SOFTWARE-BASED AND REGIONALLY-ORIENTED TRAFFIC MANAGEMENT IN NETWORKS-ON-CHIP

Thesis submitted in partial fulfillment of the requirements for the degree

of Doktor-Ingenieur (Dr.-Ing.) at the

Faculty of Computer Science and Electrical Engineering

University of Rostock, Germany

Submitted by

Philipp Gorski, born on October 8, 1981 Rostock, Germany

Rostock, Mai 30-th 2016

urn:nbn:de:gbv:28-diss2016-0138-4

Place and Date of Doctoral Defence and Disputation:

- Institute of Applied Microelectronics and Computer Engineering
  Richard Wagner Straße 31
  18119 Rostock
  Germany

- November 17-th 2016

Thesis Reviewer:

- **Prof. Dr.-Ing. Dirk Timmermann** (doctoral supervisor)
  Institute of Applied Microelectronics and Computer Engineering
  Faculty of Computer Science and Electrical Engineering
  University of Rostock, Germany

- **Prof. Dr.-Ing. habil. Christian Haubelt**
  Institute of Applied Microelectronics and Computer Engineering
  Faculty of Computer Science and Electrical Engineering
  University of Rostock, Germany

- **Prof. Dr.-Ing. Rainer Kokozinski**
  Elektronische Bauelemente und Schaltungen
  Abteilung Elektro- und Informationstechnik
  University of Duisburg-Essen, Germany

ACKNOWLEDGMENTS

Diese Arbeit entstand im Rahmen meiner Tätigkeit als wissenschaftlicher Mitarbeiter am Institut für Angewandte Mikroelektronik und Datentechnik (IMD) unter der Betreuung von Prof. Dr.-Ing. Dirk Timmermann. Zurückblickend richtet sich mein Dank vor allem an ihn für sein Vertrauen und die aufgebrachte Geduld in meine Person sowie mein Wirken. Nach diesen sechs Jahren am IMD, mit den inhaltlichen Schwerpunkten von Forschung, Lehre und Projektakquise, habe ich das Gefühl eine ganzheitliche Ausbildung genossen zu haben und kann auf einen vielseitigen Erfahrungsschatz aus dieser Zeit zurückblicken. Neben der Entwicklung eigener Ideen durfte ich viele Studenten auf ihrem beruflichen Werdegang begleiten sowie ihnen betreuend zur Seite stehen.

Zudem darf ich auf wertvolle Kooperationen mit Kollegen verweisen, welche im Rahmen meiner Forschung entstanden sind und deren Inhalte direkten sowie indirekten Einfluss auf die vorliegende Dissertationsschrift hatten. Dafür möchte ich im Speziellen Peter Danielis, Frank Golatowski, Claas Cornelius, Hendrik Bohn, Martin Gag, Tim Wegner, Hagen Saemrow, und Andreas Tockhorn meinen Dank aussprechen. Dabei habe ich nicht nur immer kompetenten Rat, sondern gleichzeitig liebenswerte Freundschaften schließen können. Allen nicht namentlich erwähnten Mitgliedern des IMD gilt ebenfalls mein Dank für die schöne Zeit und das gute Miteinander über diese sechs Jahre hinweg.

Abschließend möchte ich mich für den uneingeschränkten Rückhalt, die Liebe und die Unterstützung durch meine Familie und Freunde bedanken, ohne die diese Arbeit wohl nie zustande gekommen wäre. Ganz besonders mein kleines Hildchen, welche mir Ansporn sowie willkommene Ablenkung zugleich war :D

Rostock den 01.05.2016 Philipp Gorski

ABBREVIATIONS

2D-Mesh	Two-dimensional Mesh topology
ASIC	Application Specific Integrated Circuit
AU	Aggregation Unit
BE	Basic Event
BLP	Bit-Level Parallelism
BP	Bit Permutation
BUF	Buffer
CDF	Communication Distribution Function
CI	Core Interface
CLP	Core-Level Parallelism
CMOS	Complementary Metal Oxide Semiconductor
CMP	Chip Multiprocessor
CPI	Cycles Per Instruction
CX	Crossbar
DC-FIFO	Dual-Clock FIFO
DDR-SDRAM	Double Data Rate Synchronous Dynamic Random-Access Memory
DPDF	Discrete Probability Distribution Function
DVFS	Dynamic Voltage Frequency Scaling
EDAC	Error Detection and Correction
ETE	End-to-End
FCFS	First Come First Served
FF	Flip flop
FIFO	First In First Out
Flit	Flow Control Unit
GALS	Globally Asynchronous Locally Synchronous
HCI	Hot Carrier Injection
HL	Handshake Logic
HTH	Hop-to-Hop
ILP	Instruction-Level Parallelism
IP core	Intellectual Property core
IPC	Instructions per Cycle
ISA	Instruction Set Architecture
ITRS	International Technology Roadmap for Semiconductors
LL	Long Link
LLC	Lower Left Corner
LT	Link Traversal
LVT	Low Threshold Voltage
MM	Monitoring Memory
MOSFET	Metal Oxide Semiconductor Field Effect Transistor

MPSoC	Multiprocessor System-on-Chip
MPU	Microprocessor Unit
MTTF	Mean Time to Failure
MuGFET	Multiple Gate Field Effect Transistor
MVT	Mixed Threshold Voltage
NI	Network Interface
NN	Nearest Neighbor
NoC	Network-on-Chip
PDF	Probability Distribution Function
PE	Processing Element
Phit	Physical Unit
PO	Path Occupation
PTP	Point-to-Point
PVT	Process-Voltage-Temperature
QMesh	Quadrant-based Mesh topology
QoS	Quality of Service
RAM	Random-Access Memory
ROI	Region-of-Interest
ROM	Read-Only Memory
SA	Switch Arbitration
SAF	Store and Forward
SL	Short Link
SMT	Simultaneous Multithreading
SoC	System-on-Chip
SOI	Silicon on Insulator
ST	Switch Traversal
TLP	Thread-Level Parallelism
TPM	Temperature Monitoring
TRM	Traffic Monitoring
TSA	Traffic Sensor Array
URC	Upper Right Corner
VC	Virtual Channel
VCA	Virtual Channel Arbitration
VCT	Virtual Cut Through
VFI	Voltage and Frequency Island
WHS	Wormhole Switching

FIGURES

TABLES

EQUATIONS

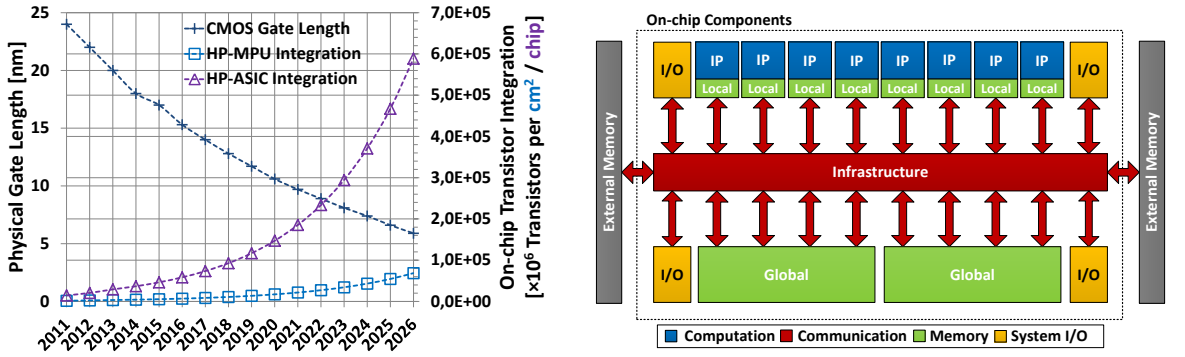
TABLE OF CONTENTS

1 INTRODUCTION

Over the last four decades, computer architects were able to exploit the technological advancements of Complementary Metal Oxide Semiconductors (CMOS) to realize complex integrated microprocessor systems on a single silicon die that contain billions of transistor units [1]. Each new system generation came along with more computational performance at shrinking or constant unit prices, which boosted the distribution of such systems over all areas of life and the overall technological progress of mankind up to date. This development is also known as Moore's Law [1] and illustrated in the chart of Figure 1.1 (a), which contrasts the shrinking gate length of CMOS transistors to the exponential growing integration density of the million transistors per cm²/chip per Microprocessor Unit (MPU)/Application Specific Integrated Circuit (ASIC) over the years 2011 up to 2026. The data is extracted from the projections of the 2012/2013 International Technology Roadmap for Semiconductors (ITRS)[2]. Historically, the performance improvements started at the level of a single MPU, named as intellectual property (IP) core or computational resource in the work at hand, whose computational capabilities were boosted by increasing the number of executable instructions per cycle (IPC) as well as the operational frequency [1]. Thereby, the IPC count was increased by the utilization of a wide spectrum of design- as well as run-time techniques that envisaged higher degrees of Instruction-Level Parallelism (ILP) exploitation per MPU. This was realized by the integration of pipelining, speculative execution and branch prediction, superscalar and out-of-order architectures, and the utilization of on-chip memory (caches, scratchpad) for lower access latencies [1]. But in consequence, this development led to diminishing returns for the ILP performance improvements, because the complexity of the MPU architectures increased, larger footprints boosted the hardware/power costs and the single thread performance does not scale with the number of invested transistors. This is also known as Pollack's law, which states that the performance of a single MPU is proportional to the square-root of its area footprint ($Performance \propto \sqrt{Area}$) [1], [3], [4]. Consequently, the architectures were evolved to exploit Thread- and Core-Level Parallelism (TLP and CLP) [1], [3], [4] to achieve higher performance gains per invested transistor at reasonable power levels. This development resulted in today's Chip Multiprocessor (CMP) or Many-Core architectures that integrate multiple MPUs, dedicated hardware accelerators as well as peripheral input/output controller, private/local and shared/global memory resources on a single die to provide massive parallelism. Figure 1.1 (b) depicts a principle schematic setup of those modern CMP architectures.

But as the number of cores increases, the on-chip communication infrastructure becomes a dominant factor for the performance of parallel and mixed workloads, because it has to provide the required data bandwidth and latency at feasible design- as well as run-time costs [1], [4]–[11]. Thus,

the communication infrastructure becomes a vital and active part for the future of optimized operations of workloads on CMPs as well as many-core systems [1], [4]–[11]. Thereby, especially the Network-on-Chip (NoC) represents the latest development of advanced infrastructure solutions and its optimization represents the main context of the work at hand [1], [3], [4], [6]–[31]. Therein, the focus of the investigations is the holistic integration of flexible traffic management mechanisms into the orthogonal arrangement of hardware-software layers in modern CMP systems, with respect to the challenges of future CMOS technology generations.



(a) CMOS Technology Trend (b) Chip Multiprocessor Schematic

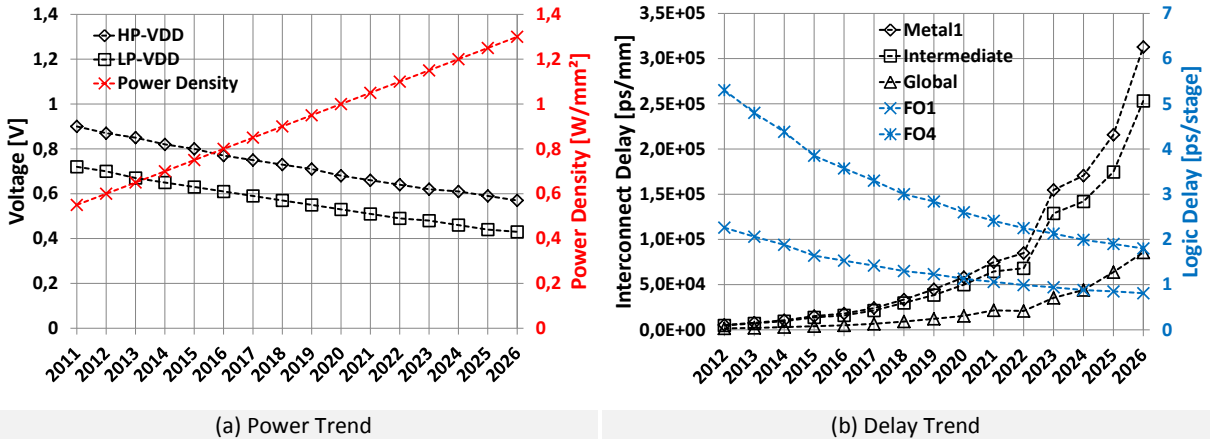
Figure 1.1: CMOS technology trend according to ITRS and common chip multiprocessor system schematic

1.1 MOTIVATION AND OBJECTIVES

As the CMOS technology is scaled down and the number of cores per CMP increases for each technology generation, new CMP architectures have to face a composition of challenges and requirements at design- as well as run-time [27].

PHYSICAL IMPERFECTION: Since 2005 the constant electrical field scaling of CMOS [32], also called Dennard Scaling [33] and the key to enable the trend specified by Moore's Law, could not be maintained anymore, because the technology approached physical limitations that prohibit the linear downscaling of the supply voltage (V_{DD}) by one half each new CMOS generation as it would be required [1], [32]–[36]. The resulting phenomenon is called the power wall and one of the most critical aspects of modern integrated circuits [1], [4], [33], [34]. With shrinking technology scales, the number of transistors per area unit grows quadratically and the frequency scales lineary. The potential net performance increase has a qubic nature. But the difference of the Post-Dennard [33] scaling is that this will not be possible if the power budget must remain constant, because the supply voltage scaling is nearly flat and the on-chip power scales quatdratically as well. Hence, each new generation of integrated circuits must operate with reduced utilization of its hardware to meet its power constraints. Otherwise, the interplay of on-chip power and temperature would result in destruction due to thermal effects [37], which is also known as temperature wall.

Additionally, the delay of integrated circuits becomes dominated by the signal traversals along interconnection wires and this trend will further strengthen the overall impact of data traversals along the infrastructure [11]. According to the technology projections of the ITRS in Figure 1.1 (a) above, the diagrams in Figure 1.2 (a) and (b) illustrate the discussed power and delay trends. Thereby, Figure 1.2 (a) shows the supply voltage trend at high performance (HP) and low power (LP) CMOS technology design styles. The depicted power density is named as "average maximum power density" by the ITRS and shows that they expect at least a linear trend under consideration of advanced power management techniques [2]. Figure 1.2 (b) contrasts the shrinking delay of simple oscillator stages with different driver strengths (FO1 and FO4) to the exponential increasing wire delay per length unit at different on-chip metallization layers (Metal1, Intermediate, Global).



(a) Power Trend (b) Delay Trend

Figure 1.2: Power and delay trends of ASIC/CMOS according to ITRS

While ongoing technology shrinking is possible at the costs of increased power budgets or intensified run-time power management, external connectivity characteristics of CMP systems are worsening due to pin/pad technology limitations [1], [4], [5], [11], [33], [34], [38]. Technology advancements in packaging are not able to keep step with those in CMOS. Hence, future systems will face the problems to move the demanded amounts of data as well as power on chip because of external connectivity limits. This bandwidth wall enforces the designers to hold data in place by increasing the on-chip memory capacities [11].

Finally, the combination of smaller structures, increased thermal stress, and signal-to-noise ratios represents a significant reliability issue at run- and design-time, because failure rates will increase and wear-out is accelerated [1], [4], [36], [39]–[43]. The resulting Process-Voltage-Temperature (PVT) variations affect the reliable CMP operations at the short- and long-term as well as they consume valuable on-chip resources to provide workarounds.

In summary, the consequences of the physical imperfection can be formulated as overall utilization wall that is built up from power, temperature, bandwidth, or reliability limitations, while the

dominant factor varies over the different use-cases of ASICs and CMPs. Hence, it is indispensable for future systems to provide active management services that deliver the requested workload performance while the resource utilization is dimmed to minimum required levels [1], [33], [34], [38].

DIMINISHING RETURNS OF PARALLELIZATION: As the number of available cores in CMP systems increases by each technology generation [27], it becomes obvious that higher degrees of parallelization for singular applications through TLP/CLP are limited by several dependencies [5], [11], [12], [44]. Power, temperature, and bandwidth wall as well as spatio-temporal data dependencies reduce the increases of speedups at certain parallelization granularities [11], [33], [34], [38]. The traditional parallelization following Amdahl's Law [45] or Gustafson's Law [46] is no longer feasible, because thread interference, synchronization, and memory coherence as well as concurrency dependencies lead to diminishing returns in achievable speedups as the parallelization granularity decreases [1], [3], [5], [11], [44], [45], [47]–[52]. Thus, every application scenario has a specific sweet spot of parallelization on a CMP that strongly correlates with its inner thread/data dependencies and the system setup. Mixed application workloads and spatio-temporal coordinated virtual instances on a singular CMP become more likely for the future [1], [4], [19], [53], [54]. Furthermore, the increasing number of resources raises the required efforts for the underlying system, application and traffic management at design- as well as run-time [1], [17], [22], [55], [56]. Thereby, the dynamic power/thermal management inserts additional dependencies and increases the complexity of the overall CMP handling to find optimized run-time configurations for the current workload conditions [33], [34], [38], [57]–[59].

THESIS OBJECTIVES: The interplay of utilization wall, growing system sizes, and increased workload variability requires a broad range of measures at all system layers to support holistic solutions. Furthermore, the corresponding mechanisms must be integrated at design-time and applicable at run-time to adjust optimized operating conditions for current workloads. Hence, dedicated profiles for the run-time behavior of applications should be known or measurable and the CMP must be enabled to act self-aware during operation to adapt sensitively to its current loads. **From the perspective of the work at hand, this desired self-awareness and –configuration of CMPs must be targeted via the systems communication infrastructure as the central instance of investigations**. All data moved for computational purposes have to traverse it. Thus, its traffic load represents a midterm indicator for computational activity and power consumption as well as resulting temperature distributions. The infrastructure already belongs to the most dominant performance factors of CMPs and efficient on-chip communication is one of the major challenges. This development becomes boosted by the increased on-chip memory resources due to the bandwidth wall, because it comes along with higher data locality and thus more communication on-chip.

Thereby, the traffic composition and load management at run-time are the most powerful options to encounter performance bottlenecks resulting from traffic interferences and to provide workarounds in the presence of failures.

To realize self-awareness and –configurability for Networks-on-Chip, the latest generation of on-chip infrastructures, the work at hand considers regularity, simplicity, observability, generality, and flexibility as the most valuable properties. **Thereby, the initial step towards advanced traffic handling is the review of the communication infrastructure's architecture and the parallel connectivity it provides to the computational resources. The predestinated entry-point for these investigations is the composition of network topology and routing policy.** The topology outlines the potential for parallel communication, because it constraints the availability of independent paths between the resources, while the routing policy defines the realizable exploitation of the potential path diversity and the determinism of traffic flows. Under consideration of the growing physical issues, regularity and simplicity are the most valuable design properties for this step. High layout regularity targets the discussed wire delay issues as well as aspects of the power wall and is a major requirement for scalable solutions as the CMP size grows further. Simplicity is the key to feasible costs and orthogonal inter-layer compatibility. In NoCs, most of the functional features are implemented in hardware and simple routing algorithms impose minor stress to the hardware costs. Furthermore, such designs support the integration of fine-grained power islands in CMP systems. Additionally, the intended adaptation mechanisms should provide enough flexibility to cover different use-cases that differ in their spatial and temporal scope. Traditional run-time traffic management often is realized as closed-loop mechanism with sophisticated use-case [60]–[64].

After the fundamental NoC architecture is set, the integration of observability is the next challenge to bring the run-time traffic management to life. Therefore, the availability of the observed traffic state information must be handled as a global concern to enable the interplay of various management service instances at different system layers [17], [19], [22], [59], [65], [66] (i.e. application mapping/scheduling, voltage/frequency level adjustments, profiling, path adaptations). **But instead of specialization of partially redundant monitoring solutions inside the NoC for dedicated services, the work at hand investigates the reusability of the gathered traffic state information as well as the needed infrastructure extensions for its collection.** Thereby, the provided quality of traffic state information must fulfill the requirements of each considered management service instance and the general availability must be ensured [17], [22]. As the IP count on the CMP grows and workload fractions are mapped/combined within regional scopes on it [11], [17], [22], the communication pattern become more and more localized as well [11], [14], [67] and spatially oriented adaptivity of observations help to tradeoff the efforts against the benefits by using it in

regions where the traffic loads occur. Furthermore, the impact of run-time path adaptations correlates to the intensity of traffic interferences, which makes high-frequency traffic monitoring obsolete if the traffic loads are too low. Thus, the temporal scope of observations must be adjustable inside spatial scopes on-demand to avoid unnecessary work.

For the flexible and holistic system-level integration of the traffic management, a hardware-centric solution is not suitable due to the targeted adaptivity to spatio-temporal workload dynamics and the static characteristics of hardware-only approaches. However, at least sensing and acting devices must be integrated in hardware to gather the traffic statistics and adjust the paths. The integration of the evaluation logic should be realized via software components to have full flexibility gains and provide more degrees of freedom. Software can be updated and thus the policies and algorithms for traffic management become adaptable in each supervised region for each supervised workload case. As other run-time services (such as the application mapping/scheduling) are typically integrated as part of the operating system, the holistic system-level integration benefits from a software-based approach for the traffic management. Furthermore, modular designed software is migratable and avoids the single point of failure as it exists in hardware-centric approaches.

The big question for the combined hardware/software integration scheme is the performance aspect of the software modules. But this must be shown by its application to practically relevant scenarios.

1.2 OWN CONTRIBUTIONS

The work at hand provides solutions for the objectives outlined in the prior section regarding the required infrastructure improvements and the spatio-temporal traffic management under consideration of the intended realization as hardware-software integration [68]–[70]. The resulting basic framework structure is illustrated in the diagram of Figure 1.3 below.

Thereby, the first major contribution is the introduction of a mesh-based NoC infrastructure [71] that provides increased connectivity for the computational resources via spatially independent network interfaces, while it integrates the capabilities for run-time traffic management by programmable path tables that work as the required actors. Furthermore, it can be shown that this multi-ported topology, called quadrant-based Mesh (QMesh), as stand-alone solution offers combined improvements regarding performance, communication locality, and reliability, while the cost overhead is moderate. As Figure 1.3 shows, the QMesh infrastructure represents the hardware baseline for the proposed management framework.

In the second step, the observability objective is addressed with the introduction of a adaptive traffic monitoring for mesh-based NoCs [68], [72], [73]. Thereby, the left part of Figure 1.3 regarding the

required system-monitoring is focussed. The provided solution is a combined hardware/software-design that is configurable regarding its spatial scope, temporal scope, and its coverage of the basic NoC components. It applies clustering to observe desired regions-of-interest (ROIs) with individual parameterization and collects the gathered monitoring data over a centrally organized flow on a single entity per cluster. The sensing and data aggregation is performed in hardware due to performance benefits, while the monitoring data evaluation is performed in software to provide maximum adaptivity regarding the implemented algorithms and policies. The presented work addresses the required hardware modules as well as the software parts. The monitoring is designed with the inherent features of reusability and flexibility to support various use case scenarios.

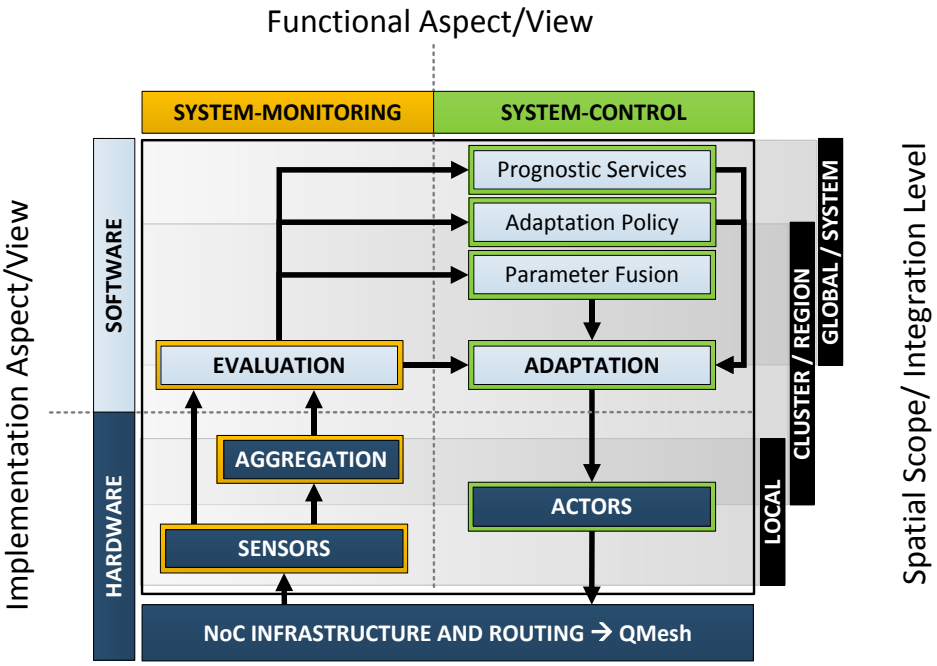


Figure 1.3: Leading structural framework for the spatio-temporal traffic management of the work at hand

Finally, a solution for the last objective is provided that applies a software-directed and spatially restricted run-time traffic management [69], [70], [72] inside the proposed QMesh [71]. Thereby, the newly introduced traffic monitoring is utilized by this traffic management to perform centralized path adaptations inside every monitored cluster/region. Hence, it represents a concrete application scenario of the framework outlined in Figure 1.3.

1.3 THESIS ORGANIZATION

The research results presented in the work at hand required the initial examination of the fundamental background regarding NoCs as well as CMPs and the exploration of related work to argue in the context of the state-of-the-art of modern NoC-based systems. Thereby, the fundamental background on NoCs and CMPs is presented as compact standalone chapter, while the related work on the specific research topics is presented in dedicated sections of the relevant chapters.

CHAPTER 1 introduces the context of the work at hand, outlines the motivation and formulates thesis objectives. Afterwards, the major contributions and the organization of thesis are discussed.

CHAPTER 2 provides the required background and related work of the work at hand regarding NoCs.

CHAPTER 3 introduces the Quadrant-based Mesh (QMesh) topology, which is the first major contribution of the work at hand according to the above formulated objectives.

CHAPTER 4 introduces the proposed solution for regional adaptive run-time traffic monitoring, which represents the second major contribution of the work at hand.

CHAPTER 5 investigates the final combination of the traffic monitoring solution proposed in CHAPTER 4 with the Quadrant-based Mesh of CHAPTER 3. Thereby, the formulated hardware/software-based management concept is applied as final solution for the targeted spatial-restricted traffic management in NoCs.

CHAPTER 6 finalizes the work at hand with a conclusive summary.

APPENDIX provides the major background information and data for:

- The modeling and simulation flow for the experimental evaluation.
- The evaluation of basic NoC component for CMOS technologies from 45nm down to 22nm.
- The parameter settings for the performed simulations are aggregated in specific tables.
- The evaluation methodology for traffic statistics to aggregate the experimental results.

2 NETWORK-ON-CHIP - BACKGROUND AND FUNDAMENTALS

This chapter provides the required background for the investigated NoC infrastructures. Furthermore, discussions of physical characteristics at the component level and major design-/run-time aspects at different system abstraction levels are presented as preliminary step for the choices made regarding the proposed solutions in the following chapters. In addition to the referenced works, the work at hand provides an evaluation of the design characteristics for the fundamental Network-on-Chip components as a consistent technological baseline for the argumentation of the impact of design decisions (see Appendix pp. 167 to 175).

The evolution of on-chip communication infrastructures is mainly driven by the technology scaling and its challenges for the efficient design and operation of complex CMPs. Thereby, research and development on those on-chip architectures have to consider a wide range of parameter dependencies and interrelations regarding performance, reliability, costs, design productivity, and scalability. With each new technology generation, the number of integrable functionality per area nearly doubles and on-chip architectures have to provide optimized design tradeoffs to make this potential of computational growth fully utilizable [1], [2], [8], [33], [35]. Historically, the complexity of interconnects and infrastructures kept up with increasing exploitation of parallelism and the growing data traffic that comes along with it [1]. Thereby, networked on-chip communication represents the last stage of ongoing research and can be seen as successor of bus-based shared medium solutions [1], [6]–[10], [25], [27], [28], [30], [74], [75]. They are used in modern CMP architectures and typically apply a ***tile-based organization*** concept. Inside each tile, the functionality is implemented as well defined portion via ***intellectual property*** (IP) cores and the data transfers are realized over ***network interfaces*** (NI) that integrates ***standardized protocols***. This tile-based encapsulation results in highly modular and reusable designs with reduced complexity and increased layout regularity [8]. Furthermore, the progressing exploitation of TLP, CLP, and DLP benefits from this modularity, which represents an essential basis for an increasing number of tiles in next generation CMPs that integrate hundreds of IPs [1], [33], [34]. But such scaled up systems must be equipped with sufficient communication capacities, otherwise inter-tile data transfers become the systems bottleneck [5]. This requirement and increasing parallelism [34], [76]–[78] in modern workloads boosted research to look for infrastructure solutions beyond shared medium communication. While a bus-based infrastructure represents an efficient solution for small sized CMPs, it cannot be scaled up to hundreds of tiles with massively parallel data transfers, because the implementation cost for power and area would exceed affordable limits to provide the required bandwidth [1], [8]. The reason results from the worsening timing characteristics of interconnections with progressing technology scaling [2] (see also Figure 1.2 (b) at p. 19). Thus, long range wiring has to be avoided and this is one

advantage of the NoC approach. Such NoC infrastructures split the global bus up into short bidirectional segments, called ***links***, by the insertion of ***router/switches***, which manage and allocate these links for passing inter-tile data flows. Furthermore, a ***packet-based communication*** is applied to enable partial link-by-link resource allocation and flow control along the inter-tile paths through the network. This avoids the blocking as well as multiplexing of the complete interconnection medium for single data flows and allows the coexistence of massively parallel data traffic, if conditions for spatial and/or temporal locality are met [11]. Additionally, the regular layout of the NoC makes the electrical parameter of the infrastructure more predictable for high-performance circuits and the short links support optimized signal flows with a high duty cycle of its wires [8]. Another major advantage of NoCs results from the pipelined segmentation, which also represents timing as well as potential voltage barriers and thus avoids the necessity for globally synchronized designs. In co-operation with the high modularity, each tile inside the NoC can represent a decoupled ***voltage and frequency island*** (VFI) [79]–[81]. This architectural concept is referred to as ***globally asynchronous locally synchronous*** (GALS) design [25], [79]–[82].

Initially, the focused NoC architectures of the work at hand should be outlined before stepping into the fundamentals of this infrastructure. The emphasis is placed on the class of mesh-based designs with applied wormhole-switching and parallel point-to-point links. The reason for this selection results from the practical relevance of this infrastructure in the research [7], [9], [10], [25], [27], [28], [30] as well as the industrial [1], [16], [18], [21], [23], [26], [29], [31], [83] domain. This mesh-based setup offers the desired regularity and simplicity. Thus, the following sections will contain more focus-related details and practical evaluations, but for completeness alternative solutions will be discussed and referenced.

2.1 INFRASTRUCTURE ORGANIZATION AND COMPONENTS

In general, the NoC is a ***packet-switching communication infrastructure*** for CMPs and contains three different types of basic components to enable data transfers between the IP cores [7]–[10], [28], [30], [75] (see Figure 2.1). Each of those types implement dedicated functionalities as follows:

- The ***network interface*** (NI) represents the terminal circuit that enables the IP core to transmit and receive data through the NoC via standardized protocols at the ***end-to-end*** (ETE) level between IPs. It has to implement functionality regarding buffering, packetization of data transfers, ETE flow control, ETE error handling, and parts of the routing.
- The ***router*** or ***switch*** processes the storing and forwarding of packets towards their destined IP core at the ***hop-to-hop*** (HTH) level. It has to implement functionality regarding buffering, input and output selection/allocation, multiplexing, link level flow control and error handling.

Routers are interconnected via links to each other and the NIs at the IPs. The traversal of data from one router via an outgoing link to another router or the NI is defined as a hop.

- The ***link*** represents an unidirectional interconnect that is responsible for the physical transmission of data between routers as well as routers and NIs. Therefore, w_{data} data and w_{ctrl} control signals are passed as data words via $w_{link} = w_{data} + w_{ctrl}$ parallel wires with integer word size of the transmitted data bits w_{data} as a multiple of the power of two [25].

The final logical structure of a NoC with N_{router} routers, N_{NI} NI terminals and N_{links} links is specified by the ***topology***. Besides the basic components of its structural hardware organization, the NoC approach specifies data units, system abstraction layers and hierarchies for the data flows inside the network.

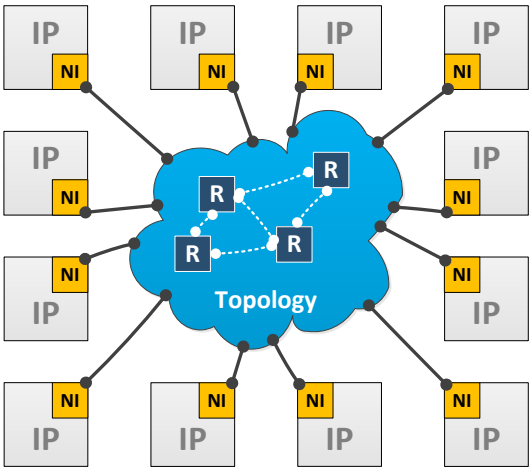


Figure 2.1: Basic components of Networks-on-Chip

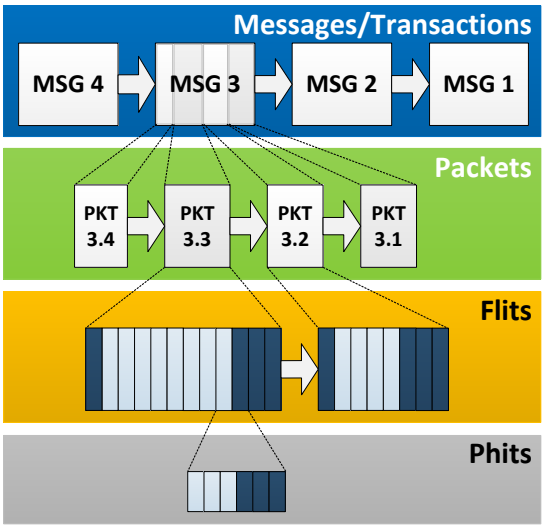


Figure 2.2: Data flow decomposition hierarchy and data units in Networks-on-Chip

As illustrated in Figure 2.2, the ***hierarchical decomposition of data flows*** contains four entities [84]:

- ***Transactions*** define the basic interoperation between distributed workloads at the application or the system protocol layer in a NoC-based CMP. Through defined application programming interfaces (APIs), these *transactions* are converted into a sequence of *messages* and pushed towards the message buffer that sources the NI.
- The NI splits these ***messages*** up into a sequence of *packets* and vice versa. Thereby, a *packet* represents the native data unit of the NoC that is handled at the path level for ETE transmissions between NIs.
- Each ***packet*** represents a container that transports one up to multiple ***flits*** (flow control units). A *flit* is the basic data unit for buffer sizing and resource allocation at the path for ETE as well as the link level for HTH transmissions.

The last data unit is the ***phit*** (physical unit) and refers to the physical data width of a link (the data word that passes a link in a single cycle). Each *flit* represents a composition of one up to multiple *phits*. Most typically, the size of a *flit* and a *phit* are assumed to be equal for practical and performance reasons regarding the avoidance of serialization overheads and faster flow control at the link level [25]. The work at hand applies this equal sizing and thus a flit is considered as smallest unit for data flows in the NoC. In the Figure 2.3, these data units are put into the context of the NoC system abstraction layers for a schematic data flow along a multi-hop path from the source to the destination IP over intermediary routers.

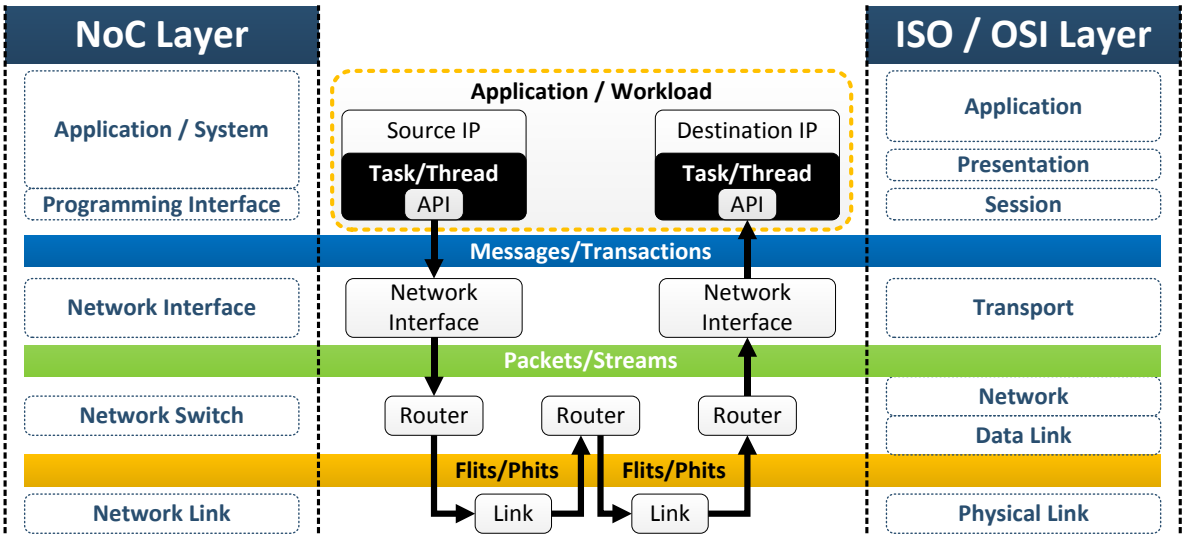


Figure 2.3: Communication data flow and abstraction layers for Networks-on-Chip

Furthermore, the NoC layers are contrasted with the ISO/OSI model [9], [30], which defines a standard reference for open system interconnection and inter-process communication in heterogeneous and distributed systems. The NoC covers all ISO/OSI layer from transport down to the physical link level within its basic components.

The packet represents the native communication entity of the NoC and its structure has major impact on the implementation cost (i.e. buffer and link sizing) as well as on the performance (i.e. flow control, error handling, or network congestions). The typical generic data layout of a routable packet, used for wormhole-switching NoCs, is depicted in Figure 2.4. Therein, a flit represents the smallest entity that passes the links with applied flow control [7], [9], [10]. To identify and interpret its data contents each flit carries a standardized type signature (TYPE) that also segments the packet into three specific sections:

- The leading section is the ***packet header*** (H) and its data word contains all relevant information fields for control and management of the packet's traversal along its path from source (SRC) to destination (DST) IP and dynamic *extensions* (EXT). Hence, the header flag

marks the beginning of a new packet inside a data flow. The additional free *payload* field holds *application*- or *system-specific* data of the communicating workload instances.

- The next section is the ***packet body*** (B) and the corresponding data words contain the *application-specific payload* data of the communicating workload instances.
- The final section is the ***packet tail*** (T) and marks the *end* of the current packet. Its data word typically contains *application-specific payload* and further dynamic *extensions* (EXT) for application control or traffic management.
- Optionally, a packet may contain additional data per flit or as final flit itself for *error detection and correction* (EDAC) mechanisms at the link and/or path level to ensure a reliable communication inside the NoC [40].

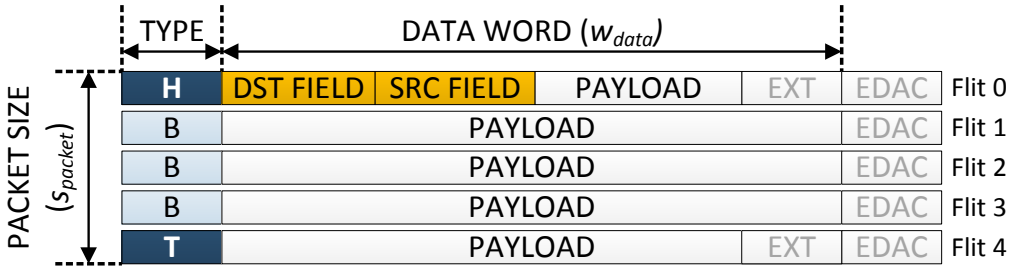


Figure 2.4: Basic structure and organization of communication packets in Network-on-Chip

The TYPE signature for the flits consumes at least two additional bits (i.e. '01'=H, '00'=B, '10'=T and '11'=H+T) and together with the optional EDAC bits it contributes to the number of control wires w_{ctrl} in the link's width w_{link}. The data word fraction contributes to the data wires w_{data} and typically represents a multiple of commonly used data word sizes inside the memory or processors of the CMP. The packet size s_{packet} in a wormhole-switching NoC is not limited, but typically transports between 32 and 64 bytes of payload [25]. Furthermore, single flit packets are possible (TYPE=H+T) and have the main purpose of transporting little amounts of data as exemplary needed for application control or cache coherency.

The resulting traffic of packetized data flows inside the NoC will be quantified via rates. Those rates (r_{flit} or r_{packet}) specify the number of flits (n_{flit}) or packets (n_{packet}) that was injected/received in a dedicated time interval ($\Delta t = t_2 - t_1$). The scope of these rates can vary between per tile/core, region, distance, or system-wide measurements. Most commonly, the mean ***injection/reception rates*** per core will be used as standardized metric for the traffic inside the NoC and the time interval is normalized to one clock cycle. Hence, the data rates quantify the number of injected/received flits/packets per time unit or clock cycle as follows:

$$\mathbf{r_{flit}} = \frac{n_{flit}}{(t_2 - t_1)} = \frac{n_{flit}}{\Delta t} \;\; with \;\; t_2 > t_1 \qquad (2.1)$$

$$\mathbf{r_{packet}} = \frac{\mathrm{n}_{packet}}{(t_2 - t_1)} = \frac{\mathrm{n}_{packet}}{\Delta t} \quad with \ t_2 > t_1 \tag{2.2}$$

As mentioned before, the tile-based organization of the CMP and the well-segmented structure of the NoC benefits in the integration of the final system as modular GALS design. Therein, the IP cores and NoC components are placed and grouped as VFIs. These VFIs support a dedicated run-time power management through *dynamic voltage frequency scaling* (DVFS) [79]–[81] and allows optimized adjustment for both parameters at dedicated locations regarding the current requirements of the systems workload. Frequency and voltage islands do not necessarily partition the NoC with the same granularity and can vary in their dimensions [57], [79], [85]–[87]. Because of the implementation and run-time cost for fine grained voltage regulation, most commonly a single voltage island spans over clusters of multiple independent frequency domains [58], [79], [85], as realized in the Intel SCC [16]. The voltage selection complexity grows exponentially with the number of voltage islands [85]; the domain-crossing FIFOs as well as voltage regulators introduce additional latency, area as well as power costs [58].

