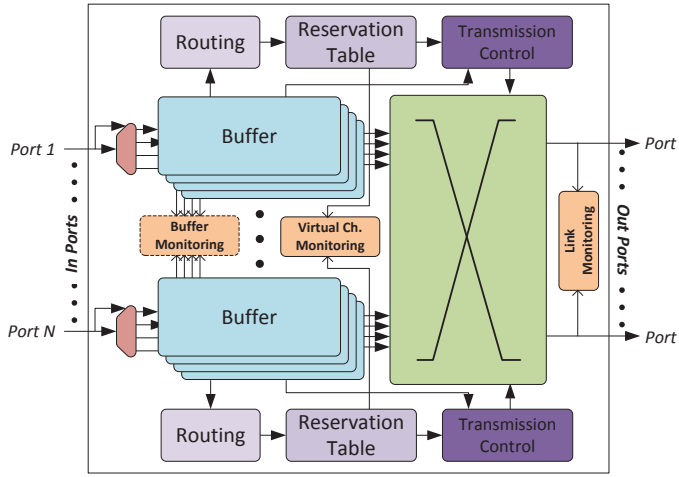


Figure 2.12: Block diagram of the invasive network router with a variable number of ports and an including monitoring infrastructure [45].

2.6.2.2 Router Architecture

The following describes the router design which was used as a basis for this work. A more detailed description of the router can be found in [45]. The network router supports both, connectionless *best effort (BE)* communication and connection oriented *guaranteed service (GS)* communication. Both transmission strategies (BE and GS) share the router resources dynamically. As a routing strategy XY routing is used in this work, other routing algorithms are also implemented, such as adaptive odd-even turn routing.
The routers are connected with each other by two unidirectional links which are divided into a parameterizable number of *virtual channels (VCs)*. A block diagram of a single router unit is shown in figure 2.12. The router input and output links, which are usually placed in cardinal directions to the neighboring routers are grouped together in this figure on the left (input ports) and right (output ports) side. Also, the router provides monitoring information about the current link utilization, the buffer fill level and VC utilization during runtime.

Figure 2.13: Network packet, consisting of multiple flits, where each flit holds its own control flag (Ctrl. Bit). Each packet starts with a header flit and is terminated with a tail flit.

Flit Format Data packets are divided into smaller portions of equal size, also referred to as flits. The size of the flits is equal to the bandwidth of the router link and can be chosen during the design process. The size of a packet is variable, depending on the amount of payload to be transferred. Each packet consists of one header flit, several payload flits and one tail flit, see figure 2.13. The header flit contains routing information, such as source and destination address as well as the connection type (GS and BE). The composition of the header and tail flit is shown in figure 2.14. The header flit is transmitted first and will be routed in each router. The routing information is stored in a lookup table in each router and all other flits follow the path which was established by the header. The VC used by this packet is reserved till the tail flit closes the connection. Figure 2.14 shows the structure of the three different flit formats. To indicate the router to establish or close a connection, header and tail flits are transferred with a control flag. For payload flits, this control flag is kept low.

Credit-based Wormhole Flow Control By the time the sending router puts data on the link, a valid signal is sent simultaneously. The transmitter has a credit counter for the buffer fill level of the neighboring routers. The counter is decremented when a flit is transmitted. The acknowledge bit is sent back from the receiver router at the moment the receiver forwards the flit to the next partner. This indicates, that the space in the input buffer of the receiving router is free again. There is no flow control implemented for the case that data gets lost or corrupted during transfer.

Scheduling Each physical link is used by a predefined number of VCs. For sharing the link resources, a scheduling strategy is mandatory. The router output port scheduling unit makes a decision based on priorities and buffer fill levels. The scheduling of VCs can also be referred to as link access arbitration.

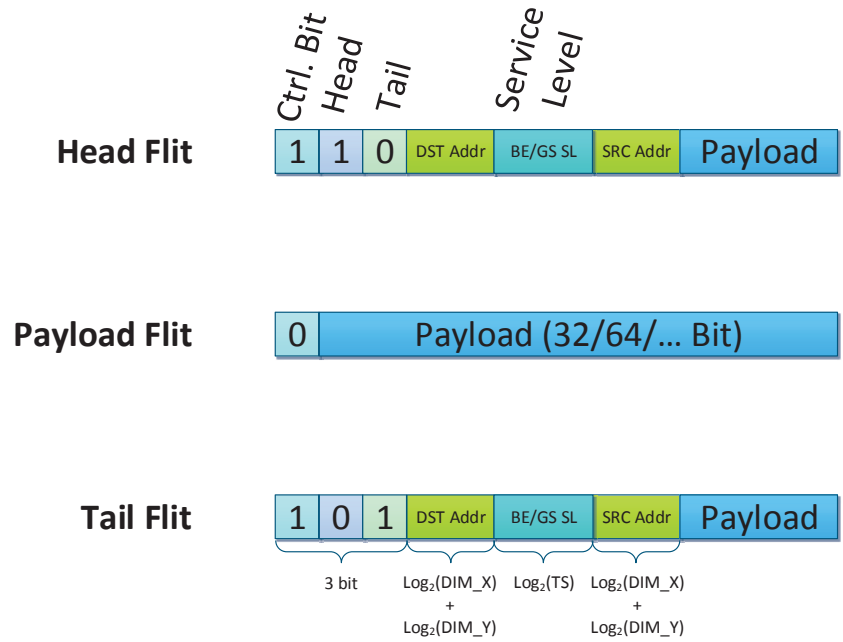


Figure 2.14: A packet consists of three different types of flits. Flit format of header and tail flits contain routing information while the payload flit comprises the actual data.

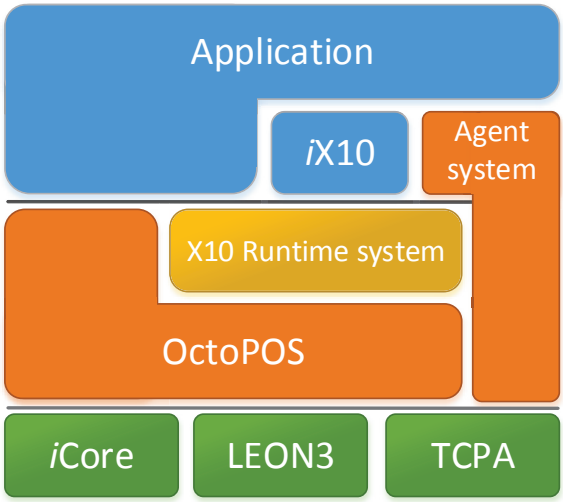


Figure 2.15: Hardware interface of invasive software layer.

To provide adaptive *quality of service (QoS)*, a weighted Round Robin scheduling scheme is used in this work. Multiple service levels for hard QoS guarantees can be chosen by the application software.

2.6.3 Invasive Software

The invasive paradigm includes a resource-aware programming model which is capable of mapping applications to various types of *processing elements (PEs)* available on the hardware platform. The mapping is done online and based on the actual workload of the PEs. The application performance can be improved by adapting the application to available resources at runtime.
Applications can dynamically acquire (invade) hardware resources according to their needs and according to the status of the hardware. Resources which can be invaded are the following: (1) computation bandwidth; (2) communication bandwidth and latency; and (3) memory space The application competes for hardware resources to get exclusive access [80].

The operating system called OctoPOS, which manages the hardware and software resources is also specialized on the resource aware concept of invasive computing [75]. Figure 2.15 shows the interaction between application (blue), runtime system

(orange) and hardware (green). OctoPOS provides operating system primitives based on Linux and in addition it provides a special execution environment for parallel applications. Providing scalability is one of the key features of OctoPOS. Resulting from that, each tile runs one instance of the operating system. This brings new challenges for the prototyping system since each tile requires a rather huge local memory.

2.7 Hardware Development

2.7.1 Hardware Description Languages

There are several *hardware description languages (HDLs)* available for designing digital hardware for FPGA or ASIC designs. The two best languages are *Very High Speed Integrated Circuit HDL (VHDL)* and Verilog. Both languages are supported by most development tools. However, new hardware description language have been introduced over the last years in an effort to raise the level of abstraction. One example is SystemVerilog, which is described in more detail in the following.

2.7.1.1 SystemVerilog

The hardware description language SystemVerilog is an extension to the Verilog language. It combines the advantages of Verilog, VHDL and C. While VHDL puts much emphasis on compile time checking, SystemVerilog provides better support for larger and more complex SoC designs. Thereby, the SystemVerilog language achieves a new level of modeling abstraction for hardware development and verification.
The Verilog language is extended by C-style data types such as struct, typedef, and dynamic types for better encapsulation and compactness of code and for tighter specification. Also, object oriented programming classes are introduced as well as new interface constructs. In SystemVerilog it is then possible to work with dynamic arrays and queues. Another new feature are assertions which can be used to validate the behavior of the design. Also, they can be used for formal functional verification. Further information about the SystemVerilog language can be found in [93, 40].

Due to the higher level of abstraction, productivity can be improved especially in the development of large-gate-count designs. In this work, the router design and its extensions as well as the testbenches for design verification have been written in SystemVerilog.

2.7.2 Design Verification and Prototyping

The wide variety of existing verification technologies can be categorized into four classes [85]: (1) simulation based technologies; (2) static technologies; (3) formal technologies; and (4) physical verification and analysis. To achieve a failure free design, a combination of these four verification technologies must be applied. Using one of the technologies exclusively makes it impossible to prove the absence of failures. However, the entire verification process is complex and far beyond the scope of this work, hence this work focuses on simulation based verification technologies only to reveal faults caused by implementation errors.
Applying simulation based verification, the *design under test (DUT)* is surrounded by a testbench. The input stimuli are applied to the testbench and output signals are compared to a reference model. The simulator can either be a software (e.g. ModelSim®) or a hardware (e.g. FPGA). Also, hybrid simulators such as Co-Simulation (see section 5.3.1) are possible.

Chapter 5 presents the verification methodology and an automated design and verification process which was developed for this work. For further information about fundamentals of verification methods see [64].

3 Adaptive Three Dimensional Router

3D chip structures are envisioned as the solution to keep up with an ever increasing number of transistors per die area. Also, communication performance benefits from the reduction of the average distance between nodes in 3D networks. A further advantage of stacked dies is that different types of silicon processes can be interconnected to heterogeneous chips. Hence, for each layer the optimal process can be chosen. Most of all, this enables the production of chips with considerably larger numbers of memory.

However, besides the advantages mentioned above, new challenges arise for three-dimensional chips, such as:

- The amount of available interconnections between layers is restricted.
- Higher latency of interconnection signals.
- Higher power consumption of inter-layer I/O pins
- Design partitioning takes additional implementation effort.

This chapter describes the extension of the existing network on chip to a third dimension and presents a solution to the interconnection challenges described above. An adaptive bandwidth router encapsulates several functions such as clock synchronization and parallel to serial conversion including flow control. This enables a fully routed three-dimensional network design with minimal resource overhead compared to 2D solutions. Chapter 4 presents a method to limit the power consumption in the network layer. The following chapter 5 focuses on the automation of the design implementation process.

3.1 Network Topologies

In section 2.5.2 the basics of 2D network topologies have been described. This section focuses on 3D network topologies and varying interconnection types.
One of the major characteristics of network performance to be 37 evaluated is the average distance between nodes or related with it, the longest path in the network. In figure 3.1 the longest path length of four network topologies is evaluated with a varying number of routers. The shortest maximum path length

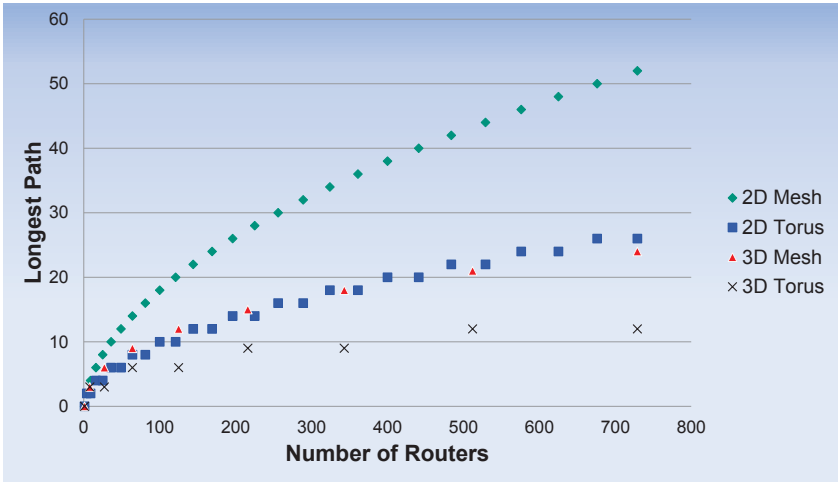


Figure 3.1: Evaluation of longest path length in different two and three-dimensional network topologies.

is always given in three-dimensional torus networks. However, torus networks require a huge number of interconnects which is antithetic to a restricted number of interconnections in 3D *integrated circuits (ICs)*. Hence, further topologies than torus designs need to be considered.

3.1.1 Interconnection Types

To cope with the restricted number of connections between the layers there are two major solutions to the problem. On the one hand, a fully routed 3D design, where all routers of one layer are connected to the other layer, but each connection has a small bandwidth. On the other hand, a design where only a portion of routers is connected vertically, but with a high bandwidth of the vertical connections. Both options have advantages and disadvantages. While a fully connected design does not require comprehensive routing algorithms, the partial routed design offers higher bandwidths.
In case of partially connected networks, there are different possibilities regarding the choice of routers which have a vertical connection (so called 3D routers). Either the connections are randomly distributed among the routers, or the 3D routers are all placed in the center or at the border of the design.

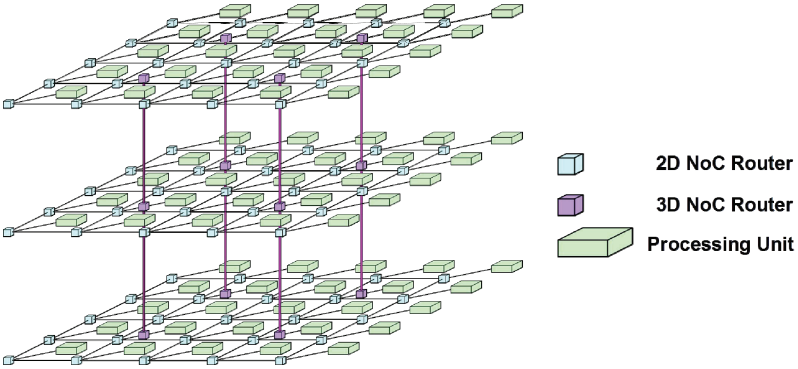


Figure 3.2: Homogeneous 3D NoC multi-core architecture with a mixture of 2D and 3D routers [103].

In this work a third option to cope with limited vertical connections will be introduced. The routers are extended by a special port with variable bandwidth and frequency.

Figure 3.2 shows one possible 3D NoC topology with a mixture of 2D and 3D routers. Only a few routers are connected vertically. On the one hand this solves the problem of restricted number of vertical interconnects. On the other hand this provokes deadlock situations, since all vertical communication has to be handled by these connections. In addition, a more complex routing algorithm than XYZ routing needs to be applied. Furthermore, two different router types need to be developed.

3.2 State of the Art

Several implementations of two-dimensional NoC are already established on the market, for example Intel® SCC [101, 51], Tilera® TILE64 [17], and Kalray MPPA [44]. The first two architectures are mesh based network structures where single processing engines are connected with a five port router. The Kalray network is also based on a mesh structure, but instead of single processing elements, a cluster containing multiple PEs is connected to a router. By progressive development of multi-layer chip technology, new network topologies become possible. A general description of basics for 3D networks on chips can be found in chapter five of [8].

Also, Pavlidis et al. [81] investigates three-dimensional network topologies on *three-dimensional (3D)* ICs.

There are different methods to construct 3D ICs [79]. The first one is chip stacking, where multiple fully processed and tested stand-alone components are stacked into one system-in-package. The second on is called transistor stacking, where multiple layers of transistors are stacked on a single substrate. As a third option, wafer-level stacking where entire wafers are stacked. Figure 2.1 in chapter 2.2 shows four different implementations of multi-chip system integration.

While the concept proposed for a 3D NoC in this work works with all construction methods, the chip stacking is not recommended due to long interconnects. According to the 2011 ITRS interconnect edition [27], the maximum number of connections between two layers can be calculated for each interconnection method. Although vertical interconnects have quite huge dimensions, the number of vertical connections between two chip layers is restricted and presents the main bottleneck in 3D designs.

The synchronization between chip layers creates another challenge in addition to restricted interconnects in 3D designs. One possibility to solve this problem are *Globally Asynchronous and Locally Synchronous (GALS)* systems with multiple clock domains. These implementations require a synchronization mechanism between clock regions. Vivet et al. presents the *FAUST* (Flexible Architecture of Unified System for Telecom) chip [104] as an example of a complex GALS NoC architecture. The network interface thereby performs the synchronization between the synchronous and asynchronous logic domains using ad-hoc decoupling *first in - first outs (FIFOs)*.

Dividing a simple planar design into different layers and then connecting the layers among themselves with short vertical interconnects has been described in [15]. The connection of multiple layers can be achieved without any other design changes. Modeling of TSV resistance, inductance, and capacitance is necessary for 3D IC designs, but this is not part of this work. Further information can be found in [60]. Also, there have been first investigations on how to cope with the restricted amount of vertical interconnects in 3D NoC designs. Liu et al. [67] describes a scheme for sharing vertical interconnects among neighboring routers in a time division multiplex mode. Xu et al. [113] investigates the impact of TSV placement to 3D NoC designs, while restricting the number of TSVs to a minimum. Vivet et al. [103] also worked on 3D NoCs and emphasized on the interconnection infrastructure. They present a 3D NoC with a combination of 2D and 3D routers. The same approach has been taken by Wang et al. [107].

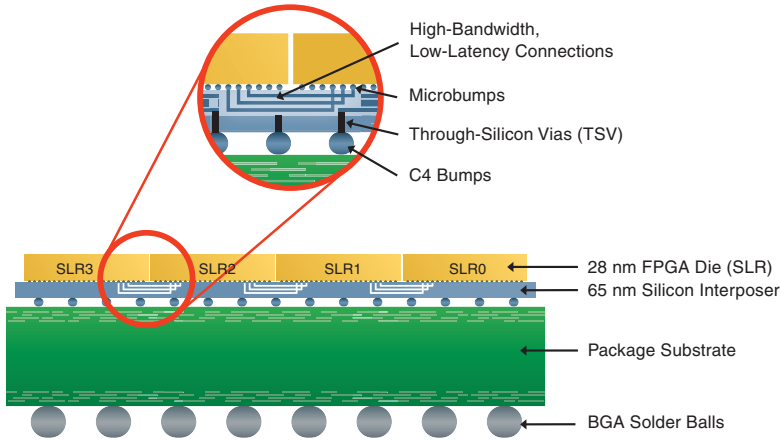


Figure 3.3: Horizontally Stacked Silicon Interconnect Technology introduced by Xilinx® [87].

3.2.1 Xilinx® Stacked Silicon Interconnect Technology

With the introduction of the Virtex® -7 FPGAs [112], Xilinx® introduced a new FPGA series with integration at 28 nm and a new *stacked silicon interconnect* (SSI) technology. The SSI technology uses passive silicon interposer with microbumps and through-silicon vias (TSVs) to combine multiple FPGA dies in a single package [87]. Figure 3.3 shows a schematic side view of the stacked FPGA design. The FPGAs are mounted on a high-performance package substrate by using TSVs. Xilinx® shows that there is a huge potential in reducing static power if multiple dies are stacked [52]. Xilinx® also reduces the dynamic power and I/O power. However, the reduction of the maximum static power delivers the most significant contribution with a reduction of over 65 %. In total, they can achieve a power reduction of up to 50 % compared to a previous generations equivalent.
Particularly due to its huge amount of logical cells, the stacked Virtex®-7 FPGAs are excellently suited for ASIC prototyping. The quad V7 prototyping platform presented in section 5.4.1 contains four of these Virtex®-7 FPGAs.

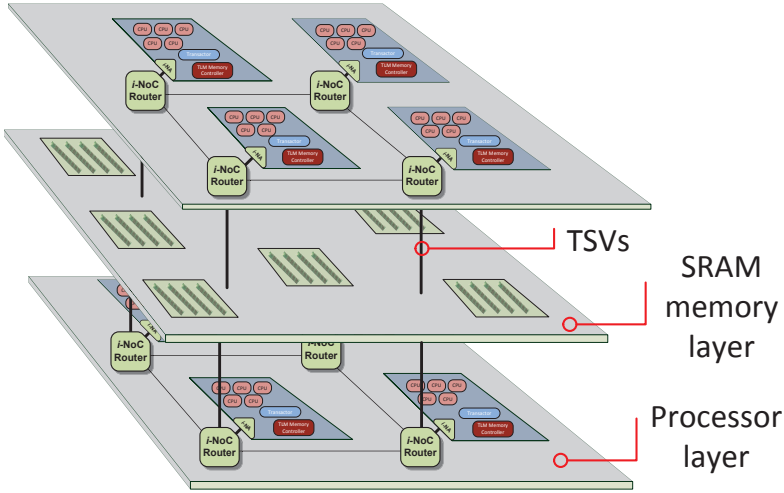
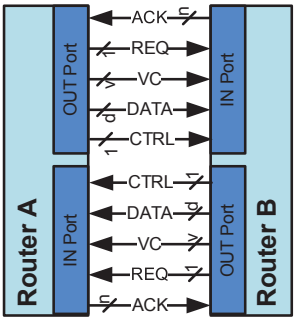


Figure 3.4: 3D mesh network structure, with all routers fully connected to their neighboring routers. The heterogeneous multi-layer network has a memory layer between two processing layers, containing routers and processing elements.

3.3 Heterogeneous Bandwidth Router for 3D Network on Chips

The goal of this work is to design a router which encloses two kinds of ports, one for vertical and one for horizontal connections. This heterogeneous router enables 3D mesh networks which are fully connected without contravening the restricted amount of interconnects between the chip layers. A first implementation of this approach has been presented in [FLB16].

Figure 3.4 shows one possible implementation of the 3D NoC described in this work. This example shows also the possibility of stacking various heterogeneous technologies on top of each other. The lower and upper layer comprise processing nodes and network infrastructure, while the layer in the middle only comprises memory cells. Due to this, the memory layer can be built up of a different technology than the layers containing routers and processing elements.



Name	Width (bit)	Description
REQ	1	Request signal, indicating payload
VC	v = 1, 2, 4, ... = ld[ceil (N_{VC})]	Virtual channel selection
DATA	d = 16, 32, 64, 128, ...	Data link
CTRL	1	Control signal, indicating header or tail flit
ACK	n = N_{VC}	Acknowledgement. One bit per virtual channel

Figure 3.5: Physical router interface, including the link width of the router ports.

3.3.1 Vertical Interconnects

Stacked chips can be connected vertically through short interconnects, called *through-silicon vias (TSV)*. The number of TSVs depend on the chip dimension and the technology. Hence, the amount of TSVs in a given chip region is limited and often the bottleneck of 3D chip designs. An example of TSV dimension is given in [42], with a 5 μm diameter per TSV.