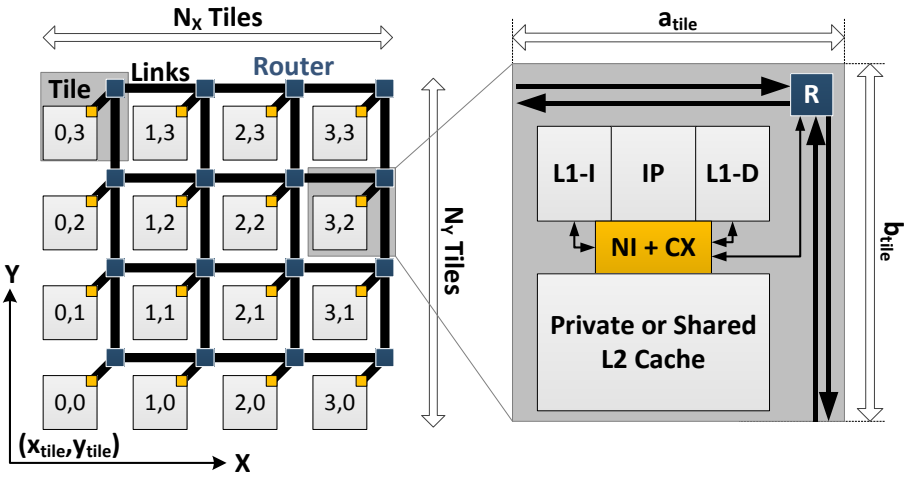


Figure 2.5: Layout of a regular 4x4 Network-on-Chip with a two-dimensional mesh topology and exemplary tile setup

The presented infrastructure organization regarding basic components, logical network structure, system abstraction layers, communication entities, and timing/power barriers offers a high-level view on the on-chip network and its architecture. But for the operation of a NoC, the NIs and routers have to implement a compatible set of functionalities regarding buffering, flow control, switching as well as routing at the ETE/HTH level. Thereby, the cost at design- and run-time must be minimized, while desired performance requirements have to be met for the scoped workload scenarios. Such cost-performance-tradeoffs and physical layout constraints enforce the system designer to focus on regular structures and algorithms with low complexities. Thus, on-chip networks are very limited in the implementation of features that already exist in other networking technologies used in the local or wide area networks [7], [9], [10]. As mentioned above, this work focusses on regular mesh-based

NoCs and Figure 2.5 illustrates it in its simplest form, the two-dimensional mesh (2D-Mesh). It fits well into the tile-based design flow and offers good scalability as well as predictable layout characteristics [6], [80]–[82], [88]. The floorplan is structured as planar array of $N_{tile} = N_X \times N_Y$ tiles with a width of N_X and a height of N_Y tiles.

Parameter	Description	Equation	
Link's width	# of wires per direction	$\boldsymbol{w_{link}} = w_{data} + w_{ctrl}$	(2.3)
NoC size	# of tiles	$\boldsymbol{N_{tile}} = N_X \times N_Y$	(2.4)
Tile area	area units	$\boldsymbol{A_{tile}} = a_{tile} \times b_{tile}$	(2.5)
CMP area	area units	$\boldsymbol{A_{CMP}} = N_{tile} \times A_{tile}$	(2.6)
Address size	# of bits	$\boldsymbol{w_{ID}} = \lceil log_2(N_X) \rceil + \lceil log_2(N_Y) \rceil$	(2.7)
Hop distance	# of passed routers (2D-Mesh)	$\boldsymbol{n_{hop}} = \lvert x_{dst} - x_{src} \rvert + \lvert y_{dst} - y_{src} \rvert + 1$	(2.8)

Each tile has a rectangular layout shape of $a_{tile} \times b_{tile}$ and contains the router, at least a pair of bidirectional inter-router links, the IP core, and the NI. Each router is interconnected via bidirectional links with up to four surrounding routers (UP, RIGHT, DOWN, LEFT) and the NI (IP, Cache, etc.). Principally, the number of connected cores per router is not limited to one and more advanced mesh-based NoCs are discussed in the following chapters. Furthermore, the number of connections between router and NI per tile is not necessarily restricted to one. The unique addressing of cores is based on the cartesian coordinate system with a minimum required data word size w_{ID} of $\lceil log_2(N_X) \rceil + \lceil log_2(N_Y) \rceil$ bits per (x,y)-identifier. The minimum hop distance n_{hop} between a source (x_{src}, y_{src}) and a destination (x_{dst}, y_{dst}) is defined as number of passed routers by the packets along the path, as calculated by $\lvert x_{dst} - x_{src} \rvert + \lvert y_{dst} - y_{src} \rvert + 1$ for the 2D-Mesh. The corresponding routing is referred to as minimal dimension-ordered xy-routing, where the packet between source and destination first traverses along the x-direction until it matches the destinations x_{dst}-position and afterwards completes this procedure along the remaining y-coordinate to reach the y_{dst}-position. This is the simplest routing for mesh-based NoC and advanced algorithms are discussed in the next sections of this chapter.

After this preliminary introduction of the NoC paradigm, its basic organization and terminology, the following sections provide details about the router, links, topologies, and further traffic management concepts like virtual channels, multi-networks, or channel slicing.

2.1.1 ROUTER, LINKS, AND PATHS

ROUTER COMPONENT BASICS: The ***router*** is a key component to enable networked on-chip communication and implements the functionalities for buffering, flow control, resource allocation, multiplexing and routing of packets from concurrent data flows. This work applies a standardized generic router for wormhole-switching NoCs that is widely used as reference design [7], [9], [10],

[20], [25], [28], [89]–[92]. It is not intended to modify the hardware units inside this structure, but for completeness dedicated publications on those topics are referenced as well.

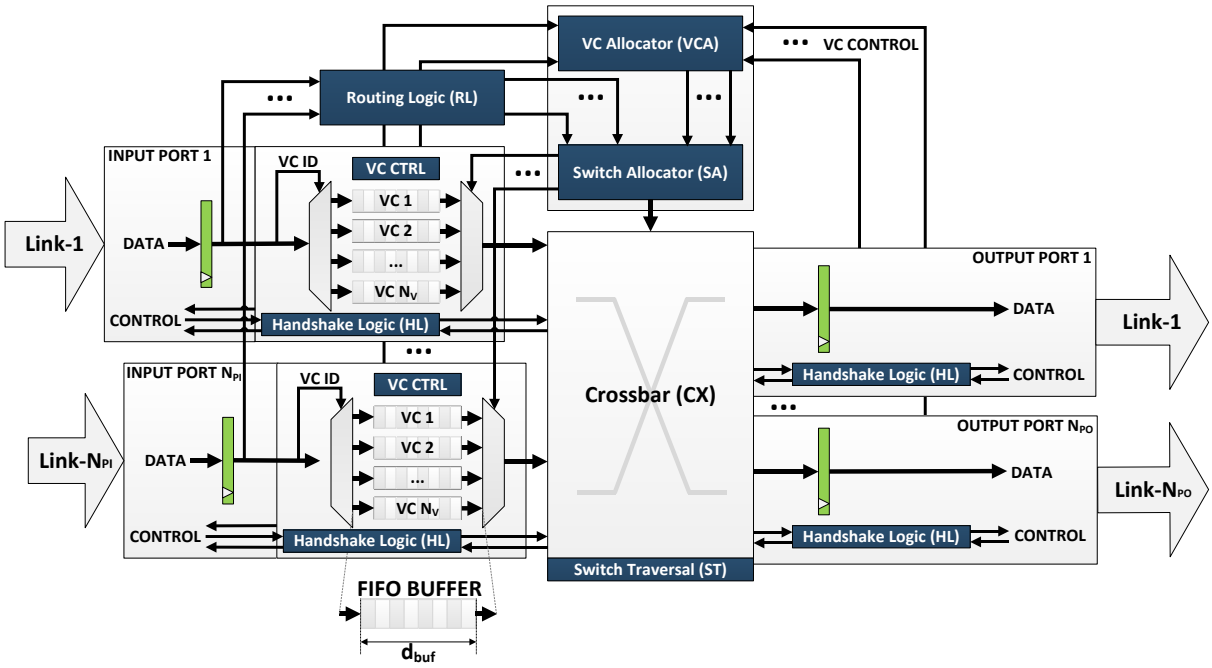


Figure 2.6: Canonical pipeline structure of a generic Network-on-Chip router

The basic microarchitecture of the generic router is illustrated in Figure 2.6 and consists of:

- ***Input*** and ***output ports*** as basic connection wrappers from incoming and to outgoing unidirectional links. These ports implement at least the buffering of flits and the ***handshake logic*** (HL) for their ***link traversal*** (LT). The buffering can be implemented solitary at the input/output port or in a dual way at both connectors. The work at hand applies the input-buffered router architecture. The number of input/output ports per router is N_{PI}/N_{PO}. Under consideration of bidirectional connections, both parameters are equally sized and simplified to the port count $N_P = N_{PO} = N_{PI}$, which describes the ***degree*** or ***radix*** of the router. Furthermore, those N_P router ports can be split into N_{PR} router-to-router and N_{PC} router-to-core connections ($N_P = N_{PR} + N_{PC}$).

- A central ***crossbar*** (CX) that connects the N_{PI} input with the N_{PO} output ports to enable the ***switch traversals*** (ST) from incoming to outgoing links. The CX is connected to a dedicated *switch allocator* (SA) logic that controls the internal multiplexing and implements the input-to-output selection function to handle concurrent traffic streams.

- Distributed (per port) or centralized control logic for the traffic management [93]. This basically includes the ***routing logic*** (RL) to perform the output selection function for incoming packets, the ***switch allocator*** (SA) to perform the input selection/arbitration function for the outputs and the ***vc allocator*** (VCA) if multiple virtual channels (VCs) are

used. Thereby, VCs represent mechanisms to map and manage multiple independent traffic flows on shared physical paths inside the NoC with reduced inferences.

The buffering of flits inside the router is realized via FIFO structures with separated read- and write-pointer and adjusted clocking regarding the synchronization issues of the mesochronous GALS design. The implementation can be realized via flip-flop-based registers (FF) or synchronous dynamic random access memory (SDRAM). Thereby, the buffer depth parameter d_{buf} specifies the maximum number of storable flits inside a single FIFO and the number of virtual channels N_V determines the amount of parallel FIFOs per port. Both parameters can vary locally for heterogeneous or irregular NoC designs (d_{buf} per port and/or VC, and N_V per port and/or dimension) [94], [95], but these application-specific optimizations are out of scope for this work and globally constant parameter settings are considered.

The functional dependencies of the router components define a ***canonical pipeline structure*** for the traffic flow from the input to the output port. Each incoming packet has to traverse these pipeline stages until it has reached the input port of the next router or NI to complete the current hop. The packet arrives flit by flit at the input port buffer and the transfer over the incoming link is regulated by the HL, which implements the needed HTH flow control for in- and external communication. The flits are screened at the TYPE signature to detect the header or tail of a packet.

STAGE 1 – ROUTING LOGIC: At the beginning of a packet, the RL is activated once if a new header was detected and performs the output selection function based on the DST field data to determine the corresponding output port for the new packet. This procedure is essential to request the correct output where all flits of this packet will be forwarded to and happens simultaneously to the buffer write. The active request for the output is released after the packet tail has passed. For the operation of the RL a wide variety of algorithms are available, which will be discussed in Section 2.2.3.

STAGE 2 – VC ALLOCATOR: The VCA handles the assignment of the packet to the correct VC based on information of the previous/next router (dynamic) or fixed as part of the header flit (static). Further control logic (VC CTRL) is needed to manage the switching between different VCs at the same port to accomplish guaranteed bandwidths or delays. ***This stage is obsolete if no VCs are implemented.***

STAGE 3 – SWITCH ALLOCATOR: The SA performs the input arbitration for the output ports to handle concurrent requests and controls the multiplexing of the CX. The activated request signals of RL and VCA serve as inputs and the SA evaluates acknowledgement signals to grant selected input ports the access to their requested outputs.

STAGE 4 – SWITCH TRAVERSAL: After the reception of the packet header the CX traversal to the correct output resulted from the interaction of RL, VCA and SA. Thus, flit by flit of the packet passes

the CX at this stage to be forwarded by the output port to the succeeding router or NI. This ST is granted until the packet tail has passed and this flow will only be partially interrupted by the controlled VC switching.

STAGE 5 – LINK TRAVERSAL: The LT stage passes each flit via the connected link from the output port to the input port of the succeeding router or NI. Similar to the ST, the flow control is handled by the HL. Regarding the growing difference at wire and logic delays, the LT stage most commonly represents the stage with the highest end-to-end signal propagation delay and determines the locally achievable clock rate of the pipeline.

This 5-staged RL-VCA-SA-ST-LT pipeline walkthrough is reduced to 4 stages (RL-SA-ST-LT) without the implementation of VCs. After the header of a packet prepared the traversal of the current hop, the remaining payload and body flits simply follow in a wormhole-switching fashion. Alternative switching algorithms like store-and-forward (SAF) or virtual-cut-through (VCT) would work similar with additional specific delays and will be discussed in the algorithms section (Section 2.2.1 p. 47).

ROUTER TIMING: The timing characteristics as well as the achievable clock rate of the presented router structure depend on the static delays inside the pipeline stages. Furthermore, traffic dependent dynamic effects regarding concurrency and serialization will contribute to the overall latency of a packet to pass a single hop. The minimum clock rate $t_{clk,min}$ of the NoC is defined by the worst critical path along the router pipeline, which is typically the LT due to the impact of the wire delay t_{wire}. The stages for RL, SA, VCA and ST are dominated by combinational logic with lower signal propagation delays. Hence, the frequency f_{NoC} and clock rate t_{NoC} of the NoC will be adjusted for these specific bounds.

$$\boldsymbol{f_{NoC}} = 1/t_{NoC} \;\; with \;\; \boldsymbol{t_{NoC}} \geq t_{clk,min} \geq t_{wire} \tag{2.9}$$

The pipeline with LT stage represents a complete hop inside the NoC from the input of the current to the input of the succeeding component. Thereby, the traversal of the header flit determines the hop delay t_{hop}, while the remaining flits of the packet follow with a static wormhole-switching offset for ST and LT. This hop delay t_{hop} contains static ($t_{hop\rightarrow stat}$) and dynamic ($t_{hop\rightarrow dyn}$) fractions, which depend on the complexity of implemented algorithms and protocols.

$$\boldsymbol{t_{hop}} = t_{hop\rightarrow stat} + t_{hop\rightarrow dyn} \tag{2.10}$$

$$\boldsymbol{t_{hop\rightarrow stat}} = t_{RL} + t_{VCA} + t_{SA} + t_{ST} + t_{LT} = n_{pipe\rightarrow stat} \cdot t_{NoC} \;\; with \;\; n_{pipe\rightarrow stat} \in \mathbb{N}^{+} \tag{2.11}$$

$$\boldsymbol{t_{hop\rightarrow dyn}} = t_{RQ\rightarrow HOL} + t_{RQ\rightarrow VCA} + t_{RQ\rightarrow SA} \tag{2.12}$$

The static fraction $t_{hop\rightarrow stat}$ represents the minimum delay for the header flit to pass the pipeline and each of the stages contributes with a specific portion ($t_{RL}, t_{VCA}, t_{SA}, t_{ST}, t_{LT}$). The stages of RL,

VCA, and SA typically need at least one clock cycle to evaluate their control signal adjustments. The static delays of ST and LT depend on the HL with typical values of one or two clock cycles. Thus, the ideal delay for the packet header to pass a single hop (with/without VC) ranges between 5/4 and 7/6 clock cycles. The dynamic fraction $t_{hop \rightarrow dyn}$ results from queuing effects regarding concurrency and serialization. At the input buffer, packets will be delayed by the head-of-line blocking $t_{RQ \rightarrow HOL}$ of leading packets in the FIFO queue. Furthermore, serialization effects occur during the requests of the virtual channels ($t_{RQ \rightarrow VCA}$) and the crossbar ($t_{RQ \rightarrow SA}$), which is why starvation-free scheduling policies for arbitration and allocation of resources are very important to guarantee the needed fairness [7], [10], [93], [96]. Thereby, round-robin and least-recently-served (matrix) arbitration represent the typical policies in the NoC domain [93]. The proportions of the dynamic and static fractions in the hop delay strongly depend on the traffic situation inside the NoC. While the ideal delay corresponds to the pure static part, increased NoC traffic with concurrent data flows will lead to the dominance of the dynamic delay.

ROUTER AREA: The area costs of the router A_{Router} are dominated by the buffers (BUF, A_{BUF}) at the input ports and the centralized crossbar (CX, A_{CX}). This is due to the dependencies of those area fractions on the complete width of the link, port and VC number. Contrary, the allocator (A_{SA+VCA} and A_{SA} when VC-free) has to serve all ports as well as VCs too, but only consumes little area due to its reduced I/O width of single bit status wires. The area costs of the RL-stages (A_{RL}) are relatively small for simple routing algorithms, but their costs will increase if more complex routing tables or adaptive strategies are applied [97],[98]. According to the block-based component area model in [20], the allocator evaluations in [93], and the modeling assumption in [92], [99], [100], the simplified area cost dependencies of the reference router can be formulated as follows:

- **Total buffer area per router** : $\boldsymbol{A_{BUF}} \propto N_P \times N_V \times w_{link} \times d_{buf}$
- **Centralized crossbar area** : $\boldsymbol{A_{CX}} \propto (N_P \times w_{link})^2$
- **Allocator area per router** : $\boldsymbol{A_{SA+VCA}} \propto N_V \times N_P^2 \;\; and \;\; \boldsymbol{A_{SA}} \propto N_P^2$
- **Routing logic area per router** : $\boldsymbol{A_{RL}} \propto N_P \times w_{ID}$

ROUTER POWER: The power dissipation on-chip is one of the major challenges to maintain the technology scaling [34] and each component contributes to it in form of a static (P_{stat}) and dynamic (P_{dyn}) power consumption, whereas both fractions will be combined to the total power [32].

$$\boldsymbol{P_{total} = P_{dyn} + P_{stat} = P_{SW} + P_{SC} + P_{stat}}$$

The dynamic part has two different components resulting from power dissipation through switching (P_{SW}) and short circuits (P_{SC}). For the dedicated parts of the power consumption below and the

assumption in [20], [92], [93], [99], [100] the simplified design parameter dependencies for the NoC router can be estimated.

$$\boldsymbol{P_{SW}} = \alpha \cdot C_L \cdot V_{DD}^2 \cdot f_{NoC} \quad ; \quad \boldsymbol{P_{SC}} = \frac{t_R + t_F}{2} \cdot \alpha \cdot V_{DD} \cdot I_{SHORT} \cdot f_{NoC} \quad ; \quad \boldsymbol{P_{stat}} = I_{LEAK} \cdot V_{DD}$$

The activity α generated in the router depends on the data streaming through it and thus on the mean flit injection rate per input port $\bar{r}_{flit \to in}$. Thereby, capacitances (C_L) will be charged and discharged during the write/read operations of w_{link}-wide flits in each of the $N_P \times N_V$ buffers of the router as well as the traversal of those flits along the crossbar. Furthermore, shorts circuit currents (I_{SHORT}) occur in the driver and buffer circuits during those router operations.

- **Switching power dependencies** : $\boldsymbol{P_{SW \to Router}} \propto \bar{r}_{flit \to in} \times N_P \times w_{link} \times N_V$
- **Short circuit power dependencies** : $\boldsymbol{P_{SC \to Router}} \propto \bar{r}_{flit \to in} \times N_P \times w_{link} \times N_V$

The main source of the static power dissipation inside the router is the leakage of the $N_P \times N_V$ buffers, whereas each those buffers has $d_{buf} \times w_{link}$ cells.

- **Static power dependencies** : $\boldsymbol{P_{stat \to Router}} \propto d_{buf} \times N_P \times w_{link} \times N_V$

At a first glance, the static power dissipation seems independent of the activity inside the router induced by the injected data stream, but the leakage currents (I_{LEAK}) inside the buffer cells depends on the operating temperature (T_{junc}) at the router which will be impacted by the activity itself as follows:

$$\boldsymbol{T_{junc}} = T_{amb} + R_{th} \cdot P_{total \to Router} = T_{amb} + R_{th} \cdot \left(\boldsymbol{P_{dyn \to Router}} + P_{stat \to Router} \right)$$

$$\Longrightarrow \quad \boldsymbol{T_{junc}} \propto \bar{\boldsymbol{r}}_{\boldsymbol{flit \to in}} \quad with \quad P_{dyn \to Router} \propto \bar{r}_{flit \to in}$$

Thereby, T_{amb} describes the ambient temperature and R_{th} the thermal resistance for the heat conduction from the chip to the heat sink. The resulting activity-induced temperature variations impact the leakage current and thus the static power dissipation. Hence, the activity inside the router influences the leakage of the router as follows:

$$\boldsymbol{I_{LEAK}} = f\left(T_{junc}\right) \propto T_{junc}^{\gamma} \quad \Longrightarrow \quad \boldsymbol{I_{LEAK}} \propto \bar{r}_{flit \to in}^{\gamma} \quad with \quad \gamma \geq 1$$

$$\boldsymbol{P_{stat \to Router}} \propto \boldsymbol{I_{LEAK}} \propto \boldsymbol{T_{junc}^{\gamma}} \quad \Longrightarrow \quad \boldsymbol{P_{stat \to Router}} \propto \bar{\boldsymbol{r}}_{\boldsymbol{flit \to in}}^{\boldsymbol{\gamma}} \; with \; \gamma \geq 1$$

The feedback of this additional increase in the static power consumption to the operating temperature results in a self-accelerating process known thermal-runaway [36], [37], [101], [102].

$$\boldsymbol{T_{junc}} \propto \boldsymbol{P_{stat \to Router}}$$

It has to be noted that this is only a simplified assumption for the thermal dependencies at the chip-level [36], [101], [102] and does not consider any non-linear saturation effects over time [37]. But the resulting temperature variations over the increase of the injection rates at the targeted reference

router by the simulations of *Aisopos* in [43] indicate that such approximation seem sufficient for coarse dependency estimates (see Appendix p. 175).

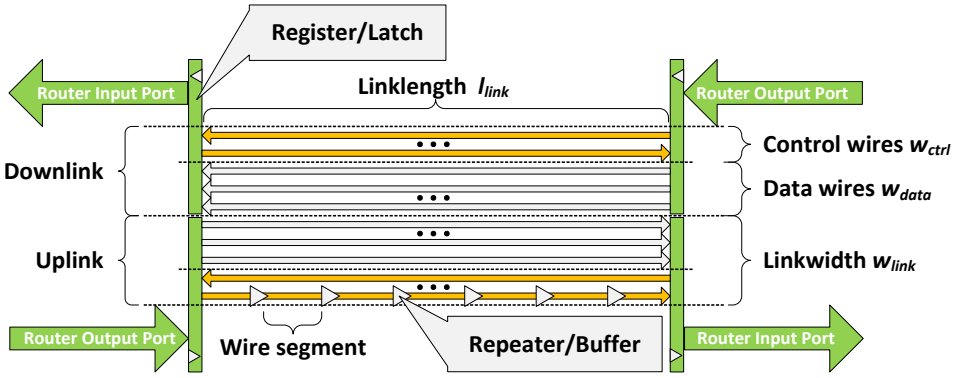


Figure 2.7: Structure of a bidirectional Network-on-Chip link that connects two router/switches

LINK COMPONENT BASICS: ***Links*** represent the ***point-to-point interconnection resources*** that will be managed by the routers and have major impact on the performance- as well as on the costs-side of the NoC infrastructure. Most commonly, adjacent routers are connected in a ***bidirectional*** scheme via two links (up- and downlink) as illustrated in Figure 2.7 and each link consists of w_{link} parallel wires with a length of l_{link}. The wires of the link are composed of w_{ctrl} control and w_{data} data wires, which are integrated at the intermediate metal layer of the chip. The control wires are responsible for the transport of data regarding the flit type, VC identifier, EDAC, flow control (i.e. request/acknowledge), and further information for dynamic traffic management mechanisms. The data wires transport the data word section of the flits.

LINK TIMING: Regarding the signal propagation delay of on-chip interconnects proportional to the square its length ($t_{wire} \propto l_{link}^2$), those links are typically split up into n_{seg} segments by the insertion of repeater or fully-registered pipeline stages to achieve a linear delay scaling ($t_{wire} \propto l_{link}$) [84], [103], [104]. Thereby, repeated links represent the standard [34] while pipelined links are considered for longer global interconnects, because pipeline stages comes along even higher area, power and round-trip latencies in HTH flow control [103], [104]. The segmentation of interconnects via repeaters follows a tradeoff between delay, throughput and power dissipation. Based on distributed RC [34] or RCL [103] wire models, the optimized number of segments, sizing of wires as well as repeaters can be calculated. A simple optimization for repeater insertion, based on the distributed RC-model, can be formulated as follows [84]:

$$\boldsymbol{t_{wire}} = n_{seg} \cdot t_{seg} \quad with \quad t_{seg} = t_{rep} + t_{lseg} = t_{rep} + r \cdot c \cdot \left(\frac{l_{link}}{n_{seg}}\right)^2 \tag{2.13}$$

Thereby, r and c represent the wire resistance and capacitance per unit length. This approach represents straight-forward segmentation of the link in n_{seg} repeater-driven sections, where the

delay of each segment t_{seg} contributes to the total wire delay t_{wire} by the signal propagation delay of the repeater t_{rep} and the succeeding wire segment t_{lseg}.

LINK AREA: The area cost estimation for the NoC links is straight forward regarding the regular layout. According to the model of *Balfour et. al.* in [20] the formula will be as follows:

- **Total link area per direction** : $\boldsymbol{A_{LINK}} = l_{link} \times (w_{link} \times (W_{wire} + S_{wire}) + S_{wire})$
- **Total bidirectional link area** : $\boldsymbol{A_{BI-LINK}} = 2 \times A_{LINK}$

Thereby, W_{wire} is the width of a single wire and S_{wire} the spacing between two wires. This area calculation assumes the planar integration of all link wires inside a single metal layer.

LINK POWER: The dynamic and static power dissipation dependencies of the link are similar to those formulated for the router. The activity α generated on the link with w_{link} segmented wires depends on the data streaming through it and thus on the mean flit injection rate $\bar{r}_{flit \to in}$. Thereby, capacitances will be charged and discharged on the n_{seg} segments per wire. The size of the capacitance per segment is dominated by its length l_{seg} $(= l_{link}/n_{seg})$. The short circuit currents occur in the n_{seg} repeater circuits per wire whose rise/fall times will be dominated by the capacitance of the driven wire segment as well. Thus, the dynamic power dependencies for the link can be expressed as follows:

- **Switching power dependencies** : $\boldsymbol{P_{SW \to Link}} \propto \bar{r}_{flit \to in} \times w_{link} \times (n_{seg} \times l_{seg})$
- **Short circuit power dependencies** : $\boldsymbol{P_{SC \to Link}} \propto \bar{r}_{flit \to in} \times w_{link} \times (n_{seg} \times l_{seg})$

The main source of the static power inside each link is the leakage of the $w_{link} \times n_{seg}$ repeaters.

- **Static power dependencies** : $\boldsymbol{P_{stat \to Link}} \propto w_{link} \times n_{seg}$

Furthermore, there is the same relation between the operating temperature, dynamic power and static power dissipation. Thereby, links suffer from activity-driven self-heating [37], [105], [106] and the wires itself represent a thermal conductor [37].

NETWORK PATHS BASICS: Router and links, as discussed above, will be used to realize the NoC under consideration of a specified topology and the ETE communication of the connected IP cores is established via packetized data flows over paths through this network. Thus, hop count, delay, availability, redundancy as well as concurrent utilization of those ***network paths*** have major impact on the performance and reliability of the communication in the CMP. Depending on the applied routing algorithm, such *network paths* can be ***static*** or ***dynamic***. Static paths remain constant for each packet transfer between source (s) and destination (d), while dynamic paths will be adapted at the ETE or HTH level at run-time. The resulting hop distance $n_{hop,s \to d}$ of a path (# of passed router between s and d) further allows the classification as ***minimal*** or ***non-minimal***. In a ***minimal path***

setup, the packets of a data flow are forwarded to an adjacent router whose position decrements the remaining distance to the destination at least by one hop. ***Non-minimal paths*** are not subjected to this constraint.

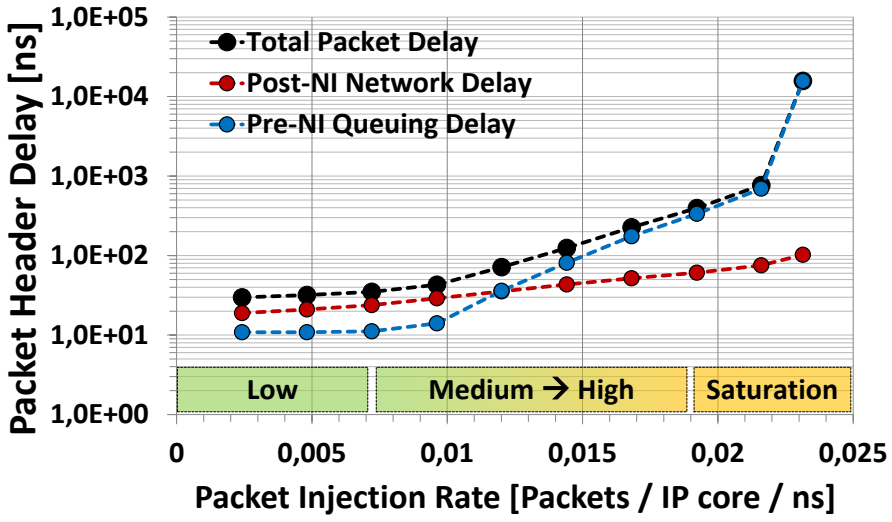


Figure 2.8: Progression of the end-to-end packet header delay as function of the mean traffic injection rate per core along a path inside an 8x8 tile mesh-based Network-on-Chip

The ETE latency of the paths is one of the most important performance evaluation measures for the communication in the NoC. As discussed above in the equations 2.10 to 2.12, each of the $n_{hop,s\to d}$ hops along a path ($s \to d$) contributes with a static ($t_{hop\to stat}$) and a dynamic ($t_{hop\to dyn,i}$) delay, whereas the dynamic fraction depends on the local traffic situation in each router ($1 \leq i \leq n_{hop,s\to d}$) along the path and the static fraction per hop is constant ($n_{hop,s\to d} \cdot t_{hop\to stat}$). Furthermore, in a wormhole-switching NoC the final reception delay ($t_{recv,d}$) for the remaining $(s_{packet} - 1)$ flits of a packet, after the header reached the destination, is constant as long as the receiver buffer at the NI is not full and depends on the HTH flow control ($n_{HL} \cdot t_{NoC}$). In addition to those in-network delays, further queuing delays (t_{queue}) occur at the transmission buffer of the source before the packet enters the network. Such queuing delays are highly dynamic and result from ***head-of-line*** blocking inside the transmission buffer. Thereby, the traffic situation inside the NoC has major impact, because the degree of interferences, blocking, and concurrency of traffic flows determines the intensity of the ***backpressure*** on waiting packets inside the transmission buffer. According to the discussed composition above, the final expression for the path delay of a packet ($t_{path,s\to d}$) and the packet header ($t_{path\to header,s\to d}$) can be expressed as follows:

$$\boldsymbol{t_{path,s\to d}} = t_{queue} + n_{hop,s\to d} \cdot t_{hop\to stat} + \sum_{i=1}^{n_{hop,s\to d}} t_{hop\to dyn,i} + t_{recv,d}$$
$$= t_{path\to header,s\to d} + (s_{packet} - 1) \cdot n_{HL} \cdot t_{NoC} \qquad (2.14)$$

$$t_{path\rightarrow header,s\rightarrow d} = t_{queue,s} + n_{hop,s\rightarrow d} \cdot t_{hop\rightarrow stat} + \sum_{i=1}^{n_{hop,s\rightarrow d}} t_{hop\rightarrow dyn,i} \quad (2.15)$$

For the evaluation of the NoC performance, the mean packet header delay ($\bar{t}_{packet\rightarrow NoC}$) for the total number of received packets ($N_{packets}$) will be used most commonly.

$$\bar{t}_{packet\rightarrow NoC} = \frac{1}{N_{packets}} \cdot \sum_{p=1}^{N_{packets}} t_{path\rightarrow header,p} = \bar{t}_{queue,s} + \bar{t}_{network,s\rightarrow d} \quad (2.16)$$

This measure can be further refined in its focus by the selection of spatial or temporal attributes. The principle curve progression of the mean packet delay over the variation of the mean injection rate per core inside an 8x8 2D-Mesh NoC is depicted in Figure 2.8.

Furthermore, this curve (Total Packet Delay) is sectioned into the two main fractions that contribute to the total packet delay (as captured by Equation 2.15 and 2.16 as well). The pre-NI network delay ($\bar{t}_{queue,s}$) represents the backpressure-induced fraction at the transmission buffer before the packet enters the network. The post-NI network delay ($\bar{t}_{network,s\rightarrow d}$) corresponds to the static and dynamic delays that affect the packets during their path traversals. This curve progression indicates that the knowledge of the current traffic situation inside the NoC is indispensable to initiate effective traffic management mechanisms at run-time. While in lower and medium traffic regions the latency is dominated by the intra-network fraction and thus by the mapping of interacting workload entities regarding their mean path lengths, the higher traffic regions suffer from increased backpressure that drives the NoC into saturation and reconfiguration of concurrent data flows on different paths as well as dedicated application compositions can reduce such phenomena [17], [19], [53], [98], [107]–[111]. The exponential slopes of the delay makes traffic monitoring and the timing of the transition between different management strategies become critical challenges [59].

2.1.2 NETWORK TOPOLOGY

The selection of the ***network topology*** has major impact on the costs, performance, and computational models of the resulting CMP architecture. It defines the connectivity of the IP cores and thereby sets the design constraints, capabilities as well as requirements for the NoC. Over the last decade, researchers proposed various topologies to improve the communication performance of NoCs [7], [8], [10], [20], [25], [28], [30], [56], [80]–[82], [84], [88], [95], [104], [112]–[120]. The first feature for their classification can be seen in the general way the resources are interconnected with each other and their resulting roles in the network. This allows the differentiation between ***direct*** or ***indirect topologies*** as follows:

- In ***indirect topologies***, a router can be connected solely with other routers and act as intermediary node in the network, but not as communication endpoint. Thus, the network consists of router that can only forwards packets and router with one up to multiple terminal connections to IP cores.
- In ***direct topologies***, each router is connected to at least one IP core and other routers of the network. Thus, a router represents a possible communication endpoint of the network and an intermediary node.

The second feature for the classification is the ***regularity*** of NoC topologies, which has a great impact on the diversity of network paths connecting the computational resources and the performable workloads. Thereby, ***regular*** and ***irregular topologies*** differ as follows:

- ***Regular topologies*** provide a non-constrained connectivity for the computational resources under consideration of general-purpose communication pattern to cover a wide range of possible workloads and application scenarios.
- ***Irregular topologies*** provide networked communication infrastructures that are tailored for dedicated applications and workload scenarios. Thus, the irregularity most commonly results from the specialization of communication pattern and the reduction of non-utilized connectivity.

The third way to classify topologies results from the parameter combination of the router's radix ($N_P = N_{PC} + N_{RC}$), its composition of router-to-router (N_{RC}) and router-to-core (N_{PC}) ports, the total number of routers (N_{router}) and the diameter ($n_{hop \rightarrow max}$) of the NoC. Thereby, the network structures mainly separate in two groups, termed as ***low-radix*** or ***high-radix topologies*** with or without ***concentration***. They differ as follows:

- ***Low-radix topologies***, as given by the two-dimensional mesh NoC, have low router degrees (i.e. $N_P = 5\ or\ 6$) with a constant port composition (i.e. $N_{PC} = 1\ or\ 2$ and $N_{PR} = 4$), a flat-structured connectivity of spatially adjacent routers and the diameter as well as the router count scale proportionally with the size of the NoC.
- ***High-radix topologies*** focus the reduction of the network's diameter and its scaling for growing NoCs. While the number of router-to-core ports remain constant (i.e. $N_{PC} = 1$), the number of router-to-router connections increase as a function of the network's size to enable the communication within low hop distances for a maximized number of IP cores. Therefore, additional router-to-router links are integrated that spans over greater spatial areas on-chip to reduce the communication distances from the logical network perspective.

- ***Concentration*** is a generic clustering strategy for reduction of the router count of the NoC through the connection of multiple cores (C=4, 8, …) per router. This increases the radix of the remaining routers (N_{PC} goes up), but simultaneously decreases the number of routers, links, path diversity, and network diameter as well.

Depending on the integration and utilization of VCs in the NoC, the implementation strategy of topologies can be further differentiated between ***virtual-*** and ***physical-express*** [112]. While the class of high-radix topologies, as introduced above, uses *physical-express* links to shorten the logical communication distances through hop minimization, VCs as traffic management mechanism can be applied to integrate specific data flow prioritization and reduce the blocking for dedicated flows instead in a *virtual-express* manner.

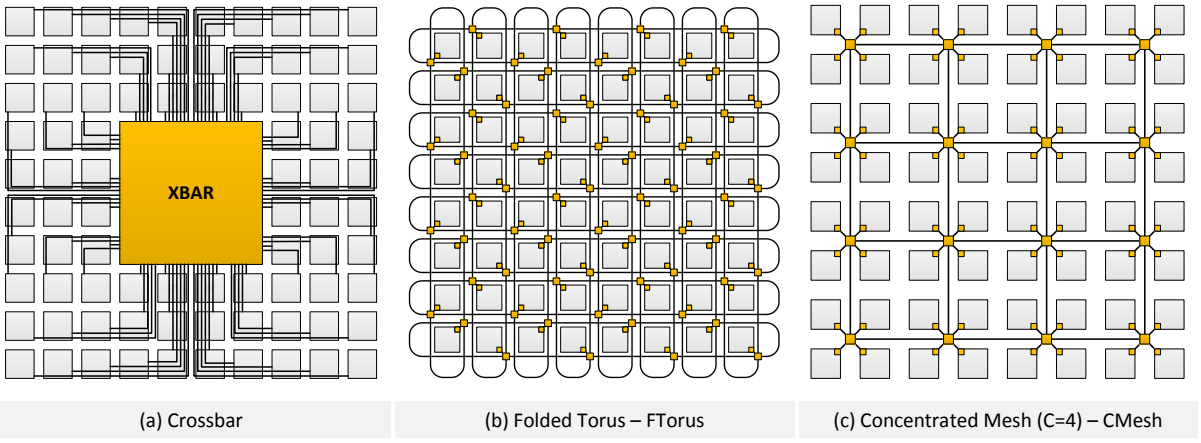


Figure 2.9: Alternative 8x8 NoC topologies: (a) Crossbar, (b) Folded Torus and (c) Concentrated Mesh

Figure 2.9 illustrates a selection of three different topologies for a *regular* CMP platform with 64 IP cores. The centralized crossbar in (a) represents an indirect high-radix topology in its extremist version. The crossbar itself is typically built from multiple switches that can be stacked/layered to save silicon area and reduce the design complexity. Such monolithic crossbar solutions are proposed in [116] or in [115]. Each IP core is reachable within one hop, but this comes at the price of higher hop and wire delays. Moreover, those delays highly depend on the geometric shapes of the tiles and the width of the links. According to [115] and [116], special design strategies allows the realization of a crossbar whose area scales as $O(N_{tile}^2)$ and its wire delay as $O(N_{tile})$, but suitable solutions that outperforms lower radix networks may not exceed the range of 64 to 128 small in-order processing cores [116] and the potential improvements highly depends on spatio-temporal locality of workloads [115], [116]. Furthermore, the monolithic crossbar introduces a single point of failure [116].

Another indirect topology is the tree-based NoC in using an h-tree layout [81], [88]. Depending on the growth of the number of inter-router links, the radix can remain low through all layers of the tree or increase with each layer by the number of aggregated IP cores, which is called fat-tree design. The problem of such aggregative tree-based structures is the low path diversity, high vulnerability due to

permanent router/link faults and the increasing traffic concentration in the upper layers of the tree. The flattened butterfly topology proposed by *Kim* in [119], is an adapted high-radix butterfly using concentration to reduce the number of router as well as links and to become a direct topology. Each router is fully connected to its four surrounding IP cores (C=4) and to all other routers in the same dimensions (row and column). Thus, the maximum hop distance between two locations in the NoC can be reduced to two at the cost of complex wiring. The 2D-Mesh (see Figure 2.5 at p. 30) in is a direct low-radix topology and its concentrated version (C=4) is presented in (c). The folded torus in (b) is an adapted version of the simple torus topology to rebalance the length of the links. In the normal torus the edge routers will be connected via long-range physical-express links to its opposites on the contrarian edge. Unfortunately, this leads to links that need to span over the complete chip with long delays and the NoC does not benefit from the reduced diameter. The folding rearranges the router positions to avoid this long link impact.

According to the focus of this work on direct and regular topologies, Table 2.1 provides an overview and comparison of basic parameters for the characteristic set of such topologies (2D-Mesh, CMesh, FBFly, FTorus). The parameter values were calculated under consideration of square-shaped CMP layouts ($N_X = N_X = N = 2^k$ and $k \in \mathbb{N}^+$) with $N_{tile} = N^2$ tiles, whereas each tile itself is assumed to fulfill this layout constraint as well ($a_{tile} = b_{tile} = d$). Furthermore, it is assumed that the geometric dimensions of a router are negligible and side length d of a tile represents the minimum length unit of the links ($l_{link} = n \cdot d$ and $n \in \mathbb{N}^+$). In addition to the parameters already introduced in the sections above ($N_{tile}, N_{router}, N_{links}, N_P, N_{PR}, N_{PC}, n_{hop \to max}$), new ones and extension are introduced as follows:

- N_{TRPC} is the number of terminal routers per IP core and thus the number of independent access points to the network.
- $N_{V \to min}$ is the minimum number of VCs to realize a specific topology with guarantee of deadlock avoidance.
- $l_{link \to min/max}$ is the minimum/maximum length of the link for a specific topology and an indicator for the possible operational frequency of the NoC.
- $n_{hop \to min}$ is the minimum hop distance for the communication with one of the nearest neighbors in the proximity of an IP core.
- $\bar{n}_{hop \to edge}$ is the mean hop distance to the four cores or routers with the equal row/column index at the perimeter of the CMP. System I/O as well as memory controller are typically placed along the perimeter.

- $N_{links \rightarrow bisec}$ is the number of unidirectional links that crosses the bisection border of the NoC. This parameter, or the derived bisection bandwidth, is often used as throughput indicator for the NoC under consideration of a uniformly distributed random communication pattern. Furthermore, it is an indicator for the global wiring complexity of the NoC.

Table 2.1: Basic design parameter comparison for an exemplary set of different Network-on-Chip topologies

PARAMETER		TOPOLOGY							
		2D-Mesh		CMesh		Fbfly		FTorus	
N_{tile}		N^2		N^2		N^2		N^2	
N_{router}		N^2		$N^2/4$		$N^2/4$		N^2	
N_{links}		$4N(N-1)$		$N(N-2)$		$N^2(N-2)/4$		$2N^2$	
N_P		5		8		$N+2$		5	
N_{PR}		4		4		$N-2$		4	
N_{PC}		1		4		4		1	
N_{TRPC}		1		1		1		1	
$N_{V \rightarrow min}$		1		1		≥ 2 ([119])		≥ 2 ([10], [121])	
$l_{link \rightarrow min}$	$l_{link \rightarrow max}$	d	d	$2d$	$2d$	$2d$	$(N/2-1)d$	$2d$	$2d$
$n_{hop \rightarrow min}$	$n_{hop \rightarrow max}$	2	$2N-1$	1	$N-1$	1	2 to $N-1$	2	$N+1$
$\overline{n}_{hop \rightarrow edge}$		$(N+1)/2$		$(N+2)/4$		2 to $(N+2)/4$		$(N+4)/4$	
$N_{links \rightarrow bisec}$		$2N$		N		$N^3/16$		$4N$	

2.1.3 Virtual Channels and Multiple Networks

Traffic management is very important in NoCs to reduce the negative performance impact of data flow concurrency and to enable higher utilization levels. The interference of concurrent packet traversals through the NoC results in backpressure and early network saturation. As solution to this problem, three major traffic management mechanisms exist [7], [9], [10], [28], [30], [74]:

- ***Adaptive routing algorithms*** target the reconfiguration of paths used by the concurrent data flows. Interference of data flows is reduced by spatial separation of their traversal through the same NoC. Adaptive routing is discussed in Section 2.2.3 (pp. 49-55) and furthermore composable with the both remaining solutions below.
- ***Virtual channels*** use a logical separation of the shared NoC resources for concurrent data flows. Thus, VCs integrate temporal assignment of the shared resources along a path to dedicated data flows and enable a multiplexing of the available bandwidth.
- The integration of multiple NoCs as ***multi-network solution*** or ***channel slicing***. This implies a full physical/spatial separation of concurrent data flows through their assignment to independent networks.

Section 2.1.1 (pp. 31-40) already provided the basics of the integration of VCs as part of the router pipeline. The additional VCA stage processes the allocation of resources at the VC-level, which

virtually splits up the physical NoC into multiple logical ones that coexist. The available bandwidth of the links will be allocated and multiplexed by it for these logical networks. For that purpose, the packets inside the NoC need to be assigned to independent buffer structures that correspond to a specified VC. This can be realized via logical segmented monolithic buffers with multiple independent read-/write-ports or physically separated buffers. Thus, concurrent packets inside different VCs have minimized interferences and the propagation of backpressure is limited to singular VCs instead of affecting the complete physical resources. The VC solution applies resource sharing for the links and the crossbar. The resulting constraints are that the physical available bandwidth does not scale up with the number of VCs, an additional pipeline stage per hop has to be integrated, network structures as well as algorithms are unified for all of the logical networks, and the transported data in each VC is bound to the same data word formats. The integration of virtual networks is mainly purposed for ***traffic management*** in ***quality-of-service*** (QoS) scenarios [122], the minimization ***busy-waiting-cycles*** in links caused by ***head-of-line blocking*** [54], and ***deadlock avoidance*** for concurrent data flows inside the NoC [60]. The assignment of packets to VCs can be realized as static or dynamic mechanism [123].

Contrary to the shared resource solution via VCs, the implementation via ***multiple physical networks*** offer full ETE traffic separation and the available bandwidth will potentially scale up with the number of integrated networks. Thereby, ***channel slicing*** refers to a constrained version of the multi-network approach, because the total bandwidth remains equal to the original single NoC implementation [20], [124]. In principle, a ***multi-network*** solution is not bound to this bandwidth constraint and offers the full freedom of customizability as well as independence of its sub-networks. The proposed traffic management scenarios do not differ from those the VC approach is used for, but the ***customizability*** at design- and run-time of each single network for its dedicated traffic classes minimizes the necessity for tradeoffs. The sub-networks can vary in their structures, data widths, algorithms, buffer sizes, clock rates, voltage levels or even apply VCs by their own [122] if it is beneficial. The degree of customization in the sub-networks increases with their specialization to dedicated traffic classes, whereas monitoring and control mechanisms in CMP are typical areas with a high degree of customization [60], [68], [72], [73], [125]. The assignment of data flows to a sub-network is realized at the ETE level and most commonly static.

Comparisons between both approaches performed by in [126]–[128] with focus on CMP show, that the multi-network solution supports valuable performance and power benefits for run-time managed systems that are able to exploit the full advantages of the physical independence of sub-networks. Furthermore, the routers will be simpler and the reliability is increased. One practical example for the application of the multi-network approach are the processors of Tilera, which integrate up to five

physical sub-networks [31] assigned to dedicated traffic classes to improve the overall performance. In [129], a dual mesh-based NoC, called CCNoC, is proposed to improve cache coherency. The sub-networks in the CCNoC are specialized for request and response traffic. In [53], a similar solutions for the separation off bandwidth- and latency-sensitive traffic classes via two mesh-based sub-networks is proposed. In addition to the differences of the design parameter, the latency-sensitive sub-network is able to operate at a tripled frequency. In [122] as well as [130] the utilization of two independent networks for the separated communication of data and control information is applied.

2.2 ALGORITHMS AND STRATEGIES

The basic components and structures of the NoC have to implement a composable set of functionalities to provide the needed protocol stack for concurrent ETE data transfers. This section discusses the different algorithms and techniques for this layered composition of ***switching, flow control*** and ***routing*** in the NoC. The ***switching*** specifies the principle technique of resource multiplexing and segmentation/packetization of data flows in the network. The ***flow control*** defines the resource allocation and management on top of the switching and is tightly interrelated with it. Furthermore, it covers the error handling and can be spread over different system layers according to the ISO/OSI model. The ***routing algorithm*** specifies the functionality for the selection of the subset and order of NoC resources to establish a communication path in the NoC. The interplay of this composition defines the data transfer capabilities and has to protect the network from ***starvation***, ***deadlocks*** as well as ***livelocks*** to guarantee robust ETE data transfers between IP cores. Thereby, these critical phenomena can be specified as follows [7], [9], [10], [60]:

- ***Starvation*** is the inability of a data packet for the allocation of required resources to proceed along its path through the network. It is a typical result of unfair allocation policies that enable unlimited preemption of packets.
- ***Deadlocks*** result from cyclic dependencies of resource allocations from concurrent data flows inside the network, where the data flows mutually block the required resources and thus any progression. Such deadlock can happen at the packet-, message- or transaction-level and might further impact the complete NoC as a kind of infarct.
- ***Livelocks*** are situations where data transfers proceed through the network, but do not reach the defined destination and move around unterminated. They result most typically from unrecognized side effects of adaptive run-time traffic management that allows misrouting.

Most commonly, NoCs will be designed and verified very carefully to avoid these critical phenomena, because the integration of mechanisms to detect and resolve them comes along with undesirable hardware overhead as well as performance limitations.

2.2.1 SWITCHING

The selected switching technique of the NoC defines the basic resource multiplexing and determines additional requirements for buffering, path setups and resource allocation flows. The multiplexing technique can be classified into two basic approaches, called ***circuit switching*** and ***packet switching*** [7], [9], [10], [25], which differ as follows:

- In the ***circuit switching*** approach, a physical ETE path between source and destination is established before, reserved during the complete time of the data transmission, and released afterwards. Each communication necessitates an allocation and de-allocation of the corresponding NoC resources and concurring data flows are blocked. The complete message is transmitted during a single session. The path allocation flow has a high delay and the blocking of the path resources prohibits the efficient exploitation of the inherent parallelism in the NoC for concurrent communication flows [9], [25].
- In the ***packet switching*** approach, messages are segmented into packets and transmitted as independent entities. The resource allocation works at the HTH level and only the currently utilized resources are blocked. Thus, full parallelism of the network can be exploited and dynamic adaptation to packets along the path at the HTH level becomes applicable. Therefore, buffers along the paths are required to store full packets or packet segments. NoCs are most commonly implemented as packet-switched infrastructures [25], [75].

For the integration of ***packet switching*** with applied HTH flow control three basic solutions exist, which work as follows:

- ***Store-and-forward*** *(SAF)* ***switching***: The complete packet must be received by a router, before the routing logic calculates the corresponding output for the downstream. Furthermore, the requirement for the packet-forwarding is that the packet fits the available buffer space at the adjacent router or it has to wait. This packet-level allocation at the HTH flow control comes along with increased delays and buffer sizes. In minimum, the buffers must provide enough space to store a packet at maximum allowed size.
- ***Virtual cut-through*** *(VCT)* ***switching***: Unlike SAF, in VCT switching the output calculation and the forwarding of flits to the adjacent router is performed without the necessity of a completed packet reception at the current router. This improves the delay of the packet traversals in comparison to SAF. But the resource allocation of the HTH flow control still works at the packet-level and the forwarding of packets is stalled until the downstream router preserves enough buffer space for the complete packet.

- ***Wormhole** (WH) **switching***: This technique improves VCT switching by the reduction of the HTH flow control requirements to the flit-level. Thus, resource allocation at the buffer of the downstream router operates flit-by-flit and forwarding starts immediately if a single flit can pass. WH switching enables smaller buffer sizes and reduced packet delays, but on the other hand single packets might block multiple buffers and links simultaneously (what is prohibited in SAF as well as VCT). The combined benefits of reducible hardware costs and optimized packet delay makes WH switching the most popular choice for NoC infrastructures [25], [28].

This overview of the different switching techniques for NoCs indicates their major impact on the flow control characteristics and the hardware costs regarding the buffering requirements.

2.2.2 FLOW CONTROL

Efficient flow control is important for the performance of parallel communication infrastructures with multiplexed shared resources and must be applied at different system abstraction levels to guarantee undisturbed data traversals of concurrent data flows [7], [9], [10], [25], [28], [30], [75]. The selected switching technique defines the basic flow control constraints regarding the resource allocation granularity and the spatial scope of blockings. The flow control itself can be classified by different aspects, such as the spatio-temporal scope, system layer, and resource management policies. Thereby, spatio-temporal scope specifies the composition of mechanisms at the HTH and the ETE level. Most typical, the HTH flow control covers a wide range of the required functionality itself, but if additional EDAC mechanisms or bandwidth management at the ETE level are applied protocols for such interactions between source and destination (like retransmission or QoS contracting) must be integrated. The system layer aspect specifies the scope of communication entities for the flow control, such as messages/transactions, packets or flits. The work at hand only considers packets and flits, but especially for deadlock treatment or error handling at the application layer, message and transactions are important [40]. The resource management policies branch into two major classes called ***bufferless*** and ***buffered flow control***. ***Bufferless flow control*** focusses the avoidance of packet stalls caused by buffer capacity misses. Therefore, packet dropping with retransmission or packet deflection (hot-potato) with optional misrouting is applied to provide workarounds for such situations at the costs of possible retransmission overhead or packet-reordering [24], [28], [131], [117], [132], [133]. ***Buffered flow control*** provides dedicated mechanisms for buffer allocation and resource management at the HTH level. The different proposals mainly differ in the management of buffer resources and the ability to handle errors.

- ***Credit-based flow control***: The adjacent downstream router propagates the number/level of free buffer capacities back to the forwarding router as credits. Thus, the forwarding router

knows the allocable buffer space ahead-of-time and updates this information continuously. This flow control mechanism is optimized for adaptive distributed routing algorithms to prevent stalls. It requires additional wiring and detection logic for the information exchange.

- ***Stall-and-go*** *(or* ***xon/xoff****)* ***flow control***: This is a single wire version of the credit-based mechanism. The downstream router signalizes to the forwarding router via a single bit if free buffer capacities below a given threshold are available (*go/xon*). Otherwise, the buffer is marked as full (*stall/xoff*).
- ***ACK/NACK flow control***: This mechanism extends the flow control with signaling states to initiate retransmissions. After data reception at the adjacent router, the transfer of correct data is acknowledged (*ACK*) or incorrect data is detected and non-acknowledgement (*NACK*) is signalized to trigger a retransmission. This flow control extension is typical for error detection with retransmission, but due to the resulting traffic overhead through retransmissions NoCs typically apply forward error correction to avoid this additional traffic up to a defined failure rate [40].

The traversal of flits along the crossbar and link is managed by ***req/ack-handshaking*** to control the pipeline flow with two wires. New flits initiate a request (*REQ*) and after the flit passed the pipeline stage the reception is acknowledged (*ACK*) by the next stage. The required time for that procedure is typically two clock cycles and can be combined with simple ***stall-and-go*** flow control.

2.2.3 ROUTING

The routing algorithm is a key issue for the design of the NoC regarding performance as well as reliability of the CMP operations. It determines the paths along the network resources for the data flows of computational resources. Hence, the freedom of the route selection sets the limits for the exploitation of the inherent parallelism in the NoC and the ability to avoid interferences of concurrent data flows. These features make the routing as important as the application mapping of workloads for the operation of the CMP. Both mechanisms are highly interrelated, because they have the ability to mutually limit or expand their search space for optimized solutions. This section discusses the basic features of routing algorithms, provides a general taxonomy and a representative selection of state-of-the-art solutions. Thereby, the focus is set on routing algorithms for two-dimensional mesh-based topologies with applied wormhole-switching, which represents the majority of published solutions in the NoC domain [25], [28], [30] and the background of the work at hand.

As mentioned during the introduction of this chapter, the routing algorithm has major impact on the occurrence of ***deadlocks*** and ***livelocks*** in NoCs. While livelocks are tightly related to dedicated adaptive routing algorithms, deadlocks are a problem that must be tackled in general for fully or

partially adaptive routing algorithms. Thereby, ***deadlock avoidance*** is the preferred choice and has to be verified at design-time. To break cyclic dependencies that result in deadlocks, two major design principles exist, which can be verified under consideration of the turn model [7], [9], [10], [134]. Turns describe the spatial direction changes of paths along the network and the total variety of allowed turns will be used to evaluate the ability of paths to form cyclic dependencies. If such situations are possible, they can be solved as follows:

- The ***removal of turns*** restricts the variety of turns to break cyclic dependencies. The exploitable path diversity is limited by the removal of turns (i.e. all yx-turns) and the resulting algorithm is deadlock-free (i.e. simple xy-routing). This strategy reduces the adaptivity of routing algorithms to a subset of paths and has minimal impact on the hardware costs.
- The ***separation of turns*** allows retaining the maximum exploitable path diversity. Thereby, critical turns are assigned to separate physical or logical sub-networks. Hence, the implicit resource sharing of virtual channels or physical separation of multi-networks break up cyclic dependencies. The drawback of this solution is the increase of the hardware costs. The most prominent case to separate xy- and yx-turns via the integration of two VCs.

Beside this general requirement of adaptive routing algorithms for the deadlock-free exploitation of path diversity, a wide set of features/properties exist and the proposed state-of-the-art solutions differ in the composition of them. The design target is to gain run-time flexibility and optimization of traffic management for performance/throughput/fault-tolerance at feasible hardware costs, power dissipation and complexity. The most important features for the ***classification of routing algorithms*** are as follows:

Algorithmic Structure: The structure of the routing algorithms specifies the granularity of path adjustments in the NoC. In ***centralized*** algorithms, a single entity inside the NoC or dedicated regions processes, adapts and assigns paths. In ***source-based*** solutions, the source node performs path selections for all of its destinations at the ETE level. ***Distributed*** routing algorithms run in each router and adjust the path at the HTH level. ***Hybrid*** algorithms realize a mix such as a ***hierarchical*** version of centralized routing for local and global paths.

Path Evaluation: The evaluation of path adaptations or routing decisions can be performed ***online*** and ***offline***. Thereby, *online* algorithms operate at run-time and evaluate path updates ***continuously*** or in periodic/sporadic time ***intervals***. *Offline* algorithms evaluate optimal path compositions or routing decisions for fixed workload sets in advance and will only configure the calculated adjustments at run-time to match the current workload case.

Information Scope: Adaptive routing algorithms evaluate their adjustments on a defined information base that most commonly correlate with the algorithmic structure as well. The simplest way considers ***probabilistic*** adaptation patterns. Thereby, the packetized traffic is spread over the available paths via probabilistic distributions without feedback. Alternatively, ***monitoring*** structures will be integrated to gather real status information with specific ***spatial scopes*** on ***local***, ***regional*** or ***global*** traffic. The routing algorithm evaluates this information to determine optimized path selections regarding communication performance, fault tolerance or power dissipation.

Virtual Channels: As discussed above, deadlock avoidance in adaptive routing algorithm may come at the price of VC integration, which increases the hardware costs and deepens the router pipeline. This overhead might grow further if traffic monitoring at the VC level must be applied for optimal adaptivity. Thus, VCs are an important feature regarding the cost side.

Hop Distances: The range of exploitable hop distances is a major feature to describe the degree of adaptivity in routing algorithms. ***Minimal*** or ***shortest path*** routing only exploits paths between source and destination that reduce the remaining hop distance during packet forwarding by at least one hop. ***Non-minimal*** routing algorithms are not constrained by this condition and ***misrouting*** of packets is supported. This further necessitates the algorithmic verification or integration of additional mechanisms for livelock treatment. ***Minimal routing is livelock-proof by design***. Figure 2.10 illustrates these different path classes.

Path Diversity: This feature defines the number of available path adaptations as well as the size of the search space for their exploration. Thereby, ***complete*** exploitation of all available paths offers the highest flexibility and complexity. ***Partial*** path diversity is constrained to a subset of independent paths and most commonly exploits major alternatives (i.e. xy or yx). The limitation can result from deadlock avoidance through turn removal as well. ***None*** path diversity offers a single path per source-destination-pairing. Such algorithms are called ***static*** or ***deterministic*** as well.

Implementation Strategy: The routing mechanisms and traffic evaluation can be implemented solely in ***hardware*** (HW) or as a ***hardware/software-combination*** (HW/SW).

Deterministic dimension-ordered routing algorithms, such as *xy-* or *yx-routing*, are the preferred choice in mesh-based NoCs due to their simplicity [16], [25], [31]. They deliver packets in-order along pre-known paths, reduce the search space for the exploration of the application mapping, and data flow interferences can be reduced through the integration of VCs. But the static characteristics also prohibit the optimal utilization of inherent parallelism in the existing network and fall short in presence of faulty links/routers. Alternatively, the *integration of adaptive routing algorithms* offers higher throughputs, lower packet delays, and possible workarounds for several network faults [28], [30], [97], [133], [135]–[137]. The resulting feature composition represents a tradeoff between

hardware overhead, complexity, and possible performance gains. Thereby, *distributed minimal routing algorithms* with *full* or *partial adaptivity* are the most popular approach in state-of-the-art literature. They adapt the path for each packet at the HTH-level and differ in their information scope for the routing decisions. Continuous regional traffic monitoring of buffer or crossbar utilization is proposed by *Ebrahimi et. al.* called CATRA [63], *Ma et. al.* called DBAR [62], *Gratz et. al.* called RCA [61] or *Chen et. al.* called RAIR [54]. The *local* traffic situation is considered by *minimal partial adaptive* solutions like odd-even [138] of *Chiu et. al.*, DyAD [139] of *Hu et. al.* or west-first/ negative-first/north-last adaptations deviated from the work of *Glass et. al.* in [134]. *Local* traffic monitoring with *non-minimal* hop distances is used by *Ebrahimi et. al.* in LEAR [64], while similar proposals like DyXY [140] of *Li et. al.* as well as EDXY [141] of *Lofti-Kamran et.al.* are limited to *minimal* hop distances. The HPRA [142] approach of *Kakoulli et. al.* uses statically-assigned regional neural-networks to predict traffic situations with feedback from local buffer utilizations as *hybrid* mechanism and is constrained to the *source-based* selection between *minimal xy-* and *yx-routing*. The neural-networks are trained *offline* to detect hotspot pattern. A representative with *probabilistic* and *source-based* selection of between *minimal xy-* and *yx-routing* is the O1TURN algorithm [143], [144]. It offers near optimal mean throughput results without additional costs [143].

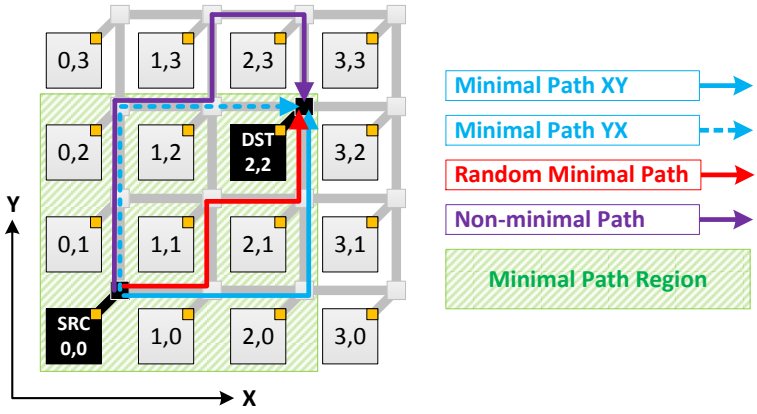


Figure 2.10: Path categories for routing algorithms in mesh-based Network-on-Chip

Contrary, the ATDOR approach of *Manevich et. al.* in [60], [125] show that *centralized online-based adaptation* of *minimal xy/yx-routing* is a cost-effective solution (up to 12x12 meshes) that outperforms distributed algorithms like O1TURN or RCA. *Source-based adaptive* routing strategies with *queue-formatted addressing* are applied by *Mubeen et. al.* in [145] and *Kim et. al.* in [146]. Thereby, *Kim et. al.* in [146] focus on *minimal paths* based on *address queues* that will be discovered via requests *online* by the source itself with the *regional* scope on the possible paths to the destination. *Mubeen et. al.* in [145] apply *direction-based path queues* with *offline path evaluation* using *global traffic* information, support *non-minimal* routes and *complete* exploration of the *path diversity*. Another *offline path evaluation* with *global communication awareness* is proposed by *Palesi*

et. al. called APSRA [98], [147]. This solution generates sets of routing tables with *partially adaptive minimal* route selection features for dedicated application workloads. For the integration of *multicast* capabilities the approaches of *Carara et. al.* in [148] and *Samman et. al.* [149], [150] use *address-based path queues* to communicate with multiple destinations along the defined path [148] or realize tree structured branching [149], [150] to communicate with defined regions. While the proposal in [148] applies *source-based minimal* routing the tree-based solution in [149], [150] works with *distributed partially adaptive minimal* routing. The complete feature composition of the presented state-of-the-art routing selection in mesh-based NoC can be obtained from Table 2.2. Routing solutions with a more sophisticated focus on increased fault-tolerance are referred in the works of *Radetzki et. al.* in [39] and do not represent the major scope of the presented selection.

Table 2.2: Feature composition matrix for the selected state-of-the-art set of adaptive routing algorithms

FEATURE	OPTIONS	EXAMPLES									
Algorithmic Structure	Centralized							■			
	Source-based						■		■	■	
	Distributed	■		■	■	■					■
	Hybrid		■								
Path Evaluation	**Online Continuously**	■	■	■	■	■	■			■	■
	Online Interval							■			
	Offline		■						■		■
Information Scope	Probabilistic						■				
	Local Traffic			■	■	■					
	Regional Traffic	■	■							■	
	Global Traffic							■	■		■
Virtual Channels		**≥2**	**≥2**	**≥2**		**≥2**	**≥2**	**≥2**	**≥2**	**≥2**	
Hop Distance	**Minimal/Shortest Path**	■	■		■	■	■	■		■	■
	Non-Minimal/Misrouting			■					■		
Path Diversity	**Complete**	■		■		■			■	■	■
	Partial		■		■		■	■			
	None										
Implementation Strategy		**HW**	**HW SW**	**HW**	**HW**	**HW**	**HW**	**HW**	**HW SW**	**HW**	**HW SW**
		[63] [62] [61] [54]	[142]	[64]	[138] [139] [134]	[140] [141]	[143] [144]	[60] [125]	[145]	[146]	[98] [147]

The presented state-of-the-art overview on routing algorithms indicates that the preferred choice for run-time traffic management in mesh-based NoC is a hardware-centric closed-loop implementation. Furthermore, the majority of these solutions act via distributed street-sign adaptation strategies at the packet level with non-cooperative routing decisions at the source or routers along the path. The main purpose for such design decisions can be found in the reduced complexity as well as the autarchic operation of the traffic management. But these concurrency and distribution characteristics prohibit global optimization as well as deterministic coordination at the system level. Therefore, the ATDOR approach in [60], [125] presents an attractive starting point to investigate an advanced run-time traffic management and adapt it to a flexible hardware-software approach.

3 MULTI-PORTED RESOURCES IN NETWORK-ON-CHIPS

As part of the thesis objectives in Section 1.1, the first step of the investigations focusses on the composition of topology and routing, because it outlines the basic capabilities for traffic management and physical implementation characteristics. The background on topologies in Section 2.1.2 (pp. 40-44) shows a wide range of proposed state-of-the-art solutions on NoC topologies focus the optimization by combining adaptations of router radix (N_P), router-to-router connectivity (N_{RC}), and concentration. In this context, the multi-ported solution proposed in this chapter can be outlined as follows:

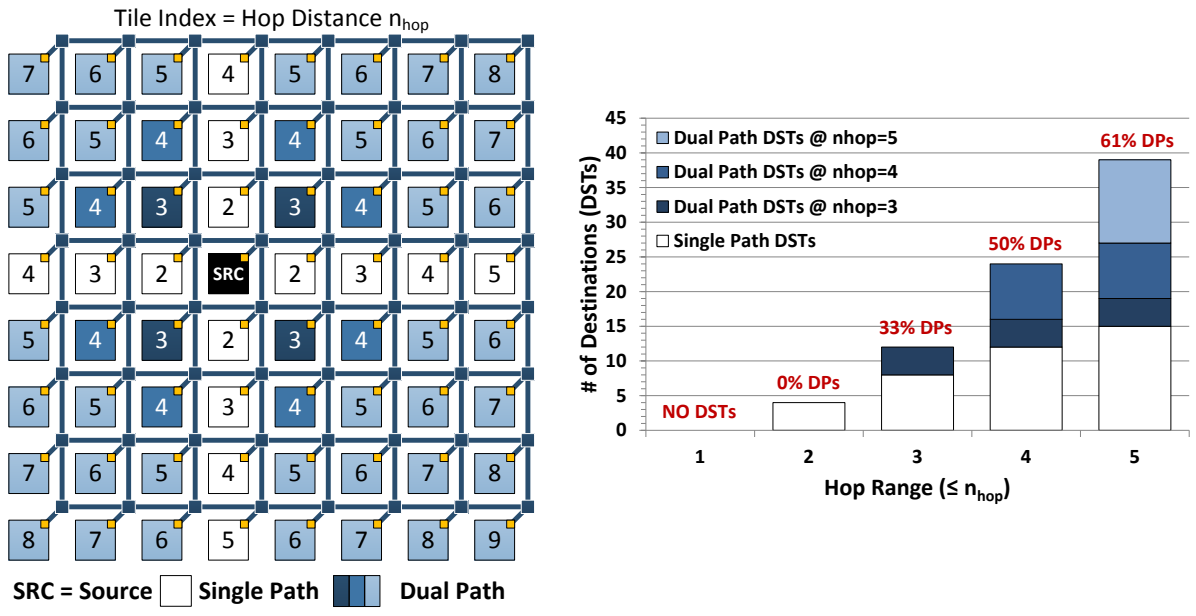
- The aspect of the physical implementation highlights that layout matters and the 2D-Mesh benefits most due to its high regularity and simplicity [151]. In different publications, the 2D-Mesh represents a reference case for new topologies and a comparison is performed under the assumption of equal clock rates or on a cycle base [13], [14], [20], [112], [113], [116], [118], [119], [152], [153]. But under consideration of layout-aware timing [81], [82], [114], [154]–[157] as well as the reliablity [43] the 2D-Mesh provides a good entry point.
- One key idea of the proposed multi-ported approach is that the connectivity characteristics of the resources (IP cores) offer great potential for optimizations, while the inherent 2D-Mesh advantages regarding simplicity as well as regularity can be maintained.
- Especially, the number of spatially independent terminal interfaces to the neighboring router of each IP core (N_{NTPC}) offers modifications that provide holistic improvements for performance, reliability, and flexibility at reasonable costs.
- The availability of multiple independent NIs, connected to different routers, enables multi-path communication by the recombination of endpoints and the generation of fully independent paths. Thereby, multi-networks as well as multi-VC approaches or router level optimization [7], [90], [158]–[162] are orthogonal to this basic topology modification.
- The multi-ported topology is suited for source-based routing to exploit the multi-path communication capabilities.

The presented topology in this chapter focusses on this dedicated modification and it is shown that it combines the expected characteristics. Before, the following introductory analysis outlines the different potential advantages of this regarding the major design- and run-time aspects (performance, costs, and reliability).

ASPECT 1 - PERFORMANCE: Workload applications can be classified as bandwidth- or latency-sensitive regarding their performance requirements. While bandwidth sensitivity needs NoCs with

optimized data and system I/O throughput features, latency sensitivity requires modifications to optimize end-to-end communication delays. Furthermore, the optimization for one class does not necessarily come along with improvements for the other one [53]. But the integration of multi-ported resources enables the provision for optimizations heading both performance criteria, if the corresponding benefits regarding the spatio-temporal locality of communication are exploited accordingly. The potential improvements can be summarized as follows:

- Multiple independent access points increase the potential data reception and injection rates of the corresponding resources [132]. Furthermore, traffic hotspots/bottlenecks at critical resources as well as the possible blocking infarcts in the NoC can be relaxed.
- The assignment of traffic flows to the NIs under consideration of spatio-temporal metrics can help to resolve interferences and reduce the impact of backpressure on queuing delays at the traffic source. Furthermore, the hop distance between source and destination can be reduced.



(a) Dual path and hop distribution in 2D-Mesh

(b) Dual path fractions over hop range in 2D-Mesh of (a)

Figure 3.1: Exemplary single and dual path distribution inside a mesh-based Network-on-Chip with xy/yx-routing

- Adaptive multi-path routing will benefit from multiple NIs if the additional injection/ejection points can be utilized to create spatially independent path alternatives for a greater number of source-destinations-pairs. Figure 3.1 (a) and (b) illustrate the common dual path situation for the basic 2D-Mesh with applied xy/yx-routing. Figure 3.1 (a) shows that xy/yx-dual-paths (blue resources) only exist, if both coordinates (x and y) of source and destination differ from each other. The labels at each tile specify the hop distance for the exemplary case of the communication with the SRC-labeled core. The stacked bar-chart in Figure 3.1 (b) indicates

that 2D-Mesh destinations with available xy/yx-dual-paths become the dominating fraction at a minimum hop range of 5 (> 50%). Under consideration of locality optimized application mapping policies, this fraction should be increased for lower hop ranges and all possible coordinate constellations. Providing multiple NIs is a potential measure to achieve this.

The above listed issues for potential performance improvements through an increase of resource connections (N_{NTPC}) target both delay fractions along the network paths (Post- and Pre-NI as discussed in Section 2.1.1 see Figure 2.8 at p. 39) through the combined handling of backpressure, data flow interferences and hop distances. Coincidently, the underlying regular 2D-Mesh topology as baseline for the intended modifications will support that potential improvements from the logical perspective should be visible in the physical chip design as well. A topology comparison based on the data in Table 2.1 (see p. 44) and the results of the physical evaluation for the basic NoC component (see Appendix pp. 167-175) underline such expectation.

- The reduction of the logical hop distance (see $\bar{n}_{hop \rightarrow edge}$ and $n_{hop \rightarrow max}$) in Fbfly, CMesh and FTorus doubles the link's minimal length (see $l_{link \rightarrow min}$) compared to the 2D-Mesh. Hence, in worst case the maximum frequency is halved and doubles the static delay per hops. Alternatively, longer links must be pipelined at the price of additional stages per link traversal. Moreover, the geometrie of IP cores impacts the link length in Fbfly, CMesh and FTorus with a factor of two. The gathered data for such link relations show that even for short links (SL) as present in the 2D-Mesh the link delay outweigh the achievable router delay by a factor of 1.7 at 45nm and by a factor of 2 at 22nm for the slowest router configuration (8P-2VC) without the additional consideration of PVT phenomena. Hence, the selection of the 2D-Mesh as baseline topology is the right approach regarding achievable frequencies.
- The main advantages of the CMesh over the 2D-Mesh are reduced hardware costs and hop distances by around 75%. But as result, the traffic has to be handled by the remaining hardware structure, which increases the traffic load per component and decreases the path diversity. The 2D-Mesh provides a regular structure with a balanced mix of path diversity and options to spread the traffic.
- The Fbfly consists of a core network that equals the CMesh and is enhanced with additional physical express links for improved router connectivity. The exploitation of short hop distances and path diversity in the Fbfly necessitates the integration of at least two VCs and adaptive routing mechanisms. Further, the complexity growth of the network and its two dimensional layout compatibility limits its scalability [14], [104], [114], [152], [153], [163].

ASPECT 2 - COSTS: An increase of the resource connectivity via additional NIs (N_{NTPC}) impacts the cost side at design- and run-time. Based on the NoC component evaluation in the Appendix (see pp. 167 - 175) the main cost issues are as follows:

- Multiple interfaces per resource increases the radix ($N_P = N_{PR} + N_{PC}$) of the routers via additional router-to-core ports (N_{PC}). But the 2D-Mesh topology for global communication remains and the number of router-to-router connections (N_{PR}) will not be affected. As the data for different router configurations show, the additional hardware efforts for each router remain feasible at the global CMP level at below 1.5% of the total anticipated tile area (see Figure I.4 p. 170), whereas the buffers and the crossbar will take a share of around 94% and more regarding the area cost of the router. The hardware costs for the integration of more NIs mainly depend on the complexity of ETE traffic management protocols. For the simplest scenario, each NI implements functionality similar to a router port and offers additional ETE flow control capabilities.
- From the run-time cost perspective, increased power dissipation and degraded frequency margins will result from a higher router radix. As shown in Figure I.3 (see p. 169), the step from five to eight ports (at 5/8P-1VC) reduces the maximum frequency by around 10%, while the impact of the increase of virtual channels (at 5P-1/2VC) would lead to a higher frequency reduction. But such orders of frequency reduction will not impact the achievable NoC timing due to the link delay dominance observed for the evaluated configurations. The run-time impact on the power dissipation depends on the consolidation of the above mentioned performance benefits and the operating conditions. The power dissipation of the router configurations is evaluated for a constant traffic throughput scenario (see results depicted in Figure I.5 p. 171 and Figure I.6 p. 172). The dominant sources of power dissipation are the buffers and crossbars of the NoC routers. Furthermore, the static power dissipation covers a fraction of around 60% at 45nm and around 80% at 22nm from the total power per NoC router. Furthermore, the static power dissipation is impacted exponentially by the operating temperature. The expected decrease of mean hop distances in communication and possible load balancing through adaptive multi-path routing can help to reduce the mean load per router/link. Thus, dynamic as well as static power dissipation (via lowered operating temperature) will benefit and this may partially compensate the additional power dissipation. Furthermore, the total workload processing can be accelerated if the applications have the ability to fully utilized reduced packet delays as speed-up.

ASPECT 3 - RELIABILITY: The above mentioned mechanisms for potential performance gains through multiple spatially independent injection/ejection ports per resource can be exploited in a similar way

to improve the reliability characteristics of the NoC. The selected topology affects the reliability of the NoC and its ability to degrade gracefully. The exploitable potential of path diversity can help to react in the presence of transient as well as permanent faults and to tackle wear-out in the long term perspective. Alternative paths that traverse the NoC via independent components can be used to circumvent permanent faults in links/router or rebalance the historical loads of all components to equalize possible wear-out effects. Furthermore, the number of components and their connectivity determine the impact of faults and thus the vulnerability of the NoC. Permanent faults in links can disconnect regions inside the NoC or lead to isolated IPs in a worst case scenario. Permanent faults in routers will isolate those IP cores the affected routers provide the only terminal for. Transient faults in links and routers depend on the intensity of PVT variations, the operating conditions of the NoC, and the increase of wear-out through activity-driven aging/stress. Under consideration of the reliability, the applied concentration in CMesh and Fbfly increases the impact of faults on the functionality of the CMP. The CMesh reduces router count, link count, and path diversity. Thus, permanent faults in links and routers will affect the communication of more IP cores and the NoC provides less workarounds to solve such situations. Moreover, the concentration of the traffic on fewer components will accelerate their wear-out due to stress effects and higher rates of transient faults. The Fbfly has the same concentrated structure of routers, but its path diversity can benefit from the additional links if the applied routing algorithm supports a high degree of exploitation. The problematic aspect for workarounds via alternative paths in the Fbfly might arise from the reduced router count, because it decreases the availability of independent paths as well. Additional to the path-level influence of the topology selection, the resulting complexity of the router and links matters, because the more critical timing paths exist inside them, the more vulnerable they are for transient errors and wear-out based degradation of the critical paths [43]. This will be the case for the CMesh and FTorus as well.

A simplified analysis for network paths is performed and shows the resulting differences for basic path setups in a 2D-Mesh topology. The needed fundamentals of the applied reliability modelling can be obtained from *Koren et.al.* [164]. The reliability $R_i(t)$ of a specific component i over a dedicated time period t can be approximated via Equation 3.1. Thereby, $\lambda_i(t)$ describes the expected failure rate (failure over time) of this component in the anticipated time period. The reciprocal of the failure rate λ is also known as mean-time-to-failure (MTTF). The plotted curve of the failure rate over the component's lifetime has a typical shape of a bathtub [36], [43], [164]. Thereby, the useful lifetime represents the middle section and is approximated with a constant failure rate. Furthermore, the future reliability trend of CMOS indicates higher failure rates as well as accelerated wear-out defects, which results from the increased hardware vulnerability through technology scaling and higher PVT stress impacts [36], [43]. A path inside the NoC is represented by a series of N components (NIs, links

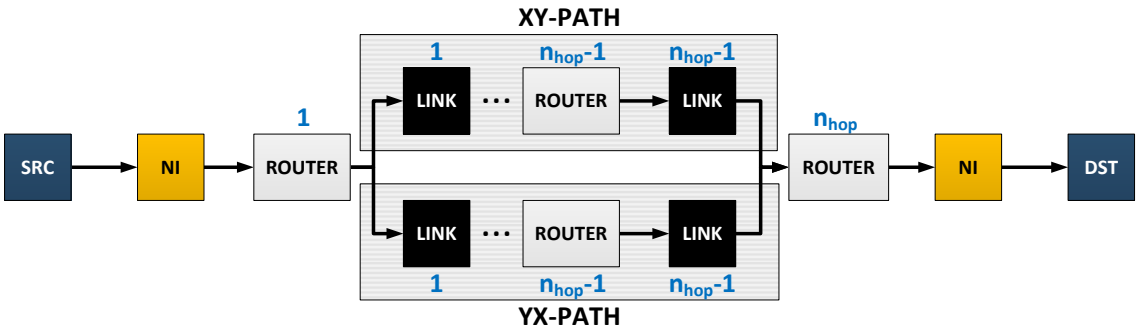
and routers) and its reliability can be modelled as $R_S(t)$ via Equation 3.2. If the system offers multiple (P) parallel options, which is typical for a multi-path situation, the overall reliability $R_P(t)$ can be modelled via Equation 3.3. In the next steps Equation 3.1 to 3.3 are applied to model specific path options inside the NoC and to enable a simplified comparison.

$$\boldsymbol{R_i(t)} = e^{-\lambda_i(t)} \tag{3.1}$$

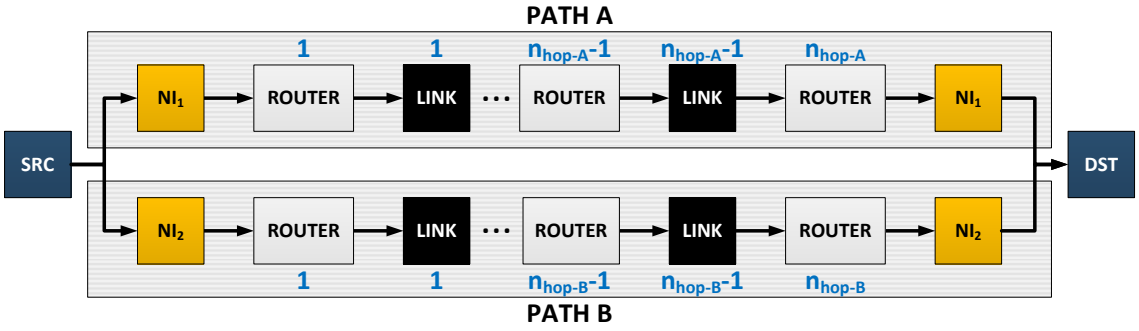
$$\boldsymbol{R_S(t)} = \prod_{i=1}^{N} \mathrm{R}_i(t) \tag{3.2}$$

$$\boldsymbol{R_P(t)} = 1 - \prod_{i=1}^{P} (1 - \mathrm{R}_i(t)) \tag{3.3}$$

(a) Single Path



(b) Dual XY/YX-Path with common endpoints



(c) Dual A/B-Path with separate endpoints

Figure 3.2: Applied path options in mesh-based Networks-on-Chip for the simplified reliability analysis

Figure 3.2 illustrates the three path options between a source (SRC) and a destination (DST) resource (with a hop distance of n_{hop}) inside the scoped NoCs. Figure 3.2 (a) contains the normal single path situation of the 2D-Mesh with static minimal xy-routing and a minimum distance of two hops ($n_{hop} \geq 2$). Figure 3.2 (b) depicts the dual path situation inside the 2D-Mesh if adaptive minimal xy/yx-routing is applied. Thereby, both paths share the same injection/ejection routers and NIs, whereas the

remaining intermediate components are separated into the yx- and the xy-segments. Inside the 2D-Mesh, such dual xy/yx-paths become available at minimum hop distances of three ($n_{hop} \geq 3$). Finally, Figure 3.2 (c) illustrates the dual A/B-path situation if spatially independent injection/ejection terminals ($NI_{1,2}$) are available.