One possible solution to cope with limited inter layer connections is to connect only a portion of routers across the chip layers. However, this would require a more complex routing algorithm than XYZ routing and result in longer distances between nodes and hence a larger average packet latency. The routing mechanism within a router takes more time than transmitting data over links with lower bandwidth. Therefore, the data link width of vertical connections is reduced in this work. The link width of connection within one layer can differ from the link width of inter-layer connections.

Figure 3.5 shows the basic physical interface between two routers. The bit width of the links depends on the link configuration. Each router has one unidirectional input and one output link per port. The number of connections per link is defined as shown in equation 3.1. Where N_{VC} and n are defined as the number of *virtual channels (VCs)* and d represents the data link width.

$$N_{link} = ld[ceil(N_{VC})] + n + d + 2 \tag{3.1}$$

The number of connections between two neighboring layers hence is made up of the product of number of routers and the double of the link width.
Taking the mesh based design discussed in section 3.4 as an example, with 4x4

routers per layer, a total of 4,352 links between two layers are needed, if a design is taken where each router has four VC per port and a data link width of 128 bit. Therefore, the number of links between two layers could be calculated as follows:

$$N_{link} = (2 + 4 + 128 + 2) * 16 * 2 = 4352 \tag{3.2}$$

3.3.2 Clock Constraints in Higher Dimensional Chip Design

Global synchronous clocks are not possible in 3D designs because clock synchronization across multiple dies is hard to achieve while holding up the frequency in the Gigahertz range. Otherwise, high clock skews will cause timing problems.
One approach to avoid global synchronous clocks are GALS systems. In case of three-dimensional systems, the most suitable solution is to have synchronous clocks at each chip layer, but the layers are not synchronous with each other. Resulting from that, a clock synchronization between the layers is necessary. The synchronization logic represents an overhead in terms of resource consumption and adds latency to the communication and is further described in the following section.
The *FAUST* chip [104] is an example of a complex GALS NoC architecture. In this implementation the synchronization is located in the network interface between the router and the processing element. Following, the routers are all within the same clock region. The design proves that GALS techniques are feasible and mature enough for NoC systems. However, the 2D design cannot be extended to a third dimension, since this would require that routers are asynchronous with each other.

3.3.2.1 Clock Domain Crossing

To realize logic designs with multiple frequency islands, a clock domain crossing needs to be performed. Transferring signals between the clock domains may lead to setup and hold violations if no additional synchronization step is included. If there is a large difference between two clock frequencies, data can get lost or interpreted twice. Following, a clock domain crossing logic is mandatory in multi-clock designs to prevent such failures.

There are several possibilities to address clock domain timing problems. A simple mechanism for clock synchronization are flip-flop chains with at least two flip-flops. Other opportunities are handshaking protocols and FIFOs. In this work, a dual-clock FIFO concept has been applied. Figure 3.6 displays the structure of the router link and the connection to the router input and output ports. The

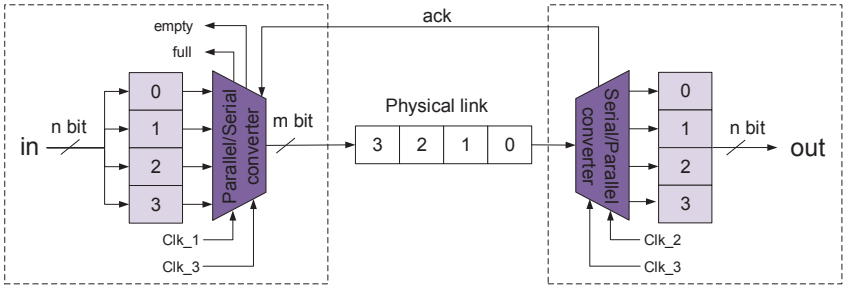


Figure 3.6: Block diagram of the parallel to serial and serial to parallel conversion at the router output and input ports. With flit size n and serialization factor of four.

clock domain crossing is handled by the parallel to serial converter. Hence, the converters act like dual clock FIFOs with different width of input and output port.

Beyond the clock domain crossing, the restricted number of vertical connections requires a serialization mechanism. Instead of using synchronous input buffers for the router as described in [45], the clock synchronization and parallelization at the input port is handled by the serial to parallel converter. The input bit width in figure 3.6 is equal to the size of one flit. The serialization factor can be chosen during design time. In this example, the serialization factor equals four. Following, size m is equal to $n/4$. Other works have refrained to use serialization schemes [103, 107] because of the increase in complexity and area of the router. To keep the resource overhead as small as possible, the flow control mechanism of the converter has been combined with the input port buffer flow control.
Two control signals indicate whether the converter is empty or full. Other control signals are not necessary due to the flow control already integrated in the router input port buffers (see ack signal in figure 3.6). The flow control is always aware of how much free space is in the neighboring input buffer. A counter value is decreased every time a flit is transferred and increased when an acknowledgment (ack) signal is received. Following, the complexity added by the serialization is insignificant.

3.3.3 Implementation

In order to avoid a reduction of the router bandwidth in the entire network, due to limitations given by the vertical connections, a router with optional low link

width ports has been developed. In addition, these links use their own clock domain to smooth clock synchronization problems due to a partitioned design across different chip layers. The router can enclose both types of ports at the same time. During design phase the developer must choose which ports should have the usual link width and which ports should provide special low link width and asynchronous serialization mechanism. Figure 3.7 shows a simplified design with only two routers connected with each other. The serial to parallel converter at first handles the clock domain crossing and secondly are connected to the input buffers for flow control reasons (see detailed in figure 3.6). This connection between the converter and input buffers is necessary to assure that new data is not transmitted, as long as the converter is busy with the serial transfer of the data. Henceforth, there are no additional registers mandatory for flow control at the low bandwidth ports. Flits which shall be transmitted through the low link width port, are split in the converter and transmitted sequentially. At the neighboring router input port, the flit fragments are reassembled before the entire flit is stored in the input buffer. The designer can specify the bandwidth of usual and through-silicon router links during design time. Figure 3.6 shows an example implementation of the converters with a split factor of four. Also, the clock frequency of all three clock domains can be chosen by the designer depending on physical parameters of inter layer connections. Although the vertical connections are shorter than router links on one chip layer, it is possible to use a higher frequency for the TSV. Accordingly, it is possible to reduce the delay which originates from serial data transfer. Depending on the size of the through-silicon router links and the size of header information, it is possible to start the routing process with flit fragments arriving at the input port of the layer border. This avoids increasing latencies due to sequential transmission.

Flit extension: The general flit composition is described in section 2.6.2.2. The space for the source and destination address depends on the size of the network. If the design is extended by a third dimension, there is more space necessary for the source and destination address, due to the additional Z coordinate. Thus, it is more likely that the flit size of 32 bit is not sufficient for the complete header flit information.

3.4 Evaluation

In order to prove the approach of the adaptive bandwidth router and investigate the systems' functionality including extensive software benchmarks, an FPGA based prototype of the 3D network has been implemented. Although the technology of FPGAs differ tremendously from 3D ASIC implementations, it could only be used as a prototype of the design. To realize a design with millions of

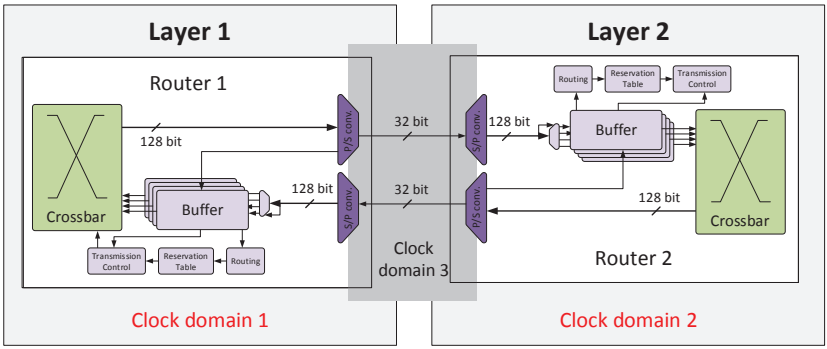


Figure 3.7: Structure of inter layer connection between two routers, connected by parallel to serial and serial to parallel converters running in a different clock domain.

gates, multiple FPGA devices are necessary. Therefore, a platform with six Virtex5 LX330 FPGAs has been used. Each FPGA represents one layer of the final chip design. The different FPGAs are interconnected via cables. While the memory builds a separate layer in the target chip, the memory on the prototyping system is realized as extension boards. Chapter 5 describes in more detail the prototyping and debugging setup.

Table 3.1 shows the resource consumption of different network implementations. Routers which are placed at the border of the network do not need to have input buffers for each port. Only those ports which are connected with another router hold input buffers. Hence, the resource consumption of one router within the network varies for mesh based networks. The additional ports per router result in a 32.06 % higher *look up table (LUT)* consumption for a 3D mesh network, compared to an equivalent 2D implementation with the same number of nodes. In torus networks, all ports of the border routers are connected and following from that, all routers in the network consume the same resources. This results in an even higher resource consumption than 3D mesh networks. A 3D torus network consumes 87.85 % more LUT than an equivalent 2D mesh implementation.

For performance measurements HDL simulation models of networks with 64 router nodes have been used. For network topology evaluation three-different designs have been implemented. First a 8x8 2D mesh network, second a 4x4x4 3D mesh, and third a 4x4x4 3D torus network, all with constant router links of 128 bits and four VC. To evaluate the different topologies, a test pattern with synthetic

Component	Resources			
	absolute		relative	
	in LUTs	in Register	in LUTs	in Register
8x8 design	**677,894**	**88,073**	+/- 0 %	+/- 0 %
Primary 2D router	12,793	1,534	+/- 0 %	+/- 0 %
4x4x4 mesh design	**895,203**	**92,613**	+ 32.06 %	+ 5.15 %
4x4x4 torus design	**1,273,452**	**116,448**	+ 87.85 %	+ 32.26 %
3D router	20,137	1,885	+ 57.41 %	+ 22.88 %

Table 3.1: Resource consumption of a 2D and 3D router design.

uniform random traffic has been used, in which the injection rate at all router nodes is equal. The basic definitions of latency and throughput measurements are described in section 2.5.3.1 and 2.5.3.2. The results of the latency and throughput are shown in figure 3.8. Since higher dimensional networks have a lower average distance between nodes, the average latency of packets is lower in the 3D mesh network implementation, see figure 3.8b. The latency is even further reduced if a three-dimensional torus network is applied. In equal measure, the throughput is increased for torus networks and the saturation point is at an injection rate of 0.6 instead of 0.4 for two-dimensional mesh networks (see figure 3.8a).
However, since the torus network consumes a much higher resource overhead than the mesh network and the throughput gain is negligible higher, the best trade-off to minimize the resource consumption while maximizing the throughput is to take a 3D mesh network. Also, the number of vertical connections is too high in torus designs. Following, the prototype presented in chapter 5 only comprises 2D and 3D mesh designs.

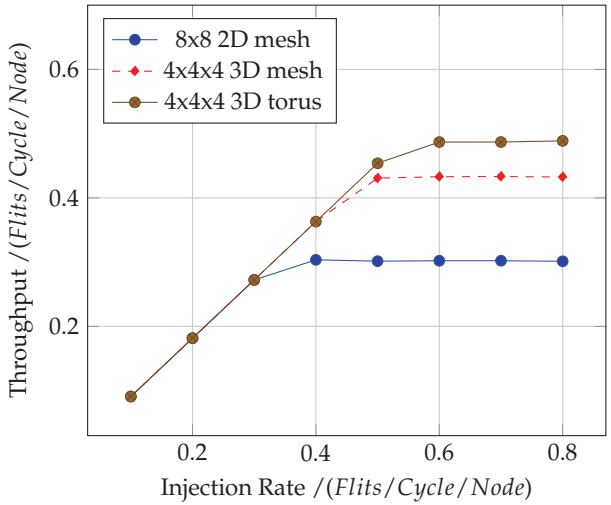
Current state of the art 3D NoC implementations are not fully routed due to limited number of vertical interconnects. There are several implementations to spread the vertical connections and to partition the design across the chip layers. One possibility described in literature are neighboring routers sharing the interconnection resources [67]. Designs with different router implementations for horizontal and vertical connections are a further possibility to realize 3D NoCs. Thereby only the 3D routers are connected vertically [103]. Kim et al. [62] suggests a partially connected crossbar to deal with the limited number of vertical links. These approaches have all in common that they require an enhanced routing algorithm, while the design presented in this work can use a simple XYZ routing algorithm. An enhanced routing results in additional hardware resources as well as in higher latencies due to longer routes. The results presented in literature do not mention the resource overhead. In addition, the 3D design presented in the

submitted work automatically include a clock domain crossing, in contrast to other work where additional synchronization mechanisms are required. A fully routed design with network multiplexer is presented by Said et al. [89], but only 2 to 1 multiplexers are used in this work and hence scalability is not provided. In [30] the network components are placed on a separate layer. Nevertheless network resources only represent a fraction of the overall design resources and following the present work introduces a different floorplanning. Each layer comprises a number of tiles and their direct connected routers. To meet high memory capacity requirements, a separate memory layer is placed between two of these layers. To provide flexibility, the design of this work can be applied to any 3D manufacturing process. Parameters which are specific for the process can be adapted during design time.

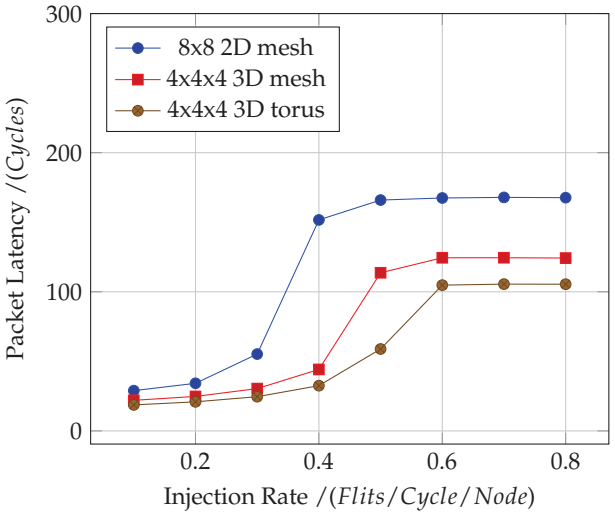
3.5 Summary

With the introduction of 3D integrated circuits and an ever increasing number of transistors per chip, the extension of network topologies to a third dimension become focus of research. Though, the extension to a third dimension costs about one third more resources for mesh networks and more than 85 % for torus networks. However, due to shorter average distances between the nodes in 3D networks, the latency could be decreased and hence communication performance rises. The performance gain of torus networks is only marginal higher than using a mesh design, whereas the resource overhead is exceedingly higher.

The issue with a restricted number of vertical connections in 3D designs have been solved in this work by a router which possesses two different kinds of ports. The bandwidth and frequency of the ports can be chosen independently. Due to the link between the output port parallel to serial converters and the input buffers, the resource overhead ca be kept to a minimum. Hence, it is possible to compensate the restricted number of inter layer connections without performance degradation in the other network segments. Further, there is no need of additional clock synchronization between the chip layers because it is handled by the new router ports as well.



(a) Throughput



(b) Packet Latency

Figure 3.8: Network performance of two-dimensional mesh and three-dimensional mesh and torus networks, each with 64 nodes and homogeneous bandwidth of 128 bit, using uniform random traffic with equal injection rates at all routers.

4 Multi-Granularity Network Power Optimization

Low power designs are becoming increasingly important, on the one hand because of limited battery power of portable devices. On the other hand because of high heat generation due to power dissipation in *integrated circuits (ICs)*. In future systems built up of hundreds of cores, power dissipation becomes critical due to massive thermal output. The increasing chip integration density raises thermal issues even in 2D implementations. Stacking chips on top of each other for 3D designs increases these issues all the more. Especially leading away thermal power from the middle of the chip causes problems.
This chapter focuses on the minimization of energy consumption and power management of network architectures. Thereby, the continuous increase of overall system performance is the fundamental goal while optimizing the power dissipation. A hardware power management controller monitors the router workload and takes the decision of fine or coarse granular power gating or clock scaling of network resources.

The first section of this chapter describes the fundamentals of the power saving methods for on chip networks and highlights different general techniques to save power within ICs. This is followed by section 4.2 which lists current research in this topic. Section 4.3 presents the implementation of the power management controller designed for this work and is followed by the evaluation of the implementation.

4.1 Power Saving Methods for Network on Chips

There are different principles to reduce power in ICs. They all have in common that the value of at least one of the following parameters needs to be decreased:

- V_{dd} (supply voltage)
- Frequency
- Load capacity
- Switching activity

With rising package density and shrinking technology sizes, the power problem is becoming worse. Reducing the supply voltage has a quadratic effect on the dynamic power. However, the reduction is only possible as long as the supply voltage is substantially higher than the threshold voltage of the gates.
There are many approved methods for power reduction in integrated circuits. The following chapter describes six low power design methodologies. First clock and power gating is covered followed by frequency and power scaling methods. Further it discusses different levels to save power within the design. For further information about low power design, see [61, 77].

4.1.0.1 Clock Gating

The clock net of an integrated circuit consumes a huge percentage of the overall power. In order to reduce the clock net power consumption it is possible to include an enabling signal to stop the toggling of local clock trees, whenever the modules in this region are inactive. The resolution of clock tree regions needs to be chosen during design time. It is differentiated between fine-grained and coarse-grained clock gating depending on the size of clock regions. Applying fine-grained clock gating, single registers are equipped with an enabling signal. However, this will result in a huge amount of additional logic. While coarse-grained clock regions consume less additional resources, it is less likely that the whole region is inactive and hence it can be isolated from the clock signal.
The biggest disadvantage of clock gating is that only dynamic power can be reduced, while the static power dissipation still remains.

4.1.0.2 Power Gating

The power gating technique works similar to clock gating, but now the enabling signal is used to disconnect a module from the power source or from ground. As opposed to clock gating leakage power can be removed as well, but the wake-up time adds a bigger delay to the system the moment the module is needed again. However, the system performance is not affected during normal operation.

In order to ensure that the system resumes operation based on its last state before power gating, special cells need to be added which can store the state during shut down. These retention cells are consuming power during shut down. There is also additional energy required for the wake-up process. Hence, the decision to power gate a module needs to be made more carefully. Therefore, power management controllers are needed to evaluate the measured utilization. This method offers the highest possible power saving, but on the other hand adds the highest wake-up latency.

4.1.0.3 Power Scaling

Since the dynamic power is proportional to V_{dd}^2, lowering the supply voltage has a quadratic effect on the power dissipation. On the other hand, the reduction of the supply voltage has a huge negative impact on the switching delay of the gates. Hence, power scaling method can only be applied if the performance is not effected.
The solution are design installations with multiple power regions. Each region has the lowest possible voltage without violating the system timing. Providing multiple clock regions however requires additional logic and more complex power grids [61]. The instantiation of level shifters is necessary to drive signals between different power domains.

4.1.0.4 Clock Scaling

The power consumption depends on the supply voltage as well as on the frequency. For frequency change, a divider value is used to select operating frequency by dividing the global frequency. The frequency range is restricted.
The advantage of this method lies in the relatively small amount of additional hardware compared to power gating, since no isolation is required. The supply voltage for a stable operation depends on the frequency. Hence, if the frequency is reduced, it is also possible to scale down the voltage. The combination of clock and voltage scaling is described in the following section.
In order to get an overview about the realization of dynamic clock frequencies in FPGAs see [111].

4.1.0.5 Dynamic Voltage and Frequency Scaling (DVFS)

The most efficient power management technology, combining the strategies introduced above, is *dynamic voltage and frequency scaling (DVFS)*. The minimization of power is achieved by dynamically adjusting the voltage and frequency of components. For DVFS, *voltage regulator controllers (VRCs)* are necessary to provide several voltage islands on the die. Also, controllers are needed to switch between the different voltages and frequencies. However, typically these controllers are implemented in software.

4.1.0.6 Fine Grain and Coarse Grain Power Switching Techniques

There are different levels of power switching techniques. Also, one can distinguish between chip perspective and design perspective.
First, looking at fine grain power switching at chip level, there is one power switch placed next to each standard cell. The overhead of this method is quite large.
In coarse grained power gating, a complete block of gates is power gated by one switch. Using this method, the resource overhead added by the power switches is much smaller compared to fine grained power gating, whereas the design implementation is more challenging since the power gates of a power region must be physically placed next to each other. In order to weigh up the disadvantages and the advantages of both methods, the penalty of resource overhead overweight and hence current designs usually use coarse grained power gating [61].

Looking at the design view for example of a router, also different levels of power switching can be applied. Either a complete instance of a router can be power gated, or single modules of the router are switched off. A first implementation of disabling single VC is described in [110, 73].

4.1.1 Power Evaluation Methodology

In order to evaluate the different settings of the architecture in terms of power dissipation, a power simulation was created using Synopsys® *PowerCompiler*, since there is no real chip available which can be used for power measurement. The tool *PowerCompiler* [95] is integrated in the Synopsys® *DesignCompiler* [94] for power estimation at RTL level. In order to get power estimation results at gate level instead, Synopsys® *PrimePower* can be used. Since a rough appraisement of the system power consumption is sufficient for evaluation, only RTL level power estimation is considered in this work.

In order to get power estimations, standard cell libraries are used. They contain the basic building blocks for modern *integrated circuit (IC)* design. The *Taiwan Semiconductor Manufacturing Company (TSMC)* digital standard cell model library *tcbn45gsbwpwc* contains timing and area information for each standard cell at 45 nm. The switching activity of all gates needs to be identified to calculate the power consumption of a component.

Figure 4.1 describes a methodology for power analysis using a combination of RTL simulation and ASIC synthesis. The gate-level power analysis uses switching activity from RTL code to analyze net switching power, cell internal power, and leakage power.

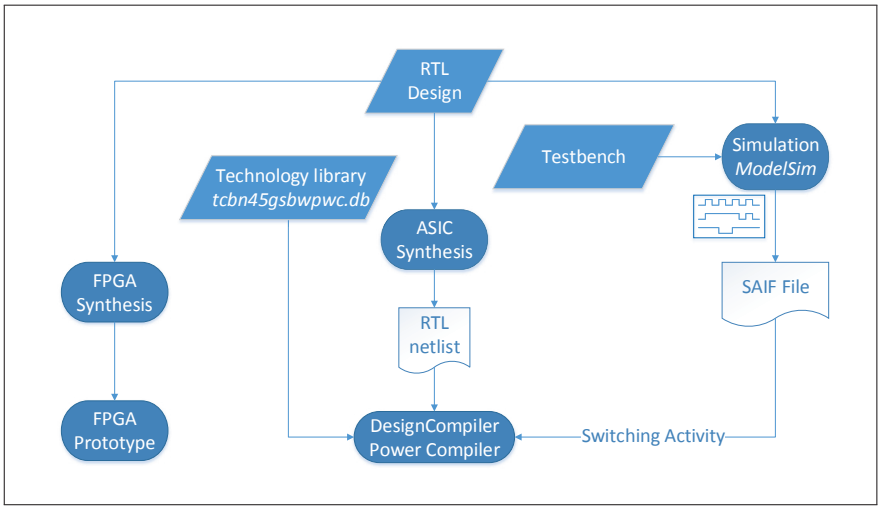


Figure 4.1: Design tool flow for power simulation using ModelSim® simulation in combination with DesinCompiler power estimation.

The following series of items shows the steps of the power evaluation process:

1. System elaboration
2. Synthesis
3. Design simulation to generate switching activity
4. Power analysis

The HDL simulation tool ModelSim® is used to generate the *Switching Activity Information Format (SAIF)* file, containing the switching activity of the design depending on the corresponding testbench. The designer can choose between optimizing the design with regard of timing, power or area.

4.1.2 Thermal Issues

With shrinking technology sizes, the power density is increasing. Especially with transistor sizes below 90 nm, power dissipation and the resulting heat generation cannot be disregarded any longer. One of the main reason is the exponentially growing sub-threshold leakage current due to increasing temperature.

Thermal management gets even more important with stacked designs and 3D architectures. Even with 2D dies heat sinks to regulate the on chip temperature are no longer sufficient. This effect is increasingly strengthened with stacking dies and further pushing of density.

The rising temperature provokes multiple failure mechanisms and in the last resort causing permanent damage. To prevent hot spots, power saving techniques for NoCs need to be applied, as described in section 4.1.

4.2 State of the Art

The following section describes state of the art of power saving techniques for NoCs. General descriptions of fundamentals of low power designs are given are [77, 61]. An overview of low-power designs for NoC architectures, including power and energy aware design techniques from several perspectives and abstraction levels is given in [91].

The number of switches between on and off states should be limited. Firstly, since each wake-up process results in power penalty and secondly because of high wake-up latencies [99, 100]. Both values highly depend on the applied technology and the depth of the sleep state [10].

There are several methods for power saving for NoC systems. They can be divided into two groups, focused either on dynamic power reduction, or on the reduction of the static power. Reduction of dynamic power can be achieved by task or thread manipulation [36] or DVFS methods. The reduction of static power is always linked with power or clock gating mechanisms and is described in the following section.

The following works are inspired by dark silicon research [49] and propose a different kind of power gating granularity, where multiple physical networks run in parallel and each layer can be power gated separately [24, 28]. The sub-networks can be power gated without compromising the connectivity of the primary network. Although the total power consumption of the multi-layer network can be reduced compared to single-layer networks, the resource consumption is not considered at all. Bokhari et al. [21] takes the idea of multi-layer networks and describes the delay which is inserted when switching between the different layers. All these works have in common that they suffer a disadvantage in regard of resource overhead by utilizing to the design of sub-networks. To keep the resource overhead to a minimum due to power management, is one of the key features of this work here.

Another method to achieve power reduction by reducing the dynamic power is DVFS. Lee at al. [66] presented a frequency and voltage scaling method based on the network workload. Another system-level DVFS technique is presented in [25], where higher level caches are comprised as well as network resources. Bogdan et al. [20, 76] present a control approach for NoC designs with multiple voltage and frequency islands. Their approaches focus on an analytic solution of the power management problem, which results in a high resource utilization and long decision times. As opposed to this, the method described in this work pursues a minimal size hardware implementation for the power management, with fast response to changing communication load since the complete logic is implemented in hardware. In addition, the centralized approach in [20] does not scale well, in contrast to my approach.
The Intel® Single-chip Cloud Computer is equipped with power management. Tiles, mesh network, and memory controllers all have separate clocks and power sources. Nevertheless, the frequency and voltage of network resources cannot be adjusted at runtime [58].

For fine grained power gating on the system level Mirhosseini et al. [73] describe a power gating scheme for virtual channels in on chip networks, that uses an adaptive method to dynamically adjust the number of active *virtual channels (VCs)* based on the on chip traffic characteristics. However, the proposed method does not comprise guaranteed service connection within the network. Hence, if virtual channels are reserved but currently not used, they cannot be power gated. Also, other resources of the router than buffers will not be power gated if they are inactive.

Jevtic et al. [54] describes a methodology to measure FPGA power consumption and separate the total power into static and dynamic power. The measurements results are more accurate than FPGA power estimation tools. However, due to the complexity of the measurement proceeding, estimation tools are usually sufficient to get an overview of the system power consumption.
It is also possible to reduce the static power in FPGAs [52], but in this work the FPGA only serves as a prototype.

Since there is no real chip available of the NoC architecture described in this work, the power measurements for evaluation of the design are based on simulation models. A basic performance and power evaluation model for NoC based interconnection networks is described in [16]. Another example for an existing power model for network routers is ORION, which comprises a set of architectural network power models [56, 106]. The power model was validated using real power measurement values from the Intel® 80-core Teraflops chip.

H. Matsutani and M. Koibuchi investigated power saving methods for on chip networks and presented their work in several publications. In [68] an ultra fine-

grained runtime power gating method for on chip routers is described. Each router is divided into 35 micro power domains. A similar work of the same author [69] presents a look-ahead routing strategy to switch on routers in advance. However, this technology requires additional computational effort. In addition to a not negligible resource overhead, additional computation overhead is necessary inside the router to produce the information required to generate a wake-up request signal. In the present work, the routers are powered on in advance as well, but the required information does not require sophisticated routing algorithms, only an additional 1 bit link between the routers to transmit the pre-request signal is required. Low power networks often conclude in a lack of performance. The goal of this work was a reduction of the power consumption while keeping the communication performance high. The following describes the implementation of a power management for networks on chips controlled by hardware logic.

4.3 Multi-Granularity Power Management Controller

The goal of the online power management unit for on chip networks is to decide whether it is reasonable to switch off single virtual channels, a complete router or to scale down the frequency without affecting the communication performance. Therefore a *power management controller (PMC)*, composed of the controller and a *Load Detection Unit (LDU)*, has been developed. A simplified version of the controller has been presented in [FNB16], including only coarse grained power gating. For efficient power gating functionality the PMC needs to observe the current utilization of the on chip routers. Each router is already supplied with a monitoring unit which is utilized by the PMC. The following section describes the hardware implementation in detail. The design implementation is written in *SystemVerilog* which offers the possibility to use the same files as for a FPGA based prototype on the one hand. On the other hand the design files can be used for an ASIC synthesis and the associated power estimation.

In order to come to a fast decision for every router while keeping the design scalable, there is one *power management controller (PMC)* connected to every router. The resulting resource overhead is proportionally small in contrast to the router design and is further evaluated in section 4.5. Figure 4.2 shows a schematic of a 3x3 mesh network, based on the architecture described in section 2.6.2, where all routers have a connection to their direct neighbors, the local cluster, and to one PMC. In order to improve clarity, the wake-up and sleep signals are only given for the router in the center of figure 4.2. In the actual implementation all PMC have these two signals to all adjacent routers.

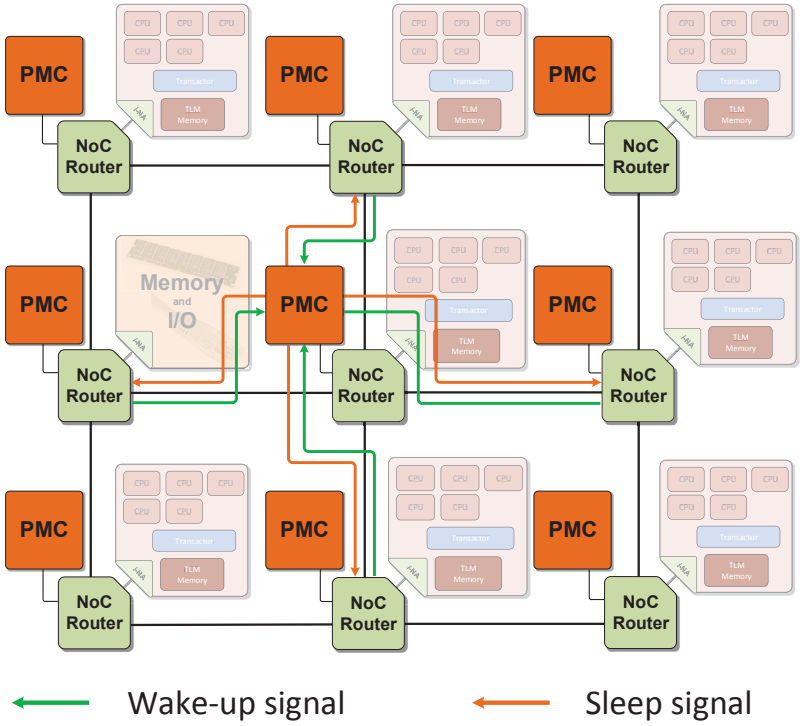


Figure 4.2: Schematic of a 3x3 mesh network, including one power management controller (PMC) for each router. The PMC sleep and wake-up signals are only shown for the PMC in the middle, all others are connected accordingly.

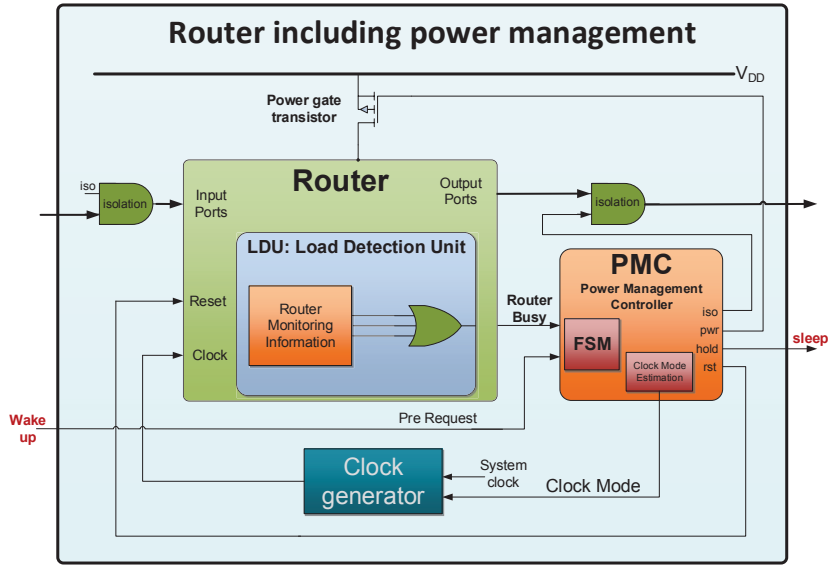


Figure 4.3: Block diagram of the power management unit, consists of the load detection unit inside the router, a clock generator, and the power management controller.

In order to motivate the design of a multi-granularity power management concept, router power consumption divided into the most important components is listed in table 4.1. The biggest share of the total power (almost 80 %) is attributed to the input buffers. The measurements have been executed with the same setup as described in table 4.2. With five ports per router and each port providing four VCs, the design holds in total 20 input buffers. Following power gating, a single input buffer can reduce the router static power by 4 %. Hence, the approach presented in this work provides the possibility to power gate a complete router or single *virtual channels (VCs)* of a router.

The power management unit consists of the following three components. At first, the load detection unit which analyzes the monitoring information. Secondly, the transmission control unit which extends the existing control unit so that no flits are transmitted as long as the following router is power gated. The third component, the power management controller which decides whether the corresponding router can be power gated or if the clock frequency should be

Power Consumption in W / Router Component	Dynamic		Static	Total	Percentage
	Switching	Internal	Leakage		
Router Core	$1.24 * 10^{-4}$	$4.85 * 10^{-4}$	$4.67 * 10^{-6}$	$6.14 * 10^{-4}$	10.85 %
Buffer Structure	$9.15 * 10^{-5}$	$4.38 * 10^{-3}$	$1.74 * 10^{-5}$	$4.49 * 10^{-3}$	79.40 %
OPS (Output port selection)	$3.78 * 10^{-5}$	$1.04 * 10^{-4}$	$9.10 * 10^{-7}$	$1.43 * 10^{-4}$	2.52 %
VCRT (Virtual channel reservation table)	$4.46 * 10^{-5}$	$3.61 * 10^{-4}$	$2.63 * 10^{-6}$	$4.08 * 10^{-4}$	7.22 %

Table 4.1: Static and dynamic power consumption of router components.

scaled. An overview of the power management unit, integrated into the router, is given in figure 4.3. Of course the router still includes the components shown in figure 2.12, but for reasons of simplification they are not shown here. In the following section, the implementation of the load detection unit and the PMC is described in detail.

4.3.1 Load Detection Unit (LDU)

The output of the monitoring unit is continuously evaluated by the *Load Detection Unit (LDU)*. While a router is active, the monitoring unit output values are greater than zero. The LDU checks during every single clock cycle if its monitoring value is greater than zero. If the monitoring value is equal to zero for a complete observation time T_{idle_detect}, the router is marked as idle and hence the router could be switched off completely. Otherwise, the busy signal of the detector is set high to signalize a busy router device. The observation time T_{idle_detect} is parametrizable and can be set during design time.
The LDU is integrated into the router. The advantage of this approach is that, while the router is shut down, the monitoring unit is also powered off. While the appropriate *power management controller (PMC)* must always be active even when the router is shut down to watch out for newly arriving flits. Since the power management unit cannot be power gated, the goal is to keep it as small as possible with regard to resource and power consumption.

4.3.2 Power Management Controller (PMC)

The applied network provides two different communication types, *best effort (BE)* and *guaranteed service (GS)* communication. In case of BE all flits of one packet are sent right after each other. Hence, the reserved channel will be utilized the whole time. In case of GS communication a channel will be reserved for a longer period

and might not be utilized for any length of time.
The PMC has to signal the power state of its router to all adjacent devices to prevent the loss of data packets. The PMC sends a *hold-signal* immediately when it detects an idle router to block further outgoing transmissions by neighboring routers and its local tile. The *hold-signal* interrupts the reservation process in the transmission control of the out port *virtual channels (VCs)*. Hence, flits are held back in the sending router till the neighboring router is activated again.
On the other hand, as soon as the arbitration module determines a new packet arrival, a *transmission request signal* is sent to the router in the direction the packet should be forwarded, even before the actual arbitration process is finished. Consequently, power gated routers start with their wake-up process as early as possible to minimize the performance penalty due to power saving mechanisms.
Isolation cells are integrated at the input of the router to generate a known logic value during shutdown of the supply power.

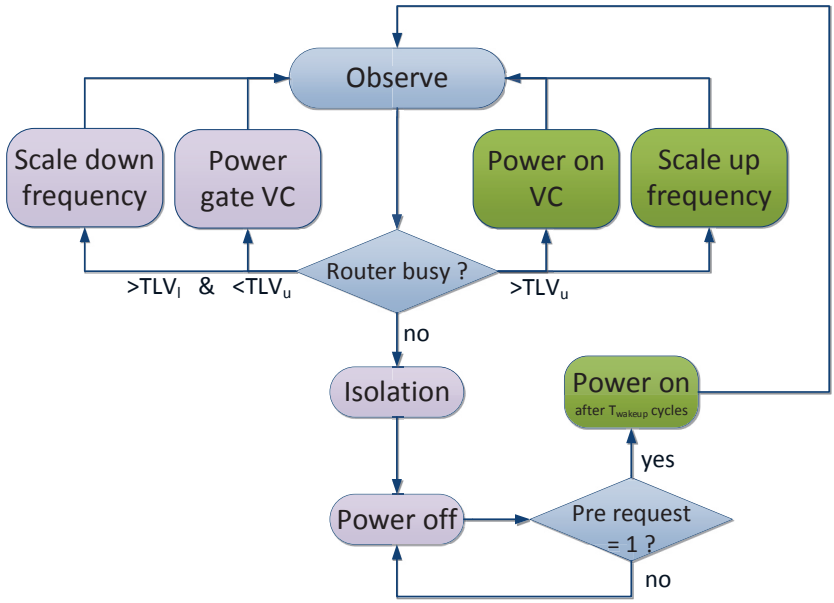


Figure 4.4: State chart of the power management controller. Depending on the router utilization (TLV - threshold limit value), the control loop suggest a frequency scaling factor or power gates single virtual channels or the complete router.

Figure 4.4 shows the state diagram of the *power management controller (PMC)* behavior. The following four paragraphs describe the main states and switches between states of the controller.

Monitoring The monitoring unit continuously evaluates the virtual channel utilization of a router. If at least one VC is utilized, the router is marked as active. In addition, each VC has its associated utilization signal. Theses signals are used by the Load Detection Unit (see section 4.3.1).

Power down If the *Load Detection Unit (LDU)* signals that the router is inactive, the power management controller monitors for *transmission request signals* of adjacent routers to avoid a false shut down while a transmission is already queued. Also, the LDU checks whether there are channels reserved for GS communication. Only if both are cases are false, the PMC can take the next state for isolating the circuits of the router. Before the router is finally shut down on the next rising clock edge, the PMC checks again for incoming requests. If a request arises at this point in time (*Pre request* state in figure 4.4), the shutdown is canceled and the router stays active.
Instead of isolating the router from the supply voltage, it is also possible to disconnect the clock source. Consequently, the leakage power will not be reduced, but on the other hand the wake up delay will be much shorter. In order to evaluate which gating techniques suites better, the wake-up time is implemented as a parameter and can be changed during design time.

Wake-up process In case that a flit should be routed to a power gated region, the component needs to be woken up before. The initiating source router sends a *transmission request signal* to the sleeping routers' PMC. This causes the PMC to switch the routers sleep transistor and connect it to the systems supply. In the next step the PMC resets the virtual channel reservation table at the output port of the requesting router to release the blocked VCs. Afterwards, the transmission is executed as usual. The time a router requires for waking up and starting its usual operation is defined as $T_{wake-up}$ and is crucial for the added latency by the power gating functionality. A mechanism for early wake-up procedure is presented in the following paragraph.

Power on Delay After the power on signal arrives at the power gated router or VC, it takes time till all cells of the module reach a stable state to be fully functional. This time is defined as $T_{\text{wake-up}}$ and highly depends on the applied power gating technology, chip technology, and sleep state of the module. Firstly,

since the chip technology is not known at early development stages and secondly to keep the design flexible, the $T_{\text{wake-up}}$ is implemented as a variable parameter. One possibility to power gate CMOS circuits is the insertion of sleep transistors. Tovinakere et al. [100] describe in their work a model to estimate the wake-up latency for circuits, power gated with sleep transistors. A similar method is described in [99]. In that case a *p-channel metal-oxide semiconductor (PMOS)* sleep transistor is inserted between the supply voltage (V_{dd}) and the node of the circuit where the supply voltage would be connected usually. Hence, the wake-up time is equivalent to the switching time of a PMOS transistor. Depending on the technology and transistor size, the wake-up time varies between 40 ns and 0.26 ns. However, the sleep transistors are bigger than the rest of the circuit since they need to handle the required amount of switching current. If the power gating is part of the power distribution network instead of being part of the standard cell, the switching time will be higher. This is the case if a complete router should be power gated. In case of smaller granularity, e.g. to power gate individual virtual channels, the wake-up latency will be shorter. Thus, there are two variables which need to be specified at design time, $T_{\text{wake-up router}}$ and $T_{\text{wake-up VC}}$.

In order to decrease the performance penalty due to wake-up times, a look-ahead wake-up strategy is implemented within this work. The wake-up signal (green line in figure 4.5) indicates the *power management controller (PMC)* of the power gated router that it should start its wake-up process. Whereas the orange sleep signal indicates the neighboring routers that packets should be held back until the router is fully functional again. This prevents packets being lost if they should be routed towards a power gated region.
The look-ahead routing strategy does not require additional logic due to the simple XY routing algorithm which is used in the design. The routing of a packet header from arrival at the input port, till the header flit leaves the router requires fife clock cycles. The port a packet has to leave a router is already known after two clock cycles while the other clock cycles are needed for the VC reservation process. Since the information of the port is already sufficient, the router can start the wake up process already two clock cycles after the arrival of the packet by giving notice to the neighboring PMC in direction of this port to wake up the neighboring router.
Matsutani et al. [69] introduced a look-ahead sleep control to reduce the performance penalty. In contrast, the approach described in the work can achieve the same results without additional resources.

Frequency scaling In case of low loads or if channels are reserved by *guaranteed service (GS)* connections, the router cannot be power gated. In order to reduce power also in these cases, the PMC comprises a clock scaling unit in addition to the power gating mechanism.

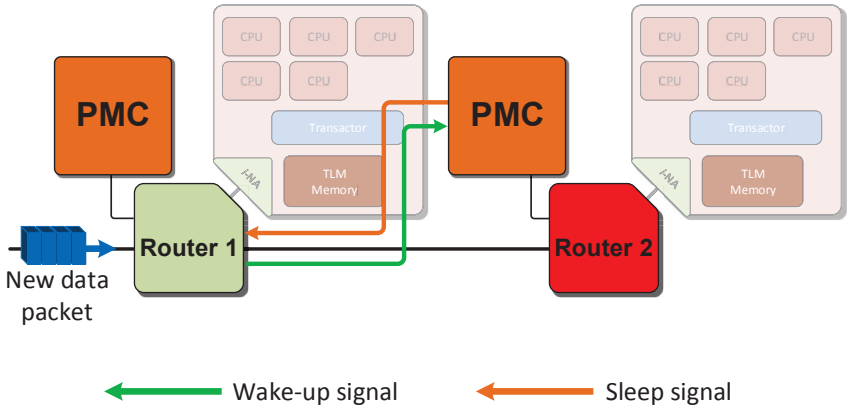


Figure 4.5: Clipping of a NoC architecture including power management. A new packet arrives at router 1 which should be transferred to the power gated router 2.

The *observe* state is the default state of the power management controller, see top of figure 4.4. Depending on the degree of router utilization, the controller either suggest a frequency scaling factor (scaling the frequency up or down) or power gates the complete router in case of no utilization. The router utilization *threshold limit value (TLV)* can be chosen during design time. Depending on this parameter, different scaling factors are calculated.
The realization of multiple clock domains requires a clock domain crossing between the different clock regions. The approach described in this work replaces the buffers at each input port of the router by asynchronous FIFOs to reduce the resource overhead. Thus, neighboring routers can run in different clock domains without any further design changes. As a result of this, clock synchronization becomes redundant.

Parameter Setting At the beginning of the design process it is usually not clear which specific standard cell library shall be used. Simulation libraries contain information about electrical, operating, and power characteristics. In particular, tables about the propagation delay and power consumption as well as cell width information are given in detail. In addition, even if the technology is known, the delay times vary widely. In order to be flexible at the one hand and to explore advantages and disadvantages of different settings on the other hand, the design should not be bound to a specific delay time. Hence, the implementation of the

power management controller introduces an adjustable delay which can be set during the system design phase. Thus, the designer gets the opportunity to test its implementation with different delay times to access the impact of this factor onto the network performance. As well, it is possible to react to short term changes in the standard cell library selection process.
The standard cell libraries are usually optimized either for maximum frequency, low power, or minimum area with highly varying delay times. After an excessive benchmarking process where different design configurations are tested against each other, the most suitable technology can then be chosen.