As part of the reliability model it is assumed that all components of a path have the same constant failure rate and therefore the component reliability can be simplified to R ($\mathrm{R}_{NI,i} = \mathrm{R}_{Link,i} = \mathrm{R}_{Router,i} = R$). This results in simplified expressions and should not affect the general tendencies of the analysis results, because it is applied to all considered path situations and the targeted mesh-based topologies have similar component setups. The single path of Figure 3.2 (a) contains two NIs, (n_{hop}-1) links and n_{hop} routers. Through the application of Equation 3.2 to this series of components the total path reliability R_{PATH} can be obtained via the Equation 3.4.

$$R_{PATH} = R_{NI} \cdot R_{Link} \cdot R_{Router}$$

$$R_{NI} = \prod_{i=1}^{2} \mathrm{R}_{NI,i} \qquad R_{Link} = \prod_{i=1}^{n_{hop}-1} \mathrm{R}_{Link,i} \qquad R_{Router} = \prod_{i=1}^{n_{hop}} \mathrm{R}_{Router,i}$$

$$\Longrightarrow \boldsymbol{R_{PATH} = R^{2n_{hop}+1}} \tag{3.4}$$

The dual xy/yx-path situation of Figure 3.2 (b) represents a series of four endpoint components (two NIs and two routers) and the aggregated component of the two parallel xy/yx-path segments (with (n_{hop}-1) links and (n_{hop}-2) routers at each). Its resulting path reliability $\mathrm{R}_{PATH \rightarrow XY/YX}$ can be obtained via the Equation 3.5. Thereby, it is assumed that the xy/yx-path segments do not share any resources and have an equal hop distance.

$$R_{PATH \rightarrow XY/YX} = R_{NI} \cdot R_{Router \rightarrow Series} \cdot \left(1 - (1 - R_{PATH \rightarrow XY}) \cdot (1 - R_{PATH \rightarrow YX})\right)$$

$$R_{NI} = \prod_{i=1}^{2} \mathrm{R}_{NI,i} \qquad R_{Router \rightarrow \boldsymbol{Series}} = \prod_{i=1}^{2} \mathrm{R}_{Router,i}$$

$$R_{PATH \rightarrow XY} = R_{PATH \rightarrow YX} = \prod_{i=1}^{n_{hop}-1} \mathrm{R}_{Link,i} \cdot \prod_{i=1}^{n_{hop}-2} \mathrm{R}_{Router,i}$$

$$\Longrightarrow \boldsymbol{R_{PATH \rightarrow XY/YX} = 2R^{2n_{hop}+1} - R^{4n_{hop}-2}} \tag{3.5}$$

The alternative dual A/B-path situation of Figure 3.2 (c) can be modelled as parallel system of two fully independent path segments (A and B) without component sharing. The Equation 3.6 formulates two alternative solutions for the total path reliability. In the general formulation $R_{PATH \rightarrow A/B}$ it is

assumed that the hop distances ($n_{hop\rightarrow A}$ and $n_{hop\rightarrow B}$) of both path segments do not necessarily have to be equal. The formulation $R^{*}_{PATH\rightarrow A/B}$ considers equal path lengths ($n_{hop\rightarrow A} = n_{hop\rightarrow B} = n_{hop}$).

$$R_{PATH\rightarrow A/B} = \left(1 - (1 - R_{PATH\rightarrow A}) \cdot (1 - R_{PATH\rightarrow B})\right)$$

$$R_{PATH\rightarrow A} = \prod_{i=1}^{2} \mathrm{R}_{NI,i} \cdot \prod_{i=1}^{n_{hop\rightarrow A}-1} \mathrm{R}_{Link,i} \cdot \prod_{i=1}^{n_{hop\rightarrow A}} \mathrm{R}_{Router,i}$$

$$R_{PATH\rightarrow B} = \prod_{i=1}^{2} \mathrm{R}_{NI,i} \cdot \prod_{i=1}^{n_{hop\rightarrow B}-1} \mathrm{R}_{Link,i} \cdot \prod_{i=1}^{n_{hop\rightarrow B}} \mathrm{R}_{Router,i}$$

$$\Longrightarrow \boldsymbol{R_{PATH\rightarrow A/B} = R^{2n_{hop\rightarrow A}+1} + R^{2n_{hop\rightarrow B}+1} - R^{2n_{hop\rightarrow A}+2n_{hops\rightarrow B}+2}} \quad (3.6)$$

$$\Longrightarrow \boldsymbol{R^{*}_{PATH\rightarrow A/B} = 2R^{2n_{hop\rightarrow A/B}+1} - R^{4n_{hop\rightarrow A/B}+2}}$$

To compare the resulting path reliability, the constant component reliability R was set to 0.99 and the equations (R_{PATH}, $\mathrm{R}_{PATH\rightarrow XY/YX}$ and $R^{*}_{PATH\rightarrow A/B}$) were solved for the hop distance range up to six. The resulting curves are plotted in the chart of Figure 3.3. For the dual A/B-path situation was assumed that the utilization of multiple spatially independent injection/ejection points support dual path source-destination pairings at lower hop ranges beginning at one hop. The results show that designs with multi-ported resources potentially outperform typical 2D-Mesh constellations with a mix of single path and dual xy/yx-path regarding the obtainable reliability of its network paths.

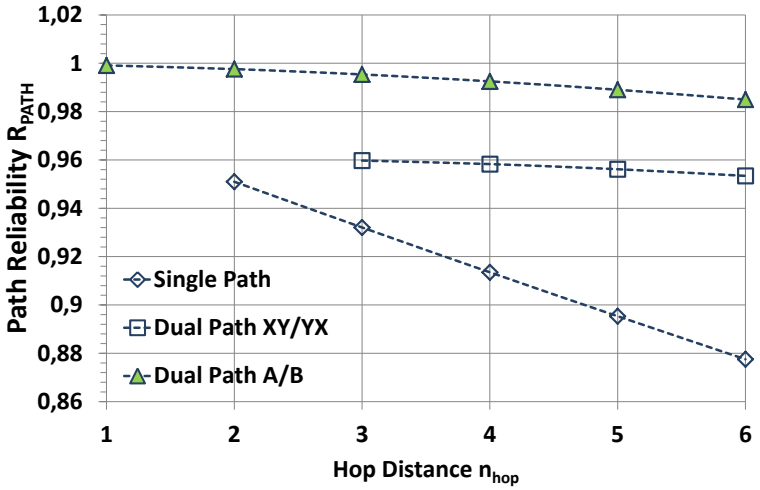


Figure 3.3: Results of simplified path reliability analysis (R=0.99 per component)

In summary, this initial analysis and discussion outlined the potential benefits of the multi-ported resources in NoCs according to performance and reliability gains. Therefore, the following investigations of this chapter will focus on design-time approaches to consolidate these promising advantages in a realistic solution at reasonable costs.

3.1 RELATED WORK

In the proposed state-of-the-art approaches, the application of multi-ported resources in NoCs is driven by the separation of traffic classes for improving the communication performance. Thereby, traffic classes are defined via dedicated spatio-temporal characteristics. The assignment of the additional resource connectivity differs between specialized and generalized policies.

In [165], *Zarkesh-Ha et. al.* proposes a hybrid NoC (HNoC) that combines the 2D-Mesh with highly specialized point-to-point busses to exclusively cover the nearest-neighbor communication (Figure 3.4 (b)). Each resource is suited with additional PTP bus-connections (green lines) to its direct neighbors in the same row/column of the 2D-Mesh. The main argument for this design decision is the rentian distribution of realistic communication pattern in CMPs and the main fraction of bandwidth is consumed for the interaction of resources with counterparts in the proximity. Unfortunately, the work in [165] represents a proof-of-concept and provides only few implementation details. The multi-network approach of *Volos et. al.* in [129] uses two independent NIs to connect the resources with two specialized 2D-Mesh NoCs, which transport dedicated traffic classes (request and response messages) in a cache-coherency scenario. Both approaches [165],[129] apply multi-ported resources as mechanism to integrate additional infrastructure into the NoC.

The general improvement for the operation of a single NoC is focused in [166]–[172]. In [166], [167], *Hu et. al.* propose a diagonally-linked 2D-Mesh (DMesh) as depicted in Figure 3.4 (a). This DMesh consists of two disjoint sub-networks whose traffic assignments are managed by spatial considerations. One network provides the communication for destinations located left from the source and the other one takes the traffic for the destinations right from the source. Therefore, each resource is suited with two injection ports (left and right) but only with a single ejection port. Furthermore, all of those ports are connected to the same router component. For the improvement of the 2D-Mesh, additional diagonal links are introduced and adaptive distributed routing is applied that selects between three different path options (XY/YX/DIAGONAL). Unfortunately, the DMesh exploit only a fraction of the potential benefits regarding multi-ported resources. The spatially-oriented injection policy helps to reduce backpressure effects. But the concentration of all resource ports to a single router instance does not shorten the path lengths or improve the reliability as much due to the shared injection/ejection points. Furthermore, each resource is only suited with a single ejection port and this avoids the relaxation of potential hotspot conditions. The router radix grows up to 13 ports and the diagonal links may impact the timing due to increased wire lengths.

In [172], *Zonouz et. al.* explore a dual NI solution to increase the fault-tolerance of a 2D-Mesh. Each resource is connected with two spatially independent routers at the same row of the 2D-Mesh to provide higher path independence at minimal xy-routing capabilities. Unfortunately, this approach

cannot provide full path independence for all source-destination pairings, because the connection to routers of the same row leads to shared xy-path segments for a significant amount of destinations. Only destinations at the same column can be reached via two spatially independent paths.

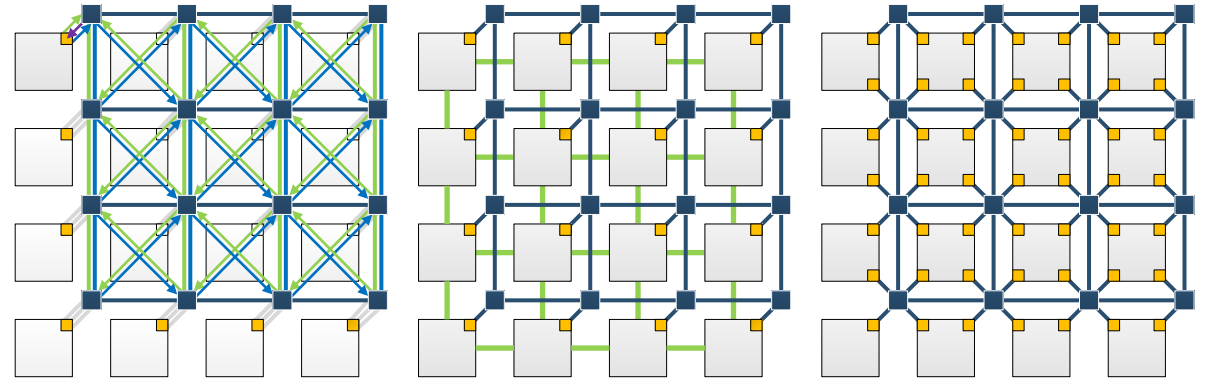


Figure 3.4: Selection of proposed multi-ported resource topologies for mesh-based Networks-on-Chip

In [168], [169], *Camacho et. al.* propose the NR-Mesh as depicted in Figure 3.4 (c). Therein, each resource is connected with up to four spatially separated routers in its proximity. It represents the generalized version of the dual NI approach presented in [172]. The traffic injection works with a random policy for the selection of a dedicated NI for each generated packet. Thereby, NIs leading to minimal paths are preferred. For the distributed routing policy, deterministic xy-routing as well as an adaptive xy/yx-routing are proposed. A dynamic ejection port selection at the destination results in the option for non-minimal paths. While the majority of proposed works apply WH switching, the NR-Mesh implements the VCT policy. In addition to the multi-ported resource concept, a port-based power management is introduced, whereas each power can be switched on/off if its connected predecessor signalizes activity-free phases. The general structure of the NR-Mesh provides many features for the full exploitation of potential improvements regarding performance as well as reliability. But the random injection port selection as well as the adaptive routing policy seems more to be a proof-of-concept study. The inherent concurrency of these distributed algorithms prohibits globally optimized traffic optimizations and random selection policy requires additional efforts to provide deterministic selections in the presence of defects. The router radix grows to eight and if adaptive routing is applied two VCs are needed. The NR-Mesh concept was further evolved to the PC-Mesh [170] and the similar HPC-Mesh [171]. Thereby, the NR-Mesh was replaced by four redundant CMesh structures and each resource is connected with up to four of them via dedicated NIs. Unfortunately, such design changes come up with all of the major drawbacks of the CMesh regarding link length and may worsen the situation for power domain crossings. Furthermore, in [128] *Das et. al.* argued that the proposed power management might not work as expected because several of the discussed issues would forbid its practical application.

Table 3.1: Feature comparison table for selected multi-ported mesh-based Networks-on-Chip

PARAMETER	TOPOLOGY				
	2D-Mesh	**DMesh**	**HNoC**	**NR-Mesh**	**QMesh**
N_{tile}	N^2	N^2	N^2	N^2	N^2
N_{router}	N^2	N^2	N^2	N^2	N^2
N_{links}	$4N(N-1)$	$N(10N-14)+4$	$8N(N-1)$	$4N(N-1)$	$4N(N-1)$
N_P	5	13	5	8	8
N_{PR}	4	10	4	4	4
N_{PC}	1	3	1	4	4
N_{TRPC}	1	1	1	**4**	**4**
$N_{V\rightarrow min}$	1	1	1	1[d] or ≥ 2[a]	**1**
$l_{link\rightarrow min}$ $l_{link\rightarrow max}$	d d	d $2d$	d d	$\boldsymbol{d}$ $\boldsymbol{d}$	$\boldsymbol{d}$ $\boldsymbol{d}$
Switching	WH	WH	WH	VCT	WH
Routing	Minimal XY	Minimal Distributed XY/YX/DIAGONAL	Minimal XY	Non-minimal XY[d] or Non-minimal Distributed Adjusted XY/YX[a]	**Non-minimal Source-based dual path XY**
Details	--	Spatially oriented sub-network (LEFT,RIGHT) assignments as function of destination	Spatial locality optimization via dedicated NN-Busses	Source-based random selection of injection terminal with prioritization of minimal path options	Spatially oriented selection of injection and ejection terminal as function of destination Programmable dual-path options

In [173], *Park et. al.* propose a multi-point network interface that provides a similar resulting structure as the NR-Mesh offers, but differs significantly in the integration and utilization of the additional connectivity. Only one NI is connected directly to the resources of the tile, while the remaining NIs are smaller and interconnected by a ring-like intra-tile subnetwork. The directly connected NI forwards packets along this intra-tile network to the corresponding NI whose router injection would reduce the remaining hop distance to the destination. But therewith, this single NI has a master-functionality and remains the major bottleneck. Furthermore, the forwarding over the intra-tile network introduces additional delays in comparison to the NR-Mesh. Table 3.1 summarizes the features of HNoC, DMesh and NR-Mesh for further discussions and comparisons with the proposed QMesh topology of the work at hand. The parameter setting of the table was already discussed in Section 2.1.2 (see Table 2.1 p. 44).

3.2 QMESH – QUADRANT-BASED MESH

The topological structure of the QMesh is similar to this one used for the NR-Mesh (see Figure 3.4 (c) or Figure 3.5 (a)) because it offers a promising feature composition. Each resource exploits its maximum connectivity to spatial independent routers in its proximity, while the original 2D-Mesh for the global communication infrastructure remains. The majority of IP cores have access to the NoC via four independent routers, which are located in the four quadrants around them. Hence, the name

quadrant-based mesh was chosen. But the applied mechanisms, policies and algorithms differ. The QMesh is designed to achieve deterministic control over ETE traffic flows without random selection policies and increased flexibility to adapt the traffic management from higher system levels.

3.2.1 TOPOLOGY CHARACTERISTICS

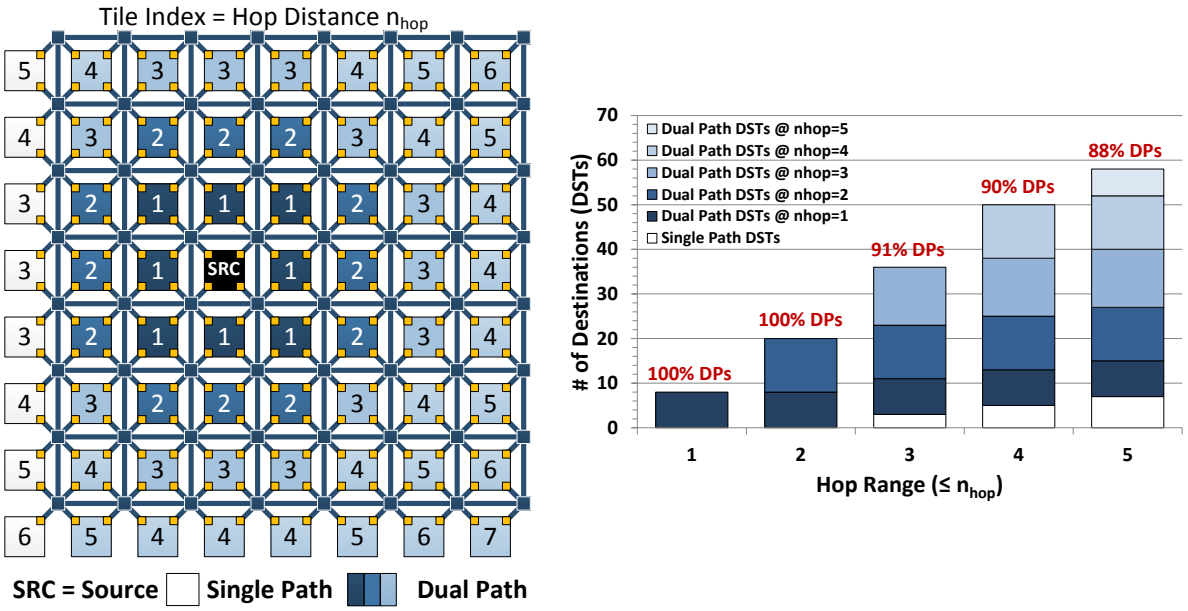
The QMesh infrastructure integrates most of the IP cores with four spatially independent terminals to the global 2D-Mesh structure. Exceptions are those resources located at the perimeter with minimal x and/or minimal y coordinate (x = 0 and/or y = 0). The majority of those IP cores along the edges are connected via two spatially independent NIs. Only a single core at the lower left corner is suited with a single NI. But if needed, this heterogeneous resource connectivity can be easily solved by extending the dimensions of the QMesh with one additional row/column along the corresponding edges. The routers integrate simple xy-routing combined with WH switching and their radix grows up to eight ports ($N_{PC} = 4 \rightarrow N_{TRPC} = 4$). These applied modifications of the basic 2D-Mesh provide a simple way to fulfill the requirements for the consolidation of the outlined performance and reliability improvements (as discussed in the introduction of this chapter).

Due to the availability of two spatially independent NI terminals to each side (UP, LEFT, DOWN, RIGHT) for a majority of the IP cores:

- Most of the source-destination pairings can communicate via two fully independent xy-paths (see dual A/B-path in Figure 3.2 (c) p. 60), which are defined by the combination of input/output terminal. Thus, adaptive dual path communication without the integration of VCs or adaptivity restrictions becomes available.
- Most of the source-destination pairings can communicate over reduced hop distances via the selection of locality optimized input/output terminal combinations.
- Most of the resources have relaxed hotspot conditions due to their accessibility via two or more fully independent NI terminals.
- The traffic injection can be improved through the consideration of locality optimized NI selection that is controlled by the destination coordinates.
- Local traffic optimization is an inherent feature without the need for specialization such as applied by the HNoC. Nearest-Neighbor traffic just passes through the corresponding router that offers a terminal port to the relevant destination. Thereby, global traffic along the row/column will not suffer from any interference.
- In comparison to the 2D-Mesh, the number of dual path source-destination pairings increases, destinations at the same row/column will not be constrained and more resources

at shorter or equal hop distances become accessible. This becomes obvious if the exemplary hop range distributions of Figure 3.1 (a) (see p. 56) for the 2D-Mesh and this one in Figure 3.5 (a) for the QMesh are compared. The dual path resources have a blue filling and their labelling carries the hop distance from the source's (SRC) perspective. The resulting advantage over the 2D-Mesh is shown in the chart of Figure 3.5 (b), where the total number of reachable dual path destinations as a function of a growing hop range is illustrated (compare to Figure 3.1 (b) at p. 56). Furthermore, the QMesh makes up to eight cores directly accessible over a single hop without any global interference of the corresponding traffic flows. This has further potential to affect the parallelization model and mapping strategy for the workload processing.

The increased resource connectivity requires design changes at the tile level. Figure 3.6 contrasts the components-level integration of the 2D-Mesh and the QMesh for an expected layout of a CMP tile.



(a) Dual path and hop distribution in QMesh

(b) Dual path destinations over hop range

Figure 3.5: Distribution of dual path destinations inside the QMesh

A ***central crossbar*** (CX) provides the needed intra-tile communication in both NoCs and connects processing element (PE), level 1 instruction/data cache (L1-I/D) as well as the level 2 cache. Furthermore, the needed NoC components, such as buffer (BUF), path table (PT) and NIs, are connected to it. For each of the PE, L1-I/D, PT, and NI components, one read and one write connection to the CX are assumed. Those pairs of parallel read/write connection to the CX are referred to as ports. The level 2 cache should be accessible via three ports to support independent data transfers with L1-I, L1-D and at least one NI. The number BUF ports to the CX is determined by its number of independent segments and thus by the number of NIs.

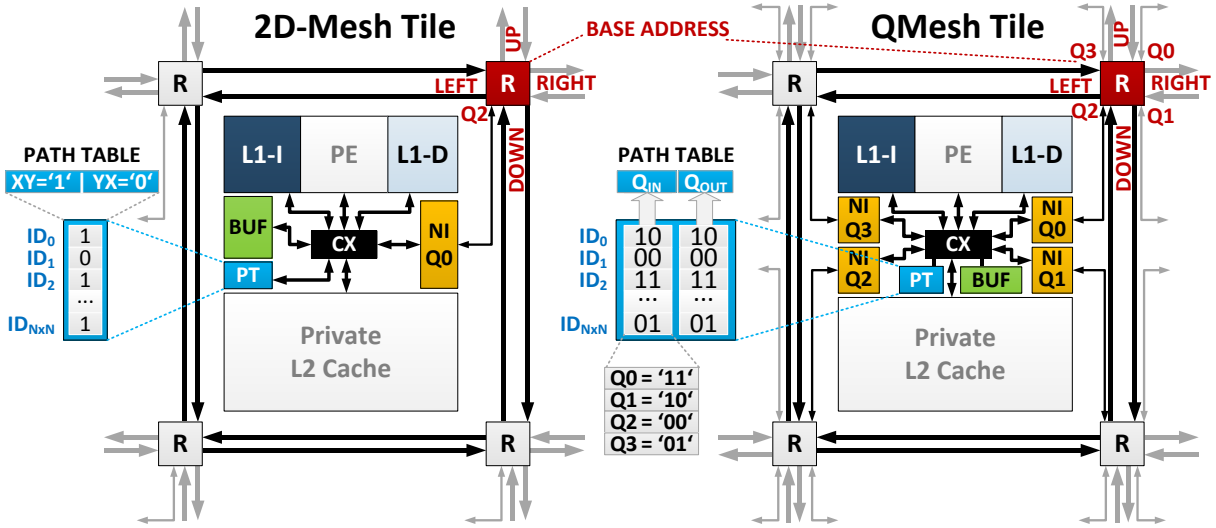


Figure 3.6: Component-level schematics of a xy/yx-Routing 2D-Mesh and a QMesh for a single tile

As crossbar arbitration policy, the round-robin strategy or fixed-priority assignment should be utilized. The expected hardware costs for the crossbar are minimal and should not be 2 up to 4 times higher than those of the CX/SA-combination of a NoC router device, which consumes 20-30 % of the router area and less than 1% of the anticipated tile area (see Figure I.4 p. 170). Alternatively, a PTP interconnection strategy has to be applied. If the radix of the crossbar increases due to higher component integration per tile, more sophisticated integration techniques like the swizzle-switch of *Sewell* [116] can be applied to reduce the costs. This was already proposed by *Abeyratne et. al.* in [114] to achieve costs reductions for high-radix NoC components. The difference of the crossbars in the tile of the 2D-Mesh and the QMesh is mainly given by the demand of additional connections for the newly introduced NIs in the QMesh and the additional buffer segments dedicated to them. This combination of multiple spatially independent NIs with the common CX connection has further potential for the intra-tile passing of inter-tile traffic flows along the NoC. As additional feature, it can be applied for reliability enhancements due to an increased number of failure workarounds and workload interaction pattern along the resources without traffic interferences along the global row/columns paths of the 2D-Mesh NoC. But the integration of required mechanisms and policy changes is out of scope for the presented approaches in the work at hand and will be left open for future investigations.

The ***buffer component*** (BUF) integrates the necessary memory resources for data transmission and reception. Furthermore, it must contain control logic for memory access and allocation management. The basic segmentation is determined by the number of NIs (N_{NI}) that must be able to read/write independently message from/to it. Thus, N_{NI} transmission segments hold the data to be transferred over the NoC and N_{NI} reception segments captures the incoming data from each NI. This segmentation results in 2× N_{NI} CX ports for the BUF. The segmentation itself can be realized logically

via read/write pointer with programmable address spaces (which supports dynamic segmentation at run-time) or physically via full separation as independent sub-units. If VCs are applied, an additional segmentation per VC of the transmission segments per NI has to be considered to encounter head-of-line blocking between different VCs. Therefore, one additional CX ports per VC and NI is necessary. The CX ports of the data reception segments can be statically assigned to them for the support fully independent data reception of all NIs and VCs without multiplexing. The destined intra-tile components (PE, L1-I/D, L2) are notified by the buffer control logic and read the incoming data from the specific reception segments. For the CX ports of the data transmission segments, static or dynamic assignment strategies are possible. The buffer control logic is connected to the PT to perform lookups for the additional routing information such as the assigned VC or output NI, which defines the transmission buffer segment to be used. This can happen prior or as part of the data write actions into the buffer. In the first case, static CX port assignments to the transmission segments are possible, because the corresponding transmission segments are known in advance. Therefore, it is the preferred solution and the PT lookup might be combined with the arbitration process of the CX. In the second case, the data needs to be dynamically multiplexed to the correct transmission segment inside the buffer for each of the CX ports, depending on the lookup results. In summary, the buffer has more similarity with a cache memory. Therefore, an estimate for the maximum area cost was performed via the CACTI-results [174] for a 32kbyte direct-mapped eight-ported cache in 45nm CMOS. The implementation area is 0.775 mm² while the anticipated tile area is 9.0 mm² (see Table I.5 p. 168). Hence, the relative area costs of the buffer remain lower than 10% of the tile area. Thereby, it has to be mentioned that the selected buffer size of 32kbyte per tile is quite generous and smaller sizes are more likely. For example, Intel's SCC provides 16kbyte buffer for a tile with two IP cores on it [29].

The ***path table*** (PT) represents a programmable lookup table that provides additional information to extend the routing data of a packet header and to specify the traversal along the NoC as well as the transmission buffer segment assignment. For the 2D-Mesh in Figure 3.6 this might be the path selection bit for each destination as applied for the adaptive xy/yx-routing approach of *Manevich et. al.* in [60], [125]. In general, each tile contains a path table with $N_X \cdot N_Y$ information words of N_{PT} bits for all possible destinations inside the NoC. This content is rewritable and will be programmed over the connection port of the path table to the CX. Hence, adaptive changes for traffic management can be initiated more flexible by local intra-tile resources (source-based) or at the inter-tile level in a centralized/regional context. Furthermore, this strategy can be applied for online, offline, or hybrid policies. The buffer control logic performs the lookup before new transmission data is written to it. This further ensures that the path information for complete data messages remains constant if it is transmitted via multiple packets. This is important to guarantee in-order

transmissions if source-based adaptive routing is applied. The concrete content of the path table inside the QMesh as well as the resulting path format are discussed in the following passage 3.2.2.

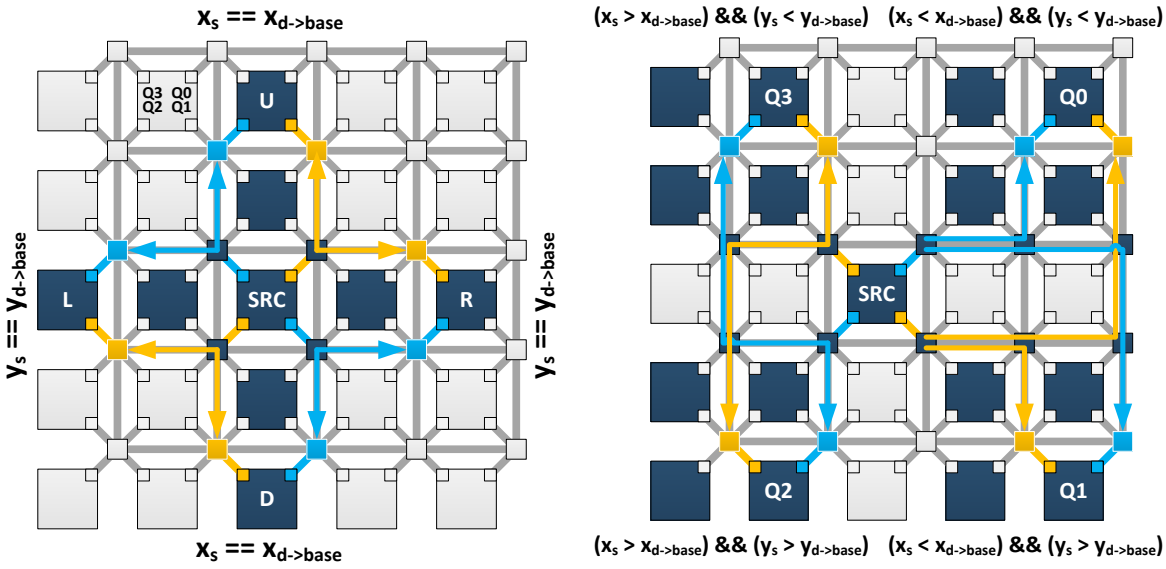
The difference between 2D-Mesh and QMesh regarding the ***NIs*** is given in their increased number per tile (up to four NIs as depicted in Figure 3.6). Thereby, the NIs are identified after the quadrant scheme of the two-dimensional cartesian coordinate system from quadrant 0 in a clockwise rotation to quadrant 3 (Q0, Q1, Q2 and Q3). The NI has a similar setup as a router port with its input/output functionalities. The input as well as the output section integrates FIFO-buffers for the packet receptions and transmissions. Thus, blocking situations regarding the concurrent tile buffer access over the crossbar are relaxed. For the application of VCs the number of FIFO-buffers per NI must be increased according to the number of VCs. The sizing of the FIFO-buffers should be similar to the input port of the router and at least one packet with a full cache block/line (see Section 2.1.1 pp. 31-40). For the control of the flit-by-flit passing in the packet reception/transmission procedure, input and output section must be suited with the corresponding handshake logic. If look-ahead routing [7] is applied the output of the NI furthermore has to integrate the necessary routing logic. Finally, each NI has to integrate ETE flow control mechanisms with corresponding buffer allocation capabilities and further EDAC logic.

In comparison to the baseline 2D-Mesh topology, the routers inside the QMesh are affected by the increased number of ports. As illustrated in Figure 3.6, each router in the QMesh provides access to multiple resources in its proximity (Q0, Q1, Q2, and Q3), while the 2D-Mesh router only serves a single resource (CORE/Q2). This will further affect the addressing concept, because the QMesh allows the resource entry of data flows via multiple terminals. Inside the 2D-Mesh the router located at Q0 of each resource and the resource itself share the same network address (logical identifier or xy-coordinate). Furthermore, the resource does not need to specify the output NI of its packets. Regarding compatibility and homogenity of the addressing inside the QMesh, the base router remains this one located at Q0. For the specification of a unique QMesh destination the packet header must contain the router address and the targeted output quadrant, which implies that each destination in the QMesh has up to four addresses.

3.2.2 NETWORK PATHS AND ROUTING

The network path concept for the QMesh needs to be extended to guarantee deterministic destination addressing and path selection. Therefore, packet injection as well as ejection terminal must be added and destination-coordinate manipulations have to be applied. This is realized via the introduction of the input and output quadrant parameter (Q_{IN} and Q_{OUT}) in addition to the xy-coordinates inside the packet header (see Figure 2.4 p. 29). The input quadrant (Q_{IN}) specifies the NI

for the packet injection into the QMesh at the source tile, while the output quadrant (Q_{OUT}) specifies the terminal port at the destination router and furthermore determines the needed coordinate manipulations of the destinations base address. Q_{IN} and Q_{OUT} are encoded with 2 bits each and have to be set for each source-destination pair. This is integrated in all tiles in form of the path tables (PTs) that contain the resulting 4 bit words ($N_{PT} = 4$) for each of the possible $N_X \cdot N_Y$ destinations per tile (see Figure 3.6 p. 68). In comparison to the path table of a 2D-Mesh with adaptive xy/yx-routing ($N_{PT} = 1$ as depicted in Figure 3.6 p. 68), the hardware efforts of the QMesh are higher, but in total they are negligible regarding the kbytes-sized efforts for memory and buffer resources per tile. For the reference case of an 8x8 NoC, the path table inside the 2D-Mesh would require 8 bytes per tile and inside the QMesh it would consume 32 bytes per tile.



(a) Same Row/Column (UP/U, RIGHT/R, DOWN/D, LEFT/L) (b) Quadrants (Q0, Q1, Q2, Q3)

Figure 3.7: Basic spatial sectors and path options inside the QMesh

Under consideration of the applied xy-routing in the QMesh, the available dual path options vary in their characteristics and depend on the spatial relation of the addresses for source (x_s, y_s) and destination ($x_{d\rightarrow base}$, $y_{d\rightarrow base}$). The condition if source and destination shares common coordinate values has the main influence on the path characteristics. Figure 3.7 illustrates the resulting spatial sectors as function of this address relationship. In Figure 3.7 (a), source (SRC) and destination are located at the same row (RIGHT/LEFT) or column (UP/DOWN) of the mesh-grid. In Figure 3.7 (b), the destination is located in one of the four quadrants (Q0 to Q3), because both addresses differ in their x- and y-coordinates. The resulting dual path options are illustrated via the blue and orange lines. For source-destination pairings with shared coordinate values, two xy-paths with equal hop distances are available. For destinations located in the quadrants, two xy-paths with different hop distances exist. Figure 3.7 further shows the necessity to manipulate the destination coordinates in the packet

header to reduce the hop distances. The general addressing and identification of the tile is done through the base address ($x_{d\rightarrow base}$, $y_{d\rightarrow base}$) as already discussed above (see Figure 3.6 at p. 68).

But during the message setup inside the buffer (BUF), a lookup on the path table (PT) is performed with this base address. The lookup result contains:

- Q_{IN} to address the correct buffer segment reserved for the corresponding NI and as header extension for the source field,
- Q_{OUT} to address the correct router output at the modified final address (x_{d*}, y_{d*}) as part of the header extension for the destination field, and
- the modified destination address (x_{d*}, y_{d*}) for the packet header, that is adapted according to Table 3.2 as function of the destinations base address and Q_{OUT}.

The alternative router endpoints in the proximity of the targeted tile are adjusted by decrementing the x- and/or y-coordinate of the base address as a function of the two Q_{OUT}-bits. The first bit controls the $\lceil log_2(N_X) \rceil$ wide decrementer logic for the x-coordinate and the second bit the $\lceil log_2(N_Y) \rceil$ wide decrementer logic for the y-coordinate. The values are decremented if their corresponding control bits in Q_{OUT} are '1'. The resulting modified destination (x_{d*}, y_{d*}) represents the final router from where the originally targeted resource tile ($x_{d\rightarrow base}$, $y_{d\rightarrow base}$) will be reached via the output port defined in Q_{OUT}. The additional hardware extensions at each lookup port of the path table are constrained to the two controlled decrementer circuits (e.g. for an 8x8 NoC each decrementer consists of three cascaded half adder elements). Alternatively, the modified destination coordinates could be stored as part of each path table entry and the costs per table entry would be increased by $\lceil log_2(N_X) \rceil + \lceil log_2(N_Y) \rceil$ bit. Thus, the decrementer option is the better choice.

Table 3.2: Base address adaptations as function of selected output quadrant at destination

	Q_{OUT} = Q0 = '11'		Q_{OUT} = Q1 = '10'		Q_{OUT} = Q2 = '00'		Q_{OUT} = Q3 = '01'	
Coordinate ID	x_{d*}	y_{d*}	x_{d*}	y_{d*}	x_{d*}	y_{d*}	x_{d*}	y_{d*}
Adaptation	$x_{d\rightarrow base} - 1$	$y_{d\rightarrow base} - 1$	$x_{d\rightarrow base} - 1$	$y_{d\rightarrow base}$	$x_{d\rightarrow base}$	$y_{d\rightarrow base}$	$x_{d\rightarrow base}$	$y_{d\rightarrow base} - 1$

Table 3.3 summarizes the dual path options for all possible spatial sectors (as specified in Figure 3.7) out of the source's perspective. Thereby, the path option A is available for all constellations of source and destination coordinates. The path option B is a valid choice for the majority of source-destination pairings that fulfill the defined coordinate constraints. These constraints affect the left/lower edge resources of the QMesh and exclude a subset of path B choices that cannot be utilized due to missing router terminals. A hop distance comparison of the resulting QMesh paths with its counterparts in the 2D-Mesh ($n_{hop\rightarrow 2D\text{-}Mesh}$) confirms the expected benefits. Along paths of source-destination pairings with shared coordinate values (UP/RIGHT/DOWN/LEFT), both options reduce the hop distance

equally by one. For source-destinations pairings with differing values, for both coordinates (Q0/Q1/Q2/Q3) the favored path option A reduces the hop distance by two, while the path option B offers hop distances equal to the 2D-Mesh.

Table 3.3: Fundamental path options and characteristics inside the QMesh

DIRECTION	PATH A			PATH B			
	Q_{IN}	Q_{OUT}	Hop distance	Q_{IN}	Q_{OUT}	Hop distance	Path B – Constraint
UP (U)	Q0	Q3	$n_{hop \to 2D\text{-}Mesh} - 1$	Q3	Q0	$n_{hop \to 2D\text{-}Mesh} - 1$	$x_s > 0$
RIGHT (R)	Q0	Q1	$n_{hop \to 2D\text{-}Mesh} - 1$	Q1	Q0	$n_{hop \to 2D\text{-}Mesh} - 1$	$y_s > 0$
DOWN (D)	Q1	Q2	$n_{hop \to 2D\text{-}Mesh} - 1$	Q2	Q1	$n_{hop \to 2D\text{-}Mesh} - 1$	$x_s > 0$
LEFT (L)	Q3	Q2	$n_{hop \to 2D\text{-}Mesh} - 1$	Q2	Q3	$n_{hop \to 2D\text{-}Mesh} - 1$	$y_s > 0$
QUADRANT 0 (Q0)	Q0	Q0	$n_{hop \to 2D\text{-}Mesh} - 2$	Q1	Q3	$n_{hop \to 2D\text{-}Mesh}$	$y_s > 0$
QUADRANT 1 (Q1)	Q1	Q1	$n_{hop \to 2D\text{-}Mesh} - 2$	Q0	Q2	$n_{hop \to 2D\text{-}Mesh}$	none
QUADRANT 2 (Q2)	Q2	Q2	$n_{hop \to 2D\text{-}Mesh} - 2$	Q3	Q1	$n_{hop \to 2D\text{-}Mesh}$	$x_{d \to base} > 0$
QUADRANT 3 (Q3)	Q3	Q3	$n_{hop \to 2D\text{-}Mesh} - 2$	Q2	Q0	$n_{hop \to 2D\text{-}Mesh}$	$y_s > 0$ && $x_{d \to base} > 0$

For the experimental evaluation of the basic QMesh approach, in the next Section 3.3 a static path table setup is applied. The default configuration of the path table uses the path option A for all Q0 to Q3 destinations. The paths to UP/RIGHT/DOWN/LEFT destinations are initiated with an alternating strategy that depends on the hop distance between source and destination. Even hop distances get the assignment of path option A, while odd hop distance utilize path option B (if constraints are met, otherwise A). But in general, different ETE path adaptation strategies (online, offline, hybrid) are orthogonal to the base structure specified by this QMesh approach and can furthermore co-exists in multiple spatial regions of the NoC. Thus, flexible traffic management is supported. Additionally, the flexible rewriting of PT-segments should be supported for the updates of paths in dedicated regions. The applied xy-routing represents one of the simplest algorithms in mesh-based NoCs. A packet is forwarded along the x-direction until the x-coordinate of the destination is reached. Afterwards, the same procedure happens along the y-direction until the current router address matches with the destination address in the packet header. This algorithm is chosen due to its simplicity, because it supports multiple orthogonal extensions/adaptations (e.g. look-ahead routing, VCs, pipeline rebalancing) without complex dependencies or side-effects.

3.2.3 BASIC FEATURE SUMMARY OF QMESH

Table 3.4 above summarizes the impact of the QMesh modifications on the basic design parameter and contrasts them to the xy/yx-routing 2D-Mesh with similar capabilities as applied by *Manevich et. al.* in [60], [125]. Thereby, the potential cost benefit ratio is comparable, while the targeted traffic management potential of ATDOR [60], [125] remains at least equal and similar use cases can be supported.