4.4 Routing Strategies for Power Gated Networks

In case of power gating one complete router or even multiple routers, the usual XY routing cannot be applied any longer without causing backpressure onto the network. Packets which need to cross this region either need to be kept back till the region is powered on again, or other routing strategies need to be applied to route the packets around the power gated region.
Keeping back packets and waking-up regions on the one hand lowers the network performance and on the other hand this might result in a high switching activity. The power on switches consume additional power and therefore should be kept to a minimum. Also, if on chip hotspots should be prevented, it might not be possible to switch on a region for a specific time.
The second possibility, routing packets around power gated regions. Therefore, different implementations can be applied such as more sophisticated routing strategies than XY routing, packet rerouting or multi layer networks.
The following sections describe two possible strategies to route packets around power gated regions. First, a rerouting strategy is presented. Second, a multi-layer network implementation is described which is used to route packets around power gated regions.

4.4.1 Rerouting

In this work only XY and XYZ routing has been applied so far. In order to avoid huge off/on switching activities in case of power gated routers, the usage of other routing algorithms provides better possibilities. In [46] a rerouting strategy is described. In this paper the rerouting was used to route packets around overloaded regions, but the same technique can also be applied to route packets around power gated regions. Accordingly, routers need only to be switched on in case that the router is within the destination region.

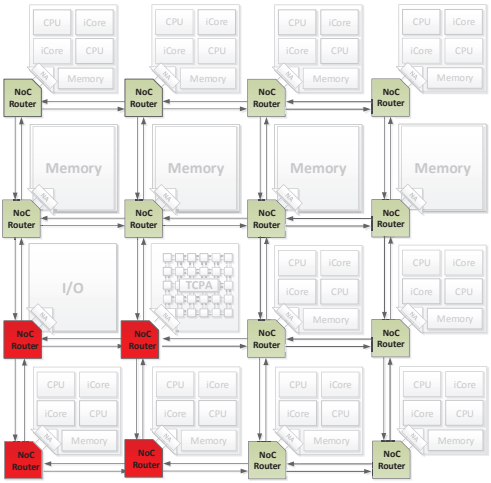


Figure 4.6: 4x4 mesh network. The red routers are power gated routers.

The new routes around the power gated region still need to be minimal routes. Non minimal routes would be possible as well, but to avoid additional latency and out of order problems, only minimal routes should be considered.

The hardware implementation of the rerouting logic adds approximately eight percent of resources. In order to adapt the concept to bypass power gated regions only congestion detection must be extended to the existing work, so that it is also sensible to power gated regions [46].

4.4.2 Multi-Layer Networks

Instead of a single layer of network resources, multiple heterogeneous network layers are connected with each other. In the present work the basic network layer is extended by a second lightweight layer, called *second layer network (SLN)*. It was initially built to provide fault tolerance for on chip networks [HFM+16] and then has been extended in this work, so that it can be applied for power saving too [HWZ+15].
The SLN is build up of switches and multiplexers. Depending on the configuration at design time, the SLN adds up to 18 % of resources to the basic network. However, as long as the SLN components are not used for rerouting packets around power gated regions, the SLN can be power gated itself and hence does

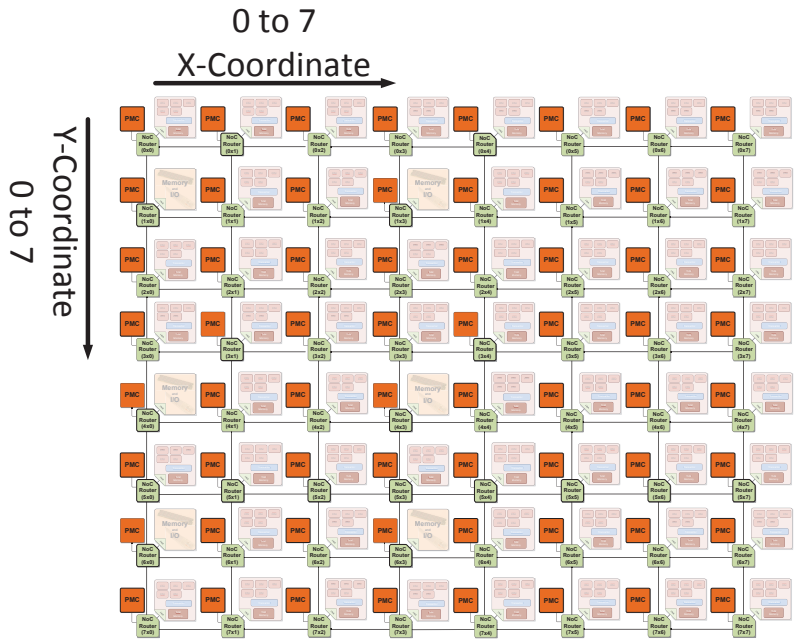


Figure 4.7: 8x8 mesh network architecture used for performance simulation. The simulation framework include routers with power management as well as the computing tiles connected to the local router port.

not consume any power. Only if basic network components are power gated, a ring is configured around the power gated region and substitute the normal network in regions. The communication is transparently redirected. Accordingly, there is no need for adaptive routing schemes to bypass power gated regions. [LWT+16]

4.5 Evaluation

For performance and power evaluation the architecture described in section 2.6.1 has been utilized and is extended by the power management. Detailed parameters of the network architecture for evaluation are given in table 4.2. The proposed router design and the power management unit are implemented in *SystemVerilog*.

Topology	8 x 8 mesh
Router Architecture	5 port router (4 neighbors + 1 local tile)
Input Buffer	4 buffers per port (due to 4 virtual channels), each 4 flits deep
Link Width	128 bit data, 9 bit control
Processor	Gaisler Leon3
Tile	5 processors per tile (cluster)
Technology	TSMC 45 nm library (tcbn45gsbwp_120a)
Frequency	25 MHz (FPGA prototype) 1 GHz (ASIC implementation, worst case)

Table 4.2: Network architecture details for performance and power simulation.

The synthesis was done on the one hand for an FPGA based prototype using one Xilinx® Virtex7 2000T FPGA. On the other hand the same HDL files have been used for an ASIC synthesis and power estimation with Synopsys® *DesignCompiler*. Therefore, the TSMC library tcbn45gsbwp_120a is applied.

The wake-up delay times for power gated circuits depend as described before on the one hand on the used design library and on the other hand on the depth of the sleep state. In order to keep flexibility and not to be assigned to one specific parameter setting, the wake-up delay time of the routers is implemented as a variable parameter. Hence, the designer can test its architecture design with different delay times to assess the impact of the switching delay on the network performance.[1]

4.5.0.1 Resource Consumption

One of the main goals of the implementation was to keep the resource overhead as small as possible when adding the power management comprising the load detection unit and the power management controller. The generated area overhead by adding the power management to the router design amounts only 2.1 %. Figure

[1] The results presented in this work have been generated with a delay of 20 clock cycles.

4.8 shows the contrasting juxtaposition of resource consumption of a single cluster (tile), a network router without power management, and a router including the *power management controller (PMC)*. The design parameter set of the tile and the router are given in table 4.2. Resource consumption for look up tables (LUT), Flip Flops (FFs), Block RAM (BRAM), and Digital Signal Processors block (DSPs) are listed in figure 4.8. The results are based on an FPGA synthesis for Xilinx® Virtex7 2000T. The overhead for an ASIC synthesis has a similar proportional outcome.

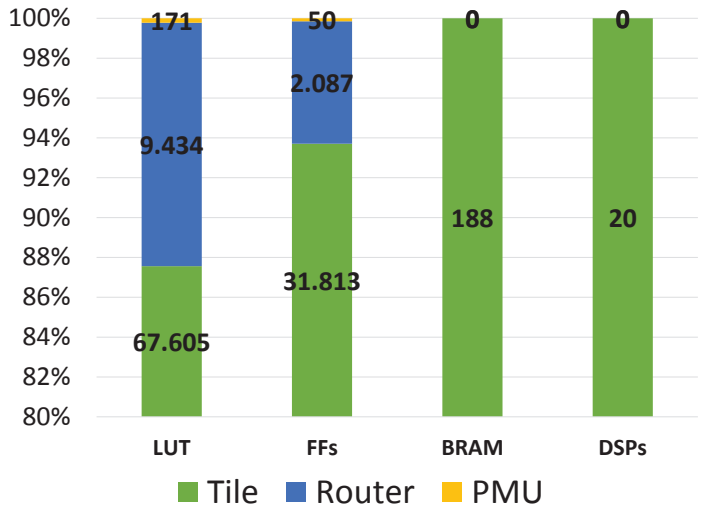


Figure 4.8: Resource consumption of a single tile, the original router, and the power management unit (PMU). The PMU consists of the load detection unit and the power management controller. The figure illustrates the synthesis results of a Xilinx® Virtex 7 2000T FPGA.

4.5.0.2 Simulation Infrastructure

In order to evaluate the performance by adding the power management, a HDL based ModelSim® simulation of an 8x8 mesh network has been used. With the purpose of reducing simulation time, the processing elements are represented by simple models. Hence, only synthetic generated traffic is applied. The injection

(x,y)	0	1	2	3	4	5	6	7
0	21%	18%	16%	15%	15%	17%	22%	27%
1	19%	15%	13%	13%	14%	16%	20%	23%
2	18%	13%	11%	11%	12%	16%	18%	23%
3	18%	13%	12%	12%	12%	15%	19%	23%
4	18%	13%	13%	12%	13%	16%	18%	22%
5	19%	15%	13%	12%	12%	16%	18%	23%
6	21%	17%	16%	15%	16%	21%	22%	26%
7	30%	22%	19%	19%	17%	20%	23%	32%

Table 4.3: Proportion of time routers are power gated in an 8x8 mesh, with synthetic uniform random traffic and an injection rate of 0.3 flits/cycle/node.

rate for each router in the design is equal. In order to evaluate the design, multiple runs with varying injection rates have been made.

The proportion of power gated routers depends particularly on the application. Taking communication intensive applications, the amount of time a router can be power gated will be much lower than having computation intensive applications. In order to evaluate the worst case scenario, a synthetic generated uniform random traffic model with varying injection rates as well as transpose traffic have been used. Even these worst case consumption still show that the approach by adding a small amount of resources can save power without major performance loss. The injection rates were measured in flits per cycle per node (router). Table 4.3 shows the percentage of power gated time of each router in an 8x8 mesh network with an injection rate of 0.3 flits/cycle/node. Building the average over all injection rates between 0.1 and 0.6, in as many as 14.2 % of the time a router is switched off.

Figure 4.9 compares the throughput and latency of an 8x8 mesh network without power management with a network of equal dimensions including the *power management controller (PMC)*, using uniform random traffic. The definition of latency and throughput measurements are given in section 2.5.3.1 and 2.5.3.2. For one simulation run, the injection rate of all routers is equal. However, rising injection rates, starting from 0.1 flits/cycle/node, are used to evaluate the system performance. The throughput (see figure 4.9a) is almost equal in both implementations. Due to the power on delay, packets which should be transmitted to a power gated router will be delayed. Hence, the latency with low injection rates differ much more since it is more likely that routers can be power gated. If the injection rate increases, it is less likely that routers can be power gated due to

(x,y)	0	1	2	3	4	5	6	7
0	4%	4%	6%	6%	7%	8%	9%	11%
1	5%	5%	5%	6%	7%	7%	9%	9%
2	6%	6%	6%	6%	6%	7%	8%	8%
3	8%	7%	6%	6%	6%	7%	8%	8%
4	8%	8%	7%	7%	7%	6%	7%	8%
5	10%	9%	8%	7%	6%	6%	7%	7%
6	11%	10%	9%	7%	6%	6%	6%	7%
7	13%	11%	10%	8%	7%	7%	6%	7%

Table 4.4: Proportion of time routers are power gated in an 8x8 mesh, with synthetic transpose traffic and an injection rate of 0.3 flits/cycle/node.

high utilization and following the latency of the two implementations is almost the same.

Figure 4.10 shows the equivalent results for transpose traffic. The latency overhead including the power management is comparatively small for transpose traffic. The reason for this is the small percentage of power gated routers as it is stated in table 4.4. Using transpose traffic, router (x,y) send packets to router (y,x). As a consequence the traffic is evenly distributed in the mesh network and routers on the margin are utilized in the same degree as routers in the center of the mesh.
In order to reduce the power consumption for such traffic patterns as well, where routers carry only low loads and hence cannot be power gated, the PMC suggest a clock scaling factor. A selection of fixed values are possible as frequency scaling factors. They can be chosen during design time by defining the *threshold limit value (TLV)* accordingly.

Instead of retarding packets if routers are power gated, it would also be possible to use a different routing scheme than XY routing. For example Heisswolf et al. [46] describe a rerouting mechanism. Nevertheless, the more complex routing algorithm comes with a resource overhead and increased latencies if non minimal routes are used. Besides that, the destination router need to be waked up anyway and hence retarding packets cannot be fully prevented. For these reasons this work abandoned the option of a different routing algorithm than XY routing.
The second routing option for power gated networks (described in chapter 4.4.2) using multiple-network layers is also not applied here since it also adds too much extra resources. Only if the design has several network layers anyway, this technology sounds reasonable.

(x,y)	0	1	2	3	4	5	6	7
0	77	69	56	55	57	60	66	71
1	66	58	51	48	53	55	61	67
2	54	56	45	43	52	55	58	64
3	55	58	45	48	51	57	60	59
4	52	52	50	49	49	57	57	59
5	62	60	54	48	52	54	63	64
6	63	57	56	52	61	55	60	66
7	82	78	69	55	59	63	70	82

Table 4.5: Power on switching activity of routers in 8x8 mesh, with synthetic uniform random traffic and an injection rate of 0.3 flits/cycle/node. Simulation time: 315 μs and 135,000 flits dispatched in total. Measurement given in total number of power on switches.

4.5.0.3 Power Analysis

There are different methods available to measure the power dissipation of FPGAs. Jevtic at al. [54] describe a methodology to measure separate values of static, clock, interconnect, and logic power in FPGAs. However, the power consumption of FPGAs differs considerably from ASIC power consumption [63]. Hence, the FPGA prototype is only used for system functionality verification and not for power evaluations.

There are two parameters which are important to evaluate the power saving potential. First, the span of time a router is powered off. Second, the amount of switches between power off and on state since each switch costs additional power. Table 4.5 shows the number of on switches for each router in an 8x8 mesh network with an injection rate of 0.3 flits/cycle/node using uniform random traffic.

For power estimations at RTL level the Synopsys® *PowerCompiler* included in the Synopsys® tool *DesignCompiler* has been used. The tool computes the leakage and dynamic power of a router based on a given test bench scenario. For generating the power estimation results, the same test bench as described in section 4.5.0.2 has been used. Table 4.6 shows the ASIC area and power consumption of a reference router without power management, compared to a router including the PMC. Thereby, the power consumption is given on the one hand for no communication (Power (Min.) columns) and on the other hand for high communication load (Power (Max.) columns).
In case of no communication, the router is inactive all the time and can be clock- or power gated. If so, the highest power savings are possible. In case of high communication load, the router is active all the time and cannot be switched off at any

time. Resulting from that, the router including the power management consumes 8.52 % additional power than the implementation without power management. The *PowerCompiler* measurements, given in table 4.6, only comprise clock gating instead of power gating. Hence, the leakage power is not reduced in this case.

Version	ASIC area (μm^2)		Leakage power (mW)	Dynamic power (mW)	Total power (mW)
Base Router	229944	Min.	2.30	10.81	13.1028
		Max.	2.06	449.2474	451.3119
Router with PM	232565 (+1.14 %)	Min.	2.13	7.29	9.4197 (-28.11 %)
		Max.	2.08	487.71	489.7825 (+8.52 %)

Table 4.6: ASIC TSMC 45 nm power estimation results for a single router including the power management. Compared to the original router. (Frequency = 1 GHz; Operating Voltage = 0.8 V)

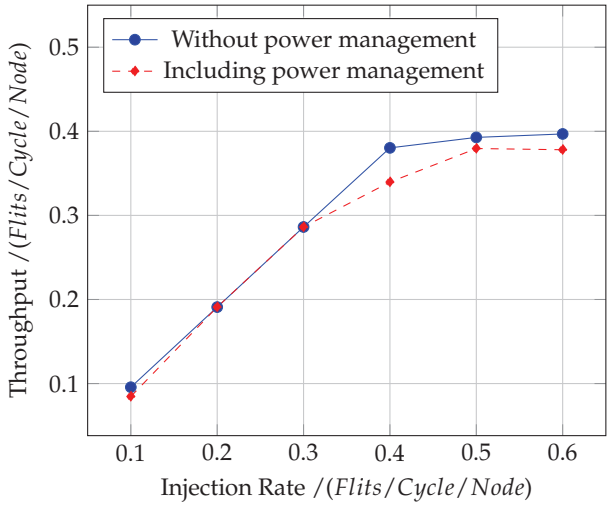
The maximum power saving potential using clock gating is 28.11 %. If power gating is applied instead, even higher power savings could be achieved. Since the same concept of the power management for network resources can be used for clock gating as well as for power gating, no additional implementation effort is required.

State of the art dark silicon NoC concepts presented in [24, 28] include a second network layer. Both works do not consider the resource overhead caused by the second network layer. Also, the switch between the network layers requires time and hence adds latency to the communication [21]. Another disadvantage of previous concepts, they do not scale well [20] and hence are not applicable for designs with thousands of processors. All previous works have in common that in contrast to this work, they did not constrain the resource overhead. This work here however presents an approach with minimal resource overhead while keeping the network performance as high as possible.
Matsutani et al. [69] introduced a look-ahead sleep control to reduce the performance penalty. In contrast, the approach presented in this work can achieve the same results without additional resources.
Compared to the network described in [73] the power management technique presented in this work covers *best effort (BE)* and *guaranteed service (GS)* communication and is also capable of switching off complete routers. Hence, depending on the actual communication an optimal energy saving measure is applied. In addition, a working FPGA prototype is available and hence more realistic communication patterns can be generated.

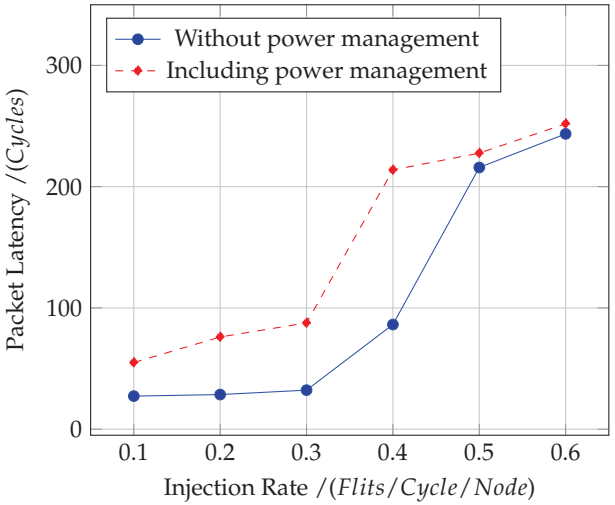
4.6 Summary

The power dissipation is massively rising when technology scales down. Hence, techniques for power reduction are essential, also for network resources. In order to figure out the modules with the highest amount of power consumption, estimation and modeling of the hardware architecture need to be done.
Since the power consumption contingents on the load capacitance, the operation frequency, and the supply voltage, a reduction of any of these three parameters results in a reduction of the power dissipation. Depending on which of these parameters should be reduced, different methods are available, such as power gating or frequency scaling.

In this chapter, a power management controller for on chip networks has been presented. Each router comprises its own controlling unit to realize a decentralized power optimization while minimizing the performance loss. Depending on the degree of router utilization, the controller either suggest a frequency scaling factor or power gates the complete router in case of no utilization.
The resource overhead of the *power management controller (PMC)* is kept to a minimum, while saving an average of 14.2 % of the network power. Also an FPGA based implementation of the approach is available for evaluation as well as an ASIC design flow for power simulation.

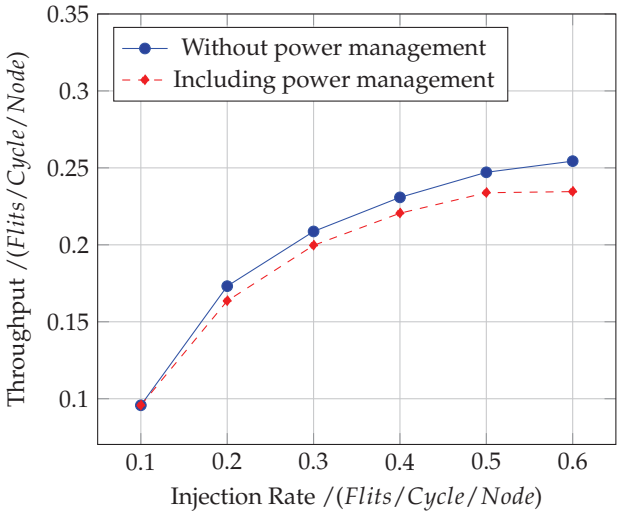


(a) Throughput

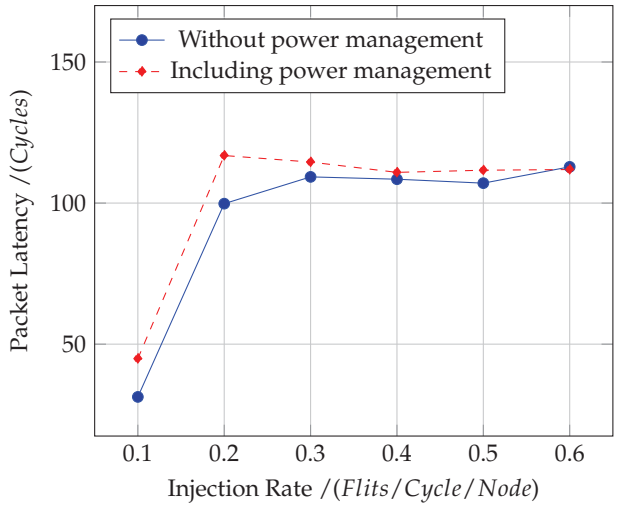


(b) Packet Latency

Figure 4.9: Network performance using uniform random traffic of 8x8 two-dimensional mesh networks, each with homogeneous bandwidth of 128 bit. The reference implementation, which does not include a power management controller is compared to the design of this work.



(a) Throughput



(b) Packet Latency

Figure 4.10: Network performance of 8x8 two-dimensional mesh networks using transpose traffic. Comparing a reference implementation with the design including power management.

5 Automated Agile System Exploration

The development of new hardware components including requirement analysis, implementation, production, and test takes several years from the first step till the final product release. To cut costs and be able to react fast on changing customer demands, a reduction of the time to market is essential. Since design verification is the most time consuming task in the development process, a reduction of verification time is crucial.

The manufacturing process of integrated circuit designs is time consuming and quite expensive. Especially critical, design faults cannot be eliminated after manufacturing has started. This is the reason why an extensive verification and validation process should be done in advance to remove as many failures as possible. The verification process can be subdivided into four classes: simulation based technologies; static technologies; formal technologies; and physical verification and analysis (see section 2.7.2). This work focuses on simulation based verification only due to the complexity of the verification process. Since some failures can only be extracted by running extensive software tests, a prototype of the design is essential. In addition, the prototype enables a parallel development of hardware and software components. So software can be released in parallel to the hardware.

The main criteria for benchmarking information processing devices are the following: (1) scalability; (2) speed; (3) energy efficiency, and (4) resource consumption. To get a rough overview of the coverage of the verification process presented in this work figure 5.1 compares ASIC designs, FPGA prototypes, and HDL simulation of single units and the complete architecture in terms of complexity of the HDL design, simulation time, and test complexity. With increasing test and design complexity, the simulation time increases dramatically. HDL based timing accurate simulation including complex software tests (for example simulating the entire operating system) is hardly usable since it take several days to run the simulation. Also, the benefit of HDL simulation to have access to all signals fail to apply due to the huge amount of data. ASIC designs offer the shortest time to run complex test, but the time and amount of money to design an ASIC is not commensurate. FPGA on the other hand have almost the same possibilities while the design can be easily changed without further costs. However, the creation

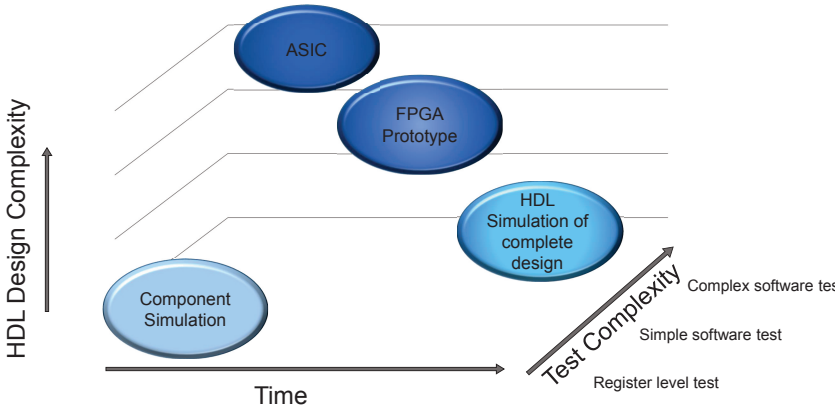


Figure 5.1: Comparison of simulation time, depending on test and HDL design complexity.

of designs for multi-FPGA platforms is also a time consuming task and requires extensive knowledge about the hardware design.

Formal verification of circuits with manageable size delivers good results in confirming the absence of failures for a given property. Although, with rising design complexity, the verification decision is reached after long runtimes. Hence, formal verification is not applicable for many-core systems with hundreds of cores. The verification process presented in this chapter bases only on simulation based technologies (see section 2.7.2). The following sections describe a combination of abstracted simulation, cycle-accurate simulation, and FPGA-based prototyping. The combination of those three methods is necessary to simulate as well as demonstrate the benefits of the new network architectures in regard of performance and power consumption. High-level simulations are used to verify the concepts at high simulation speed at an early stage. Also, this kind of simulations are used for application development. Cycle-accurate RTL simulation is used to verify the functionality of smaller architectural blocks. Furthermore, it allows developing and testing interfaces and the underlying protocols for communication between modules.

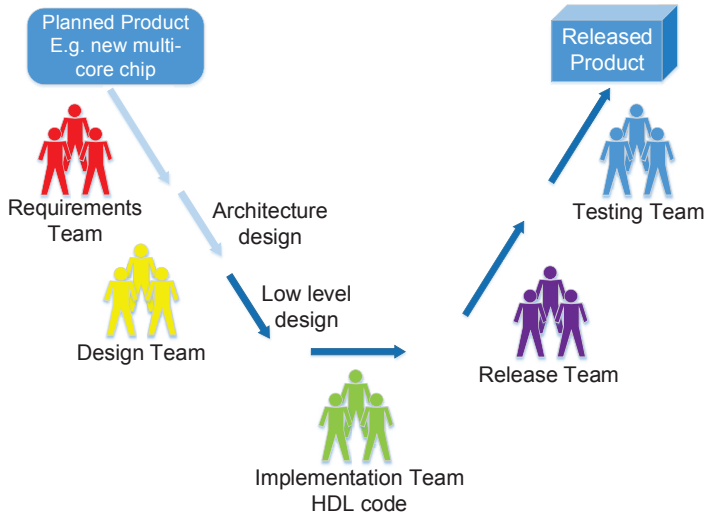


Figure 5.2: The V-model as an example for serial systems engineering process models. This work focuses on the low level design implementation and test. Requirement analysis and architecture design are not included.

5.1 System Development Life Cycle

The development of new digital systems can be divided into several stages; planning, creating, testing, and deployment of the system. To manage the development complexity, several process models have been created. In conventional development process models the stages are disposed in serial fashion. The two most well known serial models are the Waterfall model and V-model. Figure 5.2 shows a diagram of the V-model.
At the beginning of the life cycle of a new product, a developer team collects the system requirements. The decisions made at that step, will not change during the complete lifetime. However, some requirements are not identifiable during the early phases or can change due to customer requirements changes. To react on rapid changes, more flexible process models have been introduced, summarized as agile development process models. They are shortly introduced in the following section.

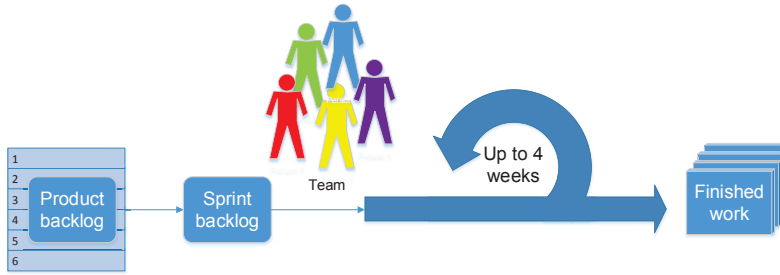


Figure 5.3: SCRUM framework, a subset of an agile design flow.

5

5.1.1 Agile Design Process Model

Agile development methods have been established in software development since many years [33]. They enable an increasing frequency in which new features and products can be released.
To adopt the benefits of agile principle also to hardware development a flexible test environment is necessary. In conventional process models each team is specialized on one topic, see figure 5.2. The teams communicate using extensive documentation. While one of the major components of agile methods are mixed interdisciplinary teams where each new developed component can directly be tested on its own or as part of the entire design (see figure 5.3). This figure represents the SCRUM framework [90], where the project is split up into small work packages which will then be finished within a month. The work packages are pooled as the product backlog.
Section 5.3 and the following describe the simulation and prototyping environment which have been part of this work. A major challenge to apply agile methods to embedded hardware development is to split the product into smaller working packages. Each working package has to be implemented and tested on its own and afterwards integrated into the design. The main achievement of this work is the integration of various scripts into a graphical user interface which fully automatically simulate, synthesize and test new hardware components.

5.2 State of the Art

Field programmable gate array (FPGA) prototyping is widely used in semiconductor design. The verification of simple and small designs can be made with common

FPGA evaluation boards. If it comes to bigger designs, prototyping platforms with multiple FPGAs are used. In case of this work the *CHIPit* prototyping platform from Synopsys®, including six Virtex5 FPGAs has been used as well as the *QuadV7* from ProFPGA. Common rapid FPGA-based prototyping hardware platforms like Haps-80 series from Synopsys® [4] and Quad KU115 Logic Module from S2C [3], as well as the CHIPit system, do not provide many interfaces. Even custom designed prototyping platforms are not capable to provide as many interfaces as a multi-processor design with hundreds of cores would require. Hence, it would not be possible to debug all processors on core level.
Previous debug modules did not give the possibility to trace the processor instructions as well as the network communication. The Gaisler LEON system originally did not include a network system and hence one connection for the debugging support unit was sufficient. Novel NoC systems require more than one debug support unit and therefore it is hardly possible to connect every unit with one physical interface, since FPGA prototypes have a limited number of debug interfaces. In [97] the network resources are used to transmit debug information through a *Joint Test Action Group (JTAG)* interface. However, this work only comprise four network nodes. This yields in two problems for huger networks with a single JTAG connection. First, the network resources are overloaded by debugging information and second, with an increasing amount of debugging data, a single JTAG interface builds a bottleneck. A different approach to handle limited number of debug interfaces is presented in [22]. The author present a virtual debug interface where all cores in the system are connected to one *universal asynchronous receiver/transmitter (UART)* interface. One drawback of the virtual UART concept is the limitation of the bandwidth to the host computer due to the use of a single UART connection. In contrast to other solutions, the transactor based debugging presented in this work only requires one physical interface while most other solution need as many interfaces as the number of debugging units in the design.

Kaisti et al. [57] present the characteristics of embedded systems and the associated challenges arising when applying agile methods to the development process. They also give guidance how to map agile principles on hardware development. However, they do not propose concrete concepts and tools to handle the new challenges. T. Punkka [84] emphasizes that prototyping is the main part of agile hardware development, including FPGA prototypes. Nevertheless, this work does not show any tools or examples how the suggested process should look like. In contrast, the following sections present a hardware development tool flow which enables an agile development process.

The university of Stanford developed a cycle accurate NoC simulator called BookSim [55]. The simulator supports different topologies, such as mesh, torus, and flattened butterfly networks. Also, it provides diverse routing algorithms and

includes options for customizing the networks' router micro-architecture.
The disadvantage of the simulator described above, it only simulates the network and do not include the processing elements. A first work combining network simulators and simulators for multi-processor systems has been described in [115].
Another automated process called ATLAS, for the design flow of the Hermes network on chip, is presented in [71]. The model focuses on the generation and evaluation of the network infrastructure. The processing elements connected to the network routers are not included and the network traffic is generated by a traffic generator. In contrast, the approach presented in the present work includes the network resources as well as the processing elements. ATLAS enables quick evaluation of the performance and power consumption of different NoC configurations. Disadvantage of all available tools, they only support homogeneous architectures or they do not include the processing elements at all.

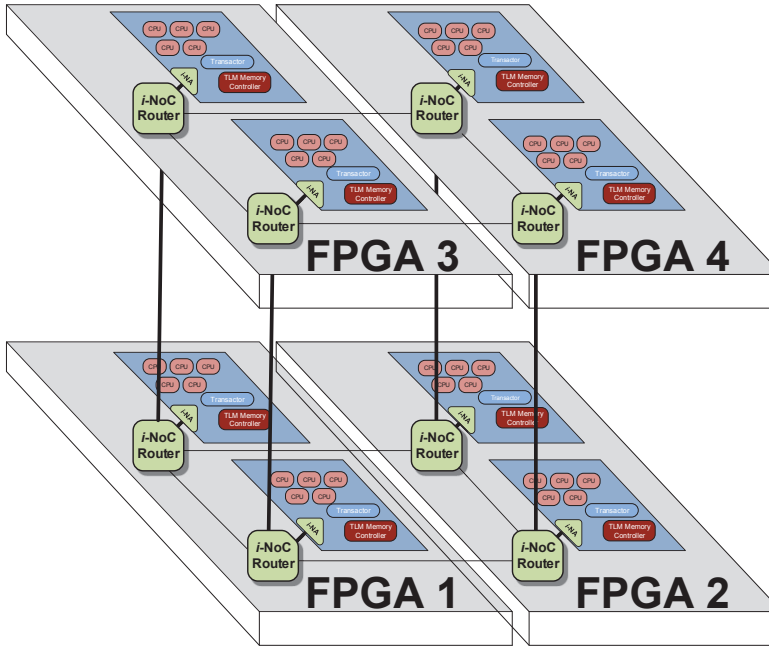


Figure 5.4: Prototype of a three-dimensional multi-core architecture. A 2x2x2 mesh is partitioned across four FPGAs.

5.3 Simulation

Software based, cycle-accurate simulators are typically used for hardware development. Cycle accurate hardware simulation delivers precise timing behavior of the hardware and software running on top but dramatically decreases the speed of the simulation. Nevertheless, for hardware development software-based RTL simulation is the only reasonable method. For software and functional simulation the common method is to raise the level by abstracting from the underlying hardware to increase simulation speed. However, raising the abstraction level lowers the simulation accuracy as timing can no longer be modeled precisely. As high-level simulations do not require precise hardware models, they are used early in the development flow for software development and debugging. Figure 5.4 shows a rough draft of the target architecture which should later be realized as an FPGA prototype.

ModelSim® is a powerful HDL simulation tool that allows to stimulate the inputs of modules and view both outputs and internal signals. However, simulation on register transfer level can not model the effect of meta-stability which can occur in asynchronous designs. In addition to the simulation a prototype of the system is essential to cover also these kinds of failures.

5.3.1 Co-Simulation

The Co-Simulation is an extension to the described simulation method above. For the Co-Simulation it is mandatory that parts of the design are already verified and running stable on FPGA prototypes. While other design components are either redesigned or newly developed. The FPGA prototype is then connected directly with the HDL simulation. This is especially beneficial to speed up simulation time. The following describes the technical realization of the Co-Simulation on the CHIPit platform, but this technique can also be applied to other prototyping platforms. The Co-Simulation process is described in detail in [FHMB14].

The HDL bridge is dedicated to move HDL modules, IPs or whole designs into the CHIPit platform and connect them to a simulation software running on a host computer. This process is also known as Co-Simulation. The DUT is moved into a hardware, and it is replaced by a simulator environment. This simulator environment will have the same I/O ports and parameters as the original HDL module. The simulator environment is directly connected with the CHIPit platform. Using this connection the signal values will be exchanged with the real HDL module implemented into the hardware. The opposite side is implemented into the CHIPit platform. There is also an environment, which

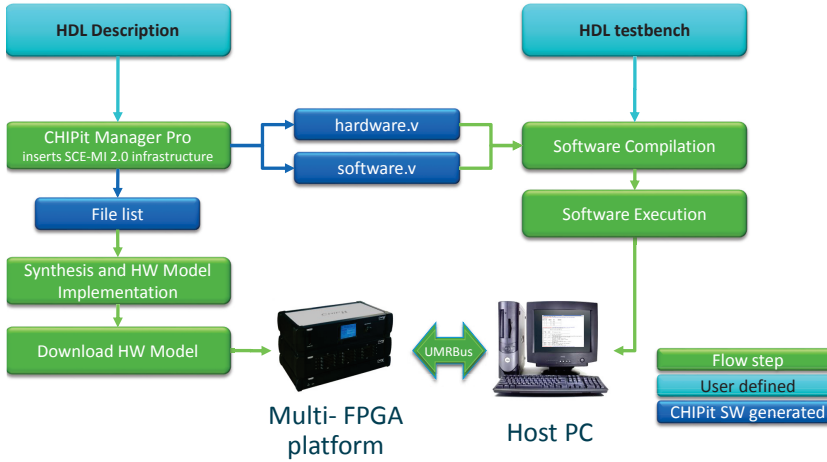


Figure 5.5: Co-simulation flow using a multi-FPGA platform in combination with an HDL simulation running on a host PC.

is called hardware environment. The hardware environment is connected to the corresponding simulator environment. The HDL module is connected to the hardware environment. It receives the signal values from the simulator, applies these values to the HDL module and sends the result values from the HDL module back to the simulator. The hardware environment is like a wrapper and is available in Verilog or VHDL. The physical communication between the software and the hardware is implemented using the UMRBus Communication System. Figure 5.5 shows the Co-simulation tool flow.

5.4 FPGA Based Hardware Prototyping

In regard of decreasing product life cycles and to react on customer individual conveniences, designers have less time to develop new hardware architectures. To hold down the development costs of an ASIC it is mandatory to extensively test and simulate the design before producing the actual ASIC. FPGA-based prototyping systems are designed for system and software validation. As FPGA

devices have become larger and faster, verifying functionality of ASIC designs in FPGAs has become an effective and economical method of verification. To map ASIC designs consisting of millions of gates to a prototyping platform, development boards with multiple FPGAs are necessary. However, mapping the design across multiple FPGAs makes the verification process quite complex. To obtain optimal performance and usability, automatic synthesis flow and testing will be introduced in this chapter.

Within this work a scalable multi-core embedded processor design is realized as a prototype on a multi-FPGA platform. The two different FPGA platforms which have been used are described briefly in the following section. For system evaluation and design space exploration, different implementations of the NoC based multi-core architecture have been realized. As an example, one realization is pictured in figure 5.4, where a three-dimensional mesh is partitioned across four FPGAs.

5.4.1 Prototyping Platforms

Prototypes of huge multi-core designs, especially with realistic cache sizes require many FPGA resources. Common FPGA development board usually comprise only single FPGAs and provide only limited number of I/O interfaces as well as a limited amount of memory. To cope with the huge resource demand and to offer large debugging possibilities a multi-FPGA prototype platform has been used in this work. The following two sections describe first the *CHIPit* platform from Synopsys® followed by the *Quad V7* platform from proFPGA.

5.4.1.1 CHIPit Prototyping Platform

The CHIPit Platinum Edition System is a high-capacity, high-speed emulation and rapid prototyping system for ASIC designs and is the predecessor of the recent HAPS systems [4]. It contains six Virtex-5 LX330T FPGAs which is equivalent to 12 million ASIC gates; furthermore the platform can be extended up to 18 FPGAs. Next to each FPGA there is an SRAM memory placed with a capacity of 8 Mbyte each. In addition, on the top of the prototyping system there are extension board interfaces to plug in up to six half-size extension boards. These boards can contain different extensions of the system as memory, Ethernet or other interfaces. Figure 5.6 shows an image of the CHIPit Platinum Edition System with extension board connectors on the top and additional logic analyzer debug connectors on the front side. In addition to the extension board interfaces, the CHIPit is connected to a host computer via the *Universal Multi-Resource Bus (UMRbus)*. This bus enables co-simulation and builds the interconnection for transactor based communication

Figure 5.6: CHIPit prototyping platform containing six Virtex-5 LX330T FPGAs.

where hardware running on the FPGAs is connected to software which is executed on the host computer.

The FPGAs are arranged on two boards connected by a main board, see figure 5.7. The FPGAs, SSRAM memory modules and extension boards are connected with each other via switches. The configuration of the switches has to be done during design time.

5.4.1.2 Quad V7 Prototyping Platform

The quad V7 prototyping platform from proFPGA is similarly constructed as the CHIPit platform. It contains four Virtex-7 2000T FPGAs (XC7V2000T) which are currently the biggest available FPGAs on the market. The quad V7 platform has the capacity to simulate designs with up to 48 M ASIC gates. The four FPGA modules can be connected with each other using cables. The same connectors can be used for extension boards containing I/O interfaces or memory. Figure 5.8 shows the architecture of the prototyping platform. The yellow lines represent the FPGA user I/Os for inter FPGA connections or for extension boards. The green

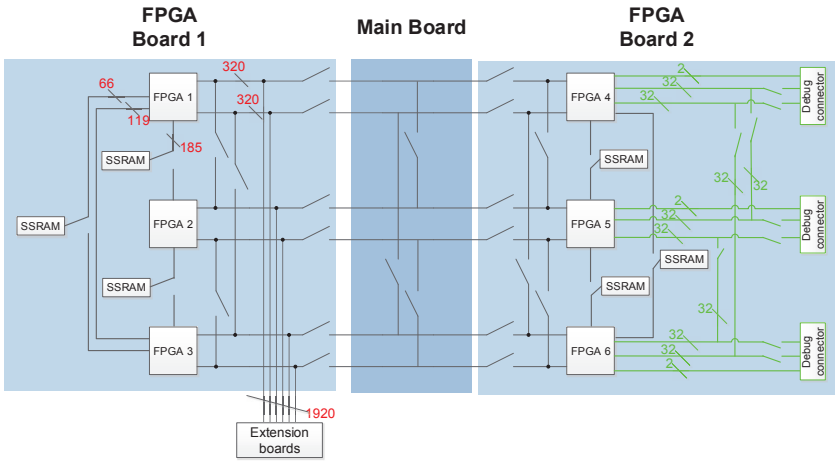


Figure 5.7: Switch matrix of the CHIPit platform. Additional to the signals shown, there are 32 fixed routed global signals, one reset resources, the clock resources and the UMR bus. The debugging connectors are placed on the left side, connected to FPGA 1-3.

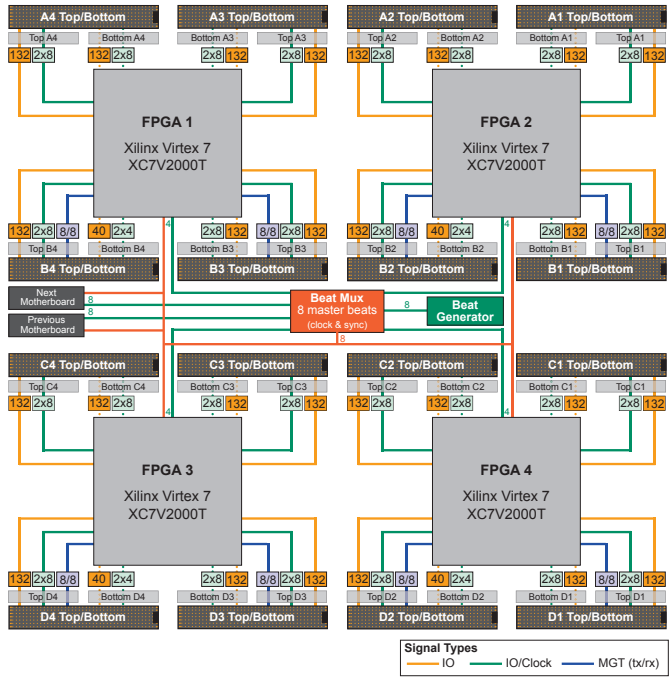


Figure 5.8: Architecture of the proFPGA quad V7 prototyping platform [2].

lines represent the clock tree. While the blue lines represent the *Multi-Gigabit Transceiver (MGT)* where each FPGA offer 16 connections.

5.4.2 Design Partitioning

When working with platforms consisting of multiple chips, the design needs to be portioned. There are different challenges arising, working with portioned designs, such as timing closure and restricted number of inter chip connections. Following, the partitioning of an FPGA design is usually an extensive piece of work which cannot be easily automated. However, the tile based network on chip hardware architecture used in this work supports the automatic generation of partitioned bitstreams quite well. The partition process will split the design only on the border between the routers and so each tile will remain as a whole on one FPGA. Figure 5.14 shows a 2x2 mesh design partitioned across four FPGAs.

The allocation of the tiles to one certain FPGA has to be written down in the tool flow script, because the placement matters for example in regards of memory extension boards and debug connectivity. However, once the placement is done, even if the internal tile structure changes, it is not necessary to change the partitioning and placement anymore. The number of tiles per FPGA is mainly restricted by the FPGA resources and external memory.
In regards of the number of signals between FPGAs, the regular network structure suits the partitioning quite well too, since only signals between two routers are transferred. Hence, pin multiplexing can be avoided with the current setup. The current router setup uses four *virtual channel (VC)* and a link bandwidth of 128 bit. This gives the opportunity to automate the complete tool flow and even a user with little knowledge about multi FPGA platforms can generate new bitstreams. The automated tool flow is further discussed in section 5.6.