Table 3.4: Design feature summary of the QMesh in contrast to a comparable 2D-Mesh configuration

DESIGN FEATURE	XY/YX 2D-MESH [60], [125]	QMESH
Router Area Overhead	67%	80%
	Compared to baseline xy-routing 2D-Mesh (see Figure I.4 p.170 45nm-5P-2VC and 45nm-8P-1VC)	
Router Ports – N_P	5	8
Router VCs – N_V	2	1
Router Pipeline	RL/VCA/SA/ST/LT → Depth = 5	RL/SA/ST/LT → Depth = 4
Routing Algorithm	XY/YX	XY
Dual Path Option	XY/YX-Paths with common endpoints	A/B-Path with separate endpoints
Path Length Reduction	None	1 up to 2 hops
	Compared to baseline xy-routing 2D-Mesh	
Links	No difference to baseline 2D-Mesh	
Path Table (PT)	N_{tile} bits	$4 \cdot N_{tile}$ bits + $\lceil log_2(N_X) \rceil$ bits decrementer for x_{d*} + $\lceil log_2(N_Y) \rceil$ bits decrementerfor y_{d*}
	Lookup of VC assignment	Lookup of Q_{IN} and Q_{OUT} + Base Address Adaptation (x_{d*}; y_{d*})
	Both programmable → Source-based/Centralized/Hybrid adaptation possible	
Network Interfaces (NI)	1	4
	Independent NI FIFOs per VC needed	Spatially fully independent + No difference to baseline 2D-Mesh
Buffer (BUF) segments	2 TRANSMIT and 2 RECEIVE	4 TRANSMIT and 4 RECEIVE
	Independent T/R segments per VC needed	Independent T/R segments per NI needed
Crossbar (CX) Ports	13	19
Packet Header Destination Format	$x_{d→base}$; $y_{d→base}$; VC → $\lceil log_2(N_X) \rceil + \lceil log_2(N_Y) \rceil + 1$ bits	x_{d*}; y_{d*}; Q_{OUT} → $\lceil log_2(N_X) \rceil + \lceil log_2(N_Y) \rceil + 2$ bits

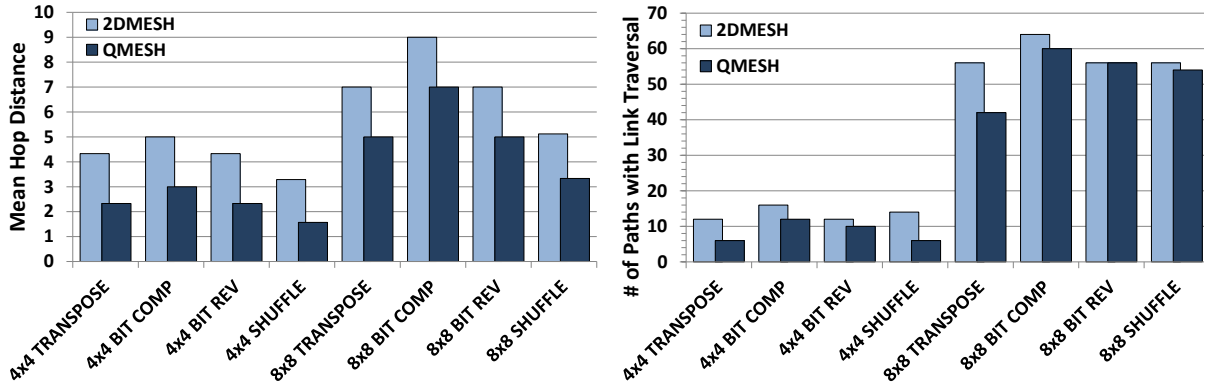
3.3 EXPERIMENTAL RESULTS

This section provides the results of the experimental evaluation for the system setups and workloads specified in the Appendix (see pp. 157-166). First, the results from the SystemC-based simulations of synthetic traffic pattern setups are summarized. Next, the reliability impact of the QMesh is presented. Finally, the results of the real application workloads simulated on Sniper are discussed.

3.3.1 SYNTHETIC WORKLOADS

Initially, the synthetic traffic patterns are evaluated on 4x4 and 8x8 NoCs. Thereby, the unmanaged QMesh (no adaptive traffic management) is compared to the reference 2D-Mesh regarding its relative influence on the network saturation ($\Delta_{Saturation}$), power dissipation (Δ_{Power}), and packet delay (Δ_{Delay}). The relevant parameter settings of the NoCs, traffic generation, and simulator are summarized in the the Appendix (see pp. 164-167 -> SystemC-based). The detailed discussion of the evaluation method for the collected traffic statistics is provided in the Appendix (see pp. 179-182). As traffic patterns, different state-of-the-art bit permutation (such as bit complement, bit reverse,

transpose, and shuffle) as well as propabilistic communication distribution functions (such as random uniform, nearest neighbor, rentian, and hotspot) are applied [12], [56], [175]–[184]. The implemented traffic generation functionality is disussed as part of Appendix (see pp. 160-163).



(a) Hop Statistics of BP Pattern

(b) Link Traversing Path Statistics of BP Pattern

Figure 3.8: Comparison of mean hop distance and number of link traversing paths for synthetic pattern in a 4x4 and an 8x8 2D-Mesh as well as QMesh

The bit permutation (BP) patterns have different properties that determine the degree for the exploitation of potential advantages offered by the unmanaged QMesh and the impact of a system scale up from a 4x4 to an 8x8 NoC. In general, BP-based patterns represent the workload scenarios running a single application. The communication of the transpose (TRANSPOSE) pattern is line symmetrical along the diagonal between lower left (0, 0) and upper right (N_X, N_Y) corner, while the communication in the bit complement (BIT COMP) pattern shows point reflection symmetry around the centre the NoC. Furthermore, the transpose and bit complement pattern have the highest mean path lengths and are more susceptible to concurrency due to symmetry-related overlapping along these source-destination paths. The bit reverse (BIT REV) and shuffle (SHUFFLE) pattern have a higher diversification of source-destination pairings without global symmetry. Thereby, the shuffle pattern operates with the lowest mean hop distance and contains a mix of communication flows addressing the same row/column. Contrary, the transpose, bit complement, and bit reverse pattern only contains source-destination pairings whose communication flows address the quadrant regions. The bar-charts Figure 3.8 depicts the mean hop distances and the number of end-to-end paths with link traversal along the simulated 4x4 and 8x8 NoCs.

Contrary to the BP-based synthetic traffic patterns, the patterns based on probabilistic communication distribution functions (CDFs) result in traffic situations that describe mixed workloads with all source-destination constellations and will not be dominated by few connections. Thereby, the nearest-neighbor (NN) pattern benefits due to reduced interferences and lower hop distances inside the QMesh. Therein, all traffic to and from the proximity resources does not traverse any router-to-router link and therefore the interferences with the random uniformly distributed multi-

hop traffic flows are reduced. The higher the NN traffic fraction becomes (20%, 40%, 60% and 80%) the higher is the degree of locality. Similar to the NN is the resulting traffic distribution of the rentian pattern (RENT), but therein the locality-oriented traffic is not focused on the direct proximity and concentrates more traffic to lower hop distances in general via a power-law based distribution with exponents (R) in the range of 0.3 to 0.7. Contrary, the random uniform (UNI) pattern assigns each possible connection inside the NoC an equal communication propability. This increases the mean hop distance of the traffic scenario and the QMesh benefits less from its nearest neighbor optimization. Thereby, the fraction of served connection is varied (20%, 40%, 60% and 80%) to cover different path occupation scenarios. The amount of injected packets per path increases for lower path occupations. The hotspot (HOT) pattern assigns dedicated resources along the perimeter of the NoC higher communication probabilities and the remaining traffic is random uniform. Such traffic scenarios represent intensive system input/output as well as memory-related communication and mainly take advantage of the multi-ported resources inside the QMesh. The following bar-charts provide a compact view on the evaluated impact of the QMesh. First, Figure 3.9 presents the relative improvements of the network saturation ($\Delta_{Saturation}$) for the QMesh over the 2D-Mesh (bar labelling: <NoC size> <traffic pattern>) and reflect the specific issues as discussed above.

In the BP scenarios, the shuffle pattern benefits most for both NoC sizes (105% gain at 4x4 and 67% gain at 8x8), while in the 8x8 NoCs with CDF-based traffic the NN pattern shows the biggest saturation gains with up to 113%. The mean relative saturation improvements of the QMesh over the 2D-Mesh over all simulated cases are 30% in a 4x4 NoC and 34% in an 8x8 NoC. The results clearly show the advantages of the QMesh. The higher the traffic locality becomes the more the saturation improvement due to the QMesh increases. Furthermore, it becomes obvious that the saturation gain in the hotspot pattern is dominated by the data throughput of the “hot” terminals along the perimeter and the resulting backpressure. The saturation gains stall at a hotspot traffic fraction of 60%. A similar effect can be observed at the BP pattern regarding bit reverse and transpose. The pattern compositions of single as well as unique path connections per tile do not exploit the benefits of the multiple terminals per tile inside the QMesh. The 4x4 NoC configurations with QMesh operate with decreased packet delays and support increased packet injection rates before the saturation occurs. Thereby, the application of the QMesh to the bit complement communication pattern has only a small impact on the NoCs saturation level due to the mix of symmetry- and path-length-related interferences of traffic flows. If scaled up to 8x8 NoCs, concurrency and path lengths increase. Thereby, the transpose, bit complement, and bit reverse patterns are highly susceptible to these scaling-related issues. Therein, only the shuffle pattern experiences a higher saturation level due to the application of the QMesh. Furthermore, increased path interferences shrink packet delay reductions in all evaluated pattern.

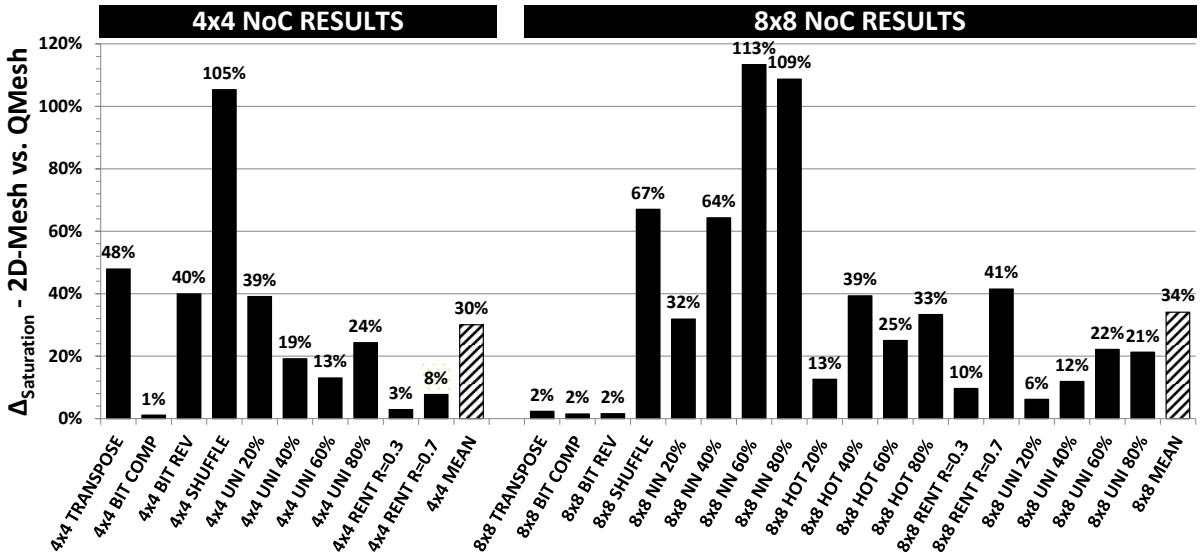


Figure 3.9: Comparison of network saturation for simulated 2D-Mesh and QMesh under synthetic traffic pattern

The bar-chart in Figure 3.10 depicts the relative impact of the QMesh on the mean total power dissipation and the mean packet header delay in comparison to the 2D-Mesh (Δ_{Delay} and Δ_{Power}) for all evaluated cases (bar labelling: <NoC size> <traffic pattern>).

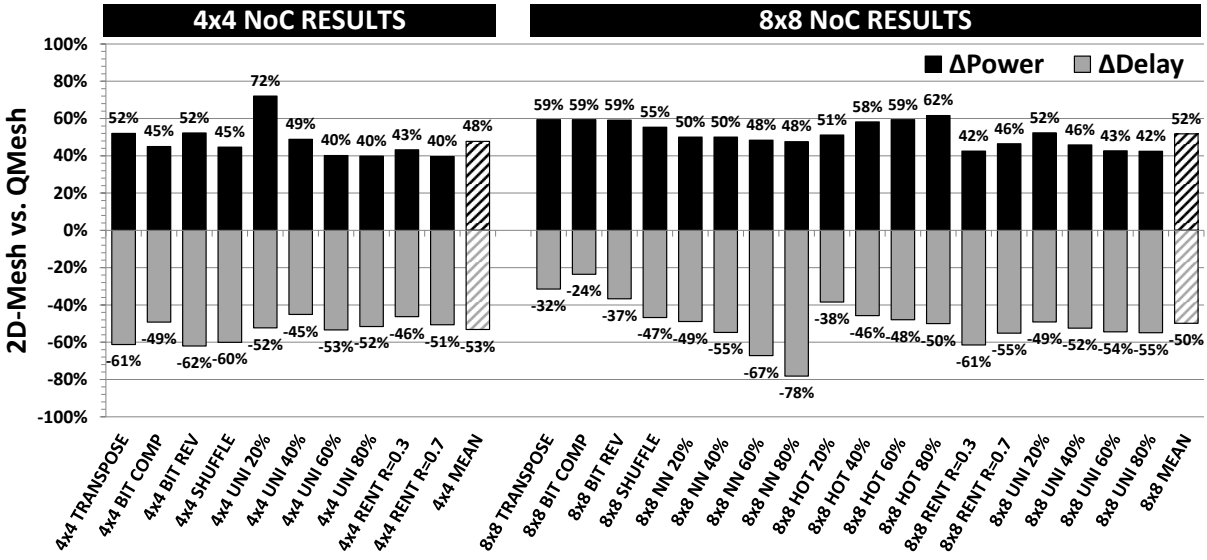


Figure 3.10: Comparison of network power/delay for simulated 2D-Mesh and QMesh under synthetic traffic pattern

Therefore, both NoC topologies are compared regarding the total power dissipation and packet header delay along their common unsaturated regions, which are called comparable regions (see Appendix pp. 179-182). It becomes obvious that the increased power dissipation in the QMesh is at least compensated by the decrease of the packet header delay for the majority of the evaluated configurations. Thereby, it has to be mentioned that for higher packet injection rate levels the packet delay savings generally outperform the power increase and the QMesh reaches higher saturation levels beyond the compared regions (see example plot in Appendix pp. 179-182). Additionally, it

must be considered that this power-delay-tradeoff only works if the delay savings can be consolidated in accelerated workload processing at the global level of the CMP (therefore no direct power-delay-product is presented).

Overall, the experimental evaluation of the unmanaged QMesh under synthetic BP and CDF patterns offer promising results. Under consideration of the basic modifications, additional costs and the absence of adaptive traffic management (simple xy-routing and static paths) the QMesh with its improvements over the 2D-Mesh provides comparable performance gains like more advanced and costly approaches that uses VCs as well as dynamic adaptation policies [60]–[62], [98], [119], [143], [163], [167]. This is very encouraging for investigations of adaptive traffic management, because multi-path routing is already an inherent feature of the QMesh that has to be exploited.

3.3.2 RELIABILITY

Regarding reliability impact of the QMesh, it must be evaluated if the improved connectivity of the QMesh has a positive effect on the thermally-related wear-out of the NoC and if it provides better degradation in the presence of permanent failures compared to alternative NoC approaches.

The evaluation of the thermal characteristics applies the modelling as specified in the Appendix (see pp. 175-179). For the comparable traffic regions of the 2D-Mesh and the QMesh at the synthetic patterns, the progress of the mean router temperature over increasing traffic loads is calculated for the simulations of Section 3.3.1. The observed reduction of the mean operating temperature of the routers inside the QMesh results from reduced hop distances in the communication paths and loads per router/link in comparison to the 2D-Mesh. The higher the locality of communication with resources inside the proximity, the more increases the temperature spread with lower temperatures in the QMesh. Furthermore, the temperature spread increases with the achievable maximum injection rates in the comparable regions due to the corresponding impact on the router activity. On the one hand, this helps to limit the impact of the QMesh modifications on the power dissipation and on the other hand the reliability of the NoC benefits. As already discussed by *Aisopos* in [43], temporary failures along critical paths are reduced. But the temperature reduction further affects the CMOS wear-out progress due to slow downs of phenomena [36], [37]. Therefore, the relation derived from the Arrhenius equation can be utilized to quantify the impact [36], [101].

The Equation 3.7 describes the acceleration factor a_{MTTF} , which is typically used to determine the mean-time-to-failure (MTTF) of CMOS devices under normal operating conditions after they were tested under harsh conditions (higher temperature) to speed-up the degradation and reduce test times [36]. But in general, a_{MTTF} describes the temperature-related increase ($a_{MTTF} < 1$) or decrease ($a_{MTTF} > 1$) of wear-out progress in CMOS. Therein, $t_{QMESH,2DMESH}$ specify the MTTF of

the QMesh/2D-Mesh component, E_a is the activation energy of the CMOS devices in electron volts (here 0.7 eV at 45nm CMOS [36]), k is the Boltzmann's constant (8.6×10^{-5} eV/K), and $T_{QMESH,2DMESH}$ describes the absolute temperatures in Kelvin (K).

$$a_{MTTF} = \frac{t_{QMESH}}{t_{2DMESH}} = e^{\frac{E_a}{k}\cdot\left(\frac{1}{T_{QMESH}} - \frac{1}{T_{2DMESH}}\right)} \quad (3.7)$$

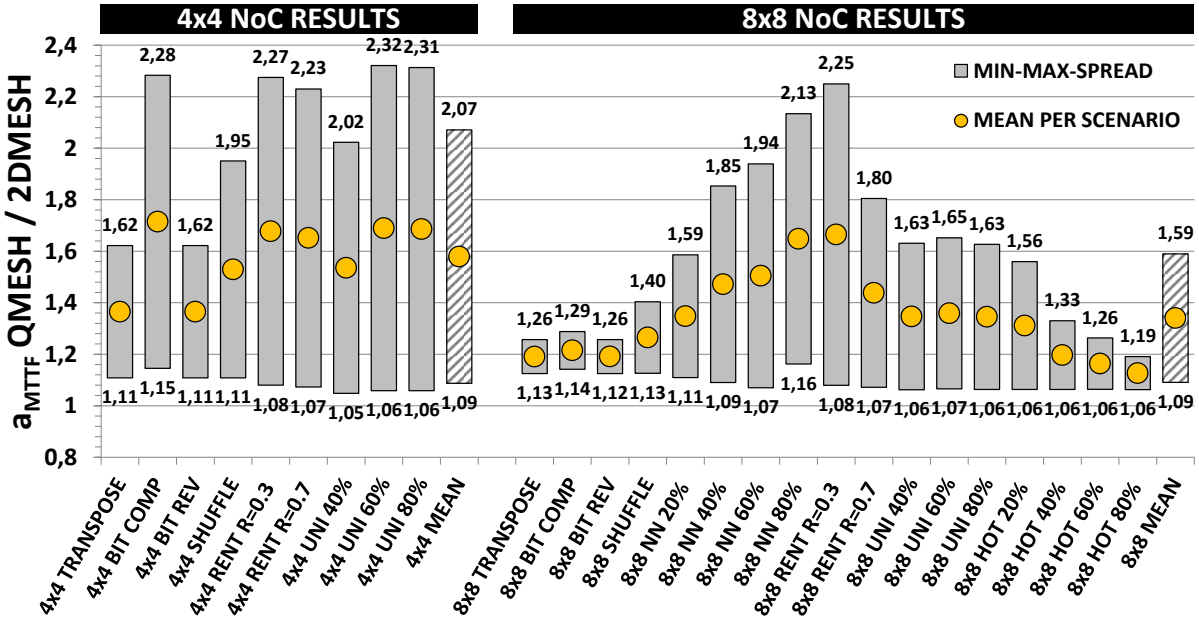
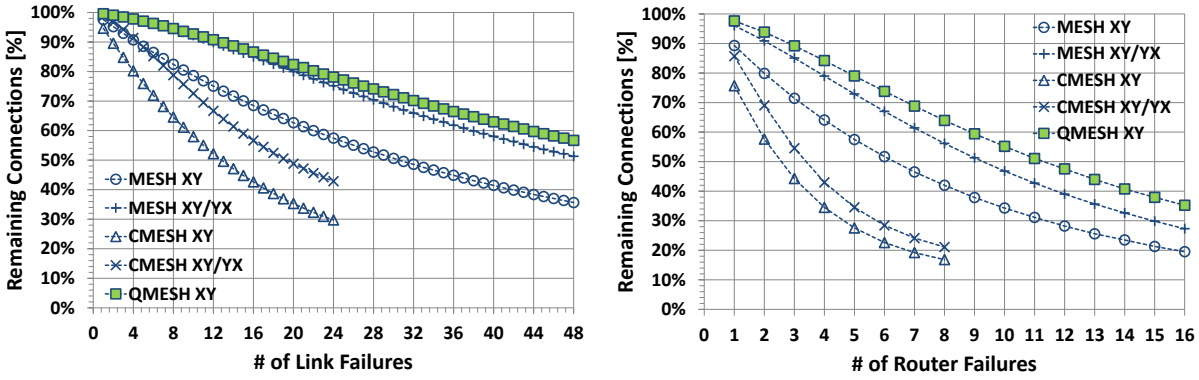


Figure 3.11: Wear-out impact of the QMesh due to decreased mean router temperatures

To quantify the potential wear-out benefits of the QMesh over the 2D-Mesh, the bar-chart in Figure 3.11 contains the reliability improvements results for all simulation setups of Section 3.3.1 (bar labelling: <NoC size> <traffic pattern>). Therefore, the mean router temperatures at the lowest simulated injection rates (MIN or PIR_{LOW} in Appendix pp. 179-182) and the highest injection rates (MAX or PIR_{HIGH} in Appendix pp. 179-182) inside the comparable regions are used to calculate the resulting reliability improvement due to the QMesh modifications. The depicted results show the direct reliability impact of the achievable temperature spreads. In comparison to the 2D-Mesh, the QMesh can increase the mean router lifetime by around 10% up to 132% over all evaluated cases. The concrete temperature related wear-out delay depends on the mean traffic intensities and pattern the NoC must perform. Higher traffic intensities can be achieved in the 4x4 NoC configurations and the mean lifetime improvement is around 58% for these cases. The 8x8 NoCs saturate at lower injection rates and the observed mean lifetime gain is around 34%. But for completeness, it has to be mentioned that this is only an estimate and furthermore the heating impact of the processing/memory resources inside the tiles has been neglected. If the temperature dissipation of these resources is dominant the thermal benefits of the QMesh can only consolidate partially into longer lifespans of the NoC due to reduced wear-out.



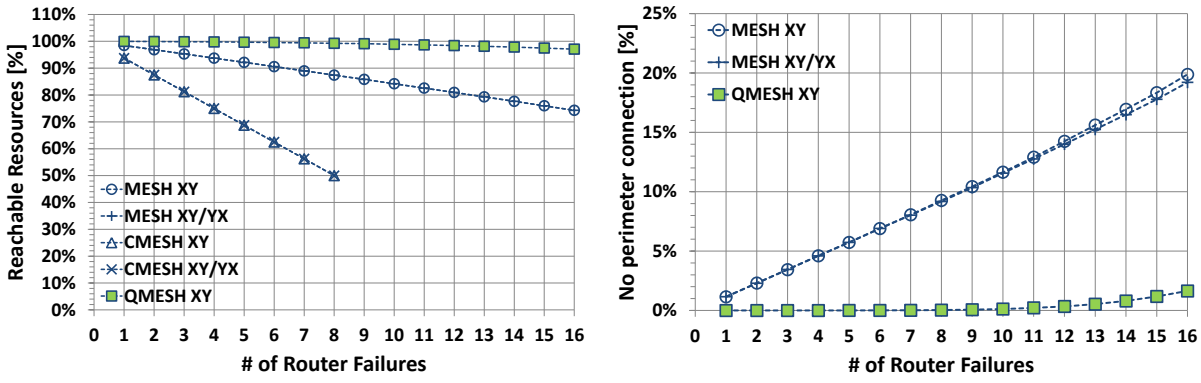
(a) Remaining ETE connections at link failures

(b) Remaining ETE connections at router failures

Figure 3.12: Remaining ETE resource connections in presence of uniformly distributed link/router failures

After the wear-out has manifested in permanent failures, the second aspects need to be evaluated to rate the degradation through possible workarounds provided by the NoC. Therefore, a fixed number of permanent failures are injected with a random uniformly distribution along unidirectional links (uplink or downlink) and routers in an 8x8 tile NoC configuration. As topology the 2D-Mesh, the CMesh (see Figure 2.9 p. 42 in Section 2.1.2 pp. 40-44), and the QMesh are applied. Furthermore, for the CMesh and the 2D-Mesh xy- as well as dual path xy/yx-routing is considered, while the QMesh operates via normal xy-routing and its inherent dual path options (see Figure 3.7 p. 71). The number of link failures is stepwise increased from 1 up to 48 for the 2D-Mesh and the QMesh, while the CMesh operated with an upper limit of 24 link failures due to its applied concentrations. A permanent failure inside a unidirectional link equals a permanent failure of an input or output port. The maximum number of full router failures is limited to 16 for the 2D-Mesh/QMesh and to 8 for the CMesh. After the random injection of the current failure pattern, all of the $N_X \cdot N_Y \cdot (N_X \cdot N_Y - 1)$ source-destination pairs are analyzed for a broken ETE connection under consideration of the integrated routing capabilities (single or dual path). This is realized in 10000 runs for each failure pattern and afterwards the mean values for each pattern-topology-routing combination is calculated.

In the first step, the general ETE connectivity degradation is evaluated for the presence of permanent link and router failures. The plots in Figure 3.12 (a/b) show the results of the remaining ETE connections over the variation of link/router failure pattern and normalized to the overall amount of $N_X \cdot N_Y \cdot (N_X \cdot N_Y - 1)$ source-destination pairs. The connectivity of the CMesh degrades fastest due to the reduced number of router and links. The xy-routing as well as the xy/yx-routing 2D-Mesh outperforms the CMesh configurations, but the QMesh is superior to all other evaluated NoCs and maintains the best degradation. The results confirm the reliability expectations formulated at the introductory paragraph of this chapter (see p. 55 and below) via the formalized comparison of the different path situations as depicted in Figure 3.2 (see p. 60).



(a) Resource-to-Resource connectivity at router failures

(b) Resource-to-Perimeter connectivity at router failures

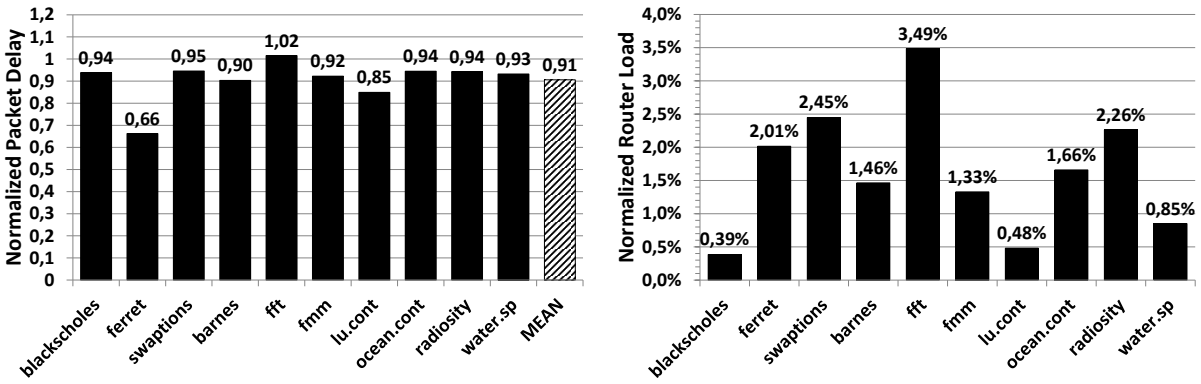
Figure 3.13: Resource connectivity characteristics in presence of uniformly distributed router failures

As next step, the resource isolation due to the presence of permanent router failures is evaluated. Therefore, the NoC configurations are analyzed for every applied failure pattern regarding the remaining fraction of reachable resources via valid ETE paths. As Figure 3.13 (a) shows, the CMesh degrades fastest if the number of router failures increases, because at least four resources are isolated with every router failure. The 2D-Mesh loses at least one resource with every router failure and the application of dual path xy/yx-routing cannot outperform the xy-routing due to the common injection/ejection terminals, which makes workarounds impossible if they are affected by the applied failure pattern. The QMesh is able to retain nearly all resources as reachable for a wide range of failure pattern und clearly outperforms the other NoC configurations. In this context, the resources along the perimeter of the NoC are very important, because most commonly they integrate the basic functionality for system input/output operations, such as memory controller or other peripheral devices. Thus, they are essential for global access to and from all NoC resources. Thereby, Figure 3.13 (b) illustrates the fraction of resources isolated from communication with the tiles at the perimeter over a variation of the failure pattern. The QMesh proves again that it is superior to the 2D-Mesh configurations and its ability to provide optimized degradation in the presence of permanent router faults. While the fraction of perimeter-isolated resources grows linearly up to circa 20% for the 2D-Mesh NoCs, the QMesh curve remains flat and does not exceed 2% even if 25% of all routers in the 8x8 NoC are unavailable due to a permanent defect.

3.3.3 APPLICATION WORKLOADS

The last part of the experimental evaluation contains the results of the full CMP simulations as outlined in the Appendix (see pp. 157-166). Thereby, the packet delay impact of the QMesh is expected to be similar to the results of the synthetic hotspot CDF pattern, because the private L2-Cache setup as well as the positioning of the memory controller to the same hotspot locations should result in dominating global traffic with these dedicated resources. Figure 3.14 (a) provides a packet

header delay comparison, where the collected values for each application case in the QMesh are normalized to those of the 2D-Mesh. As this comparison show, the QMesh lowered the mean packet delay of around 9%, but unfortunately not as much as expected (at least 38% in hotspot CDF pattern as shown in Figure 3.10 p. 77 → 8x8 HOT 20%).



(a) Normalized Packet Delay

(b) Router Load Analysis

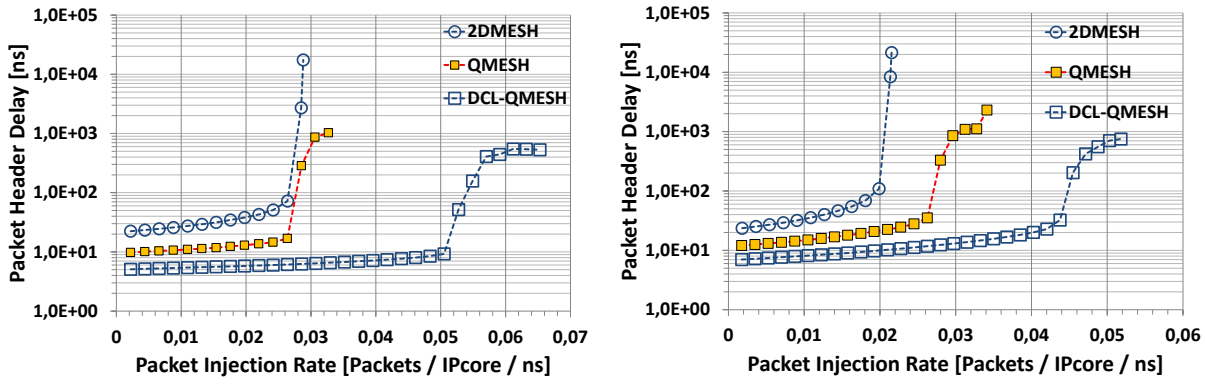
Figure 3.14: Normalized packet delay and injection rates for selected benchmark applications in an 8x8 QMesh

Furthermore, the total application processing times are only affected by minimum reductions around one up to two percent. This limited influence of the QMesh modifications on the performance was unexpected and therefore an additional analysis is processed to find the major reasons for this. The main reason for the lower packet delay reductions can be found in the different abstraction levels of the NoC infrastructure in both simulators. While the SystemC-based simulator offers a fine-grained cycle-accurate model of the router pipeline, links and constrained buffer resources, the Sniper simulator abstracts the NoC via queue models and history trees for the time slot management. Furthermore, Sniper operates with unlimited sizing of buffers and concurrency of traffic flows along the NoC will find more relaxed conditions during simulations in Sniper than in the SystemC-based setup, which diminish the QMesh advantages. Additionally, the generated NoC traffic load by the applications is very low. The bar-chart in Figure 3.14 (b) illustrates the mean router load in the 2D-Mesh due to the applications that is normalized to the mean router load resulting for the same NoC under the random uniform CDF at the minimal simulated injection rate of 0.002 packets per tile and clock cycle. In maximum, the applications reach 3.5% (fft) of the mean router loads evaluated as minimum traffic in the SystemC simulation setups.

The reason for the minimal impact of the QMesh on the total processing time arises from the fact that communication represents only a fraction of this time in addition to computation and synchronization. Thus, reduced packet delays can only contribute partially to the reduction, if the communication time represents a significant amount in comparison to the computational episodes, the synchronization delays, and the memory access times. But as the above discussed statistics of Figure 3.14 (b) indicate, this might not be the case regarding the low traffic loads.

3.4 SUMMARY AND OUTLOOK

The proposed QMesh topology of this chapter proved as a valuable adaptation of the fundamental xy-routing 2D-Mesh without losing its physical advantages, generic structure, and simplicity. The integration of multiple spatially independent terminals per resource offers a valid alternative to optimization strategies that apply the integration of more router-to-router links for improved network connectivity. **Under consideration of deterministic xy-routing and static path table setups, it is shown that the QMesh outperforms the 2D-Mesh regarding the achievable network saturation levels and packet delays for a wide range of traffic patterns at moderate increases of the costs, is superior to the 2D-Mesh regarding wear-out and degradation in the presence of permanent failures, and offers enough degrees of freedom to establish adaptive traffic management upon this structure that supports the combination of online and offline policies by reprogramming the path table.** In comparison to the NR-Mesh of *Camacho et. al.* in [168], [169] the QMesh remains deterministic and avoids any random selection policies, which is a major feature to forecast the effect of online adaptivity and guarantee traffic predictability in a global context. Random path selection at the packet-level does not provide any computational speedup if pakets does not arrive in the order required by the computational instance to proceed correctly. The ordered processes of communication and computation are tightly intertwined. Furthermore, the available routing options are reduced to the subset of two spatially independent paths, where the path table represents the entry-point for adjustments. The path option assignments and injection/ejection terminal pairing is strongly oriented on the spatial locality relation of source and destination. The focus of the next investigations in the work at hand is on the advanced exploitation of the QMesh capabilities via adaptive run-time traffic management.



(a) Rentian R=0.3 at 8x8 NoCs

(b) Rentian R=0.7 at 8x8 NoCs

Figure 3.15: Mean packet header delay as function of mean packet injection rate for the rentian CDF pattern

An alternative improvement for the evaluated unmanaged QMesh of this chapter might arise from the special case of the NN communications without router-to-router link traversal. As the discussion of the NoC timing in the NoC component evaluation (see Appendix pp. 167-175) show, these links are

limiting the achievable frequency in the 2D-Mesh and the router pipeline has further potential to operate at least at a doubled frequency. The NN communication in the QMesh represents a potential case for such a scenario, because only a single router and the short NI links need to be passed. Thus, if the source would be able to switch between the normal multi-hop clock and a NN clock with halved clock of the first, the NN traffic could experience lower latencies and higher bandwidths. Therefore, the shift to source-synchronous hops in the GALS strategy can be a valid option [81], [185], [186].

For the initial estimate of the potential of this dual-clocked approach (DCL-QMesh), the same rentian CDF traffic scenarios for the 8x8 NoC, as used for the results in Section 3.3.1, are simulated. But this time the normal clock period of 1ns is reduced to 0.5ns for the NN communication along single hop paths. The resulting curves of the mean packet header delay as function of the mean packet injection rate per resource are illustrated in Figure 3.15 and show that the improvements due to this adapted DCL-QMesh are very promising. While the single-clocked QMesh increased saturation levels around 10% (8x8 RENT R=0.3) and 43% (8x8 RENT R=0.7) in comparison to the 2D-Mesh (see Figure 3.9 p. 77), the dual-clocked QMesh raises those levels by 125% (R=0.3) and 141% (R=0.7) in both evaluated scenarios. Thereby, the impact for the router architecture is neglected and other side-effects are not considered anyway. But as outlook for future investigations on this topic of multi-ported NoCs, the proposed research scope of multi-clocked designs could mark an interesting path, which further correlates with the intra-tile traversal of packets over the crossbar as mentioned in Section 3.2.1 (see pp. 66-70).

4 REGIONALLY ADAPTIVE RUN-TIME TRAFFIC MONITORING

Monitoring mechanisms are essential for the consolidation of self-awareness in CMP systems and the aggregated observation data represents the basis for coordinated adaptation as well as the required feedback in proactive/reactive run-time management [59]. Due to the growing share and impact of the communication on the workload processing in modern CMPs [11], [47], [49], especially traffic monitoring becomes indispensable for coordinated optimization along different management services. This will be underscored by the following issues:

- The characteristic behavior of workload applications can be abstracted as alternating sequences of communicational and computational episodes (load → compute → share/update/store data) with variable durations/intensities [53], [67] and dominating communication flows [19]. The communication timing indicates begin/end of computational activity, the amounts of transferred data outline the bandwidth requirements, and the locality of data transfers indicates the affected resource.
- The knowledge of communication characteristics is required by application management services (i.e. in the format of application communication graphs) that handle the composition, mapping and scheduling of workload applications for optimized operations on the CMP [17], [22], [98], [111], [130], [182], [187], [188].
- Run-time traffic management services operate on observed network status information to provide load balancing [60]–[63], [125], quality-of-service guarantees [59], [122], [189], [190], degradation optimized workarounds in the presence of component failures [39], [40], and reliability tradeoffs by adjustment of the EDAC intensities [191], [192].

By now, as the proposed QMesh of Chapter 3 already offers the necessary actors/switches for the applied traffic management via its programmable path tables, the next step of investigations has to focus on the observability of the communication begavior inside the NoC at run-time. Therefore, an adaptive traffic monitoring solution is introduced and explored in the following section.

4.1 RELATED WORK

The survey of *Kornaros et. al.* provided in [59], shows that monitoring is a general concern for many run-time mechanisms in modern CMP systems and different state-of-the-art solutions are already present. To contrast and compare existing monitoring approaches, an initial classification regarding the relevant features for the work at hand is provided. Thereby, the main criteria of monitoring in NoC-based CMPs are as follows:

Application Context: The common reason for the integration of monitoring in NoC-based CMPs is a combination of performance, power, and reliability optimizations at run-time. But therefore dedicated target domains/aspects exist. Most commonly, ***traffic monitoring*** (TRM) is applied to improve the communication characteristics via ***adaptive load balancing*** (ALB) or ensure ***quality-of-service*** (QoS) for designated workload fractions. Alternative domains are given by the ***power management*** (DVFS), ***mixed parameter observations*** (PVT) for reliability/thermal management, or ***debugging/profiling*** of workloads (DBG/PRF).

Spatial Scope: It specifies the spatial range of the data the monitoring entities cover. Therein, the monitoring may be constrained to observations that are gathered ***locally*** by each tile (or the NN proximity), ***regionally*** along constrained multi-hop areas, and ***globally***.

Temporal Scope: This feature describes the frequency and periodicity of observations. Thereby, in ***continuous*** monitoring the gathered data is provided each clock cycle and this is a typical use case for distributed adaptive routing. ***Periodic*** monitoring provides the data in fixed time intervals. ***Sporadic*** monitoring refers to event-based observations (i.e. event-code and timestamp) if a defined threshold or counter limit (CNT) is crossed.

Component Coverage: This feature describes the covered component levels for the monitoring by sensors or probes. ***Tile*** and ***path*** represents the levels with the highest abstraction. Thereby, at the *tile* level the traffic injection rates (IR), performance counter for designated instruction events, or voltage/temperature/delay observations for DVFS/PVT are favored measures. Inter-tile communication is most commonly covered along ***paths***, ***routers*** and ***links*** as abstraction levels. *Paths* can be further differentiated as simple ETE connection, as concrete ETE connection between dedicated threads/tasks (CNX), and as router to tile (RTT) connection. The *router* level aggregates monitoring data for a hop or dedicated directions across the *links*/buffer of its in/output ports. The *link* represents the lowest level covered by typical monitoring solutions via direct sensor/probe measurements.

Adaptivity: The monitoring might allow adjustments to reconfigure the ***spatial***/***temporal*** scope and the component ***coverage***. Those adaptations can be supported only ***statically*** during design-time (DT) or ***dynamically*** during run-time (RT). Furthermore, hybrid approaches may fix the allowed limits/ranges for run-time adaptations at design-time (DRT) due to tradeoffs between observability, flexibility, and costs.

Organization: This feature determines the interplay of monitoring data flows, their sensing, aggregation/reduction, and storage/evaluation along the responsible entities. Thereby, typical options are ***distributed***, ***hierarchical***, and ***centralized*** organization schemes. In *distributed* or *flat* monitoring structures, many entities are exclusively responsible for the combined sensing,

aggregation, evaluation, and storage activities. No global instance exists. Contrary to this, in *centralized* structures all observations flow to a central instance, which is responsible for the collection, evaluation and storage. Thereby, aggregation or reduction may occur at intermediate entities between the origin of an observation and the central instance. *Hierarchical* structured monitoring represents an approach between the previously mentioned points and introduces multiple successive aggregation/reduction/evaluation levels along the responsible entities.

Infrastructure: In addition to the structural organization of the monitoring data flows along the responsible entities, the traversals must be mapped onto a physical infrastructure to bring the corresponding data from one location to another. This can be realized via ***shared*** communication over the existing infrastructure, a ***specialized*** infrastructure that is customized for this single use case, or a ***unified*** infrastructure via the multi-network approach as general communication instance. *Shared* communication can be realized as normal traffic, as piggy-back (PB) approach by appending monitoring data to application packets, or prioritized traffic via assignments to designated VCs.

Data Evaluation: The integration of the processing of monitoring data is mainly influenced by the organization, spatial/temporal scope, and application context. In general, the data evaluation can be performed ***online*** and ***offline***. Normally, run-time based management applies *online* processing, but additional data evaluations may be performed *offline* as well for profiling scenarios. Other major aspects in the *online* evaluation are its implementation and if it is integrated as exclusive out-of-band resource (OFB). Therefore, the use of specialized or partially programmable ***hardware*** as well as flexible ***software*** solutions can be considered.