5.4.3 Debug Connection Interfaces

The number of available physical debug interfaces on prototyping platforms is quite restricted. FPGAs of the Xilinx® Virtex series offer approximately 1000 user IO pins. However, these pins need to be used for inter FPGA connections as well as for connections to external memory. To avoid issues with severely limited pin counts, while at the same time offering a huge amount of debugging interfaces, virtualized debug interfaces will be introduced as shown in [FHB14]. The following section describes one possible implementation of a virtualized debug interface.

5.4.3.1 XACTOR Gen Package

The Synopsys® transactor library for *Advanced Microcontroller Bus Architecture (AMBA)* is the link between AMBA based user designs and a software environment on a PC host machine. With respect to the AMBA-based LEON-Cores that are used in the prototype, these transactors play an important role as detailed later. The library packet includes several AMBA transactors, such as master and slave components for different bus systems, including *Advanced High-performance Bus (AHB)* and *Advanced eXtensible Interface (AXI)*.
The connection between the AHB bus on the FPGA side and software running on a host PC is provided by the UMRbus. The library package includes the hardware IP for each transactor as well as a C++ library; hence the user does not need to care about the UMRbus implementation. Hence, the user only needs to configure the transactor for its own purpose and integrate it into the project. The transctors library could be used to realize the connection for hybrid prototyping as well

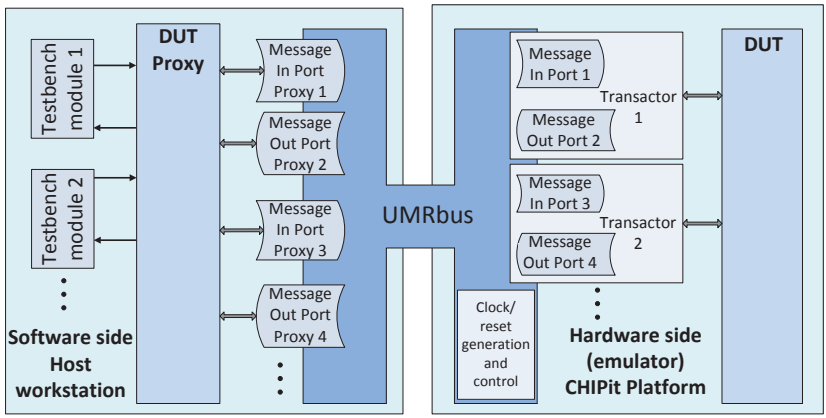


Figure 5.9: UMRBus transactor model.

as in place for all kind of interfaces. For more information about the Transactor Library see [96].

The UMRbus based transactor model is shown in figure 5.9. The left side represents the software part with multiple software instances and the right represents the hardware adapter and the DUT. In the middle lies the UMRBus driver which connects the hardware and software side. Since the communication over the UMRbus is carried out in a serial way, each transactor includes two FIFOs to store incoming and outgoing messages. The inport FIFO holds data packets while the AHB bus fabric processes them. While the outport FIFO queues the messages during the UMRbus is busy. Both FIFO sizes could be customized in the range of 16 to 4095 32-bit words.

Figure 5.10 shows the allocation of UMRBus components across FPGAs of the CHIPit platform. Each UMRbus component inside one FPGA is connected in a serial way, as well as the communication between FPGAs and the UMRbus communication interface is also serialized. Hence, each FPGA requires only one set of UMRbus signal independent of the number of transactor instances inside the FPGA. Up to 64 UMRbus components are allowed within the platform. However, due to limited FPGA resources, only designs with a maximum of 9 tiles can be realized on the CHIPit platform. The Quad V7 platform provides space for mesh based multi-core designs with up to 20 tiles.

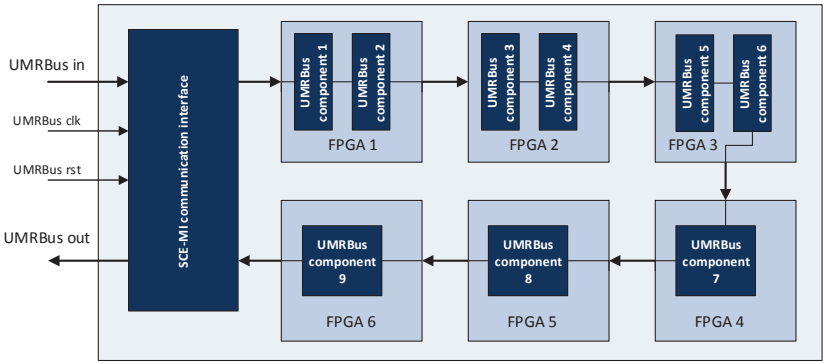


Figure 5.10: Allocation of UMRBus components across FPGAs of the CHIPit Prototyping Platform.

5.4.3.2 AHB Transactor Overhead

Including one AHB transactor into each tile results in a small overhead of resources in contrast of using only serial interfaces. The size of inport and outport FIFOs inside the transactor module could be custom parameterized and hence the consumption of BRAM resources varies. Table 5.1 shows the resource consumption of the AHB transactor implementation in contrast to a serial UART. The number of pins used by one transactor component is also higher than for one simple serial interfaces, but if more than one transactor component is placed into one FPGA, they all share the same pin resources. Figure 5.11 shows the pin utilization, in terms of different number of tiles, for implementation with serial interfaces, the UMRbus based concept, and a reference design with virtual UART interfaces [22].

5.4.4 Gaisler Debug tool GRMON2

The Aeoroflex Gaisler tool GRMON2 [9] is a general debug monitor for LEON3 processors, and for SoC designs based on the GRLIB IP library. The *Debug Support Unit (DSU)* provides a non intrusive debug environment for the Leon cores on real target hardware. The LEON3 DSU can be controlled through any AMBA AHB master in a *system on chip (SoC)* design. The debug interface can be of various types: serial UART, Ethernet or a user defined interface can be used. The GRMON2 monitor interfaces to the on chip debug support unit DSU, implementing

	Virtex5LX330	UART interface	AHB IO transactor
LUTs	189,183	336	3744 (1,98 %)
Register	207,360	184	3699 (1,78 %)
BRAM	288	0	7 (2,43 %)
IO pins	1,200	2	20 (1,67 %)

Table 5.1: Design space exploration of debugging alternatives.

a large range of debug functions. These functions are for example download and execution of software applications, built-in disassembler and trace buffer management, and read and write access to all system registers and memory which are accessible in the tile local address range. Since the GRMON2 debug monitor is intended to debug SoC designs but not *network on chip (NoC)* designs, it is necessary to have multiple instances of GRMON2 in NoC designs such as the invasive hardware architecture.
In addition to the usual prototyping challenges there are some specific issues arising from invasive computing technology. Since hardware and software is built up from scratch, one debug interface in each tile is necessary. For example software loading from the main memory is not yet implemented, and so the software must be loaded into each tile. However, this amount of debug interfaces usually exceeds the number of interfaces available on prototyping systems. Another issue is the amount of required memory. Since the goal is to run parallel applications competing with execution on current multi-core computer, the size of memory is crucial. FPGA development boards cannot cover this demand.

For optimal design debugging and loading software to each individual tile in the design, one DSU is required in each tile. Following from that, as many interfaces are necessary as the design tile count. For designs with more than two tiles, the available physical interfaces are not sufficient. Consequently, virtual interfaces are used to connect the DSU with host PC. One possible implementation of virtual interface is described in section 5.4.3.1. A loadable module must be compiled into a shared object library which includes AHB read and write functions as well as an initialization of the transactor. The user than can start GRMON2 in a common way with the shared library and the tile ID as a parameter:

./grmon -dback my_io.so -dbackarg Tile_ID

After starting GRMON2, the UMRbus interface will be initialized. Afterward data could be read and written to any memory location which is implemented and accessible through the AHB bus.

5.4.5 Debugging System Performance

Considering the performance of system components, the software on the host computer side runs at GHz speed. The FPGAs design runs with a lower frequency at 25MHz and the UMRBus communication link is running at 70MHz, but because of a long latency of the link, the transmission rate could be translated roughly to 100 KHz. However, it is possible to achieve a maximum data rate of 100 Mb/sec, by using asynchronous communication across the link and by transferring large packets over the UMRBus.

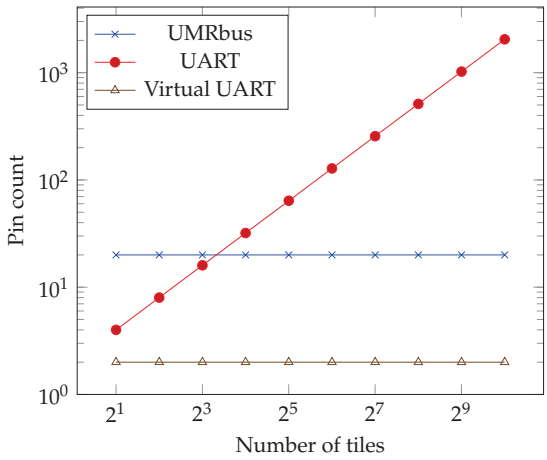


Figure 5.11: Required FPGA or ASIC pins for realization of different debugging. alternatives.

Since the transactor data is transmitted serially, the data rate decrease if all transactors are transferring data at the same time. However, since parallel debugging of all processors is not necessary on the one hand and on the other hand the debug information does not compromise huge data sets, the data transmission rate is not that affected during debugging. Only if huge amount of data is transferred at the same time, the transmission rate of a single transactor decreases. Figure 5.12

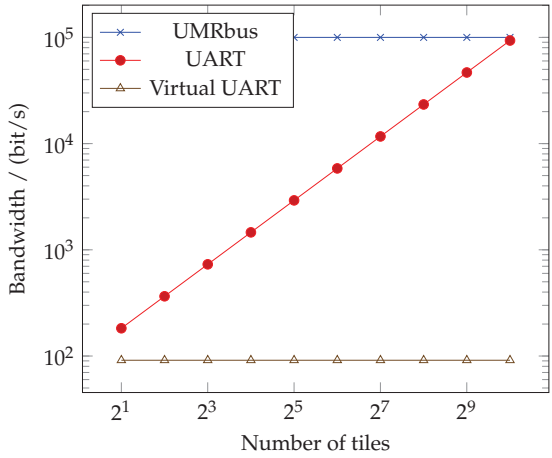


Figure 5.12: Bandwidth of debugging alternatives

illustrates the bandwidth of the debugging interfaces according to the number of tiles in the design. The experimental tests only considered designs with maximum of nine tiles. Thereby, it could be observed that the accumulated bandwidth decreases if one UMRbus connection for multiple FPGAs is used.

Figure 5.13 shows an example view of the GRMON2 connections. At start-up, information of tile internal components are reported. Because one GRMON2 connection for each tile need to be opened, a simple script to open all monitors at once in one screen window has been developed. At the bottom of the screen in figure 5.13, the tile ID of the current view is highlighted. It is also possible to load and run applications automatically after opening the connection. Thus, the user only need to start the script once and then only need to interpret the different monitoring data printed in the debugging window.

The debugging process of complex NoC systems is still a time consuming task, but thanks to the given comprehensive monitoring information, also bugs could be detected whose cause of error happens a huge space of time before the bug is visible to users.

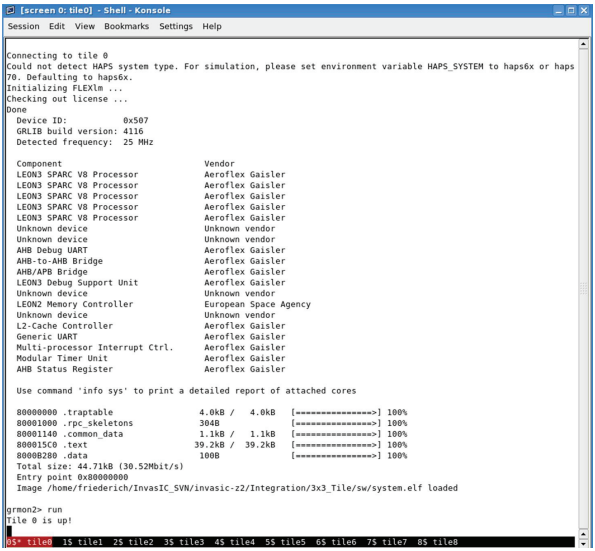


Figure 5.13: GRMON2 debug window at system start.

5.5 Verification and Testing

With rising design complexity the test process complexity also increases. After successful simulation, the design functionality need to be verified with more comprehensive software tests. Also, to reduce the time to market, software development is done in parallel to the embedded hardware development. As soon as the first prototype is available, first real applications can be tested. This offers the advantage that software can be improved already in an early stage of the product development process. Furthermore, interaction of hardware and software can be investigated.

5.5.1 Design Space Exploration

Evaluation of competing system architectures and diverse parameter settings is done by comparing the trade-offs given by results from performance and power measurements. Therefor different testbench scenarios of multiple architectures with varying parameters and different communication patterns have been created. Based on this, an HDL based simulation has been used for first evaluations.

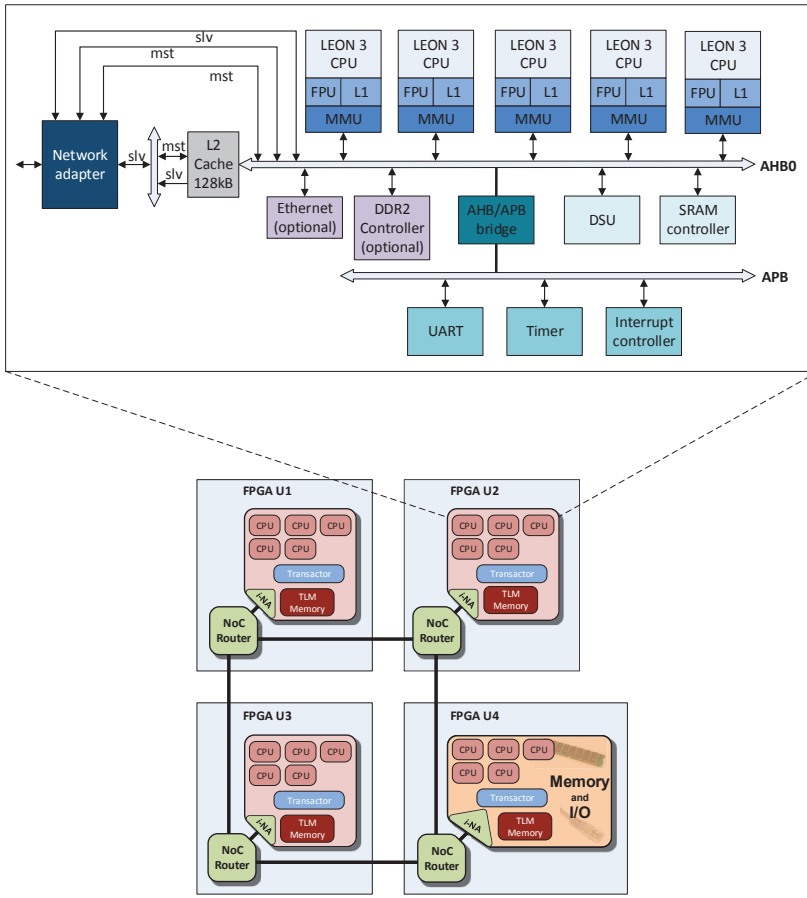


Figure 5.14: Hardware architecture setup for multi-FPGA prototype. The bottom half shows a 2x2 mesh network distributed across four FPGAs. The upper half shows details of the tile structure.

Afterwards, the same HDL files are used to synthesize the design on the one hand for area and power assessment using an ASIC synthesis flow. On the other hand using these files for a FPGA based prototype.

During this work, different extensions to the original network design have been implemented. To contrast the different implementations with each other for example in terms of resource consumption, synthesis results have been analyzed. Table 5.2 shows the resource consumption of different modules of the NoC architecture.

Configuration	Xilinx® Virtex7	
	Cells	RAM64
2D router	6,361	–
3D router	9,865	–
2D router including power management (PM)	6,440	–
Network adapter (NA)	6,246	8
Processing node (tile)	61,667	240
2x2 design, 5 cores	303,236	994

Table 5.2: Resource consumption of different network and tile components. Details about the design parameters are given in table 4.2.

5.5.2 Hardware Regression Test

Every time, when new hardware components are added to a design or existing units are updated, it need to be verified that the system functionality has not been influenced by the changes. In software development, regression tests are in wide use. This principle is now mapped onto hardware development. The implementation need to verified to a certain level before the regression tests can start. Continuous and self-triggered test iterations are necessary that the test can run autonomous.
The following described the benchmarks which have been applied for the hardware regression test, followed by the results. The scripts for automated testing have been part of a collaborative work that arose from the *InvasIC* project [7].
Since the FPGA prototype is quite similar to the intended chip, the test software

produced for these FPGA based hardware regression tests, can later be reused for production tests on the produced chip.

5.5.2.1 NAS Parallel Benchmarks

In the early '90s the *Numerical Aerodynamic Simulation* program developed a multi-processor system, consisting of more than 1,000 processor cores to simulate an entire aerospace vehicle system within a computing time of few hours [14]. At that time there has been no feasibility to test and evaluate such huge systems and prove functionality before production. For performance evaluation of such huge multi-processor systems the *NASA Advanced Supercomputing (NAS)* parallel benchmarking system has been developed then. The benchmarks do not include any hardware specifics, so they can be used for evaluation of any parallel architecture, so it can also be applied to the *InvasIC* network architecture. The software code of the benchmarks can be configured to fit the architecture requirements.

Table 5.3 lists the eight NAS applications which have been used to evaluate the *invasive* architecture. The first five benchmarks in this table are parallel kernel benchmarks. The last three applications in table 5.3 solve discretized versions of the unsteady, compressible Navier-Stokes equations in three spatial dimensions [13]. Each benchmark focuses upon different matters; (1) computation intensive application with no inter-processor communication; and (2) communication intensive applications with inter-tile communication.

The NAS parallel benchmarks have actually been developed for performance evaluation but in this work they have been applied for hardware regression tests. After each hardware update, it is necessary to verify the system is still functional the same way as before. Different hardware versions can be compared with each other and it is possible to visualize the improvement.

5.5.2.2 Regression Test Results

Figure 5.15 shows part of the regression test result page. The user need to specify the applications which should be tested as well as the number of iterations each test is repeated. After successful finishing the regression test, an HTML file with the test results is generated. The coloring of the tests gives a fast overview of the result. Green coloring imply that all iterations ended successful (UMR bus failures do not count in this case), yellow coloring imply that a portion of test iterations ended without errors and red coloring imply that all test iterations have failed.
If failures are noticed, the script directly tries to assign them to known issues,

Results for bitfile integration-ver_2015_08_17

	ok	hw	trap	DMA	sysilet	res	ctx	cic	heap	MPI	x10	UMR	timeout	unknown
bt-4t5c-chipit	8	0	0	0	0	0	0	0	0	0	0	2	0	0
bt-4t5c-hwcic-chipit	4	0	0	0	0	0	0	0	0	0	0	2	4	0
bt-4t5c-hwcpy-chipit	6	0	0	1	0	0	0	0	0	3	0	0	0	0
bt-4t5c-hwdma-chipit	0	0	0	1	0	0	0	0	0	0	1	0	8	0
cg-4t5c-chipit	9	0	0	0	0	0	0	0	0	0	0	1	0	0
cg-4t5c-hwcic-chipit	8	0	0	0	0	0	0	0	0	0	0	2	0	0
cg-4t5c-hwcpy-chipit	9	0	0	0	0	0	0	0	0	0	0	1	0	0
cg-4t5c-hwdma-chipit	0	0	0	0	0	0	0	0	1	1	0	2	6	0
ep-4t5c-chipit	9	0	0	0	0	0	0	0	0	0	0	1	0	0
ep-4t5c-hwcic-chipit	7	0	0	0	0	0	0	0	0	0	0	3	0	0
ep-4t5c-hwcpy-chipit	7	0	0	0	0	0	0	0	0	0	0	3	0	0
ep-4t5c-hwdma-chipit	0	0	0	3	0	0	0	0	0	4	1	2	0	0

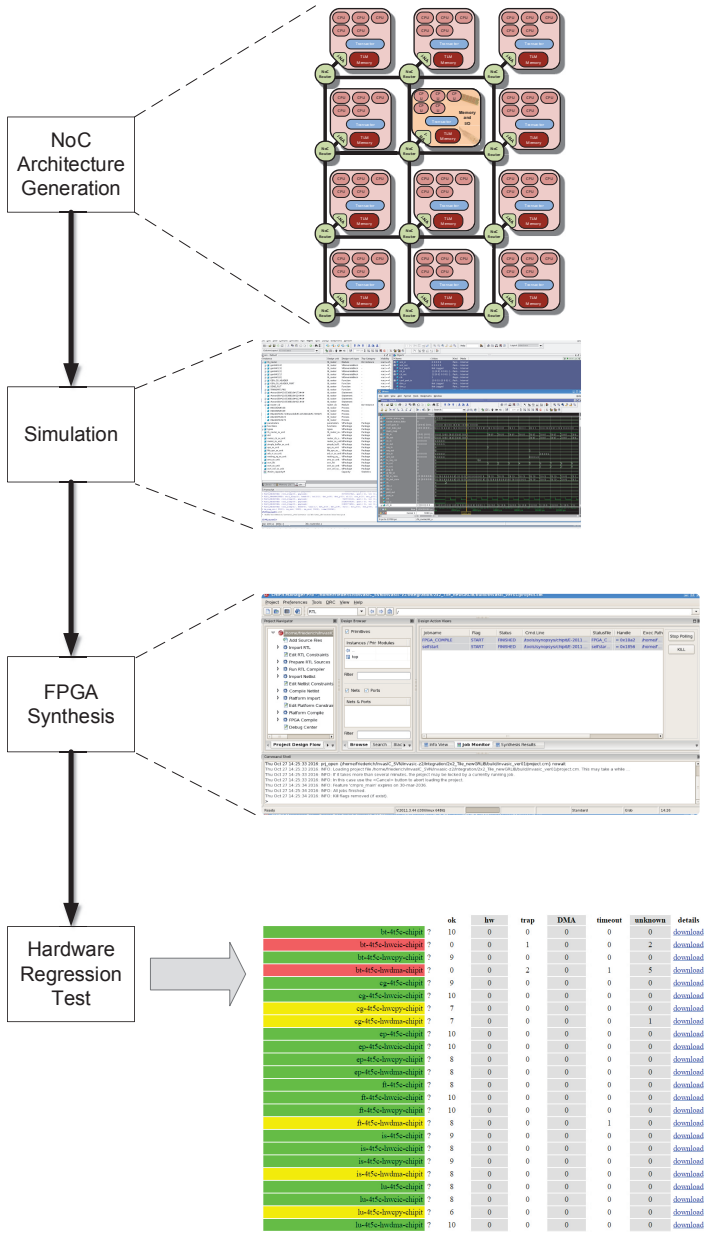
Figure 5.15: Regression test result page.

such as *DMA* (operating system detected a problem with DMA request) or *heap* (C library detected a problem during free).