Reusability: The integration of monitoring mechanisms comes along with hardware and software efforts that represent an overhead. Regarding the demand of system states at different management services, ***data*** *reusability* and *accessibility* would increase the cost-benefit-ratio. The same issue is relevant for the reusability of *dedicated* or *unified* ***infrastructures***. Reuse might be an option for the *data evaluation* if the processing is performed on *exclusive out-of-band resources*, but the workload situation does not require any active monitoring.

This collection of criteria shows that there is a potentially wide-ranged design-space of monitoring solutions, but not all combinations would fit the constraints imposed by the management services that apply the monitoring. Therefore, this classification is applied to a representative selection of state-of-the-art monitoring solutions (the corresponding publications has to provide enough details) to screen possible combinations and the results are summarized in Table 4.1.

It becomes obvious that traffic management is the preferred application context in NoC-based CMPs. Thereby, in distributed adaptive strategies the monitoring is outlined by the requirements of the routing. The monitoring works on a statically assigned local [64], [140] or regional spatial scope [61]–

[63] with continuous timing. The infrastructure is maximally specialized [61]–[64], [140] to reduce the implementation costs and most commonly the hardware probes at routers propagate aggregated indirect property measures from their own [64], [140] and/or merge them with incoming values from the routers inside assigned regions [61]–[63]. The monitoring is tightly merged with routing mechanisms as autonomous system layer. Data or infrastructure reusability is not intended [61]–[64], [140]. A similar, but globally oriented approach with periodical timing intervals is given in [135], where each router contains a table with path delay estimates to every destination tile in the NoC. Therefore, every router periodically senses local delay estimates via the number of blocked buffer slots at each port, updates its internal delay estimates to all destinations, and propagates the updated data to routers in its proximity through a specialized network. Thus, after certain periods each router has a global view on the network from its individual location (another scalability optimized version is reduced from global to square-shaped regions). In [187], a distributed dynamic cluster/region monitoring applies a periodic diffusion of locally sensed network traffic status information to propagate it into the cluster as well as between clusters via reserved VCs in a shared infrastructure. Another periodic monitoring is applied in [142] to observe buffer utilizations at tiles/routers and forecast traffic hotspots in spatial regions of fixed sizes. Therefore, the monitoring data is transmitted through a specialized infrastructure to a centralized out-of-band prediction hardware controller.

Alternatively, event-based distributed monitoring of path-based connections can be found in [189], [190], [193], [194]. In [189], [190], each router along a path observes its buffer utilization and compares it with a defined threshold for a dynamically contracted ETE connections (CNX). If the needed buffer space is not met, an indexed congestion bit for each affected router along the path is set. This happens in a piggy-back manner at traversing data packets that belongs to this connection. The NI at the destination evaluates this path congestion information against a given QoS threshold and notifies the source at violations. The solution in [193] observes the routers and links along contracted path connections (CNX) for a defined set of failures via programmable threshold counter. On violations at the routers or destination, the source of the connection is informed via shared communication along a reserved virtual channel. In [194], the monitoring of [193] is integrated into a hierarchical cluster-based run-time solution with additional application mapping and management via cluster- and global-agents, whose action are triggered by the generated events and processed in software. The monitoring in [53] focus application specific traffic classification as bandwidth- or latency-sensitive by logging parameter for intensity and length of communication episodes as well as cache-miss events at the tile if an application run/change occurs. The classification is used to assign traffic of specific applications to one of two networks, where each network is optimized for one of

the classes. Furthermore, the applications are ranked for VC prioritization along their assigned networks.

Table 4.1: Classification for a representative selection of state-of-the-art monitoring solutions in NoC-based CMPs

FEATURE	OPTIONS	EXAMPLES													
Application Context	TRM	ALB	ALB	ALB	ALB	QoS	ALB		QoS	ALB	ALB			QoS	QoS
	DVFS							■							
	PVT							■							
	DBG/PRF											■	■		
Spatial Scope	Local											A			
	Regional	■	■	■		■	■		■	■	■				
	Global				■			■		■		B	■	■	■
Temporal Scope	Continuous	■	■	■											
	Periodic				■		■	PVT		■		A/B	■	■	■
	Sporadic					CNT		DVFS	CNT		CNT				
Component Coverage	Tile						■	■							
	Path					CNX			CNX	RTT				CNX	
	Router	■	■	■		■	■		■		■	■	■		
	Link				■	■				■	■	■	■		■
Adaptivity	Spatial	DT	DT				DT		■	DT	DRT	B			
	Temporal				RT	DRT		RT			RT	A/B	■	RT	RT
	Coverage														
Organization	Distributed	■	■	■		■			■	■	■	A			
	Hierarchical					■									
	Centralized				■		■	■				B	■	■	■
Infrastructure	Shared					VC			PB		VC	■			VC
	Specialized	■	■	■	■		■	■		■			■		
	Uniform													■	
Infrastructure Reusability														■	
Data Evaluation	Online	■	■	■	■	■	■	■	■	■	■	■		■	■
	Offline						ANN						■		
	Hardware	■	■	■	OFB	■	OFB		?	■	■				
	Software					■		■	?			■	■	■	OFB
Data Reusability								■			■			■	■
TRUE	OPTIONAL	[62]	[61]	[63]	[60]	[193]	[142]	[195]	[190]	[135]	[187]	[196]	[197]	[122]	[198]

Traffic monitoring solutions with global spatial scope, centralized organization, and periodic time scope are applied in [19], [60], [122], [125], [198]. In [60], [125], each router senses and aggregates the loads of its outgoing links, which is transported and merged into maps via a specialized infrastructure to a central out-of-band hardware controller that evaluates path updates. In [122], each tile measures traffic injections for contracted thread-to-thread connections along deterministic paths and reports those observations to a central software instance. There, the observations are periodically compared to reference data from application graphs. Similarly, the approach in [19] tracks the per destination packet injection rate via performance counter at each tile and aggregates it in a central service instance of the operating system to classify traffic pattern, find dominant flows and predict changes periodically. The traffic data is used to provide a communication-aware globally-coordinated NoC that is partially reconfigurable for different traffic classes. The solution in [198] applies an event-based monitoring from [199] to capture the link utilization, but the evaluation of the aggregated events is performed periodically in software on a dedicated out-of-band resource.

Multiple use cases with central organization of monitoring at the tile level in NoC-based CMPs are provided for the MNoC solution in [195], [200]. The MNoC is a specialized static NoC that was reduced to a tree-like topology and serves a fixed central instance with observation data of core delay and temperature from all tiles. The proposal in [201] applies thermal sensing in cooperation with DVFS as well, but utilizes a static hierarchical organization with per tile, per cluster and centralized monitoring as well as evaluation responsibilities. The interconnection of monitoring instances is realized with a specialized tree-based infrastructure. The work in [202] introduces a similar concept, but considers an fully independent on-chip monitoring layer via three-dimensional CMOS integration.

Debugging and profiling is scoped by the approaches in [196] and [197]. In [196] each router is suited with a probe to take periodical snapshots of the detailed context in its pipeline and buffer. The snapshots are stored in a reserved fraction of the L2 cache and evaluated locally for each router in periodic timed intervals at the corresponding processor of the tile. Afterwards, a global evaluation on all aggregated snapshots at a central instance takes place if errors were detected. The local and global evaluation steps are implemented in software. While the approaches in [196] and [197] scope prototype debugging and verification, the event-based run-time monitoring in [199] was originally intended for the support system-level and application debugging in final products. Partially programmable monitoring probes with event-generators are attached to each router and propagate observed events via shared communication along reserved VCs to a central monitoring instance. Splitting the NoC in different observed regions will be possible as well with static placement of the central aggregation units. The subsequent work in [203] extends the solution of [199] by the introduction of different event abstraction levels for overhead tradeoffs.

In addition to the specific monitoring proposals in the works above, related publications on the evaluation of important aspects exist. The infrastructure is important for the costs, performance, and flexibility of the monitoring. Therefore, this issue is targeted by different work as primary topic. In [204], [205] *Guang et. al.* evaluate the shared (with and without VC), unified and dedicated (tree-based) infrastructure solutions in the context of a hierarchical organized scenario with four levels of monitors statically placed. The works show that physically separated infrastructures are more energy-efficient and provide lower communication latency. Due to the optimization of the dedicated infrastructure for the static spatial traffic scenario, this choice performs best. In [195], *Zhao et. al.* provide similar results regarding the benefits of a physically separated monitoring infrastructure in comparison to shared communication, but a unified approach was not considered. The works of *Ciordas et. al.* in [206] and [207] focus the integration of monitoring as part of the system design flow and prefer shared communication via VCs. But the evaluation is debugging centric and unfortunately

the reuse of physically separated infrastructures for general control flow scenarios as in [122], [130] was not considered. Another important aspect is the organization/management and focused by the investigations of different works with major focus on hierarchical agent-based hardware-software integration strategies and corresponding application scenarios [194], [208]–[213].

4.2 TRAFFIC MONITORING CONCEPT

Prior to the introduction of the monitoring solution, the targeted requirements, capabilities and constraints are outlined. In the work at hand, the ***basic data flow of a monitoring solution*** is considered as ***pipeline of three subsequent stages*** called ***sensing, aggregation and evaluation***. These stages have to be arranged and integrated into the NoC-based CMP in a way that the resulting monitoring supports enough ***observability***, ***flexibility***, and ***adaptivity*** to meet the ***requirements*** of modern dynamic workloads. Thereby, observability is considered as a feature that must be provided at run-time to realize a modern self-aware management. This enables the system to integrate reactive and proactive self-configuration functionality. The provided flexibility and integrated measures to adapt the monitoring behavior specify the exploitable use case scenarios. Thereby, the ***flexibility aspect*** is targeted as follows:

- The run-time traffic monitoring must provide sufficient status information to be utilizable for adaptive traffic management [60], [125], traffic classification [19], [53], application mapping and profiling [17], [110], [130], [214]. Hence, the ***reusability of captured observation data*** as well as ***adequate resource coverage*** via sensing is required.

- ***Unified physically separated monitoring infrastructures*** with general connectivity to all tiles have to be applied [122], [130], [204], [205]. This is indispensable to support different use cases with spatio-temporal variability in their corresponding monitoring data flows. Furthermore, such infrastructures are highly reusable and customizable for the control data traffic domain, with minimal implications/interdependencies to the main infrastructure for workload data transfers.

- The stages of the traffic monitoring pipeline should be implemented preferably in software to create a solution that is adjustable regarding the evaluation policies of observation data. Furthermore, software modules are migratable at run-time and offer better prospects for bindings with other system management services. The key challenge is the balanced splitting between ***hard- and software*** regarding the required monitoring performance.

Contrary to the majority of the presented solutions in the related work (see Section 4.1 and Table 4.1 p. 89), the traffic monitoring of the work at hand is not maximally optimized for a single use case. Instead, the implementation costs are addressed by the increased reusability of data and

infrastructure, which improves the cost-to-benefit ratio of the necessary investments and allows different management services to profit.

Sufficient ***adaptivity*** is required to provide adjustments of the monitoring regarding the dynamic run-time behavior of workloads as well as use cases. Therefore, the intended solution is configurable in its spatial scope, temporal scope, and component coverage as follows:

- ***Workload***: Multiple co-existing applications inside the NoC-based CMP are considered. Those applications may be mapped to different subsets of the CMP tiles or composed onto shared sets of tiles [17], [107], [110], [111], [130], [214]. Thereby, locality-optimized mappings into closely-shaped regions of tile-sets are typical due to the greedy nature of the applied mapping heuristics [5], [11], [12], [14], [44], [51], [67], [111], [215], [216]. The assigned workload fractions in each of those regions can behave differently and therefore imposes individual demands on monitoring, while inactive regions does not need any surveillance.
- ***Spatial Scope***: According to the assumed workload characteristics above, it must be possible to specify dedicated ***regions-of-interest*** (ROIs) or ***clusters*** that co-exist and do not overlap. Furthermore, those clusters have to be re-locatable as well as re-sizable at run-time. Hence, only the necessary parts of the CMP infrastructure become observed.
- ***Temporal Scope***: Each cluster contains a workload fraction with individual variability in its communication behavior and intensity. Therefore, the timing must be configurable by the selection of dedicated event-thresholds or monitoring intervals (continuous monitoring will not be considered due to its overhead of observation data). Furthermore, the selectable timing resolutions must satisfy the requirements of all targeted use cases.
- ***Component Coverage***: In general, full coverage of ***tile output***, ***ETE path utilization,*** and ***output port/link utilization*** is considered to provide sufficient state information for all desired use cases. But for light-weighted observation of clusters with lower traffic intensities, the coverage should be configurable to trade off observation data overhead as well as processing time for evaluation.

4.2.1 PRELIMINARY CONSIDERATIONS

Prior to the concept description, some preliminary considerations regarding the selection of temporal scope, spatial scope and organization are formulated. Afterwards, the basic concept is introduced and discussed. Initially, basic parameter and relations will be introduced as follows:

- The CMP with the mesh-based NoC contains N_{tile} tiles and a cluster represents a closely shaped subset of N_{CS} tiles. A cluster is allowed to contain a maximum number of N_{CSmax} tiles. ($N_{tile}, N_{CS}, N_{CSmax} \in \mathbb{N}\ and\ N_{CS} \leq N_{CSmax} < N_{tile}$)
- Each tile in the CMP contains N_S sensors that provide single observations. The maximum number of provided path sensors per tile is N_{CSmax}, whereas N_{CS} sensors are used for the current cluster. The number of output port/link sensors per tile is equal to the number of ports N_P per router. The number of tile output sensors is N_{OS}. ($N_S = N_P + N_{CS} + N_{OS}$)
- For all of the $N_S \cdot N_{CS}$ sensors inside the cluster, a homogeneous configuration is assumed. Heterogeneous setups with individual configurations per sensor would exceed affordable limits for adaptive reactions.
- The final utilization values for NoC components are considered in the range of 0 up to 100 percent with at least one configuration mode that resolves this scale in steps of one percent.

Before the concept is introduced, the choices for temporal scope and organization scheme must be made as entry-point. Regarding the temporal scope, the periodically timed monitoring approach has advantages over the event-based solution.

- The periodic timing provides lower data overheads per observed sensor when compared to the event-based approach. A fixed monitoring cycle/interval (T_{MC}) determines the moment when each sensor provides its raw observations data and the data of all sensors per tile can be aggregated into a single vector structure with index-based identification. These characteristics make the data overhead of the event-based approach [59], [199], [203] obsolete and avoids synchronization issues [199].
- The periodic timing provides fixed time intervals for the monitoring data evaluation, which comes along with lower computational efforts compared to the event-based approach. The event-based solution would require prior parsing of the collected events to detect if the data must be considered for the current time window.

For the organization scheme a centralized approach per cluster is favored over the distributed one, because one single master-tile is responsible to provide the global data view and monitoring evaluation for the N_{CS} tiles of the cluster.

- Each of the $(N_{CS} - 1)$ slave-tiles transmits its monitoring data to this master-tile as many-to-one traffic pattern and the data evaluation follows at this designated cluster instance. The communication efforts, storage requirements and the number of occupied tiles will be lower than in the distributed organization due to the lower redundancy as well as minor data

sharing efforts, but the centralized instance might be the potential bottleneck if the allowed cluster size exceeds affordable limits.

- The many-to-one traffic pattern inside each cluster makes the communication behavior and path traversals more predictable. Thus, the infrastructure reuse can be improved by interference reductions if other services have similar pattern and the mapping choices of the corresponding central instances allow diametrical placements.
- If the monitoring evaluation running on a master-tile is integrated as software module, the single point of failure issue of the centralized organization can be avoided. Software is migratable and the master-role can be delegated to another tile.

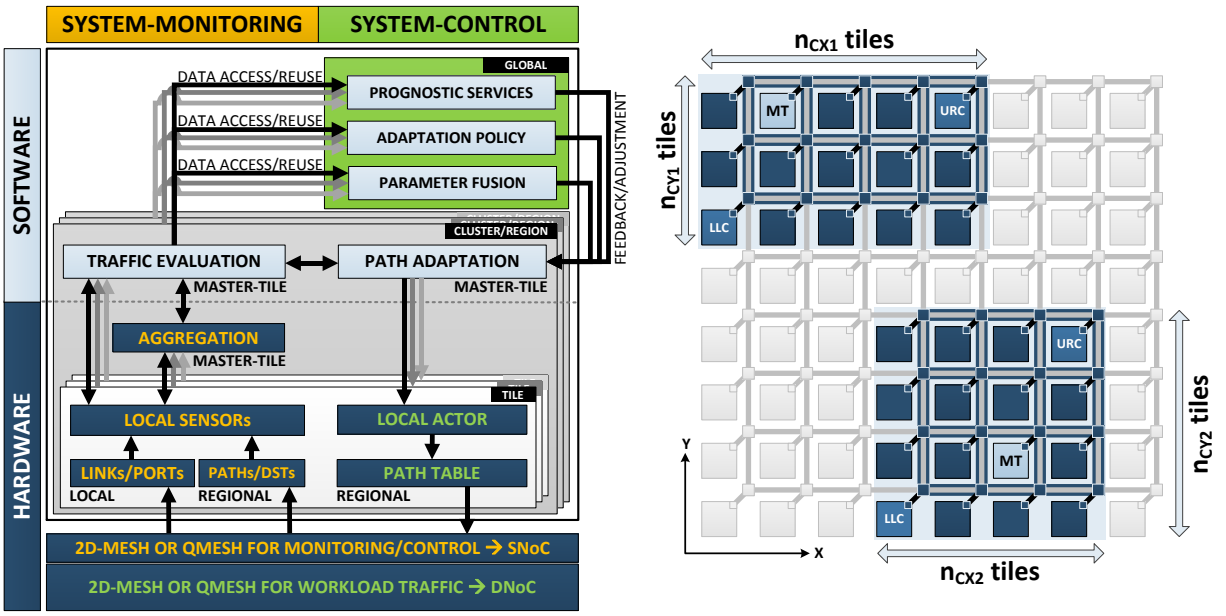
As result, the combination of a periodic time scope with a centralized organization is the best choice for the targeted traffic monitoring in the work at hand due to the expected resource-efficiency. But this is only the entry-point for further investigations, because different challenges remain.

- The periodic data transfers of $(N_{CS} - 1)$ slave-tiles to a central master-tile reveal that the monitoring data sources dominate the timing configurability and the time window for possible computations. An adaptation of the monitoring requires N_{CS} full transactions to change the monitoring cycle globally for the cluster. Hence, modifications of the sensor data and aggregation flow should enforce that timing adjustments are supported locally and fast in the master-tile as well.
- After a monitoring cycle is finished, the reception and data takeover for all $(N_{CS} - 1)$ monitoring packets represents a non-negligible offset before aggregation and/or evaluation can be performed. Hence, modifications to support the rebalancing of those final reports as part of the previously completed monitoring cycle are preferable over of burst-transfers with high bandwidth requirements in short time windows at the end.
- Finally, the computational efforts must be reduced to enable a centralized monitoring for greater clusters, to decrease the resource utilization of the software module on the master-tile, or to decrease the latencies prior to the processing of subsequent data evaluations.

4.2.2 CONCEPT

The prototypical framework for the basic system management functionalities was already sketched in Figure 1.3 of Section 1.2 (see p. 23). The diagram in Figure 4.1 (a) below, maps the outlined traffic monitoring/management on this framework with its intended spatial integration, organization structure, and implementation characteristics. While Figure 4.1 (a) shows the complete approach, the traffic monitoring context itself is covered by the left half of this diagram. Now, the basic

functionalities and data flow concepts for the traffic monitoring pipeline stages will be introduced and discussed. This starts with the specification and formal description of the ROI or cluster for the traffic monitoring within the background of mesh-based topologies.



(a) Traffic Monitoring Concept (b) Two cluster in an 8x8 2D-Mesh

Figure 4.1: Functional framework of regional adaptive traffic monitoring and clustering

CLUSTER: A group of N_{CS} tiles that form a closed and rectangular shaped region inside the NoC-based CMP, which is specified by the diametrical opposed tile positions of the lower left corner (LLC) and the upper right corner (URC) along its perimeter tiles (see Figure 4.1 (b) above).

- **LLC** $\rightarrow \{x_{LLC}; y_{LLC}\} \Longrightarrow 0 \leq x_{LLC} \leq (N_X - 1) \quad and \quad 0 \leq y_{LLC} \leq (N_Y - 1)$
- **URC** $\rightarrow \{x_{URC}; y_{URC}\} \Longrightarrow x_{LLC} < x_{URC} \leq (N_X - 1) \quad and \quad y_{LLC} < y_{URC} \leq (N_Y - 1)$
- **Sizing** $\rightarrow N_{CS} = n_{CX} \cdot n_{CY} \Longrightarrow n_{CX} = |x_{URC} - x_{LLC}| \quad and \quad n_{CY} = |y_{URC} - y_{LLC}|$
- **Constraints** $\rightarrow n_{CX} > 0 \quad and \quad n_{CY} > 0 \quad and \quad N_{CS} \leq N_{CSmax}$

A cluster is centrally managed by a dynamically assigned master-tile (MT) out of the spanned region and is reconfigurable at run-time by this MT-instance regarding its size, location and assigned traffic monitoring job. The remaining $(N_{CS} - 1)$ tiles have a slave status.

- **Master-Tile** $\rightarrow \{x_{MT}; y_{MT}\} \Longrightarrow x_{LLC} \leq x_{MT} \leq x_{URC} \quad and \quad y_{LLC} \leq y_{MT} \leq y_{URC}$

The master-tile performs the final monitoring data aggregation and traffic evaluation in each cluster. Moreover, dynamic reassignment of this status to another cluster tile is required for the intended spatial adaptivity. In the traffic monitoring/management context, each tile of the CMP can either be an exclusive member of one or no cluster member at all. Hence, multiple independent clusters with this context can co-exist and work in parallel, but they are regionally disjoint and do not share

common tiles. The example in Figure 4.1 (b) illustrates the case of two independent clusters inside an 8x8 2D-Mesh. Overlapping with clusters of another monitoring context, such as temperature monitoring, is permitted as long as it doesn't affect the traffic monitoring functionality.

The next part of this ***concept*** description targets the ***centralized monitoring flow*** applied in the cluster introduced above. Therefore, the integrated functionality and timing dependencies of the ***traffic monitoring pipeline*** are specified under consideration of the critical issues formulated in the prior Section 4.2.1.

LOCAL-SENSORS-STAGE: The traffic sensors must be designed to register basic events (BE) along ports/links and paths that indicate data flow activities along these components to provide the measurement of utilization statistics. In the case of NoCs, the smallest data word unit that passes ports, links or NIs is given by the flit with a payload of w_{data} (see Equation 2.3 in Section 2.1.1 p. 31). Furthermore, the occurrence of those basic events is discretely timed within intervals Δt_{BE} equal to the clock period t_{NoC} of the NoC. Thus, the traffic sensor timing follows Equation 4.1.

$$\boldsymbol{t_{i+1}} = t_i + \Delta t_{BE} = t_i + t_{NoC} \tag{4.1}$$

Thereby, the occurrence of basic events for each sensor j of a tile ($0 \leq j < N_S$) is indicated by a binary control signal $s_{ctrl,j} \in \{0, 1\}$ that follows the same clocking scheme. For the registration of flit traversals, $s_{ctrl,j}$ can be sourced by the existing request/acknowledge signals of the handshake logic (HL), which is part of the flow control along the router pipeline (see Section 2.1.1 pp. 31-40 and Section 2.2.2 pp. 48-49). The number of clock cycles each flit requires for a stage traversal and before the next one can be handled, depends on the implementation of the handshakes. Typical values are one or two clock cycles. The traffic sensor itself is a simple counter $cnt_j \in \mathbb{N}$ that is incremented by one for each clock cycle the control signal $s_{ctrl,j}$ is active/high. Following the normal central organized scheme, the sensor would proceed this way over the full monitoring cycle duration T_{MC}, report the final counter value cnt_j and would reset/wrap-around afterwards. Contrary to this scheme and as first step towards an optimized monitoring flow, the work at hand introduces a defined threshold $\boldsymbol{v_{thresh}} \in \mathbb{N}$ as maximum value for the sensor counter for those resets. Thus, the counting procedure follows Equation 4.2 below.

$$\boldsymbol{cnt_j(t_{i+1})} = \begin{cases} cnt_j(t_i) + 1 & (s_{ctrl,j} = 1) \wedge (cnt_j(t_i) < \boldsymbol{v_{thresh}}) \\ cnt_j(t_i) & (s_{ctrl,j} = 0) \\ 0 & (s_{ctrl,j} = 1) \wedge (cnt_j(t_i) = \boldsymbol{v_{thresh}}) \end{cases} \tag{4.2}$$

AGGREGATION-STAGE → LEVEL 1: The first level of the aggregation stage utilizes the threshold value $\boldsymbol{v_{thresh}}$ as mechanism for data volume reductions and to enable the spreading of monitoring data reports over the monitoring cycle. Each time the sensor counter reaches the defined threshold, an

overflow flag $OFG_j \in \{0, 1\}$ is generated as defined by Equation 4.3, which is externally resettable by the signal $rst_{OFG,j} \in \{0, 1\}$. Hence, the overflow flag reduces the counter value cnt_j and thus the number of registered basic events into one single bit.

$$\boldsymbol{OFG_j(t_{i+1})} = \begin{cases} 1 & (s_{ctrl,j} = 1) \wedge (cnt_j(t_i) = \boldsymbol{v_{thresh}}) \\ 0 & (rst_{OFG,j} = 1) \\ OFG_j(t_i) & else \end{cases} \quad (4.3)$$

This OFG_j bit will be checked and reset ($rst_{OFG,j} = 1$) within a periodic time interval called sensor period t_{sp}, whose periodicity is defined by the product of NoC's clock cycle with the threshold value $\Delta t_{SP} = v_{thresh} \cdot t_{NoC}$. Hence, the monitoring cycle T_{MC} is redefined as an integer multiple $N_{sp} \in \mathbb{N}$ of this sensor period and the timing of the aggregation stage operations follows Equation 4.4 below. This does not change the timing of the traffic sensor, because t_i and t_{sp} work with the NoC clock cycle t_{NoC} as fundamental time base (see Equations 4.1 and 4.4 → $\Delta t_{SP} = v_{thresh} \cdot \Delta t_{BE}$). Moreover, this change brings the needed flexibility for the interpretation of sensor data and for timing adjustments into the monitoring flow.

$$\boldsymbol{t_{sp+1}} = t_{sp} + \Delta t_{SP} = t_{sp} + \boldsymbol{v_{thresh}} \cdot t_{NoC} \implies \boldsymbol{T_{MC}} = N_{sp} \cdot \Delta t_{SP} = N_{sp} \cdot \boldsymbol{v_{thresh}} \cdot t_{NoC} \quad (4.4)$$

An overflow bit per sensor ($OFG_j = 1$) can be generated in each sensor period Δt_{SP} at a maximum traffic load. With a configuration of $N_{sp} = 100$ sensor periods per monitoring cycle T_{MC}, this would result in a maximum of 100 overflow bits reported per sensor at full traffic load. Hence, each reported overflow bit can be interpreted as one percent of the components utilization inside this within the defined time window of the monitoring cycle T_{MC} if it contains $N_{sp} = 100$ sensor periods Δt_{SP}. The time window is still configurable via the threshold value assignment v_{thresh}. Moreover, the functional relations of overflow flag, utilization, sensor period, and monitoring cycle have the potential to reduce the computational effort per sensor down to simple overflow counting in the next level of the aggregation stage. Expensive scaling operations for sensor counts via division and multiplication would be obsolete. The bandwidth challenge of the centralized traffic monitoring is targeted with the local integration of this first level of the aggregation stage into each tile. This step comes along with the desired spread of the many-to-one communication towards the central master-tile to $N_{sp} = 100$ overflow reports over the monitoring cycle. Each tile communicates its N_S overflow bits every sensor period. One additional option for reduction of the data overhead is a functional unit that checks for every expired sensor period, if overflow reports are necessary. Equation 4.5 describes the performed data operations of this check unit. Only if overflows are present ($OFG_j = 1$) a monitoring packet must be send.

$$\boldsymbol{CHECK(t_{sp})} = \sum_{j=0}^{N_S-1} OFG_j(t_{sp}) > 0 \quad (4.5)$$

AGGREGATION-STAGE → LEVEL 2: While the first level of the aggregation stage reports the overflow bits, the second level is responsible for their counting via $load_j \in \mathbb{N}$ for each sensor of all N_{CS} tiles over the time of N_{sp} sensor periods. If the current monitoring cycle is expired, the counter values are read out and reset afterwards via an external signal $rst_{LD,j} \in \{0,1\}$. With the intended modifications those counter values would represent the utilization values for the observed components from 0 to 100 percent with a scale resolution of one percent. This level of the aggregation stage can be integrated locally per tile or centrally in the master-tile. But the centralized option is preferred to exploit the avoidance of the final communication bursts. The functional description for this second level of the aggregation stage is given in Equation 4.6 below.

$$\boldsymbol{load_j(t_{sp+1})} = \begin{cases} load_j(t_{sp}) + 1 & (OFG_j(t_{sp}) = 1) \wedge (rst_{LD,j} = 0) \\ load_j(t_{sp}) & (OFG_j(t_{sp}) = 0) \wedge (rst_{LD,j} = 0) \\ 0 & (rst_{LD,j} = 1) \end{cases} \quad (4.6)$$

An additional benefit of the centralized integration is in the flexible handling of the timing to start a new monitoring cycle. Each tile sends the overflow packets every Δt_{SP} sensor period to the central level 2 aggregation instance. Thereby, this master-tile with its reset capabilities for the overflow counter solely determines when the first sensor period of a new monitoring cycle starts. Thus, the timing of the central aggregation with successive evaluation is decoupled from the monitoring data arrival and can be configured more flexibly by demand.

TRAFFIC-EVALUATION-STAGE: With the static assumption of $N_{sp} = 100$ sensor periods per monitoring cycle the evaluation stage for the processing of the final representation as utilization would be obsolete, because the necessary computational steps are already an implicit part of the monitoring flow. But there is an additional option to exploit the timing flexibility of the central master-tile by trading off the duration of a monitoring cycle against the scale resolution of the component utilization, without the requirement of a global period adjustment at the assigned cluster tiles. Instead of a fixed scale resolution, where every counted overflow bit represents one percent of utilization (0:1:100 in %), the variable parameter $\boldsymbol{k_s} \in \mathbb{N}$ is introduced for it (0: $\boldsymbol{k_s}$:100 in %) and accordingly each counted overflow bit can be interpreted as a weight of k_s percent on the utilization scale. If k_s is increased, the granularity increases and the number of registered sensor periods per monitoring cycle changes as follows in Equation 4.7.

$$\boldsymbol{N_{sp}} = 100/\boldsymbol{k_s} \quad with \quad \boldsymbol{k_s} \in \{1,2,4\} \quad (4.7)$$

Herewith, the master-tile can influence the time intervals between monitoring data evaluations on its own. The co-domain of k_s is fixed to the numbers 1, 2, and 4, which all represent a power of two. This simplifies the indispensable utilization value conversions of the counted overflows in $load_j$ to the correct values in $sload_j$. Each counted overflow bit in $load_j$ must be interpreted as k_s percent on the scale and therefore multiplied by the factor k_s. The co-domain selection of $k_s = 2^p$ ($p = 0, 1, or\ 2$) reduces this operation to an arithmetical left shift by p bits per overflow counter as described in Equation 4.8 below.

$$\boldsymbol{sload_j} = \begin{cases} load_j & k_s = 1 \implies N_{sp} = 100 \\ load_j \ll 1 & k_s = 2 \implies N_{sp} = 50 \\ load_j \ll 2 & k_s = 4 \implies N_{sp} = 25 \end{cases} \tag{4.8}$$

Thus, if the traffic situation in the cluster has a high spread Δ in the component utilizations (Δ> k_s) and needs higher monitoring frequencies to capture the workload dynamics, this modification is the right choice. On the other hand, it has to be mentioned, that thereby the possible monitoring error per missed overflow bit increases to k_s percent as well.

INTERIMS SUMMARY: The monitoring flow described above, addresses all challenges issued in the considerations of Section 4.2.1 and provides the required flexibility for a run-time solution in modern NoC-based CMPs as described by the requirements in the introduction of Section 4.2.

- The proposed clustering offers the freedom to operate independently in ROIs and adapt them to the spatial requirements of the addressed workload fractions.
  - **Spatial Scope** → $\{x_{LLC}; y_{LLC}\} \Leftrightarrow \{x_{URC}; y_{URC}\}$ and $N_{CS} \leq N_{CSmax}$
- For each cluster, the timing of the monitoring can be adapted via global threshold reconfiguration at all tiles or via adjustments of the local data interpretation at the master-tile. Thus, two parameters allow effective control over the monitoring flow behavior.
  - **Temporal Scope** → $T_{MC} = N_{sp} \cdot \boldsymbol{v_{thresh}} \cdot t_{NoC}$ and $N_{sp} = 100/\boldsymbol{k_s}$
- The ***computational efforts*** per sensor value for the traffic evaluation could be effectively ***reduced*** to simple shift and add instead of multiply and divide operations.
- The proposed structuring of the monitoring pipeline addresses a ***balanced*** monitoring data ***communication*** and inherently includes parts of the scaling operations for the generation of utilization values.
- The counter-based sensing and overflow aggregation offers regular structures for potential hardware implementations of those parts.

The implementation and integration of this traffic monitoring concept will show if it works as expected or if tradeoffs must be made. Especially, the relation of cluster size and timing remains a concern due to the hotspot characteristics of the centralized organization. The proposed concept only relaxes some critical conditions to increase its applicability for practical problem sizes.

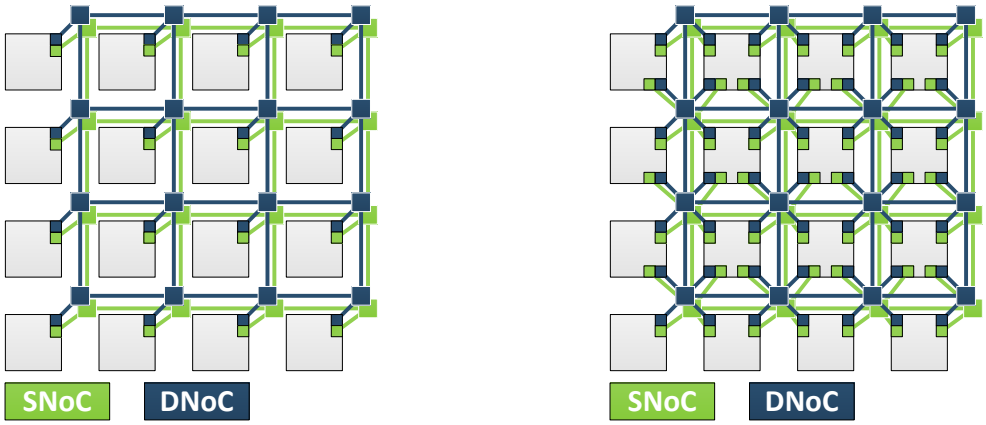
4.3 IMPLEMENTATION AND INTEGRATION

Subsequent to the conceptual work above, this section describes the implementation and integration characteristics of the intended traffic monitoring. Furthermore, it outlines the proposed concept with realistic approximations for the performance/costs for its hardware and software components. Thereby, the integration on the base of the 2D-Mesh and QMesh NoC will be discussed in parallel.

4.3.1 INFRASTRUCTURE

As partially discussed in Section 4.2, the communication of the traffic monitoring/management is realized over a physically separated unified infrastructure to provide a regular, flexible, and reusable solution for this aspect. As different works could show, this solution is suitable and less intrusive to normal workload operations [195], [204], [205]. The illustrations in Figure 4.2 (a) and (b), show that each tile has access to two independent networks, called ***Data-NoC*** (DNoC) and ***System-NoC*** (SNoC), which serve different traffic classes and were already proposed for such general use in [122], [130].

Data-NoC → DNoC: This network is responsible for data transfers and flows that belong to the workload operations for inter- as well as intra-application, memory or other input/output communication. Sizing and parameterization of the DNoC follow those specifications described in the Appendix (see pp. 164-167).



(a) 2D-Mesh (b) QMesh

Figure 4.2: Unified monitoring infrastructure integration for 2D-Mesh and QMesh

System-NoC → SNoC: All data transfers and flows relevant for system control/management operations are mapped onto this network. As Figure 4.2 (a) and (b) above show, the topology is equal

to this one applied for the DNoC to ensure a general connectivity that covers the same interaction patterns the workload applications will have. Furthermore, this regularity does not fix the infrastructure for a dedicated organization of the management, such as applied in [195], [204], [205], and the SNoC can follow the GALS/VFI organization of the DNoC. As the depicted component schematic for the 2D-Mesh and the QMesh in Figure 4.3 shows, the input/output components at the tile level (NIs, BUF, PT) are independent of the DNoC and thus the SNoC can be fully customized for the requirements of its own traffic classes regarding buffer, the width of the links, and clock rate. The router and link components of the SNoC are not depicted in Figure 4.3.

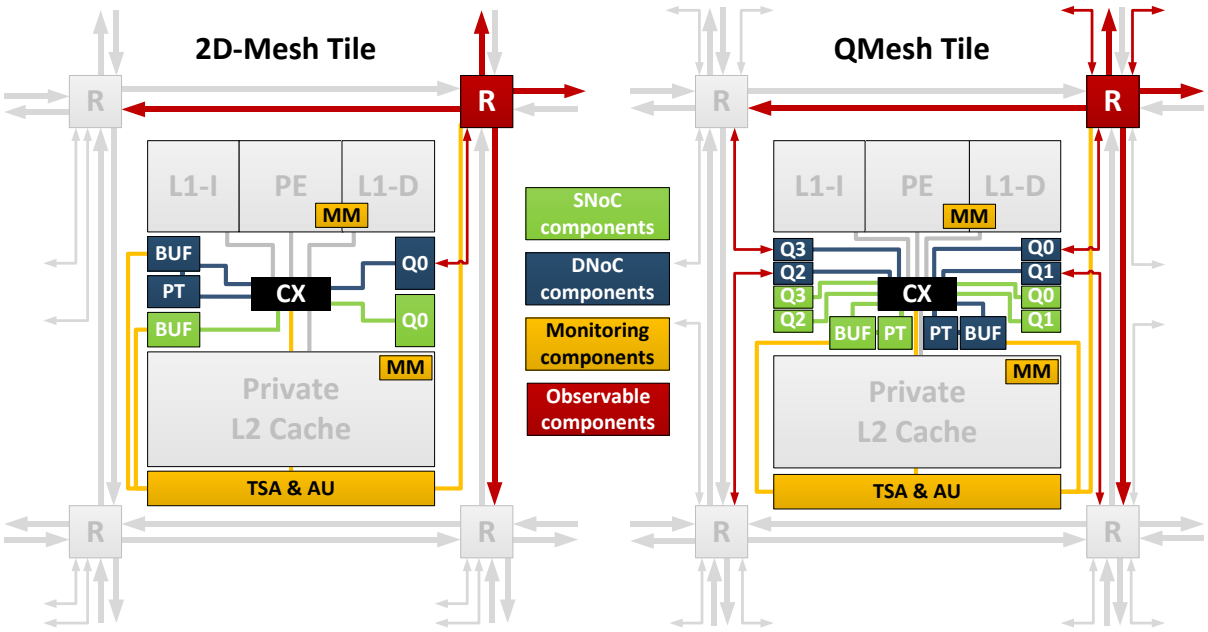


Figure 4.3: Component schematic at tile level on integration of the traffic monitoring in 2D-Mesh and QMesh

For the design of the SNoC, the work at hand generally favors an approach with minimized and simple hardware. But if more performance/throughput would be needed, the networks parameterization can be adjusted to the specific requirements. The 2D-Mesh as well as the QMesh configuration of the SNoC apply VC-free wormhole-switching, matrix arbitration, input port buffering, and request/acknowledge handshake logic for their router pipelines. The routing policies differ for both topologies, whereas the 2D-Mesh SNoC uses simple dimension-ordered xy-routing, while the QMesh SNoC has full dual-path xy-routing capabilities as described in Section 3.2.2 (see pp. 70-73). The data width $w_{data \rightarrow SNoC}$ of the links/flits must be sized to enable the header flit of a packet to carry at least the complete routing information. Otherwise, the performance would be impacted, because more than one leading flit would have to be received at each router input per hop to determine the required output port. Thus, the data width sizing has to fulfill the following condition for the 2D-Mesh/QMesh:

- **2D-Mesh** → $w_{data \rightarrow SNoC} \geq \lceil log_2(N_X) \rceil + \lceil log_2(N_Y) \rceil$
- **QMesh** → $w_{data \rightarrow SNoC} \geq \lceil log_2(N_X) \rceil + \lceil log_2(N_Y) \rceil + 2$

The utilized parameterization and feature composition for the SNoC of the work at hand is summarized in the Appendix (see Table I.4. at p. 166).

4.3.2 CLUSTERING

The formal characteristics of a cluster were already introduced in Section 4.2.2 (see pp. 94-100). But to build, maintain, and adapt these clusters, a communication flow and messaging protocol is needed. Therefore, specific details, packet types and responsibilities will be provided. The assumed organization for the management follows the hierarchical concept of *Guang et. al.* in [204], [205], [209]–[211], [213], with cooperative management service instances responsible for the global system level and the cluster level. Furthermore, these services are implemented as software modules. The initial decision for a cluster creation is performed by a global service instances. Thereby, the application mapping represents the preferred entry-point due to its global knowledge/planning of the workload as well as its correlation with the traffic management. Afterwards, the cluster building will be delegated to the cluster agent that operates on a selected tile out of the spanned region. This entity operates as master-tile and requests, organizes as well as maintains the remaining slave-tiles via the following messaging scheme:

CLUSTER-CREATION-ASSIGNMENT (CCA): Initially, the cluster agent receives its task definition for the new cluster creation process. The corresponding packet addresses the selected master-tile $\{x_{MT}; y_{MT}\}$ and contains a unique context identifier ID_{CTX} to classify the contents. Moreover, it contains the required configuration data D_{CTX} for the traffic monitoring such as the parameter for v_{thresh}, k_s, component coverage, and the region specification $\{x_{LLC}; y_{LLC}; x_{URC}; y_{URC}\}$ for the cluster. The cluster agent acknowledges this packet and immediately starts the cluster setup via the transmission of cluster requests (CRQ) to the remaining $(N_{CS} - 1)$ tiles of the spanned region.