5.6 InvasIC Project Generator

The development and evaluation tool flow as described in the previous sections consists of many steps, e.g. simulation, synthesis, and test. Figure 5.16 shows the complete tool flow which builds the framework of the *graphical user interface (GUI)*. For automation and reusability reasons, scripts for each of these steps has been developed. The user does not need a detailed view of each step any longer. However, it is still necessary to customize the hardware files by hand and to type the commands to start the scripts. Instead of command line interfaces and to simplify the usage, a GUI combining all scripts into one tool has been developed in this work. Also, this opens up the usage of FPGA prototypes for users without any knowledge about hardware design, while they are still able to customize the hardware architecture.

Besides the better usability a second issue came to the fore. One of the major difficulties in designing new heterogeneous devices is testing the devices with multiple configurations. A test infrastructure for comparison of the different configurations is necessary in order to maintain a time and cost efficient design verification.



	ok	hw	trap	DMA	timeout	unknown	details
bt-4t5c-chipit	10	0	0	0	0	0	download
bt-4t5c-hwvic-chipit	0	0	1	0	0	2	download
bt-4t5c-hwcpy-chipit	9	0	0	0	0	0	download
bt-4t5c-hwdma-chipit	0	0	2	0	1	5	download
cg-4t5c-chipit	9	0	0	0	0	0	download
cg-4t5c-hwvic-chipit	10	0	0	0	0	0	download
cg-4t5c-hwcpy-chipit	7	0	0	0	0	0	download
cg-4t5c-hwdma-chipit	7	0	0	0	0	1	download
ep-4t5c-chipit	10	0	0	0	0	0	download
ep-4t5c-hwvic-chipit	10	0	0	0	0	0	download
ep-4t5c-hwcpy-chipit	8	0	0	0	0	0	download
ep-4t5c-hwdma-chipit	8	0	0	0	0	0	download
ft-4t5c-chipit	8	0	0	0	0	0	download
ft-4t5c-hwvic-chipit	10	0	0	0	0	0	download
ft-4t5c-hwcpy-chipit	10	0	0	0	0	0	download
ft-4t5c-hwdma-chipit	8	0	0	0	1	0	download
is-4t5c-chipit	9	0	0	0	0	0	download
is-4t5c-hwvic-chipit	8	0	0	0	0	0	download
is-4t5c-hwcpy-chipit	9	0	0	0	0	0	download
is-4t5c-hwdma-chipit	8	0	0	0	0	0	download
lu-4t5c-chipit	8	0	0	0	0	0	download
lu-4t5c-hwvic-chipit	8	0	0	0	0	0	download
lu-4t5c-hwcpy-chipit	6	0	0	0	0	0	download
lu-4t5c-hwdma-chipit	10	0	0	0	0	0	download

Figure 5.16: Framework of the *InvasIC Project Generator*.

Figure 5.17: Start window of *InvasIC Project Generator* GUI.

The following chapter describes the structure of the *InvasIC Project Generator* GUI and a test environment to accelerate the test process.

5.6.1 GUI Implementation

A *graphical user interface (GUI)* gives the opportunity that the user does not require deep knowledge about the hardware code to customize the architectural design. As well as it does not require command-line knowledge. The following describes the implementation of the GUI and the different windows which are visible for the user.

To implement a multi-platform GUI, the tool QT creator [5] has been used for development. Qt is an object-oriented user interface written in C++. The generated software code can be compiled for Windows and as well for Linux systems which was one of the basic requirements to be platform independent.

When the user start the tool, the first window which opens is the "start up screen" to open an existing project or to create a new architecture, see figure 5.17. Currently only mesh based architectures are possible, but with an adjustable size. The user has to specify the X and Y dimension of the architecture. Afterwards the next window will open to select the tiles.

Figure 5.18 shows the window view to define the heterogeneous architecture. The user must choose the content and location of processing as well as memory and I/O tiles. Prefabricated tiles are shown on the right side of the window, depicted by an appropriate image. The content of each tile can be chosen either through a

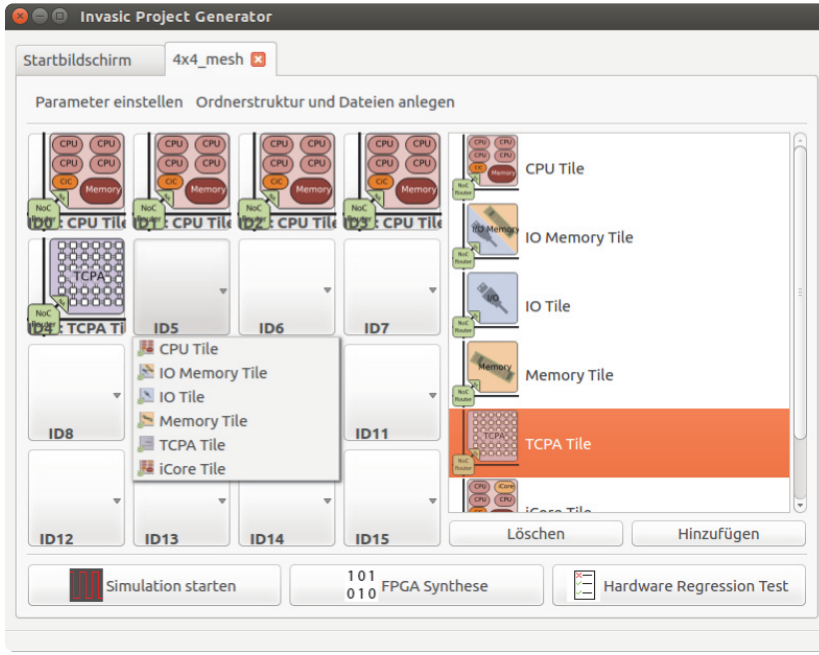


Figure 5.18: Window to define the content of the processing tiles.

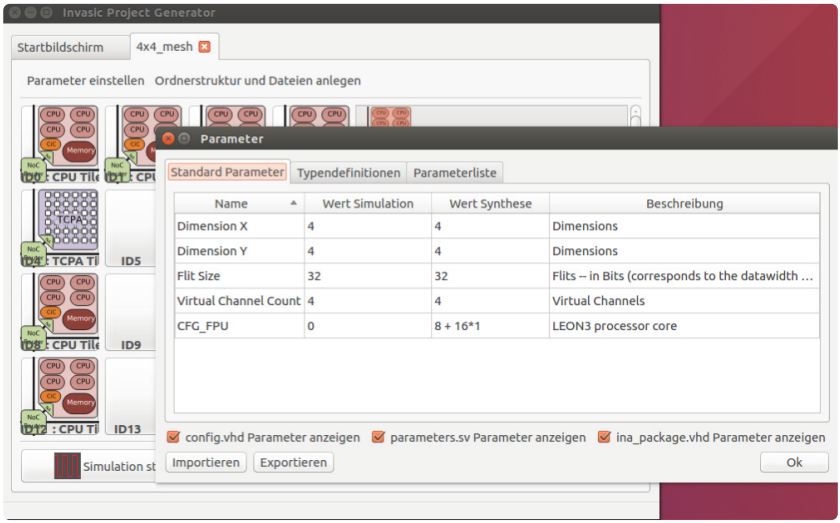


Figure 5.19: Parameter selection window. Neatly arranged, since more frequent used parameters are in a separate list.

drop down list or by drag and drop one of the images into the respective location of the mesh design. It is also possible to add other tiles in addition to the listed ones.

Additional parameters, such as the number of cores per tile can be set in the parameter setting window, see figure 5.19. Each parameter has a default value, so that the user only need to change the ones necessary for its project. Since the number of parameters is quite huge and it would be too confusing to list all parameters in one list, a separate list containing the most frequently used parameters is available. This list is called standard parameters. It is possible to add or delete parameters from that list and save the settings.
One of the most important advantage of the parameter setting in this tool, the parameters are set equally in all hardware files where they are required.
Also, since the configuration for simulation and synthesis can vary, there are two columns for each parameter, one for settings for HDL simulation and a second one for synthesis. For example the floating point unit of the LEON processor can not be used in simulation, while it is required for the FPGA prototype (see last row in figure 5.19).

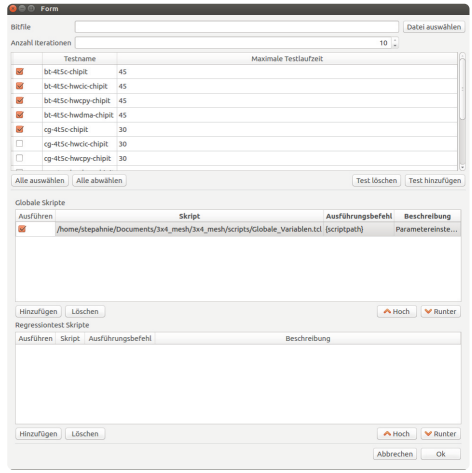


Figure 5.20: Regression test window. The user can choose between predefined tests.

After defining the architecture and modifying the parameters, the top level and configuration hardware files are automatically generated. Following the simulation and also the FPGA synthesis can be started without further assistance of the user.

If the synthesis has finished successfully, the design can be tested on real hardware. Therefor a FPGA prototyping system need to be connected to the machine where the tool is running on. The only modification of the user is the selection of iterations and tests which should be included in the current setting. Figure 5.20 shows the window to select the tests.

In addition to existing NoC simulators, the framework described in this work comprises architecture definition, automatic file generation, hardware simulation with a variety of network traffic patterns as well as regression tests on real hardware. The hardware regression test gives the opportunity to test network and the *processing elements (PEs)* on an FPGA prototype.

Kaisti et al. [57] present the characteristics of embedded systems and the associated challenges arising when applying agile methods to the development process. They also give guidance how to map agile principles on hardware development.

However, they do not propose concrete concepts and tools to handle the new challenges. T. Punkka [84] emphasize that prototyping is the main part of agile hardware development, including FPGA prototypes. Nevertheless, this work does not show any tools or examples how the suggested process should look like. In contrast, this work here presents a hardware development tool flow which enables an agile development process. Another challenge arising with FPGA prototypes are limited number of debug interfaces. In [97] the network resources are used to transmit debug information through a JTAG interface. However, this work only comprise four network nodes. For prototypes with huger networks, this yields in two problems. First, the network resources are overloaded by debugging information and second, with an increasing amount of debugging data, a single JTAG interface builds a bottleneck. A different approach to handle limited number of debug interfaces is presented in [22]. The author present a virtual debug interface where all cores in the system are connected to one UART interface. One drawback of the virtual UART concept is the limitation of the bandwidth to the host computer due to the use of a single UART connection. In contrast to other solutions, the transactor based debugging presented in this work only requires one physical interface while most other solution need as many interfaces as the number of debugging units in the design. Another advantage of the transactor based debugging to conventional interfaces is the considerable increase of the bandwidth.

5.7 Summary

This chapter presented a simulation environment and an FPGA based debugging methodology for large scale many-core architectures. The resulting prototype of the network architecture was used for proof of concept and data rate measurements. In contrast to other solutions, the transactor based debugging only requires one physical interface while most other solution need as many interfaces as the number of debugging units in the design. Thus, there will be no problems because of the lack of interfaces for debugging. Another advantage of the transactor based design to conventional interfaces is the considerable increase of the bandwidth.

For automated agile system exploration and to summarize the significant number of scripts of the simulation, synthesis and test flow, a *graphical user interface (GUI)* tool has been implemented. The user friendly interface enables an easy selection of user defined heterogeneous architectures and automatic file generation.

Abbr.	Benchmark Name	Description
EP	Embarrassingly Parallel	Evaluation of an integral by means of pseudo-random trials. Virtually the kernel does not require inter-processor communication.
MG	MultiGrid	Requires highly structured long distance communication.
CG	Conjugate Gradient	Tests irregular long-distance communication, using unstructured matrix-vector multiplications.
FT	Fast Fourier Transform	A three-dimensional partial differential equation.
IS	Integer Sort	Tests integer computation speed and communication performance.
LU	Lower-Upper symmetric Gauss-Seide	Represents computations associated with the implicit operator of a newer class of implicit computational fluid dynamics algorithms.
SP	Scalar Pentadiagonal	Solution of multiple, independent systems of non-diagonally-dominant, scalar, pentadiagonal equations.
BT	Block Tri-diagonal	Solution of multiple, independent systems of non-diagonally-dominant, block tridiagonal equations.

Table 5.3: The eight NAS parallel benchmarks introduced by Bailey et. al [14].

6 Conclusion and Outlook

6.1 Conclusion

Gordon Moore predicted in 1965 that the transistor count per integrated circuit will double approximately every two years. The prediction is still valid but an end of "Moores law" is in sight. For further performance increase of processors, laptops, mobile phones, and other electronics, new technologies such as three-dimensional chip designs are required. At the same time, bus based multi-core designs have been replaced by network on chips and these architectures need to be mapped onto the new 3D chips.

The goal of this work was to face these new challenges by implementing and mapping a three-dimensional *network on chip (NoC)* onto a 3D chip. One of the main criteria of the new design has been the restriction of vertical interconnects. The development of a new router port with adaptive bandwidth enables fully routed 3D network designs partitioned across several chip layers and still satisfy the restricted number of vertical interconnects. Thereby, the bandwidth of each router port can be individually chosen during design time.

When focusing on high integrated 3D chip designs one major issue arising is the heat generation due to power dissipation. A decentralized, communication aware power management concept have been designed in this work. One *power management controller (PMC)* is integrated into each router to use communication monitoring information to generate fast decisions if a router can be power gated. A main criterion, the communication performance should not be affected by the power management. The results given in section 4.5 show that the performance is only marginal affected including the power management unit to the design while it is possible to save an average of 14 % of power.

To validate the proposed 3D network architecture and the concept of power management, an FPGA based prototype have been build up. To place the huge many core design, a platform comprising multiple FPGAs had to be taken. Each FPGA represents one chip layer of a target architecture. One of the biggest challenges in the development process of new digital hardware designs is the verification. The lack of debug interfaces usually constitutes a problem of FPGA platforms. The concept of a transactor based debug interface solves this problem. Since the

design flow for multi-FPGA platforms is time consuming and requires broad knowledge of the hardware architecture, a *graphical user interface (GUI)* tool has been designed. The tool enables a fast generation of new heterogeneous architecture versions. After defining the architecture, the FPGA design is automatically created and synthesized. As well, it includes a verification process for automatic hardware regression testing.

6.2 Outlook

The Latest silicon transistors have dimensions around a few nano meters [31]. Since further technology shrinking is hardly possible in a few years, completely new ideas are required. The end of pure silicon based chip design is within reach. There are a lot of new ideas on the horizon, but not yet fully applicable. Quantum computing and carbon nanotubes are two possible concepts for further performance improvement. Also, other materials than silicon will be applied for integrated circuits. Materials such as gallium arsenide (GaAs) or silicon-germanium (SiGe) have higher electron mobility than silicon and hence provide a major transport enhancement.

Chapter 4 described the implementation of a power management unit for adaptive hardware controlled power management of the routers. Also, first rerouting algorithms are given in section 4.4, but further enhancement of the routing algorithms is possible. Power aware communication can further decrease the power consumption of the network resources.

The network on chip discussed in this work as well as the FPGA prototype are part of the Transregional Collaborative Research Center *Invasive Computing* [7]. Both will continuously be used for further research. Also, the FPGA prototype forms a basis for software developer as well as it can be used for verification of further hardware integration.

List of Figures

List of Tables

I

Abbreviations

3D	three-dimensional
AHB	Advanced High-performance Bus
AMBA	Advanced Microcontroller Bus Architecture
ASIC	application-specific integrated circuit
AXI	Advanced eXtensible Interface
BE	best effort
BFM	Bus Functional Model
BIST	built-in self-test
CAPIM	Client Application Interface Module
CMOS	complementary metal-oxide-semiconductor
CMP	Chip Multi-Processor
DC	DesignCompiler
DIMM	Dual in-line memory module
DMA	direct memory access
DPI	Direct Programming Interface
DSU	Debug Support Unit

DUT	design under test
DVFS	dynamic voltage and frequency scaling
EDA	Electronic Design Automation
EVC	express virtual channel
FIFO	first in - first out
FinFET	Fin Field Effect Transistor
FPGA	field programmable gate array
FSM	finite state machine
GALS	Globally Asynchronous and Locally Synchronous
GS	guaranteed service
GUI	graphical user interface
HDL	hardware description language
HIM	Hardware Interface Module
IC	integrated circuit
ITRS	International Technology Roadmap for Semiconductors
JTAG	Joint Test Action Group
LDU	Load Detection Unit
LUT	look up table
MCP	Multi-Chip Package
MGT	Multi-Gigabit Transceiver

NA	network adapter
NAS	NASA Advanced Supercomputing
NoC	network on chip
OCP	Open Core Protocol
OS	operating system
PCB	printed circuit board
PE	processing element
PLI	Programmable Language Interface
PMC	power management controller
PMOS	p-channel metal-oxide semiconductor
QoS	quality of service
RB	ring bus
RTL	register-transfer level
SAIF	Switching Activity Information Format
SCC	Single-chip Cloud Computer
SCE-MI	Standard Co-Emulation Modeling Interface
SDLC	systems development life cycle
SiP	system in package
SL	service level
SLN	second layer network
SLR	super logic region

SoC	system on chip
TCPA	tightly-coupled processor array
TDM	time division multiplexing
TDP	thermal design power
TLV	threshold limit value
TMR	triple modular redundancy
TSMC	Taiwan Semiconductor Manufacturing Company
TSV	through-silicon vias
TWG	technology working group
UART	universal asynchronous receiver/transmitter
UMRbus	Universal Multi-Resource Bus
VC	virtual channel
VHDL	Very High Speed Integrated Circuit HDL
VLSI	very large scale integrated
VRC	voltage regulator controller

Bibliography

[1] Intel corporation. www.intel.com.

[2] proFPGA. www.prodesign-europe.coms.

[3] S2c inc. www.s2cinc.com.

[4] Synopsys, inc. www.synopsys.com.

[5] The Qt Company. www.qt.io.

[6] Gartner says worldwide semiconductor revenue declined 1.9 percent in 2015. www.gartner.com/newsroom/id/3182843, 2016.

[7] Invasive Computing (InvasIC). http//invasic.informatik.uni-erlangen.de/, 2016.

[8] A. B. Abdallah. *Multicore Systems On-Chip: Practical Software/Hardware Design*, volume 2. Springer, 2013.

[9] Aeroflex Gaisler. *GRMON2 User's Manual*, 2015.

[10] K. Agarwal, K. Nowka, H. Deogun, and D. Sylvester. Power gating with multiple sleep modes. In *Proceedings of the 7th International Symposium on Quality Electronic Design*, pages 633–637. IEEE Computer Society, 2006.

[11] G. M. Amdahl. Validity of the single processor approach to achieving large scale computing capabilities. In *Proceedings of the April 18-20, 1967, spring joint computer conference*, pages 483–485. ACM, 1967.

[12] W. Arden, M. Brillouët, P. Cogez, M. Graef, B. Huizing, and R. Mahnkopf. More-than-moore. White Paper, International Technology Roadmap for Semiconductors, 2010.

[13] D. H. Bailey, E. Barszcz, J. T. Barton, D. S. Browning, R. L. Carter, L. Dagum, R. A. Fatoohi, S. Fineberg, P. O. Frederickson, T. A. Lasinski, R. S. Schreiber, et al. The NAS parallel benchmarks. 1994.

[14] D. H. Bailey, E. Barszcz, J. T. Barton, D. S. Browning, R. L. Carter, L. Dagum, R. A. Fatoohi, P. O. Frederickson, T. A. Lasinski, R. S. Schreiber, et al. The NAS parallel benchmarks. *International Journal of High Performance Computing Applications*, 5(3):63–73, 1991.

[15] K. Banerjee, S. J. Souri, P. Kapur, and K. C. Saraswat. 3-d ics: A novel chip design for improving deep-submicrometer interconnect performance and systems-on-chip integration. *Proceedings of the IEEE*, 89(5):602–633, 2001.

[16] N. Banerjee, P. Vellanki, and K. S. Chatha. A power and performance model for network-on-chip architectures. In *Design, Automation and Test in Europe Conference and Exhibition, 2004. Proceedings*, volume 2, pages 1250–1255. IEEE, 2004.

[17] S. Bell, B. Edwards, J. Amann, R. Conlin, K. Joyce, V. Leung, J. MacKay, M. Reif, L. Bao, J. Brown, et al. Tile64-processor: A 64-core soc with mesh interconnect. In *Solid-State Circuits Conference, 2008. ISSCC 2008. Digest of Technical Papers. IEEE International*, pages 88–598. IEEE, 2008.

[18] L. Benini and G. De Micheli. Networks on chips: a new soc paradigm. *Computer*, 35(1):70–78, 2002.

[19] T. Bjerregaard and J. Sparso. A router architecture for connection-oriented service guarantees in the mango clockless network-on-chip. In *Design, Automation and Test in Europe*, pages 1226–1231. IEEE, 2005.

[20] P. Bogdan, R. Marculescu, S. Jain, and R. T. Gavila. An optimal control approach to power management for multi-voltage and frequency islands multiprocessor platforms under highly variable workloads. In *Networks on Chip (NoCS), 2012 Sixth IEEE/ACM International Symposium on*, pages 35–42. IEEE, 2012.

[21] H. Bokhari, H. Javaid, M. Shafique, J. Henkel, and S. Parameswaran. dark-noc: Designing energy-efficient network-on-chip with multi-vt cells for dark silicon. In *Design Automation Conference (DAC), 2014 51st ACM/EDAC/IEEE*, pages 1–6. IEEE, 2014.

[22] P. Bomel, K. Martin, and J.-P. Diguet. *Virtual UARTs for Reconfigurable Multiprocessor Architectures*. IPDPSW '13. IEEE Computer Society, Washington, DC, USA, 2013.

[23] K. M. Bresniker, S. Singhal, and R. S. Williams. Adapting to thrive in a new economy of memory abundance. *Computer*, 48(12):44–53, 2015.

[24] A. Canidio. A power gating methodology to aggressively reduce leakage power in networks-on-chip buffers. Master thesis, Politenico di Milano, 2015.

[25] X. Chen, Z. Xu, H. Kim, P. V. Gratz, J. Hu, M. Kishinevsky, U. Ogras, and R. Ayoub. Dynamic voltage and frequency scaling for shared resources in multicore processor designs. In *Proceedings of the 50th Annual Design Automation Conference*, page 114. ACM, 2013.

[26] P. Cichowski, J. Keller, and C. Kessler. Modelling power consumption of the intel scc. In *The 6th Many-core Applications Research Community (MARC) Symposium*, pages 46–51. ONERA, The French Aerospace Lab, 2012.

[27] I. R. Committee et al. International technology roadmap for semiconductors, 2011 edition. *Semiconductor Industry Association*, 2011.

[28] R. Das, S. Narayanasamy, S. K. Satpathy, and R. G. Dreslinski. Catnap: Energy proportional multiple network-on-chip. *SIGARCH Comput. Archit. News*, 41(3):320–331, June 2013.

[29] G. De Micheli and L. Benini. *Networks on chips: technology and tools*. Academic Press, 2006.