CLUSTER-REQUEST (CRQ): The master-tile sends $(N_{CS} - 1)$ request packets to all tiles inside the spanned region between $\{x_{LLC}; y_{LLC}\}$ and $\{x_{URC}; y_{URC}\}$. Thereby, a request packet contains the destination address of the tile $\{x_{tile}; y_{tile}\}$, the address of the master-tile $\{x_{MT}; y_{MT}\}$, a cluster-internal identifier ID_{GROUP}, and the context information via ID_{CTX} and D_{CTX}. Afterwards, the master-tile waits for the response packets (CSP), which acknowledge the transmitted requests.

CLUSTER-RESPONSE (CSP): The $(N_{CS} - 1)$ cluster tiles receive request (CRQ) or update (CUP) packets of the master-tile, perform the data takeover and return a response packet to the master-tile as acknowledgement. The packet addresses the master-tile $\{x_{MT}; y_{MT}\}$, and contains a context identifier ID_{CTX} to classify the packet as well as the address of the responding tile $\{x_{tile}; y_{tile}\}$. After

the reception of $(N_{CS}-1)$ CSP packets the cluster agent has established or updated the full cluster, while the source tiles of those packets become slave-tiles and begin with their configured monitoring data reports (MDR).

CLUSTER-UPDATE (CUP): During cluster operation, the configuration data (i.e., monitoring periods or component coverage) need to be adapted, the cluster agent migrates to another master core, or the cluster will be deleted. Thus, the $(N_{CS}-1)$ slave-tiles are informed via update packets, which have the same format like the CRQ and acknowledge those updates via CSP packets to complete these transactions.

MONITORING-DATA-REPORT (MDR): Each slave-tile transmits packets with its monitoring data to the master-tile in periodic intervals. Thereby, each packet contains the address of the master-tile $\{x_{MT}; y_{MT}\}$, its cluster-internal reference ID_{GROUP} for unique monitoring data assignments, the defined component coverage and the monitoring data.

The transfers of those packets are realized via the SNoC. While the size of the MDR packets depends on the selected component coverage and therefore is partially dynamic, the contents of the remaining four packet types (CCA, CRQ, CSP, CUP) are sized as follows:

- Each NoC Address → $\lceil log_2(N_X) \rceil + \lceil log_2(N_Y) \rceil$ bits $+2$ bits for Q_{OUT} at QMesh
- D_{CTX} Cluster Region (LLC, URC) → $2 \cdot \lceil log_2(N_X) \rceil + 2 \cdot \lceil log_2(N_Y) \rceil$ bits
- D_{CTX} Threshold → $\lceil log_2(8) \rceil = 3$ bits
- D_{CTX} Scale Resolution → $\lceil log_2(3) \rceil = 2$ bits (CCA only)
- D_{CTX} Component Coverage → $4\ modes = 2$ bits
- ID_{CTX} Context Identifier → $\lceil log_2(5) \rceil = 3$ bits
- ID_{GROUP} Group Identifier → $\lceil log_2(N_{CSmax}) \rceil = 6$ bits for $N_{CSmax} = 64$

Thereby, it is assumed that eight discrete threshold values v_{thresh} and four different operation modes for the component coverage are supported. The value range for the scale resolution k_s was already fixed to {1, 2, 4} in Section 4.2.2 (see pp. 94-100) and is only part of the CCA. The unique context identifier ID_{CTX} must enable the classification of five different packet types (CCA, CRQ, CSP, CUP, and MDR). The group identifier ID_{GROUP} provides a cluster-internal referencing scheme that is independent of the global NoC address space. Its details are discussed at the end of this subsection. Under consideration of the selected data payload per flit $w_{data \rightarrow SNoC}$ of 16 bits, the corresponding packet sizes for CCA, CRQ, CSP, and CUP are in range of two up to three flits.

The next issue for the integration of the clustering is a selection of the maximum cluster size N_{CSmax}, because it specifies the number of sensors per tile for the observation of traffic flows to destinations inside the cluster. Furthermore, this parameter determines the expectable worst-case data throughput at the master-tile. But the maximum number of tiles per cluster must consider the potential mappings of workload applications as well to support practically relevant ROIs. Therefore, a modified version of Amdahl's Law [49] is applied to analyze the expectable asymptotic performance gains regarding the workload parallelization over a variation of the cluster size. It describes the expectable speedup of an application over a sequential single core solution, if its parallelizable fraction f_p is mapped onto n independent tiles [45]. The complementary fraction $(1 - f_p)$ is not parallelizable and performed sequentially [45]. But this original version does not consider important aspects regarding communication, synchronization, interferences, and memory access behavior of workload applications. Hence, different works propose extended versions to adapt Amdahl's Law for those aspects [3], [45], [47]–[49], [217]–[220]. In the work at hand, the modified model of *Yavits et. al.* [49] is considered, because it covers at least thread synchronization f_{sync} and inter-core communication f_{icc} as non-trivial fractions of the performance limiting issues. As the work of *Attiya et. al.* in [52] showed, synchronization is reducible, but not evitable at all. The models of *Eyerman et. al.* [48] or *Cassidy et. al.* [47] contain similar modifications. Thereby, it has to be mentioned that the modification of *Yavits et. al.* [49] additionally considers a variable sequential performance of CMP tiles via Pollack's Law [1] through the metric of base core equivalents (BCEs) per tile, but this aspect was out of scope for the analysis in the work at hand and therefore simplified. The resulting speedup S_{EAD} for the extended version of Amdahl's Law is given in Equation 4.9 below.

$$\boldsymbol{S_{EAD}} = f(f_p, f_{icc}, f_{sync}, n) = \frac{1}{(1 - f_p) + \frac{f_p}{n} + \frac{f_{icc}}{n} + f_{sync}} \tag{4.9}$$

The two additional parts in the denominator of this equation consider the time spent for inter-core communication T_{ICC} ($f_{icc} = T_{ICC}/T_{SEQ} = 0.001 \cdot n^{0.5}$) and the time spent for sequential-to-parallel data synchronization T_{SYNC} ($f_{icc} = T_{SYNC}/T_{SEQ} = 0.05$) in relation to the sequential execution time T_{SEQ} of the application. The utilized parameter settings for those values are extracted from [49] and represent the average case of different application workloads with varying arithmetic intensities as well as parallelization granularities. This extended version of Amdahl's Law is evaluated for single application workloads with parallelizable fractions of $f_p = \{0.5, 0.9, 0.95, 0.99\}$ mapped on 2 up to 128 tiles. The speedup results are plotted in Figure 4.4.

The chart shows that parallelization is only suitable for a specific range in the number of utilized processors, because afterwards the additional efforts would lead to diminishing returns regarding the expectable performance gains. As result of this analysis, the maximum cluster size N_{CSmax} is

chosen to be not higher than 64 tiles and this tradeoff seems sufficient to fulfill the requirements for a majority of possible application workloads. Even 32 or at least 16 tiles seem reasonable at $f_p \leq 0.95$. Under consideration of mapping/scheduling optimizations regarding spatio-temporal data and communication locality, the utilization of fewer tiles at higher utilization levels seems reasonable [11], [12], [111], [216], [221], [222] as future trend. Single- as well as multi-application workloads may additionally suffer from cache interference [221], [223], [224] that would further diminish the positive effect of higher parallelization degrees.

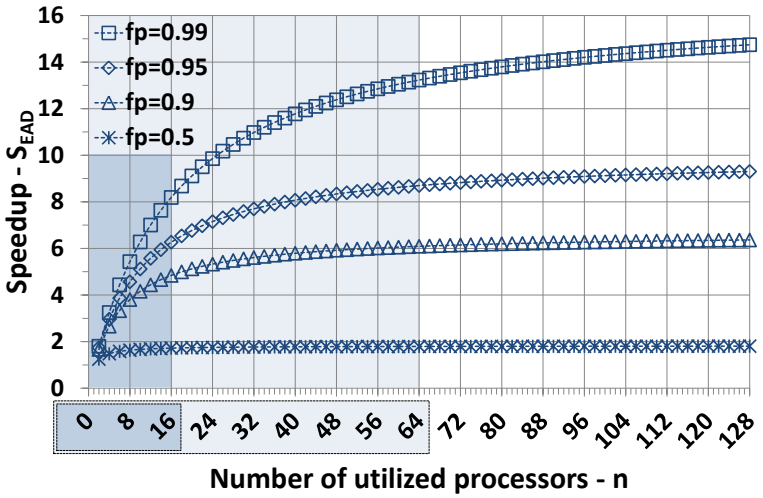


Figure 4.4: Speedup estimates for different cluster sizes using the extended version of Amdahl's Law

Finally, a cluster-internal identifier (called GROUP-ID or ID_{GROUP}) has to be introduced as consequence of the dynamically re-locatable and re-sizable traffic monitoring. The flexible sizing and positioning of clusters require a consistent cluster-internal identification of source-destination-paths and tiles. Furthermore, it has to be independent of the global xy-coordinate address space of the NoC ($\lceil log_2(N_X) \rceil + \lceil log_2(N_Y) \rceil$ bits), because the traffic monitoring inside each cluster has the maximum capabilities to sense traffic flows for N_{CSmax} paths and the path sensor assignments for the intra-cluster destinations must be uniquely indexed in this range of $0 \leq index \leq N_{CSmax} - 1$. Thereby, it is assumed that the size of the NoC exceeds the maximum cluster size and therefore the xy-addresses would violate this indexing constraint ($\lceil log_2(N_X) \rceil + \lceil log_2(N_Y) \rceil > \lceil log_2(N_{CSmax}) \rceil$).

$$x_{LLC} \leq x_{tile \rightarrow base} \leq x_{URC} \quad and \quad y_{LLC} \leq y_{tile \rightarrow base} \leq y_{URC} \tag{4.10}$$

$$x_{tile \rightarrow roi} = x_{tile \rightarrow base} - x_{LLC} \quad and \quad y_{tile \rightarrow roi} = y_{tile \rightarrow base} - y_{LLC} \tag{4.11}$$

$$\boldsymbol{ID_{GROUP}} = x_{tile \rightarrow group}[(w_{ID} - 1) \rightarrow 0] \oplus y_{tile \rightarrow group}[0 \rightarrow (w_{ID} - 1)] \tag{4.12}$$

The generation of the cluster-internal identifier ID_{GROUP} follows the functional flow described in the Equations 4.10 to 4.12.

- Each tile that is part of the spanned ROI $\{x_{LLC}; y_{LLC}; x_{URC}; y_{URC}\}$ of the cluster gets a unique ID_{GROUP}. Therefore, a location check with the global base address $\{x_{tile\rightarrow base};\ y_{tile\rightarrow base}\}$ of the tile must fulfill the conditions defined by Equation 4.10. A hardware implementation of this check would require two comparator-circuits with a data width of $\lceil log_2(N_X)\rceil$ and another two with a data width of $\lceil log_2(N_Y)\rceil$.
- The next step contains a coordinate shift for the tile coordinates $\{x_{tile\rightarrow base};\ y_{tile\rightarrow base}\}$ in the cluster region and the lower left corner $\{x_{LLC}; y_{LLC}\}$ will be the new origin. This is performed in Equation 4.11. A hardware implementation would require two adder-circuits with supported data widths of $\lceil log_2(N_X)\rceil$ and $\lceil log_2(N_Y)\rceil$.
- Finally, an address space conversion must be performed, because the shifted coordinates $\{x_{tile\rightarrow roi};\ y_{tile\rightarrow roi}\}$ are still in the range of $\lceil log_2(N_X)\rceil + \lceil log_2(N_Y)\rceil$ and must be reduced to a unique mapping on $w_{ID} = \lceil log_2(N_{CSmax})\rceil$ bits. A general way would be the conversion to the logical identifier via $y_{tile\rightarrow roi} \cdot n_{CY} + n_{CX}$ and afterwards take the w_{ID} lower bits. But this would imply costly multiplication each time the ID_{GROUP} must be generated and this is necessary locally at the sensing stage of the tiles for each time a flit is transmitted to a cluster internal destination. Hence, a light-weighted and hardware friendly way, as given in Equation 4.12, is preferable. Thereby, the w_{ID} lower bits of both cluster-internal coordinates $\{x_{tile\rightarrow roi};\ y_{tile\rightarrow roi}\}$ are used. For the cases of $w_{ID} > \lceil log_2(N_X)\rceil$ and $w_{ID} > \lceil log_2(N_Y)\rceil$, the higher bits without counterpart from the coordinate values are filled with zeros. One of these data words is reversed $(0 \rightarrow (w_{ID} - 1))$, while the other one remains in order $((w_{ID} - 1) \rightarrow 0)$. Afterwards, both data words are merged to the ID_{GROUP} via bitwise xor operations that happen in parallel. The reversing of one coordinate is hard-wired and the merging operation needs just w_{ID} xor-gates.

As a result of the modified ID_{GROUP}, the indexing does not linearly follow the logical addressing of the cluster-internal tiles and is scattered over the index range of $0 \leq index \leq N_{CSmax} - 1$. But this can be corrected at the data takeover for the traffic evaluation or via a remapping vector that hold the correct logical address representations for the generated indexing. The vector must be generated only once during cluster creations or re-sizing/-location updates. An example is depicted inside the clusters of Figure 4.8 (see p. 115), where the top numbers in each cluster tile represent its internal xy-address after the coordinate shift $\{x_{tile\rightarrow roi};\ y_{tile\rightarrow roi}\}$ and the bottom number is the resulting ID_{GROUP} value. For the presented messaging protocol, maximum cluster size as well as the cluster-internal identifier generation, there are no differences between the 2D-Mesh or the QMesh.

4.3.3 SENSORS AND AGGREGATION STAGES

After the description of the SNoC infrastructure and the clustering, the integration of the monitoring pipeline stages regarding sensing and aggregation is discussed next. Therefore, the implementation counterparts for the flow and stage composition of the concept in Section 4.2.2 (see pp. 94-100) are presented. The component schematic at the tile level in Figure 4.3 (see p. 101) shows the basic hardware parts of the traffic monitoring, whose functional capabilities can be outlined as follows:

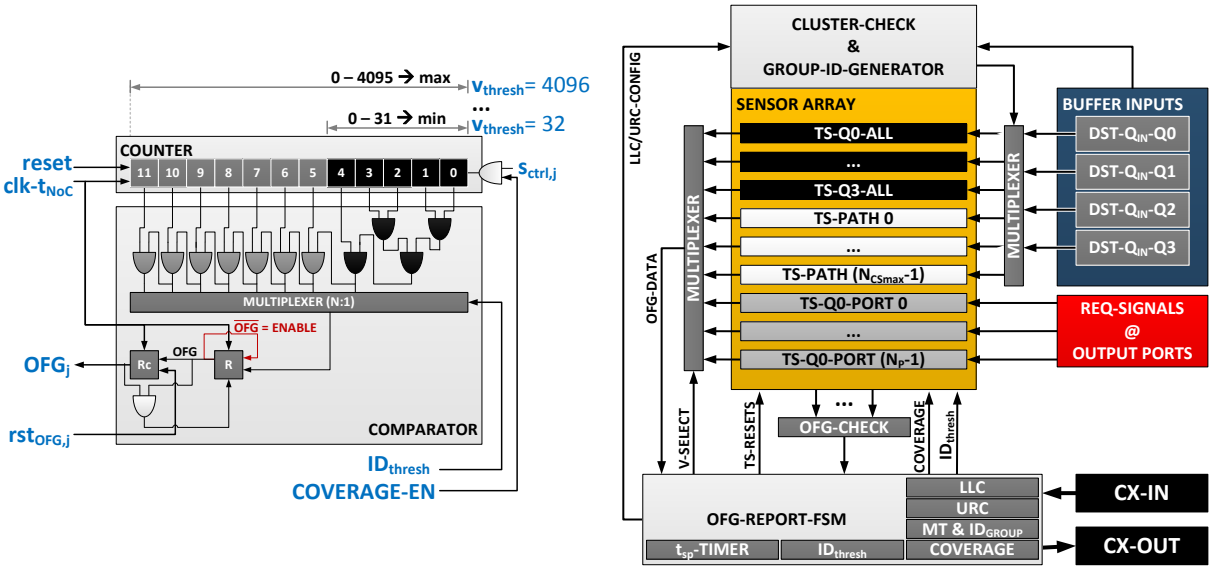
- The ***traffic sensor array*** (TSA) integrates the local sensor stage and the first level of the aggregation stage. Hence, it represents the implementation of the Equations 4.1 to 4.5 in the concept of Section 4.2.2 (see pp. 96-98). For the sensing purpose it is connected to the relevant signals of the corresponding tile router and the transmission buffer. Furthermore, it serves as local endpoint for the clustering and generates the overflow reports.
- The ***aggregation unit*** (AU) integrates the overflow aggregation and thus the second level of this monitoring pipeline stage. Hence, it represents the implementation of the Equations 4.6 to 4.8 in the concept of Section 4.2.2 (see p. 98). It aggregates all the incoming overflow reports and is only active if it is selected as master-tile.
- The ***monitoring memory*** (MM) holds the gathered utilization data for the traffic evaluation or monitoring data sharing.

The following passages below explain the implementation details and specific design decisions step by step. Furthermore, the expected implementation costs, monitoring configurability, and performance are addressed.

LOCAL-SENSORS-STAGE: The basic component of the sensing stage in the monitoring pipeline is the ***traffic sensor*** (TS) that counts discrete events occurring at flit traversals along router output ports and during traffic injections at the tile. Therefore, it has to implement the functionality as described by Equation 4.2 and 4.3 in Section 4.2.2 (see p. 96). This is realized via a combination of a simple ***counter with a comparator*** as depicted in Figure 4.5 (a) below. The counter is triggered by the activity control signal $s_{ctrl,j}$ and enabled by a register that controls the component coverage (COVERAGE-EN). Its clock signal is sourced by the DNoC clock to work synchronously with the observed events indicated by $s_{ctrl,j}$. Furthermore, it is connected to the general reset signal. The counter has a width of $\lceil log_2(4096) \rceil = 12$ bits and each bit cell consists of one half adder and one register cell.

The register outputs source the current counter value into a comparator unit that detects overflows for a defined threshold value v_{thresh} and generates the overflow flag OFG_j as output signal of the traffic sensor. As selection option for the threshold v_{thresh}, a set of eight discrete values

$\{32, 64, 128, 256, 512, 1024, 2048, 4096\}$ is provided. This threshold is controlled by an external input signal ID_{thresh} with a data width of $\lceil log_2(8) \rceil = 3$ bits. This signal controls a multiplexer that passes through the overflow detection signal, which corresponds to the selected threshold. Therefore, the register outputs of the counter are fed into a cascade of 11 and-gates that detect for all selectable values of v_{thresh} if a wrap-around at the corresponding sub-words of the counter occurs. Thereby, this overflow is captured by a first register (R), which is only enabled if its output equals zero. Afterwards, this overflow flag is taken over by a second copy register (Rc) and on success the first register R will be reset to capture new overflows. The output of the copy register Rc represents the OFG_j output of the traffic sensor. At the end of each period $\Delta t_{sp} = v_{thresh} \cdot t_{NoC}$, this OFG_j output is read out for the overflow reports and reset afterwards by the corresponding aggregation instance via $rst_{OFG,j}$.



(a) Traffic Sensor (TS) (b) Traffic Sensor Array (TSA) – Level 1

Figure 4.5: Traffic sensor (TS) schematic and traffic sensor array (TSA) for sensing and aggregation stage

AGGREGATION-STAGE → LEVEL 1: All traffic sensors of a tile are combined with the required control logic into the ***traffic sensor array*** (TSA), which is depicted in Figure 4.5 (b). It implements the complete sensing and level 1 aggregation stage for the traffic monitoring. Hence, each tile inside the NoC-based CMP is at least suited with one complete TSA. The decision for the pure hardware integration is reasoned by the resulting non-intrusiveness of the TSA on normal regular tile operations and the suitability for passive tile configurations without processor capabilities (i.e., cache-only tiles, system-input/output, dedicated hardware accelerators, or memory controller). The management, control, and overflow report functionality of the TSA is integrated as a finite state machine, called OFG-REPORT-FSM in Figure 4.5 (b), which is the only component of the TSA with a direct input/output connection to the central crossbar (CX) in the tile. Hence, it represents the

configuration endpoint for the traffic monitoring (via CRQ or CUP) as well as the source of monitoring data packets (via MDR). The performed operations are as follows:

- On arrival of CRQ or CUP packets, the OFG-REPORT-FSM performs the data takeover from the D_{CTX}-section of these packets and generates the CSP packet as response. Hence, registers for the storage of LLC, URC, COVERAGE, ID_{GROUP}, master-tile address $\{x_{MT}; y_{MT}\}$, and ID_{thresh} must be integrated. Furthermore, these configuration parameter registers source the traffic sensor configuration inputs and the path sensor selection unit (CLUSTER-CECK and GROUP-ID-GENERATOR).
- After this configuration phase, the OFG-REPORT-FSM has to trigger and perform the MDR transmissions. Therefore, an internal t_{sp}-timer with a data width of $\lceil log_2(4096) \rceil = 12$ bits is configured with the selected threshold ID_{thresh} and indicates the expiration of sensor periods. As subsequent action, the OFG-REPORT-FSM reads out the OFG-CHECK status bit and if overflow flags were generated ($OFG_j = 1$) in the expired sensor period the MDR setup is started. Therefore, the OFG-REPORT-FSM composes the required header data ($\{x_{MT}; y_{MT}\}$ and own ID_{GROUP}), reads out the OFG-DATA as function of the defined component coverage (COVERAGE→V-SELECT), performs a reset of the overflow registers afterwards ($rst_{OFG,j} = 1$), and pushes the completed MDR to the transmission buffer of the SNoC.
- The MDRs are generated as long as the corresponding tile is member of a traffic monitoring cluster, the OFG-CHECK indicates activity, and the COVERAGE configuration enables reports for specific traffic sensor fractions.

The OFG-CHECK unit of the TSA is connected to each of the OFG_j outputs of the N_S traffic sensors and performs the functionality specified by Equation 4.5 in Section 4.2.2 (see p. 98). But instead of the summation a cascade of or-gates can be used to detect if at least one OFG_j output is set to a non-zero. The SENSOR ARRAY contains all N_S traffic sensors of the tile. Thereby, these sensors are source by different control signals and cover the observation of tile outputs (TS-Q0/Q1/Q2/Q3-ALL), router output ports (TS-Q0-PORT-…), as well as the cluster-internal ETE paths (TS-PATH-…). The numbers of traffic sensors in the 2D-Mesh and in the QMesh differ. Furthermore, the activity of the specific sensor groups depends on the adjusted coverage (COVERAGE → COVERAGE-EN).

The number of ***global tile output sensors*** (TS-Q…-ALL) is one for the 2D-Mesh and four for the QMesh. While the 2D-Mesh has only a single NI, the QMesh provides output sensing for each of its four NIs.

- Thereby, the 2D-Mesh global output sensor (TS-Q0-ALL) is triggered each time a flit is acknowledged to leave the transmission buffer (BUF) towards the FIFO-buffer of the NI and the corresponding packet further addresses a cluster-internal or –external tile.
- The four global output sensors of the QMesh (TS-Q0/Q1/Q2/Q3-ALL) are configured differently, because each sensor only captures the flit transmission events for its assigned quadrant, which is defined by the two bits for the input quadrant Q_{IN} of the packet.

Thus, those global output sensors cover the sum of occurred flit transfers into the complete DNoC destined inside and outside of the cluster.

The number of ETE ***path sensors*** (TS-PATH-…) for cluster-internal flit transmissions equals the maximum allowed cluster size N_{CSmax} for both topologies. Thereby, additional steps (see CLUSTER-CHECK and GROUP-ID-GENERATOR in Figure 4.5 (b) and discussion in prior Section 4.3.2) must be performed to trigger the corresponding traffic sensor for each destination.

- Each time a flit is acknowledged to leave transmission buffer towards the FIFO-buffer of one NI, the corresponding destination must be checked according to the conditions of Equation 4.10 (see p. 105), if it is inside the cluster. If this check fails, no path sensor is triggered. Implemented in hardware, this check logic requires two comparator circuits with $\lceil log_2(N_X) \rceil$ data width for the x-coordinate and two comparator circuits with $\lceil log_2(N_Y) \rceil$ data width for the y-coordinate.
- Additionally, the path sensor must be correctly indexed by the cluster-internal identifier ID_{GROUP} of the destination. Therefore, the LLC-based coordinate shift of Equation 4.11 (see p. 105) and the xor-based index generation of Equation 4.12 (see p. 112) must be performed. Implemented in hardware, this generation logic requires one adder circuit with at least $\lceil log_2(N_X) \rceil$ full adder cells for the x-coordinate shift, one adder circuit with at least $\lceil log_2(N_Y) \rceil$ full adder cells for the x-coordinate shift, and $\lceil log_2(N_{CSmax}) \rceil$ xor-gates for the merging operation.

The control signals of the sensors for global tile outputs as well as cluster-internal ETE path traversals are sourced by the buffer component (BUF) of the DNoC. Therefore, each time a flit traversal towards the NI FIFO over the CX is acknowledged (ACK='1') by the handshake logic, the corresponding transmission buffer segment (Q0/Q1/Q2/Q3) passes a data word into a queue (DST-Q_{IN}-Q0/Q1/Q2/Q3) that contains the destination address $\{x_{tile \rightarrow base}; y_{tile \rightarrow base}\}$, and in case of the QMesh the corresponding quadrant identifier Q_{IN} for the NI. Thereby, the buffer in the QMesh serves four of these queues, while the 2D-Mesh only needs one for each VC. Each data word inside these queues indicates a single flit transmission and provides the data for the cluster check as well as the

ID_{GROUP} generation (correct sensor indexing). Thereby, the global tile sensors are incremented by each processed entry of the queue. The processing of one of these data words would result in the increase of the corresponding sensor counter values by one and takes one DNoC clock cycle. Afterwards, the next word is processed. In the case of the QMesh, the four queues are multiplexed in a fixed order and each must be able to carry at least four entries. In summary, global tile as well as ETE path traffic sensors measure the real data loads for each tile by counting the number of injected flits.

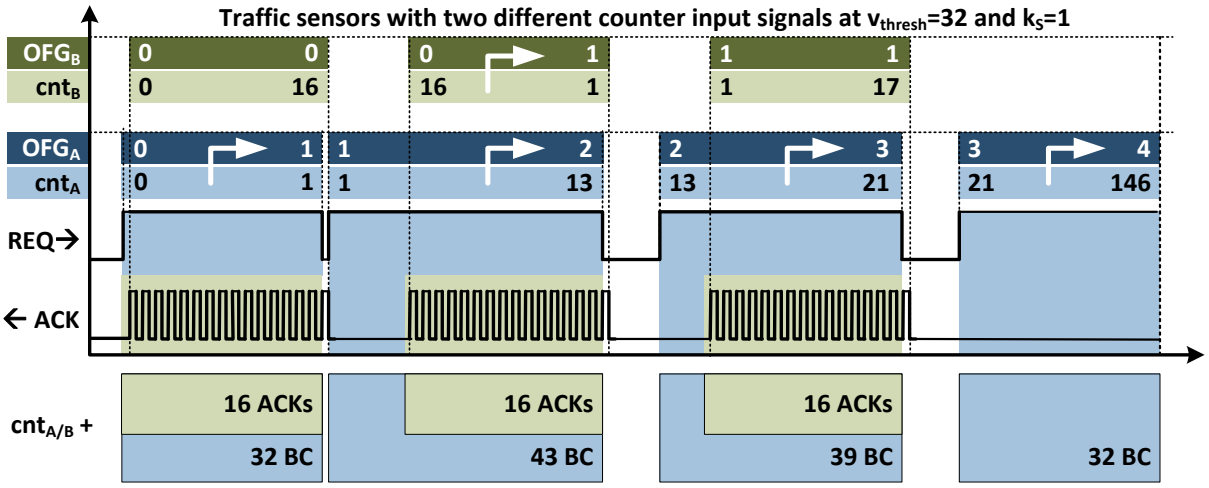


Figure 4.6: Different traffic sensor operations as function of the source for the control signal

The ***port output sensors*** (TS-Q0-PORT-…) operate differently and are intended to capture the utilization resulting from finalized flit traversals as well as blocking situations. The global tile output and ETE path sensors count acknowledged flit traversals and thus the raw data injections. In [198], *Van den Brand et. al.* proposes a similar utilization measurement via flit/packet counts for links, but this measure does not reflect the real utilization of the links, because congestions with busy-waiting are not considered. In such situations no flit transfers take place, but the bandwidth is lost as a kind of pseudo-utilization. Hence, the control signal inputs for the port output sensors are sourced by the request (REQ) of the handshake logic instead of the acknowledgment (ACK). If a router output port wants to forward flits, this request signal is set to high (REQ='1') as long as all waiting flits are passed, while the downstream component sets the acknowledgement signal (ACK='1') only for one clock cycle for each flit traversal to indicate that the next flit can be forwarded. Thus, the REQ signals from the N_P output ports of the observed routers are connected to the control signal inputs $s_{ctrl,j}$ of these traffic sensors and busy-cycles are counted instead of acknowledged flits. This works for VC-based solutions as well due to the inherent sharing of links (see Section 2.1.3 p. 44). Figure 4.6 illustrates the different sensor operations as function of the source for the control signal at the threshold adjustment of $v_{thresh} = 32$ and a scale resolution of $k_S = 1$. Configuration A (cnt_A and OFG_A) uses the REQ-signal and counts the busy cycles (BC), while configuration B (cnt_B and OFG_B) is

sourced by the ACK signal. At the depicted time window, three packets of 16 flits (= 16 ACKs) each are transferred and acknowledged flit-by-flit. Thereby, three blocking situations of varying duration take place between the transfers, where no flits can be passed at all and only busy-waiting occurs. While cnt_B captures the correct number of flits, cnt_A is enabled to count all clock cycles consumed for real transfers as well as the blocking.

Thereby has to be considered, that this kind of busy-cycle counting might be incompatible with the operation mode of the global tile output sensors and the ETE path sensors (which would work similar to configuration B). But this depends on the handshake policy and the required number of cycles n_{HL} for a handshaked flit traversal. If only one clock cycle is needed for request and acknowledge ($n_{HL} = 1$), the counted busy-cycles at the ports would have the same time base as the counted flit traversals at the tile injections. But if two clock cycle per flit ($n_{HL} = 2$) would be needed for this handshake procedure (as applied in Figure 4.6), each counted flit traversal (ACK) would have the weight of two counted busy cycles (1 ACK = 2BC) and the port sensors would generate overflows with the doubled rate for the same utilization scenario (see final values for $cnt_{A/B}$ and $OFG_{A/B}$). As workarounds for this potential incompatibility, the overflow generation capabilities for both sensor groups can be adjusted via the sensor threshold. Thereby, the threshold of the port output sensors $v_{thresh \rightarrow port}$ represents the n_{HL} multiple of the global tile/path sensors threshold $v_{thresh \rightarrow all|path}$ as described in Equations 4.13 and under the general assumption of $n_{HL} = 2^p$ ($p \in \mathbb{N}$). With the discrete selection range of $v_{thresh} = \{32, 64, 128, 256, 512, 1024, 2048, 4096, \dots\}$ and a linearly increasing ID_{thresh} over this range as selection index, this index for the threshold of the output port sensor must be increased by p. The t_{sp}-timer for the sensor periods would be controlled by $ID_{thresh \rightarrow all|path}$ and thus by the lower threshold $v_{thresh \rightarrow all|path}$. This workaround ensures that each measured flit injection would be recognized equivalent to n_{HL} busy-cycles per link along its path traversal through the NoC.

$$\boldsymbol{v_{thresh \rightarrow port}} = n_{HL} \cdot v_{thresh \rightarrow all|path} \Longrightarrow \boldsymbol{ID_{thresh \rightarrow port}} = ID_{thresh \rightarrow all|path} + p \qquad (4.13)$$

The 2D-Mesh and the QMesh topology differ in the number of integrated port output sensors. Figure 4.8 illustrates an example of a single 5x4 cluster in a QMesh to provide more insights. The corresponding tile level view can be obtained from Figure 4.3 (see p. 101).

- Each TSA inside the 2D-Mesh has to integrate $N_{P \rightarrow 2DMesh} = 5$ output port sensors to observe all outgoing connections of a single router that belongs to the same tile (located at Q0).
- Each TSA inside the QMesh has to integrate $N_{P \rightarrow QMesh} = 8$ output port sensors to observe all connections of its base-address router (located at Q0).

The additional router endpoints (Q1/Q2/Q3) inside the QMesh along the tiles at the left and the bottom edge of the cluster perimeter are not monitored to avoid the overlapping of adjacent clusters.

Finally, the total number of traffic sensors per TSA for both topologies can be formulated as follows (with concrete number is for the case of $N_{CSmax} = 16\ or\ 64$ tiles):

- **2D-Mesh Sensor Array** → $N_{S\rightarrow 2DMesh} = N_{CSmax} + N_{P\rightarrow 2DMesh} + 1 = 22\ or\ 70$
- **QMesh Sensor Array** → $N_{S\rightarrow QMesh} = N_{CSmax} + N_{P\rightarrow QMesh} + 4 = 28\ or\ 76$

Furthermore, this TSA sensor count N_S specifies the worst case monitoring data payload in bits that is transmitted by a single MDR at the end of each sensor period. But the addressed use case might have different requirements and would utilize only a subset of these sensors. Therefore, the component coverage of TSA can be configured via the selection of the COVERAGE out four options:

- **COVERAGE** = '00' : --- → All sensors disabled and monitoring off
- **COVERAGE** = '01' : 1 or 4 bits → Tile outputs
- **COVERAGE** = '10' : $N_S - N_{CSmax}$ bits → Tile outputs and router ports/links
- **COVERAGE** = '11' : N_S bits → Tile outputs, router ports/links, and paths

Hence, the tile outputs are activated if the first or the second bit is set to one and the router ports/links are activated when the second bit is set to one. These logical combinations are connected from the OFG-REPORT-FSM to the COVERAGE-EN inputs of the sensors. To enable MDR overflow payloads without reordering on different COVERAGE configurations the sequencing for the vector indices $0 \leq j \leq (N_S - 1)$ is fixed to BEGIN→TS-Q…-ALL| TS-Q0-PORT-…| TS-PATH-…←END. Thus, if the component coverage changes, the relevant segments are simply cut off.

AGGREGATION-STAGE → LEVEL 2: The ***aggregation unit*** (AU) provides the functionality of the last aggregation level and is integrated as hardware component as shown in Figure 4.7. The decision against the integration into the software module is reasoned by the non-intrusiveness of the hardware module. While a hardware AU provides the final monitoring data at the end of each monitoring cycle, the software based overflow aggregation of all MDRs would be too interruptive and costly regarding processor utilization. In the worst-case, for each incoming MDR the N_S payload bits have to be parsed and at the occurrence of a non-zero the corresponding overflow counter must be incremented. This has to be performed $N_{CS} \cdot N_{sp}$ times each monitoring cycle and would drive the processor utilization by the software module. The hardware AU performs this in parallel to other software operations on the processor and this frees run-time for monitoring evaluations. But this design-time decision limits the capabilities for the migration of the cluster agent, monitoring

evaluations, and other interdependent software modules as well to those tile positions with integrated AU. Furthermore, the hardware costs of the anticipated traffic monitoring/management increase and the clustering is restricted to regions that include at least one AU-suited tile. Thus, a tradeoff must be found at design-time that provides enough potential master-tiles N_{MT} suited with TSA as well as AU and an adequate distribution over the CMP.

- **TSA integration at all tiles** → $N_{tile} \Longrightarrow 100\%$ of tiles
- **AU integration at potential master-tiles** → $N_{MT} \Longrightarrow 100 \cdot \frac{N_{MT}}{N_{tile}}\%$ of tiles

The example in Figure 4.8 illustrates such a tradeoff with a master-tile ratio of 25% and a regular distribution. This presented tradeoff would ensure that each cluster with a size of N_{CS} tiles includes at least $\lfloor N_{CS}/4 \rfloor$ optional master-tiles, as long as the cluster shape fulfills the condition of $(n_{CX} > 1)\ and\ (n_{CY} > 1)$.

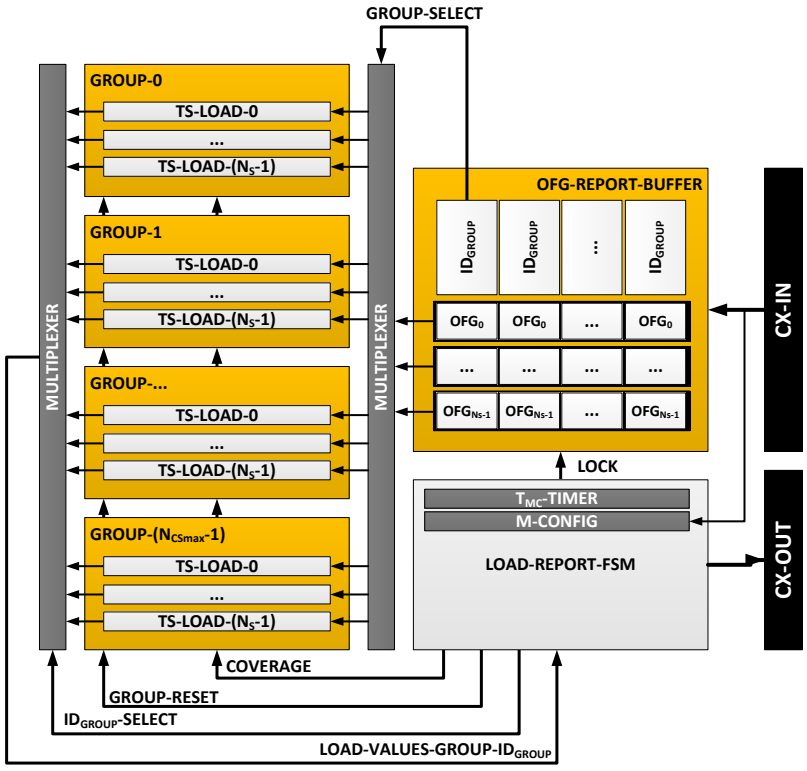


Figure 4.7: Master-tile aggregation unit (AU) for OFG reports

Additionally, it has to be noted that this partially dynamic approach with its static design-time tile mappings would correlate with advanced concepts for reliable computing as proposed by *Leem et. al.* in [225], [226]. Therein, high reliability tiles perform tasks critical for the control-flow and relaxed reliability perform less critical data flow operations. The anticipated traffic monitoring/management would be classified as critical system level service and the potential master-tiles as high reliability tiles.

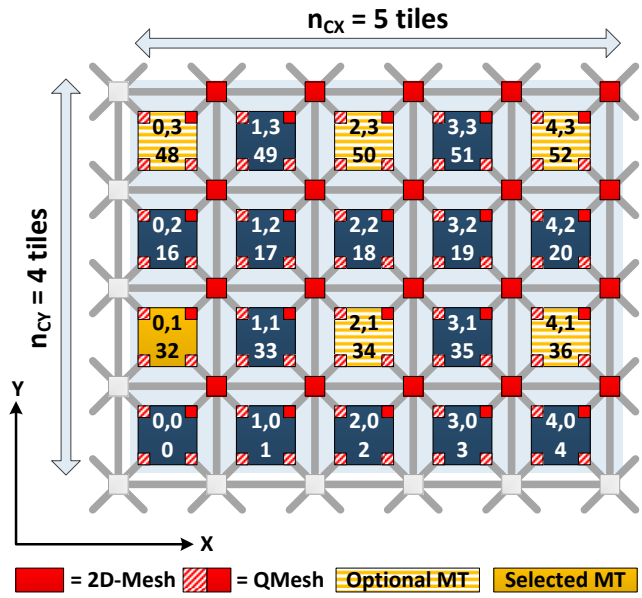


Figure 4.8: Exemplary master-tile distribution inside a 5x4 QMesh monitoring cluster

The AU itself has a fairly simple structure that is dominated by the overflow counter as basic element to integrate the functionality of Equation 4.6 from the concept in Section 4.2.2 (see p. 98). Each load counter must be able to count up to 100 overflows and therefore is suited with a data width of $\lceil log_2(100) \rceil = 7$ bits. Each counter bit includes a half adder and a register cell. The setup of a load counter is similar to the counter component of the traffic sensor illustrated in Figure 4.5 (a) (see p. 108) and the increment operations are triggered by reported OFG bits as control signal.

- The AU integrates N_{CSmax} groups of overflow counters with one group for each potential cluster member. Thereby, each overflow counter group is referenced by the cluster-internal identifier ID_{GROUP} and integrates N_S counter to gather the overflow for each sensor of a tile.
- For all incoming MDRs at the master-tile, the ID_{GROUP} and the reported overflow payloads are fed as data word into the OFG-REPORT-BUFFER, which sources the counter groups and is directly connected to the input port of the central crossbar (CX) of this tile. The ID_{GROUP} serves as counter group selection input for a multiplexer and the OFG bits source the corresponding counter inputs. The overflow counters inside each group have the same order as the traffic sensors in the TSAs. Furthermore, the COVERAGE bits determine if specific overflow counter fractions are enabled or not.
- A finite state machine, called LOAD-REPORT-FSM, observes the monitoring cycle with its T_{MC}-timer, holds relevant control information (M-CONFIG →LLC, URC, ID_{thresh}, k_s, COVERAGE) in its registers, and performs the counter value reports. The timer has a data width of $\lceil log_2(100 \cdot v_{thresh \rightarrow max}) \rceil = 19$ bits and is controlled by ID_{thresh} and k_s. When the current monitoring cycle is expired, the sourcing by the OFG-REPORT-BUFFER is temporary locked

from passing OFG words to the current read out group, the counter values are read out group by group, reset afterwards, and transmitted to the monitoring memory with required scaling operations as specified in Equation 4.8 (see p.99).

- For the final utilization data transfers, the LOAD-REPORT-FSM has a direct connection via an output port of the central crossbar (CX). Furthermore, the values for all M-CONFIG-registers are programmed by the cluster agent via the crossbar input of the CX, which sources the OFG-REPORT-BUFFER as well.

The scaling operation described by Equation 4.8 (see p.99) is integrated directly as part of the LOAD-REPORT-FSM of the AU, because the k_s-controlled shift can be performed on-the-fly during the counter read operations. Hence, this evaluation stage can start advanced processing immediately without this overhead.

All of the final utilization data must be transferred by the LOAD-REPORT-FSM into the ***monitoring memory*** (MM). At this point the data format will be converted from values with 7-bits data width to bytes, which enables the better handling regarding standard data types for the processing in software. As the component schematic at the tile level in Figure 4.3 (see p. 101) shows, different options for the integration of the MM exist:

- The MM is represented by a L1-D or L2 cache partition [222], [224] that is exclusively assigned to the software module/thread of the cluster manager. Depending on the implemented cache partition capabilities, the MM fraction could be dynamically adapted to the current component coverage. A similar approach for the collection of monitoring snapshot is used by *Abdel-Khalek et. al.* in [196].
- The MM is integrated as additional scratchpad memory [11] component next to the processing element. It must be sized for the maximum amount of utilization data and is exclusively used by the cluster agent software module.