[30] V. De Paulo and C. Ababei. 3d network-on-chip architectures using homogeneous meshes and heterogeneous floorplans. *International Journal of Reconfigurable Computing*, 2010:1, 2010.

[31] S. B. Desai, S. R. Madhvapathy, A. B. Sachid, J. P. Llinas, Q. Wang, G. H. Ahn, G. Pitner, M. J. Kim, J. Bokor, C. Hu, et al. Mos2 transistors with 1-nanometer gate lengths. *Science*, 354(6308):99–102, 2016.

[32] J. Duato, S. Yalamanchili, and L. M. Ni. *Interconnection networks: an engineering approach*. Morgan Kaufmann, 2003.

[33] T. Dybå and T. Dingsøyr. Empirical studies of agile software development: A systematic review. *Information and software technology*, 50(9):833–859, 2008.

[34] A. Gaisler and S. Goeteborg. Leon3 multiprocessing cpu core. *Aeroflex Gaisler, February*, 2010.

[35] Gaisler, Aeroflex and Göteborg, Sweden. *Leon3 multiprocessing cpu core*, 2010.

[36] Y. Ge, P. Malani, and Q. Qiu. Distributed task migration for thermal management in many-core systems. In *Proceedings of the 47th Design Automation Conference*, pages 579–584. ACM, 2010.

[37] F. Gebali, H. Elmiligi, and M. W. El-Kharashi. *Networks-on-chips: theory and practice*. CRC press, 2011.

[38] K. Goossens, J. Dielissen, and A. Radulescu. Æthereal network on chip: concepts, architectures, and implementations. *IEEE Design & Test of Computers*, 22(5):414–421, 2005.

[39] M. Gries, U. Hoffmann, M. Konow, and M. Riepen. Scc: A flexible architecture for many-core platform research. *Computing in Science and Engineering*, 13(6):79–83, 2011.

[40] I. S. V. W. Group et al. Ieee standard for systemverilog c unified hardware design, specification, and verification (ieee std 1800–2005), 2005.

[41] P. Gschwandtner, T. Fahringer, and R. Prodan. Performance analysis and benchmarking of the intel scc. In *Cluster Computing (CLUSTER), 2011 IEEE International Conference on*, pages 139–149. IEEE, 2011.

[42] W. Guo, G. Van der Plas, A. Ivankovic, V. Cherman, G. Eneman, B. De Wachter, M. Togo, A. Redolfi, S. Kubicek, Y. Civale, et al. Impact of through silicon via induced mechanical stress on fully depleted bulk finfet technology. In *Electron Devices Meeting (IEDM), 2012 IEEE International*, pages 18–4. IEEE, 2012.

[43] J. L. Gustafson. Reevaluating amdahl's law. *Communications of the ACM*, 31(5):532–533, 1988.

[44] M. Harrand and Y. Durand. Network on chip with quality of service, Dec. 31 2013. US Patent 8,619,622.

[45] J. Heisswolf. *A Scalable and Adaptive Network on Chip for Many-Core Architectures*. PhD thesis, Karlsruhe Institute of Technology, 2014.

[46] J. Heisswolf, M. Singh, M. Kupper, R. Koenig, and J. Becker. Rerouting: Scalable noc self-optimization by distributed hardware-based connection reallocation. In *2013 International Conference on Reconfigurable Computing and FPGAs (ReConFig)*, pages 1–8, Dec 2013.

[47] J. Heisswolf, A. Zaib, A. Weichslgartner, M. Karle, M. Singh, T. Wild, J. Teich, A. Herkersdorf, and J. Becker. The invasive network on chip-a multi-objective many-core communication infrastructure. In *Architecture of Computing Systems (ARCS), 2014 27th International Conference on*, pages 1–8. VDE, 2014.

[48] J. Held, J. Bautista, and S. Koehl. From a few cores to many: A tera-scale computing research overview. White Paper, Intel®, 2006.

[49] J. Henkel, H. Bukhari, S. Garg, M. U. K. Khan, H. Khdr, F. Kriebel, U. Ogras, S. Parameswaran, and M. Shafique. Dark silicon: From computation to communication. In *Proceedings of the 9th International Symposium on Networks-on-Chip*, page 23. ACM, 2015.

[50] M. D. Hill and M. R. Marty. Amdahl's law in the multicore era. *Computer*, (7):33–38, 2008.

[51] J. Howard, S. Dighe, S. Vangal, G. Ruhl, N. Borkar, S. Jain, V. Erraguntla, M. Konow, M. Riepen, M. Gries, G. Droege, T. Lund-Larsen, S. Steibl, S. Borkar, V. De, and R. Van Der Wijngaart. A 48-core ia-32 processor in 45 nm cmos using on-die message-passing and dvfs for performance and power scaling. *Solid-State Circuits, IEEE Journal of*, 46(1):173–183, Jan 2011.

[52] J. Hussein, M. Klein, and M. Hart. Lowering power at 28 nm with xilinx 7 series fpgas. White Paper, Xilinx®, Feb. 2012.

[53] ITRS. International technology roadmap for semiconductors, 2013 edition. http://www.itrs2.net/2013-itrs.html, 2013.

[54] R. Jevtic and C. Carreras. Power measurement methodology for fpga devices. *IEEE Transactions on Instrumentation and Measurement*, 60(1):237–247, 2011.

[55] N. Jiang, J. Balfour, D. U. Becker, B. Towles, W. J. Dally, G. Michelogiannakis, and J. Kim. A detailed and flexible cycle-accurate network-on-chip simulator. In *Performance Analysis of Systems and Software (ISPASS), 2013 IEEE International Symposium on*, pages 86–96. IEEE, 2013.

[56] A. B. 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 conference on Design, Automation and Test in Europe*, pages 423–428. European Design and Automation Association, 2009.

[57] M. Kaisti, T. Mujunen, T. Mäkilä, V. Rantala, and T. Lehtonen. Agile principles in the embedded system development. In *International Conference on Agile Software Development*, pages 16–31. Springer, 2014.

[58] N. Karavadara, M. Zolda, V. T. N. Nguyen, J. Knoop, and R. Kirner. Dynamic power management for reactive stream processing on the scc tiled architecture. *EURASIP Journal on Embedded Systems*, 2016.

[59] E. Kasapaki, M. Schoeberl, R. B. Sørensen, C. Müller, K. Goossens, and J. Sparsø. Argo: A real-time network-on-chip architecture with an efficient gals implementation. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 24(2):479–492, 2016.

[60] G. Katti, M. Stucchi, K. De Meyer, and W. Dehaene. Electrical modeling and characterization of through silicon via for three-dimensional ics. *IEEE Transactions on Electron Devices*, 57(1):256–262, 2010.

[61] M. Keating, D. Flynn, R. Aitken, A. Gibbons, and K. Shi. *Low power methodology manual: for system-on-chip design*. Springer Publishing Company, Incorporated, 2007.

[62] J. Kim, C. Nicopoulos, D. Park, R. Das, Y. Xie, V. Narayanan, M. S. Yousif, and C. R. Das. A novel dimensionally-decomposed router for on-chip communication in 3d architectures. In *ACM SIGARCH Computer Architecture News*, volume 35, pages 138–149. ACM, 2007.

[63] I. Kuon and J. Rose. Measuring the gap between fpgas and asics. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 26(2):203–215, 2007.

[64] W. K. Lam. *Hardware Design Verification: Simulation and Formal Method-Based Approaches (Prentice Hall Modern Semiconductor Design Series)*. Prentice Hall PTR, 2005.

[65] J. H. Lau and T. G. Yue. Thermal management of 3d ic integration with tsv (through silicon via). In *2009 59th Electronic Components and Technology Conference*, pages 635–640. IEEE, 2009.

[66] K. Lee, S. joong Lee, S. eun Kim, H. mi Choi, D. Kim, S. Kim, M. wuk Lee, and H. jun Yoo. A 51mw 1.6ghz on-chip network for low-power heterogeneous soc platform. In *Solid-State Circuits Conference, 2004. Digest of Technical Papers. ISSCC. 2004 IEEE International*, 2004.

[67] C. Liu, L. Zhang, Y. Han, and X. Li. Vertical interconnects squeezing in symmetric 3d mesh network-on-chip. In *Proceedings of the 16th Asia and South Pacific Design Automation Conference*, pages 357–362. IEEE Press, 2011.

[68] H. Matsutani, M. Koibuchi, D. Ikebuchi, K. Usami, H. Nakamura, and H. Amano. Ultra fine-grained run-time power gating of on-chip routers for cmps. In *2010 Fourth ACM/IEEE International Symposium on Networks-on-Chip (NOCS)*, pages 61–68. IEEE, 2010.

[69] H. Matsutani, M. Koibuchi, D. Wang, and H. Amano. Run-time power gating of on-chip routers using look-ahead routing. In *Proceedings of the 2008 Asia and South Pacific Design Automation Conference*, pages 55–60. IEEE Computer Society Press, 2008.

[70] T. G. Mattson, M. Riepen, T. Lehnig, P. Brett, W. Haas, P. Kennedy, J. Howard, S. Vangal, N. Borkar, G. Ruhl, et al. The 48-core scc processor: the programmer's view. In *Proceedings of the 2010 ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 1–11. IEEE Computer Society, 2010.

[71] A. Mello, N. Calazans, and F. Moraes. Atlas-an environment for noc generation and evaluation.

[72] 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 *Design, Automation and Test in Europe Conference and Exhibition, 2004. Proceedings*, volume 2, pages 890–895. IEEE, 2004.

[73] A. Mirhosseini, M. Sadrosadati, A. Fakhrzadehgan, M. Modarressi, and H. Sarbazi-Azad. An energy-efficient virtual channel power-gating mechanism for on-chip networks. In *Proceedings of the 2015 Design, Automation & Test in Europe Conference & Exhibition*, 2015.

[74] G. E. Moore et al. Cramming more components onto integrated circuits. *Proceedings of the IEEE*, 86(1):82–85, 1998.

[75] B. Oechslein, J. Schedel, J. Kleinöder, L. Bauer, J. Henkel, D. Lohmann, and W. Schröder-Preikschat. Octopos: A parallel operating system for invasive computing. In *Proceedings of the International Workshop on Systems for Future Multi-Core Architectures (SFMA). EuroSys*, pages 9–14, 2011.

[76] U. Y. Ogras, R. Marculescu, D. Marculescu, and E. G. Jung. Design and management of voltage-frequency island partitioned networks-on-chip. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, 17(3):330–341, 2009.

[77] P. R. Panda, B. Silpa, A. Shrivastava, and K. Gummidipudi. *Power-efficient system design*. Springer Science & Business Media, 2010.

[78] P. P. Pande, C. Grecu, M. Jones, A. Ivanov, and R. Saleh. Performance evaluation and design trade-offs for network-on-chip interconnect architectures. *IEEE transactions on Computers*, 54(8):1025–1040, 2005.

[79] R. S. Patti. Three-dimensional integrated circuits and the future of system-on-chip designs. *Proceedings of the IEEE*, 94(6):1214–1224, 2006.

[80] J. Paul, W. Stechele, B. Oechslein, C. Erhardt, J. Schedel, D. Lohmann, W. Schröder-Preikschat, M. Kröhnert, T. Asfour, É. Sousa, et al. Resource-awareness on heterogeneous mpsocs for image processing. *Journal of Systems Architecture*, 61(10):668–680, 2015.

[81] V. F. Pavlidis and E. G. Friedma. 3-d topologies for networks-on-chip. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, 15(10):1081–1090, 2007.

[82] S. Penolazzi and A. Jantsch. A high level power model for the nostrum noc. In *9th EUROMICRO Conference on Digital System Design (DSD'06)*, pages 673–676. IEEE, 2006.

[83] E. Pop. Energy dissipation and transport in nanoscale devices. *Nano Research*, 3(3):147–169, 2010.

[84] T. Punkka. Agile hardware and co-design. In *Embedded Systems Conference*, 2012.

[85] P. Rashinkar, P. Paterson, and L. Singh. *System-on-a-chip verification: methodology and techniques*. Springer Science & Business Media, 2007.

[86] K. Roy and S. C. Prasad. *Low-power CMOS VLSI circuit design*. John Wiley & Sons, 2009.

[87] K. Saban. Xilinx stacked silicon interconnect technology delivers breakthrough fpga capacity, bandwidth, and power efficiency. White Paper, Xilinx®, Dec. 2012.

[88] M. Sadri, A. Bartolini, and L. Benini. Single-chip cloud computer thermal model. In *Thermal Investigations of ICs and Systems (THERMINIC), 2011 17th International Workshop on*, pages 1–6. IEEE, 2011.

[89] M. Said, F. Mehdipour, K. Murakami, and M. El-Sayed. A design methodology for performance maintenance of 3d network-on-chip with multiplexed through-silicon vias. In *Proceedings of the 3rd International Workshop on Many-core Embedded Systems*, pages 9–16. ACM, 2015.

[90] K. Schwaber. Scrum development process. In *Business Object Design and Implementation*, pages 117–134. Springer, 1997.

[91] C. Silvano, M. Lajolo, and G. Palermo. *Low power networks-on-chip*. Springer Science & Business Media, 2010.

[92] D. H. Soudris, editor. *Designing CMOS circuits for low power : [European low-power initiative for electronic system design]*. Kluwer, Boston [u.a.], 2002.

[93] C. Spear. *SystemVerilog for verification: a guide to learning the testbench language features*. Springer Science & Business Media, 2008.

[94] Synopsys, Inc. *Design Compiler User Guide*, 2012.

[95] Synopsys, Inc. *Power Compiler User Guide*, 2012.

[96] Synopsys, Inc. *Transactor Library for AMBA*, 2015.

[97] S. Tang and Q. Xu. A multi-core debug platform for noc-based systems. In *2007 Design, Automation & Test in Europe Conference & Exhibition*, pages 1–6. IEEE, 2007.

[98] E. Totoni, B. Behzad, S. Ghike, and J. Torrellas. Comparing the power and performance of intel's scc to state-of-the-art cpus and gpus. In *Performance Analysis of Systems and Software (ISPASS), 2012 IEEE International Symposium on*, pages 78–87. IEEE, 2012.

[99] V. D. Tovinakere, O. Sentieys, and S. Derrien. Wakeup time and wakeup energy estimation in power-gated logic clusters. In *VLSI Design (VLSI Design), 2011 24th International Conference on*, pages 340–345. IEEE, 2011.

[100] V. D. Tovinakere, O. Sentieys, and S. Derrien. A semiempirical model for wakeup time estimation in power-gated logic clusters. In *Proceedings of the 49th Annual Design Automation Conference*, pages 48–55. ACM, 2012.

[101] R. F. Van der Wijngaart, T. G. Mattson, and W. Haas. Light-weight communications on intel's single-chip cloud computer processor. *ACM SIGOPS Operating Systems Review*, 45(1):73–83, 2011.

[102] S. Vangal, J. Howard, G. Ruhl, S. Dighe, H. Wilson, J. Tschanz, D. Finan, P. Iyer, A. Singh, T. Jacob, et al. An 80-tile 1.28 tflops network-on-chip in 65nm cmos. In *IEEE International Solid-State Circuits Conference, San Fransisco, USA, 2007*, pages 98–99. IEEE, 2007.

[103] P. Vivet, C. Bernard, E. Guthmuller, I. Miro-Panades, Y. Thonnart, and F. Clermidy. Interconnect challenges for 3d multi-cores: from 3d network-on-chip to cache interconnects. In *2015 IEEE Computer Society Annual Symposium on VLSI*, pages 615–620. IEEE, 2015.

[104] P. Vivet, D. Lattard, F. Clermidy, E. Beigne, C. Bernard, Y. Durand, J. Durupt, and D. Varreau. Faust, an asynchronous network-on-chip based architecture for telecom applications. *Proc. 2007 Design, Automation and Test in Europe (DATE07)*, 2007.

[105] M. Waldrop. More than moore. *Nature*, 530:144–147, 2016.

[106] H.-S. Wang, X. Zhu, L.-S. Peh, and S. Malik. Orion: a power-performance simulator for interconnection networks. In *Microarchitecture, 2002.(MICRO-35). Proceedings. 35th Annual IEEE/ACM International Symposium on*, pages 294–305. IEEE, 2002.

[107] J. Wang, H. Gu, and Y. Yang. Cluster mesh: a topology for three-dimensional network-on-chip. *IEICE Electronics Express*, 9(15):1254–1259, 2012.

[108] R. Weerasekera, M. Grange, D. Pamunuwa, H. Tenhunen, and L.-R. Zheng. Compact modelling of through-silicon vias (tsvs) in three-dimensional (3-d) integrated circuits. In *3DIC*, pages 1–8, 2009.

[109] D. Wentzlaff, P. Griffin, H. Hoffmann, L. Bao, B. Edwards, C. Ramey, M. Mattina, C.-C. Miao, J. F. Brown III, and A. Agarwal. On-chip interconnection architecture of the tile processor. *IEEE micro*, (5):15–31, 2007.

[110] J. Wiedmann. Intelligentes energie-optimierendes virtual channel management. Studienarbeit, Karlsruhe Institute of Technology, 2014.

[111] Xilinx®. *Virtex-5 FPGA User Guide*, 2012.

[112] Xilinx®. *7 Series FPGAs Overview*, 2015.

[113] T. C. Xu, P. Liljeberg, and H. Tenhunen. Optimal number and placement of through silicon vias in 3d network-on-chip. In *Design and Diagnostics of Electronic Circuits & Systems (DDECS), 2011 IEEE 14th International Symposium on*, pages 105–110. IEEE, 2011.

[114] A. Zaib, J. Heißwolf, A. Weichslgartner, T. Wild, J. Teich, J. Becker, and A. Herkersdorf. Network interface with task spawning support for noc-based dsm architectures. In *International Conference on Architecture of Computing Systems*, pages 186–198. Springer, 2015.

[115] Q. Zhang, M. Zhou, J. Chen, and H. Yang. A homogeneous many-core x86 processor full system framework based on noc. In *2015 4th International Conference on Computer Science and Network Technology (ICCSNT)*, volume 01, pages 794–797, Dec 2015.

Supervised Student Research

[Arl15] Alexander Arlt. Power-gating mechanismen for network-on-chips. Diplomarbeit, Karlsruhe Institute of Technology, 2015.

[Dis13] Christian Dischke. Partitionierung eines multi-fpga systems. Diplomarbeit, Karlsruhe Institute of Technology, 2013.

[Erg14] Oguz Ergin. Tool zur automatischen erstellung einer invasic architektur. Bachelor thesis, Karlsruhe Institute of Technology, 2014.

[Koo13] Mohamed Koobar. Novel multicore architecure: Vergleich von simulation und co-simulation. Bachelor thesis, Karlsruhe Institute of Technology, 2013.

[Kri15] Michael Krimm. Evaluation des energieverbrauchs eines network-on-chips basierend auf verschiedenen inter-tile-kommunikationsszenarien. Seminararbeit, Karlsruhe Institute of Technology, 2015.

[Leh15] Niclas Lehmann. 3d network on chip architektur design. Bachelor thesis, Karlsruhe Institute of Technology, 2015.

[Mat12] Lazar Mateev. Prototyping und test einer invasic many-core-architektur. Bachelor thesis, Karlsruhe Institute of Technology, 11 2012.

[Mec13] Michael Mechler. Design und implementierung von debugmethoden fuer den v-fpga. Bachelor thesis, Karlsruhe Institute of Technology, 03 2013.

[Neb15] Marco Neber. Integration einer steuerungseinheit fuer power-gating arbitrierung eines network-on-chips. Master thesis, Karlsruhe Institute of Technology, 2015.

[Ric13] Tobias Rickelhoff. Entwicklung und test einer debug-schnittstelle fuer die invasic-architektur. Diplomarbeit, Karlsruhe Institute of Technology, 2013.

[Spi15] Marko Spiric. Design und implementierung eines network-on-chip mit heterogener bandbreite. Master thesis, Karlsruhe Institute of Technology, 2015.

[Sti16] Bernhard Stiehle. Neues echtzeitfähiges hardware-in-the-loop system auf basis eines system-on-chip. Master thesis, Karlsruhe Institute of Technology, 2016.

[Wie14] Julian Wiedmann. Dynamische expressverbindungen fuer network on chips. Diplomarbeit, Karlsruhe Institute of Technology, 2014.

The supervised student research work can be inspected at the *Karlsruhe Institute of Technology (KIT), Itstitute of Information Processing Technologies (Institut für Technik der Informationsverarbeitung - ITIV), Engesserstrasse 5, 76131 Karlsruhe, Germany* and are available in electronic form as well.

Publications

Conference & Journalpapers

[BFH+12] Jürgen Becker, Stephanie Friederich, Jan Heisswolf, Ralf Koenig, and David May. Hardware prototyping of novel invasive multicore architectures. In *Design Automation Conference (ASP-DAC), 2012 17th Asia and South Pacific*, pages 201–206. IEEE, 2012.

[FHB14] Stephanie Friederich, Jan Heisswolf, and Jürgen Becker. Hardware/-software debugging of large scale many-core architectures. In *Proceedings of the 27th Symposium on Integrated Circuits and Systems Design (SBCCI)*, pages 1–7. ACM, September 2014.

[FHMB14] Stephanie Friederich, Jan Heisswolf, David May, and Jürgen Becker. Hardware prototyping and software debugging of multi-core architectures. In *Synopsys Users Group Conference*. Synopsys, May 2014.

[FLB16] Stephanie Friederich, Niclas Lehmann, and Jürgen Becker. Adaptive bandwidth router for 3d network-on-chips. In *Applied Reconfigurable Computing - 12th International Symposium, ARC 2016, Mangaratiba, RJ, Brazil, March 22-24, 2016, Proceedings*, pages 352–360, 2016.

[FNB16] Stephanie Friederich, Marco Neber, and Jürgen Becker. Power management controller for online power saving in network-on-chips. In *Embedded Multicore Socs (MCSoC), 2016 IEEE 10th International Symposium on*, 2016.

[HFM+16] Jan Heisswolf, Stephanie Friederich, Leonard Masing, Andreas Weichslgartner, Aurang Zaib, Carsten Stein, Marco Duden, Jurgen Teich, Andreas Herkersdorf, and Jürgen Becker. A novel noc-architecture for fault tolerance and power saving. In *In Proceedings of the second International Workshop on Multi-Objective Many-Core Design (MOMAC) in conjunction with International Conference on Architecture of Computing Systems (ARCS)*, 2016.

[HWZ+15] Jan Heisswolf, Andreas Weichslgartner, Aurang Zaib, Stephanie Friederich, Leonard Masing, Carsten Stein, Marco Duden, Roman Klopfer, Jurgen Teich, Thomas Wild, Andreas Herkersdorf, and Jürgen Becker. Fault-tolerant communication in invasive networks on chip. In *Adaptive Hardware and Systems (AHS), 2015 NASA/ESA Conference on*, pages 1–8. IEEE, 2015.

[LWT+16] Vahid Lari, Andreas Weichslgartner, Alexandru Tanase, Michael Witterauf, Faramarz Khosravi, Jürgen Teich, Jan Heisswolf, Stephanie Friederich, and Jürgen Becker. Providing fault tolerance through invasive computing. In *IT Journal*, 2016.