The maximum amount of utilization data per cluster occurs if all N_S sensors at $N_{CS} = N_{CSmax}$ tiles are activated. Hence, the MM must be sized as follows (with concrete number is for the case of $N_{CSmax} = 16\ or\ 64$ tiles):

- **2D-Mesh MM-Size** → $N_{CSmax} \cdot (N_{CSmax} + N_{P\rightarrow 2DMesh} + 1) = 352 \text{ or } 4480$ bytes
- **QMesh MM-Size** → $N_{CSmax} \cdot (N_{CSmax} + N_{P\rightarrow QMesh} + 4) = 448 \text{ or } 4864$ bytes

Thus, the gathered monitoring data for a complete cluster with 64 tiles and full component coverage would be smaller than 5 kbyte.

The data values are formatted to byte-sized words and represent final utilization values along tile outputs, paths and ports. Hence, the cluster agent can proceed with advanced evaluations without any further expensive operations for pre-processing. Table 4.2 introduces the naming and indexing conventions for the aggregated utilization data inside the MM. Finally, the following Table 4.3 summarized all parameter configurations and sizing dependencies for the presented traffic monitoring integration.

Table 4.2: Aggregated utilization data fields collected inside the monitoring memory (MM)

MM → Data fields	Content Description
→ $\boldsymbol{TLOAD}[\boldsymbol{1\ or\ 4}][\boldsymbol{N_{CS}}]$	Tile output loads for each cluster tile → Raw flit injections to N_{tile} tiles of the CMP/NoC Coverage of the the global output in- and outbound regarding the cluster
→ $\boldsymbol{PLOAD}[\boldsymbol{N_{CS}}][\boldsymbol{N_{CS}}]$	Path loads for each cluster tile → Raw flit injections to N_{CS} tiles of the cluster Coverage of in-bound destinations regarding the cluster
→ $\boldsymbol{LLOAD}[\boldsymbol{N_P}][\boldsymbol{N_{CS}}]$	Link loads for each router port of a cluster tile → Raw flit traversal + Busy-cyles Coverage of in-bound links/ports regarding the cluster, but the measurements contain the traffic that traverses the cluster as well

Table 4.3: Configuration and hardware summary of the traffic monitoring

PARAMETER	2D-MESH	QMESH
Traffic Sensor (TS)		
→ **TS-Counter**	$\lceil log_2(v_{thresh \to max}) \rceil$ = 12 bits data width → 12 HA + 12 REGS	
→ **TS-Comparator**	13 AND-GATES + 2 REGS + 8:1 MUX	
→ **TS-Thresholds – v_{thresh}**	$\{32, 64, 128, 256, 512, 1024, 2048, \mathbf{4096}\}$	
→ **TS-Thresholds – ID_{thresh}**	$\{'000', '001', '010', '011', '100', '101', '110', '111'\}$	
→ **Sensor Periods – t_{sp}**	$v_{thresh} \cdot t_{DNoC}$	
Traffic Sensor Array (TSA)		
→ **# of TS in Sensor Array – N_S**	$N_{CSmax} + 6$	$N_{CSmax} + 12$
→ **CLUSTER CHECK**	Two x-coordinate comparators with $\lceil log_2(N_X) \rceil$ bits data width Two y-coordinate comparators with $\lceil log_2(N_Y) \rceil$ bits data width	
→ **GROUP-ID-GENERATOR**	One x-coordinate shift adder with $\lceil log_2(N_X) \rceil$ bits/FA One y-coordinate shift adder with $\lceil log_2(N_Y) \rceil$ bits/FA $\lceil log_2(N_{CSmax}) \rceil$ XOR-GATES	
→ **DST-FIFOs**	$\geq 2 \cdot N_V \cdot (\lceil log_2(N_Y) \rceil + \lceil log_2(N_Y) \rceil)$	$\geq 16 \cdot (\lceil log_2(N_Y) \rceil + \lceil log_2(N_Y) \rceil + 2)$
→ **OFG-CHECK**	N_S OR-GATES	
→ **OFG-REPORT-FSM**	LLC/URC/MT → $3 \cdot (\lceil log_2(N_Y) \rceil + \lceil log_2(N_Y) \rceil)$ bits ID_{GROUP} → $\lceil log_2(N_{CSmax}) \rceil$ bits t_{sp}-TIMER → $\lceil log_2(v_{thresh \to max}) \rceil$ = 12 bits wide COVERAGE → $\{'00', '01', '10', '11'\} = 2_{COV}$ bits ID_{thresh} → 3 bits	
→ **MDR Header - $s_{MDR \to header}$**	$\lceil log_2(N_Y) \rceil + \lceil log_2(N_Y) \rceil + 2_{COV} + \lceil log_2(N_{CSmax}) \rceil$ bits	$\lceil log_2(N_Y) \rceil + \lceil log_2(N_Y) \rceil + 2_{COV} + \lceil log_2(N_{CSmax}) \rceil + 2_{Q_{IN}}$ bits
→ **MDR Payload - $s_{MDR \to payload}$**	1 bit	4 bits
→ **MDR Payload - $s_{MDR \to payload}$**	$N_{P \to 2DMesh} + 1$ bits	$N_{P \to QMesh} + 4$ bits
→ **MDR Payload - $s_{MDR \to payload}$**	$N_{P \to 2DMesh} + 1 + N_{CSmax}$ bits	$N_{P \to QMesh} + 4 + N_{CSmax}$ bits
→ **MDR Size - s_{MDR}**	$s_{MDR} = s_{MDR \to header} + s_{MDR \to payload}$	
Aggregation Unit (AU)		
→ **OFG-Counter**	$\lceil log_2(100 \rceil$ = 7 bits wide → 7 HA + 7 REGS	
→ **# of OFG-Counter per AU**	$N_{CSmax} \cdot N_S$	
→ **OFG-REPORT-BUFFER**	$\geq 2 \cdot \lceil log_2(N_{CSmax}) \rceil \cdot N_S$ bits	$\geq 8 \cdot \lceil log_2(N_{CSmax}) \rceil \cdot N_S$ bits
→ **LOAD-REPORT-FSM**	LLC/URC → $3 \cdot (\lceil log_2(N_Y) \rceil + \lceil log_2(N_Y) \rceil)$ bits T_{MC}-TIMER → $\lceil log_2(100 \cdot v_{thresh \to max}) \rceil$ = 19 bits wide COVERAGE → 2 bits ID_{thresh} → 3 bits k_s → 2 bits	
Monitoring Memory (MM)		
→ **MM-Size**	$N_{CSmax} \cdot N_S$ bytes	
→ **MM-Integration**	L1-D Cache / L2 Cache Partition or Scratchpad Memory Segment	

HA = Half Adder Cell; FA = Full Adder Cell; REGS = Register Cell

4.3.4 MONITORING CONFIGURATION AND APPLICABILITY

For the configuration of the traffic monitoring, the interdependencies between timing, cluster sizing and component coverage must be considered to avoid a communication bottleneck at the master-tile. Furthermore, the specific characteristics of an applicable monitoring configuration for the intended use cases must be specified. In additional to the general relations and conditions, this subsection provides practical values for a system with the following setup:

- The DNoC has a provided payload data width of $w_{data \rightarrow DNoC} = 64$ bits per flit/link.
- The SNoC has a provided payload data width of $w_{data \rightarrow SNoC} = 16$ bits per flit/link.
- Both networks operate with a clock cycle of $t_{NoC} = 1$ ns and a handshaking of $n_{HL} = 2$.
- The maximum allowed cluster size is set to the worst-case of $N_{CSmax} = 64$ tiles.
- As corner-cases for the coverage the configurations COVERAGE = ’01’/’11’ are assumed.

Initially, the allowed cluster size N_{CS}^{*} for a specific sensor period must be set as described by Equation 4.14 below. The formulated cluster size condition is straight forward under consideration of the allowed communication load of the master-tile. Thereby, the maximum flit reception rate of a SNoC NI is the limiting parameter for the 2D-Mesh and as well the worst-case in the QMesh. It can be obtained by the reciprocal value of the needed handshake cycles for a single flit traversal $1/n_{HL}$. Furthermore, in the worst case scenario each slave-tile injects $s_{MDR \rightarrow F}$ flits per sensor period with v_{thresh} cycles. As summarized in Table 4.3, the header section of each MDR $s_{MDR-header}$ is constant and the monitoring payload $s_{MDR-payload}$ depends on the configured component coverage. Equation 4.14 formulates that the quotient of the maximum flit reception rate per NI $(1/n_{HL} \cdot t_{SNoC})$ and the flit injection rate per slave-tile $(s_{MDR \rightarrow F}/v_{thresh} \cdot t_{DNoC})$ defines the allowed cluster size. Furthermore, a reception rate correction factor $0 < c_{rate} \leq 1$ is introduced, if bandwidth must be preserved for other services. But in the following examples a value of $c_{rate} = 1$ is assumed.

$$\boldsymbol{N_{CS}^{*}} \leq \left\lfloor c_{rate} \cdot \frac{\left(\frac{1}{n_{HL} \cdot t_{SNoC}}\right)}{\left(\frac{s_{MDR \rightarrow F}}{v_{thresh} \cdot t_{DNoC}}\right)} \right\rfloor \tag{4.14}$$

As inputs for the calculation of N_{CS}^{*} values for the specified system setup, Table 4.4 contains the needed MDR packet sizes for all considered COVERAGE configurations for the 2D-Mesh/QMesh.

$$\boldsymbol{s_{OFG}} = w_{data \rightarrow DNoC} \cdot \frac{v_{thresh}}{8 \cdot n_{HL}} \quad in \quad bytes \tag{4.15}$$

Another important parameter is the sensitivity per reported overflow s_{OFG}, because it specifies the required amount of transferred data in bytes to trigger an overflow at the TSA. As described by

Equation 4.15, it results from the product of the number of bits per DNoC flit $w_{data \rightarrow DNoC}$ with the number of detectable flits for the defined threshold v_{thresh}/n_{HL}. Hence, this overflow sensitivity specifies the detectable step width of the communication volumes along components and thus the quality to capture the dynamic behavior of application workloads.

Table 4.4: MDR packet sizing for different component coverage configurations

COVERAGE		Payload – $s_{MDR \rightarrow payload}$	$s_{MDR \rightarrow B}$ in Bits	$s_{MDR \rightarrow F}$ in Flits	MM Data
'01'	2D-Mesh	1	15	1	$TLOAD[1\ or\ 4][N_{CS}]$
	QMesh	4	20	2	
'10'	2D-Mesh	6	20	2	$TLOAD[1\ or\ 4][N_{CS}]$ $LLOAD[N_P][N_{CS}]$
	QMesh	12	28	2	
'11'	2D-Mesh	$N_{CSmax} + 6$	84	6	$TLOAD[1\ or\ 4][N_{CS}]$ $LLOAD[N_P][N_{CS}]$ $PLOAD[N_{CS}][N_{CS}]$
	QMesh	$N_{CSmax} + 12$	92	6	

Table 4.5 provides the complete overview of all traffic monitoring characteristics that result from the variation of the traffic sensor threshold v_{thresh} and scale resolution k_s for the prior defined system setup. The presented data serves as evaluation base for the applicability of the provided traffic monitoring for the intended use cases.

Table 4.5: Configuration dependencies of traffic sensor threshold, monitoring cycle, and observable cluster size

v_{thresh}	ID_{thresh} dec→bin	s_{OFG} $n_{HL} = 2$	T_{MC}			N^*_{CS} with $s_{MDR \rightarrow F}@'01'/s_{MDR \rightarrow F}@'11'$	
			$k_s = 1$	$k_s = 2$	$k_s = 4$	2D-Mesh	QMesh
32	0→'000'	128	3.2	1.6	0.8	16/2	8/2
64	1→'001'	256	6.4	3.2	1.6	32/5	16/5
128	2→'010'	512	12.8	6.4	3.2	64/10	32/10
256	**3→'011'**	**1024**	**25.6**	**12.8**	**6.4**	**64/21**	**64/21**
512	4→'100'	2048	51.2	25.6	12.8	64/42	64/42
1024	**5→'101'**	**4096**	**102.4**	**51.2**	**25.6**	**64/64**	**64/64**
2048	6→'110'	8192	204.8	102.4	51.2	64/64	64/64
4096	7→'111'	16384	409.6	204.8	102.4	64/64	64/64
Unit		Byte per OFG	µs	µs	µs	Slave-Tiles	Slave-Tiles

Thereby, the provided flexibility and configurability becomes obvious. The monitoring cycles T_{MC} are adjustable between 800 ns and 409.6 µs via the combination of v_{thresh} and k_s. Furthermore, in both topologies, at the threshold of $v_{thresh} = 256$ the maximum cluster N_{CSmax} size would be observable with the component coverage of '01' and '10', while full coverage '11' is enabled for at least 21 tiles and for a scaling to N_{CSmax} the threshold must be adjusted to $v_{thresh} = 1024$. The sensitivity per reported overflow s_{OFG} ranges from 128 bytes up to 16 kbyte. Under consideration of cache line transfers with sizes of 32 up to 128 bytes per packet and communicated page sizes of 2 up to 8 kbyte, the adjustable sensitivity is suitable.

Beneath the direct utilization values of the monitoring data fields inside MM as given in Table 4.2, different additional traffic states can be obtained from it directly or indirectly as follows:

- As the global data injection is directly given by $TLOAD[1\ or\ 4][N_{CS}]$, the global data reception rates can be indirectly approximated through the link utilizations at the tile output ports of the router $LLOAD[Q0/Q1/Q2/Q3][N_{CS}]$. The connected NIs work via independent buffer segments and should enable the reception of incoming data immediately. Hence, the fraction of sensed busy-cycles through congestion or head-of-line blocking will be minimal and the utilization values will be as near as possible to the raw data loads.
- In addition to the global data injection/reception measurements above, the cluster-internal values can be obtained indirectly through summations based on the path utilizations $PLOAD[N_{CS}][N_{CS}]$. The sum of injected traffic from one source to all remaining cluster tiles $PLOAD[src][0\ to\ (N_{CS}-1)]$ represents its total transmissions into the cluster, while the sum of injections over all remaining destinations towards this source $PLOAD[0\ to\ (N_{CS}-1)][src]$ represents the total receptions from the cluster.
- Furthermore, the knowledge about global and cluster-internal data volumes can be used to calculate the traffic volumes of transmissions to and receptions from outside the cluster. In CMPs the intensity of external communication might be an indicator for system-input/output as well as external memory traffic. Hence, the ratio of internal and external communication may be an additional quality indicator for mapping and clustering.

Accordingly, these additional values help to provide a complete picture of the traffic situation inside the cluster. For more application specific mapping and fractioning of the collected data, the extension of this data via internal performance counter instrumentation [53], [59], [107], [227]–[229] can be very useful, but would introduce more interferences with the normal workload operations.

Next, the monitoring timing and sensitivity must be classified in the context of the expected traffic behavior in the CMP. Therefore, the findings of different works on real application loads and CMP sizes of 32 to 64 tiles are as follows:

- The results of *Heirman et. al.* in [67] show, that the bandwidth utilization per tile and the communication locality (rentian exponent) remain stable along intervals of $5{\cdot}10^6$ up to 10^7 CPU cycles. Thereby, the traffic was further analyzed regarding the power law behavior of the communication locality and the results show that the rentian exponent follows a periodic behavior along these intervals due to the alternation of computation and communication episodes during workload processing. The interval results in another work [230] are similar. For the exemplary case with a CPU clock cycle of 0.5 ns these stable intervals would last 2.5 ms up to 5 ms.

- The results of *Jin et. al.* in [19] on the profiling of dominant traffic flows in application workloads show, that the tile injections rates remain stable over the intervals of 0.5 up to 1 billion instruction executions. Assuming that one instruction takes one CPU cycle with a clock rate of 0.5 ns, these intervals would last 250 ms up to 500 ms.
- In [53], *Mishra et. al.* profiled the length of network communication episodes as well as their intensity/height (in number of packets). The results show episode lengths in the range of 10^2 up to 10^6 NoC clock cycles and the episode heights vary between 1 and 12 outstanding packets with a payload size of 128 bytes.

These provided traffic characteristics of multiple independent works [19], [53], [67], [230] show, that the proposed traffic monitoring is well suited regarding fine grained observations as well as the expectable frequency of changes in the communication behavior. As Table 4.5 indicates for the exemplary system setup, reaction time of µs are provided for 64 tile clusters and therefore the only limiting factor might be the execution speed of the software modules.

The traffic monitoring and classification can be realized for different requirements regarding timing, cluster size, scale resolution and component coverage. Furthermore, this run-time configurability provides the opportunity for a light-weighted monitoring that evaluates if active management is required, triggers the corresponding service instances if thresholds are reached, and adjusts its configuration to the new situation. The use cases for the light-weighted traffic monitoring and evaluation would simply contain a threshold based observation of the current traffic situation and thus a preliminary stage to active traffic management. Depending on the used metric, tile output, path utilizations, and/or link utilizations might be used. Hence, the software module at the master-tile would perform computations such as the analysis of minimum/mean/maximum tile, path loads or link loads, while parsing through the gathered data. The major limitation will be the complexity of calculations regarding their execution times in relation to the required reaction times.

Application mapping and profiling are supported as well. The application mapping needs the information of the current traffic situation to evaluate optimal allocations and bindings of resources for the workload applications [17], [22], [25], [110], [111], [130], [214]. Therefore, it further utilizes the traffic requirements specified by the communications graphs for each workload application in addition to similar formats for the computational fraction. The application profiling provides the sourcing of this information. The run-time application mapping needs at least the current traffic situation along tile outputs and links ($TLOAD[1\ or\ 4][N_{CS}]$ and $LLOAD[N_P][N_{CS}]$). With integrated capabilities for path adaptations, full component coverage is required ($TLOAD[1\ or\ 4][N_{CS}]$, $LLOAD[N_P][N_{CS}]$ and $PLOAD[N_{CS}][N_{CS}]$). Those monitoring data sets enable the mapping service to compute optimized solution in the context of the dynamic workload behavior.

Otherwise the calculations would be performed on static assumptions. Additional internal tile statistics for the computational aspect might be favorable and can be provided by intra-tile performance counter instrumentation [53], [59], [107], [227]–[229]. Another highly relevant feature is the evaluation of the mapping quality as feedback for the corresponding service instances. Therefore, different heuristics exist that result from a mix of latency, throughput, component utilization, or idle time measures [17], [22], [25], [110], [111], [130], [214]. These heuristics typically address the optimization tradeoff between performance and power/energy. But the evaluations in recent works provide that the greedy nature of optimization algorithms results in a spatio-temporal communication behavior and locality relations that can be described via basic power laws [11], [12], [47], [50], [67]. Regarding these different works, a synthetic rentian traffic model was formulated in [12] and [14] to perform NoC design explorations with a traffic pattern that reflects the real workload behavior and is applied in the work at hand as well. Via the rentian exponent $0 < R \leq 1$ the traffic pattern can be adapted to a communication behavior with higher locality ($R \downarrow$) or lower locality ($R \uparrow$). Conversely, this rentian exponent R could offer a single parameter heuristic for the quality of an applied mapping inside the CMP. This strategy is already known from the circuit design, where Rent's rule originates from the analysis of wire length distributions and placement optimization in VLSI designs [231], [232]. The traffic monitoring of the work at hand has the ability to provide the rentian exponent R for the communication via evaluation of the path utilizations $PLOAD[N_{CS}][N_{CS}]$.

For the use case of the application profiling, the assigned workload fraction to the cluster should be limited to the application of interest and full component coverage must be activated ($TLOAD[1\ or\ 4][N_{CS}]$, $LLOAD[N_P][N_{CS}]$ and $PLOAD[N_{CS}][N_{CS}]$). Moreover, it requires dedicated support at the mapping service to generate isolated profiling inside the cluster. The explicit application profiling for the communication behavior can be part of preliminary workload evaluations as well as run-time mode for application executions to refine given statistics for specific user-spaces.

4.4 EXPERIMENTAL RESULTS

The experimental evaluation of the proposed traffic monitoring (TRM) focusses on different key aspects, such as the infrastructure performance, monitoring accuracy, implementation costs and software performance. The TRM is configured according to the methodology as described in the prior Section 4.3.4. The specific details are described in the following subsections on the results of the experimental evaluations. In general, the TRM is explored via simulations for different cluster sizes and shapes. Thereby, the 16 tile 4x4 and the 64 tile 8x8 clusters represent the major corner cases. The TRM is fully integrated into the SystemC-based NoC simulator of the work at hand. For the purpose of the intended infrastructure reusability, a periodic temperature monitoring (TPM) similar to this one of *Zhao et. al.* in [195] is performed in parallel to the TRM to study interferences. For this

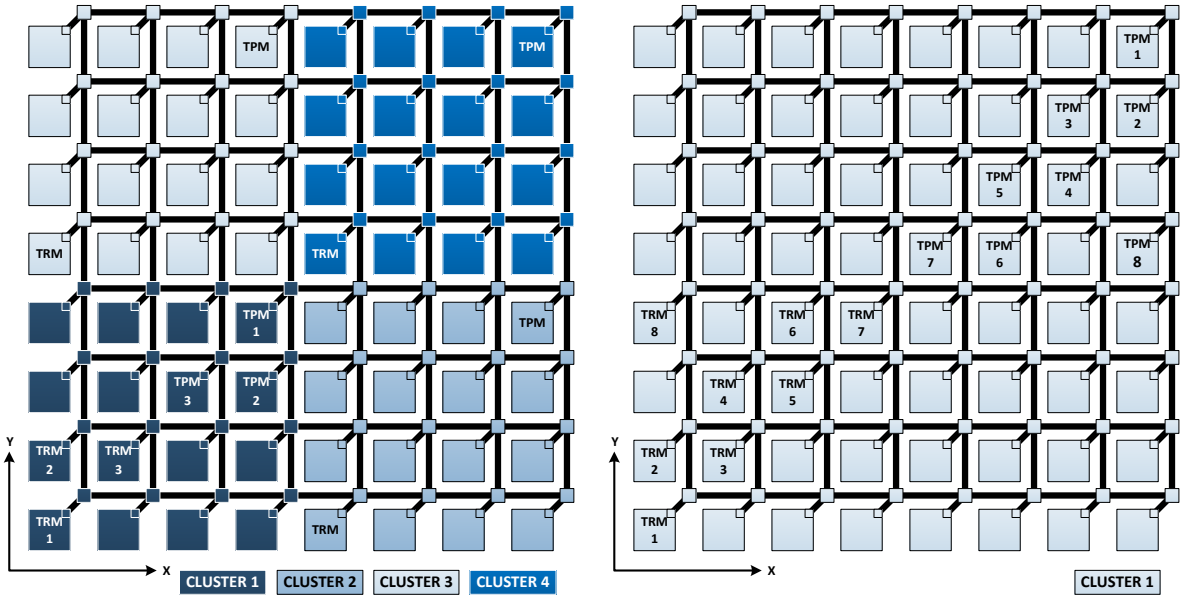
TPM each tile is suited with 8 temperature sensors, whereas each of them produces sensor data of 8 bits. Hence, each tile produces a TPM monitoring packet with 8 bytes of payload. With the additional packet header and the intended data width of 16 bits for the SNoC links, each tile reports $s_{MDR-F} = 5$ flits per TPM packet along the SNoC for each monitoring cycle T_{MC-TPM}. In [195], *Zhao et. al.* report a required monitoring cycle of 1600 cycle at a clock of 2 ns, which results in $T_{MC-TPM} \leq 3200$ ns. Hence, the selected monitoring cycle is set to $T_{MC-TPM} \leq 2048$ ns and each tile provides TPM data with higher sampling rates at 36% lower monitoring cycles. The TPM itself has a central organization and all slave tiles report to a single master core inside a cluster. TPM and TRM clusters are allowed to overlap. The evaluated parameter configurations are outlined in Table 4.6.

Table 4.6: Parameter setups for the experimental evaluation of the traffic monitoring

PARAMETER	CONFIGURATION
DNoC Setup	2D-Mesh and QMesh at 45 nm → see Table I.2 at p. 164
SNoC Setup	2D-Mesh and QMesh at 45 nm → see Table I.4 at p. 166
Cluster Shapes	Quadratic : 2x2, 3x3, **4x4**, 5x5, 6x6, 7x7, **8x8** @ N_{CSmax} = 64 tiles Rectangular : 8x2, 16x4 @ N_{CSmax} = 64 tiles
TRM Configuration	see Table 4.4 at p. 120 and Table 4.5 at p. 120
TPM Configuration	$T_{MC-TPM} = 2048$ ns and $s_{MDR-F} = 5$ flit
Processor Configuration	see setup table in Appendix at p. 165
Application Graphs	Nodes : 7 up to 70 threads per graph Edges : 5 up to 15 and 5 to 50 flits per packet (random uniform assignments) Activation Periods : 100 to 500 clock cycles (random uniform assignments) Applied Mapping : Random uniform
Master-tile Placement	TRM at LLC and TPM at URC of the cluster as worst-case for 2D-Mesh and QMesh if not explicitly reconfigured

4.4.1 MONITORING INFRASTRUCTURE, COMPONENTS AND MEASUREMENTS

Initially, the communication performance of the SNoC is explored for different cluster configurations as well as master-tile selections for the TRM and TPM inside the 2D-Mesh/QMesh. Figure 4.9 illustrates two different cluster setups for an 8x8 2D-Mesh NoC. In Figure 4.9 (a), four 16 tile 4x4 clusters coexist and each cluster contains a TRM master-tile at its LLC and a TPM master-tile at its URC. Furthermore, CLUSTER 1 indicates the different placement pairings (TRM 1 to 3 and TPM 1 to 3) for the interference evaluations. In Figure 4.9 (b), a 64 tile 8x8 cluster is depicted with the different placement pairings (TRM 1 to 8 and TPM 1 to 8) for the interference evaluations. All of these configurations are simulated and the resulting packet delays of the monitoring applications are captured. Thereby, it was assumed that each TRM slave tile injects one MDR each expired sensor period, while the TPM slave tiles perform this way by default, to cover the worst case communication. Furthermore, the COVERAGE at the TRM was set to '11', which results in a MDR packet size of $s_{MDR-F} = 6$ flits according to Table 4.5 (see p. 120).



(a) 16 tile clusters with 4x4 shape

(b) 64 tile cluster with 8x8 shape

Figure 4.9: 4x4 and 8x8 cluster configurations for the traffic monitoring evaluations in a 2D-Mesh topology

The bar-chart in Figure 4.10 below depicts the aggregated results for the 4x4/8x8 cluster with all simulated master-tile placements (TRM/TPM 1 to 8) and TRM timings. The left part in Figure 4.10 illustrates the results for the mean packet header delay inside the 2D-Mesh as well as the QMesh SNoC, if the TRM timing is configured normally to $v_{thresh} = 256/1024$ according to Table 4.5 (see p. 120) for $N_{CS}^{*} = 16/64$ tiles as allowed cluster size after Equation 4.12 (see p. 105). The highlighted limit (orange) indicates the TMR sensor period for the overflow reports (TPM limit is twice as high). The SNoC does not provide bandwidth or latency guarantees and thus the MDR packet delivery is best-effort based. The MDR packet shall be able to reach the master-tile within this sensor period to keep up with the monitoring timing. Thereby, the QMesh clearly outperforms the 2D-Mesh due to the increased number of NIs at the master-tiles as well as the shortened path lengths (see Section 3 pp. 65-83). The more the TRM and TPM master-tile are concentrated in the middle of the cluster, the lower the mean MDR packet delay becomes due to the better distribution of the MDR traffic to all of the four available NIs. As expected, the impact of the traffic interference between TRM and TPM packets is low. This results from the many-to-one communication, whose mean packet delay is dominated by the queuing at the master-tiles as central hotspots. Furthermore, the TRM/TPM 1 mapping to LLC and URC proves as worst case for the 2D-Mesh as well as for the QMesh. The right part of Figure 4.10 illustrates the mean packet header delay results for critical sensor periods ($v_{thresh} = 256\ and\ 512$) with remaining cluster size of 64 tiles at the three corner case placements TRM/TPM 1, 7, and 8. They prove the cluster size after Equation 4.12 (see p. 105) and show that MDR packets are likely to miss their sensor period deadlines for lower timings. Only the QMesh could keep up at the master-tile placement in the middle of the cluster (TRM/TPM 7 @ 256/512).

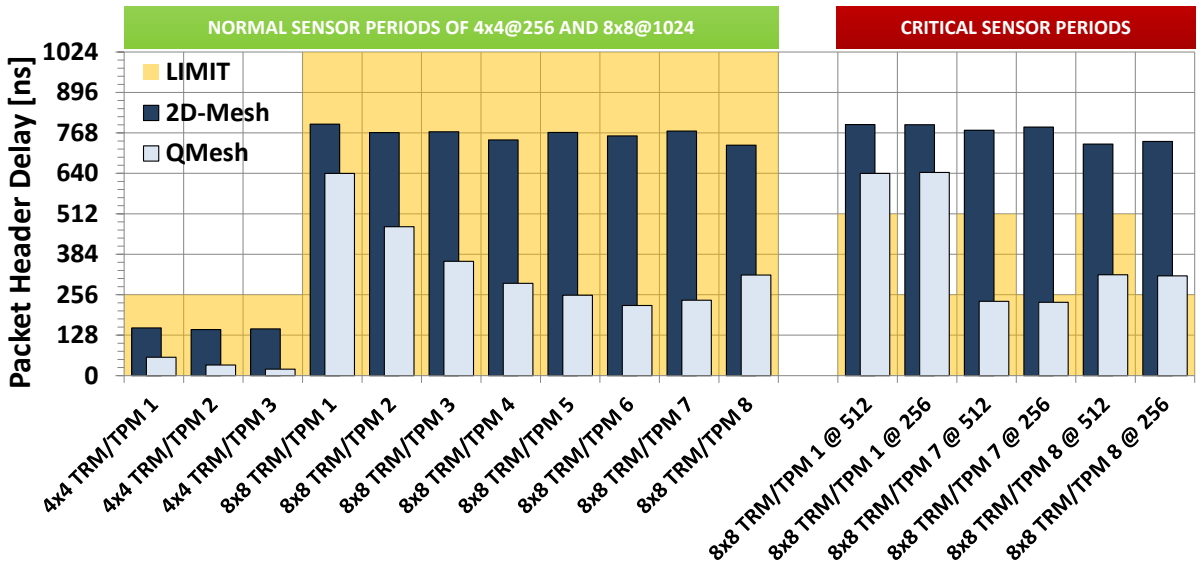


Figure 4.10: System-NoC monitoring delays for different configurations inside a 4x4 and 8x8 cluster

For the general avoidance of traffic path interference in such bipolar cluster operations with overlapping regions, the specific characteristics of the applied xy-routing can be utilized. As long as the master-tiles are placed opposed on the left and the right edge of the cluster (or beyond), their many-to-one traffic flows do not share any link along the xy-paths:

$$|x_{MT-TRM} - x_{MT-TPM}| \geq (n_{CX} - 1)$$

For a similar case to this one depicted in Figure 4.9 (a), if TRM or TPM would cover the full 8x8 region and the other one operates with the four 4x4 clusters, the master-tile placement condition must be verified for all master-tile combinations. In additional to the normal TRM/TPM monitoring traffic, the simulation of the TRM contained regular cluster update transactions to measure the required reconfiguration time of the cluster. Therefore, the TRM master-tile transmitted CUP packets to all slave tiles and afterwards collects the CSP packets from them. The CUP transmissions began with addressing the LLC and succeeded row by row up to the URC. The required time from transmission of the first CUP to the reception of the last CSP was logged and the results are as follows:

- **4x4 2D-Mesh cluster** → 660 ns at TRM/TPM 3 and 770 ns at TRM/TPM 1
- **4x4 QMesh cluster** → 270 ns at TRM/TPM 3 and 630 ns at TRM/TPM 1
- **8x8 2D-Mesh cluster** → 4500 ns at TRM/TPM 7 and TRM/TPM 1 as well
- **8x8 QMesh cluster** → 1120 ns at TRM/TPM 7 and 4500 ns at TRM/TPM 1

Hence, reconfiguration of sensor periods inside the cluster can be performed in less than 1 µs for the 4x4 cluster and less than 5 µs for the 8x8 cluster.

The next experiments evaluate the monitoring error of the proposed TRM in the work at hand. Therefore, the 16 tile cluster ($v_{thresh} = 256$) is shaped as 4x4 and 8x2 regions, while the 64 tile cluster ($v_{thresh} = 1024$) is shaped as 8x8 and 16x4 regions. As worst-case scenario the 2D-Mesh topology is chosen with TRM and TPM operating in parallel under consideration of the LLC/URC placement for the master-tiles. These monitoring configurations are simulated with two different traffic scenarios at all possible values of 1, 2 and 4 for the scale resolution k_S. Thereby, the real utilizations of all paths ($PLOAD[N_{CS}][N_{CS}]$) and links ($LLOAD[N_P][N_{CS}]$) are logged and compared to the captured data of the proposed monitoring after each finalized monitoring cycle. The resulting deltas Δ represent the absolute monitoring errors $e_{PLOAD/LLOAD} = |\Delta|$. The maximum as well as the mean values of $e_{PLOAD/LLOAD}$ for the $PLOAD[N_{CS}][N_{CS}]$ and $LLOAD[N_P][N_{CS}]$ are extracted from these statistics for each simulated parameter variation. The results for the tile outputs $TLOAD[1][N_{CS}]$ are similar to those of the path utilizations.

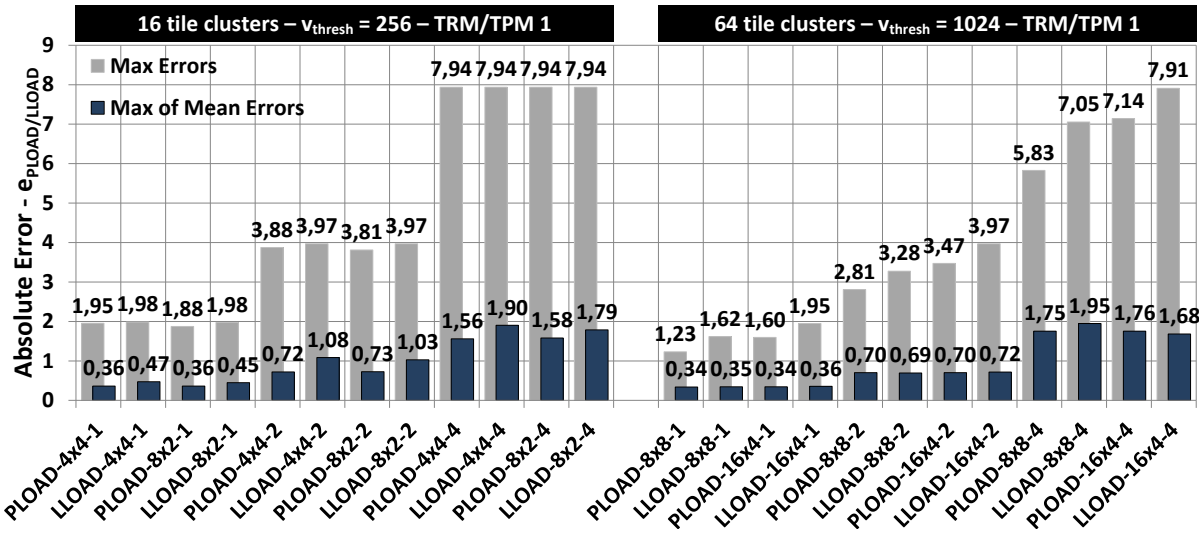


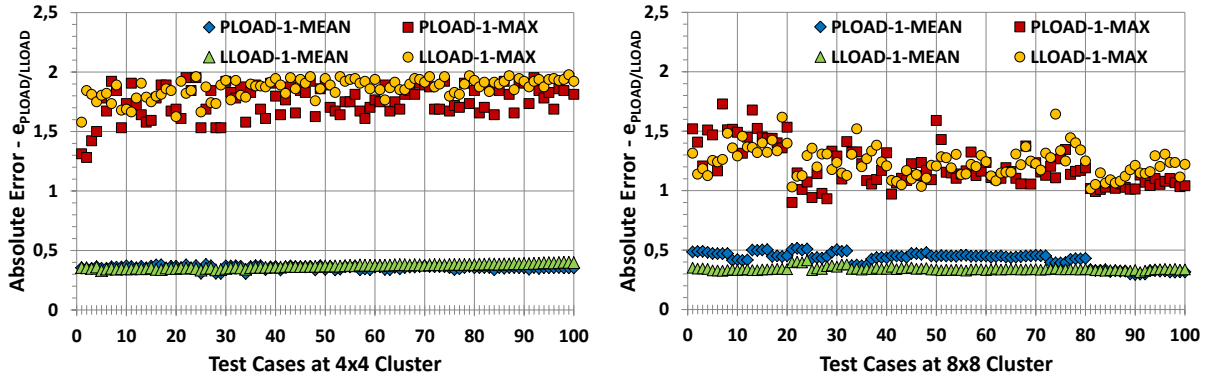
Figure 4.11: Maximum and mean monitoring errors for all simulated cluster shapes, scale resolutions and sizes of 16 and 64 tiles in a 2D-Mesh topology

The first simulated scenario covers the traffic injection per tile from low communication activities up to a fully saturated DNoC, which allows the evaluation of the traffic monitoring stability. Therefore, the uniform random traffic pattern (see Appendix at p. 161) is applied for the DNoC communication, because it enables a balanced traffic distribution throughout the complete DNoC and therewith activates most of the traffic sensors. Each tile injects packets with a uniform selection of the destination address and a size in the range of 5 to 15 flits. The bar-chart in Figure 4.11 contains the monitoring error evaluations of this first traffic scenario for all cluster shapes, scale resolutions and cluster sizes. Therefore, the maximum errors as well as the maxima along the mean errors for the detected traffic utilizations are depicted for all simulated configurations (bar labelling is <LLOAD/PLOAD>-<cluster shape>-<k_s >). As mentioned in Section 4.2.2 at the introduction of

Equation 4.7 (see p. 98), the selected scale resolution affects the potential monitoring error, because each overflow has the weight of k_s percent in the final utilization values. The higher k_s is chosen, the higher the maximum error and the lower the adjustable monitoring cycle T_{MC}. These results indicate that the errors have fixed limits if cluster size and timing of the monitoring are sized according to the constraints described in Section 4.3.4 (see pp. 119-123). These observed limits are as follows:

- **Maximum/Worst-Case** → $e_{PLOAD/LLOAD} < 2 \cdot k_S$ per monitoring cycle
- **Mean/Practical** → $\tilde{e}_{PLOAD/LLOAD} \leq 0.5 \cdot k_S$ per monitoring cycle

The explanation for the maximum error per monitoring cycle can be found in the combination of two circumstances. First, at the moment of the load data read out in the AU the last reported overflow OFG_j of a traffic sensor j is not processed right on time and remains inside the OFG-REPORT-BUFFER of the AU. Second, since this last overflow report the same sensor has reached a counter value of $cnt_j = v_{thresh} - 2$. Hence, these utilization values are not registered by the traffic monitoring inside the current cycle. But they will be registered inside the next monitoring cycle and are not lost.



(a) 4x4 cluster – v_{thresh} = 256 – TRM/TPM 1 (k_S=1)

(b) 8x8 cluster – v_{thresh} = 1024 – TRM/TPM 1 (k_S=1)

Figure 4.12: Maximum and mean monitoring errors for different clusters with 16 and 64 tiles over the variation of 100 random application graph test cases in a 2D-Mesh topology

The second scenario targets the simulation of workloads with mixed traffic characteristics. Therefore, a set of random application graphs with 7 to 70 threads per graph is used. Out of this set, 100 random workloads (test cases) with 20 to 400 threads (2 to 10 application graphs) per workload are randomly composed and simulated with a random mapping. The threads communicate via packets with a uniform random sizing from the interval of 5 up to 50 flits that is selected for each communication edge at the beginning of a simulation run. The activation of the application graphs is triggered periodically in a uniform random range of 100 to 500 clock cycles. The resulting traffic in the DNoC is unbalanced with a high spread of the utilizations at the paths and links. Thus, its communication characteristics vary from the first simulation scenario. The resulting maximal (PLOAD/LLOAD-1-MAX) and mean (PLOAD/LLOAD-1-MEAN) monitoring errors for the highest scale resolution ($k_s = 1$) are depicted in Figure 4.12 (a) for the 4x4 cluster and in Figure 4.12 (b) for the

8x8 cluster. The results confirm the observed monitoring error observations of the first simulated traffic scenario.

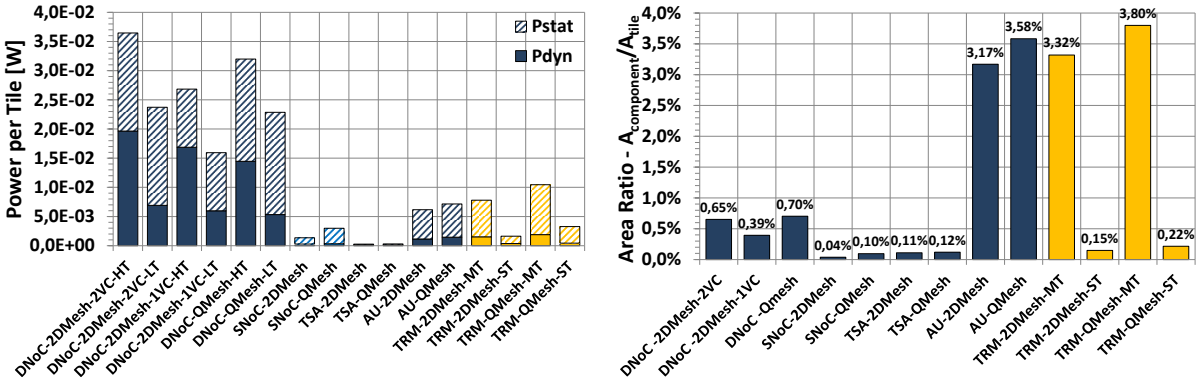
Finally, the results for the costs of the proposed traffic monitoring configuration regarding area footprint and power dissipation are discussed. Therefore, Table 4.7 contains the hardware synthesis results per tile for all major TRM and NoC components in the context of three different DNoC configurations. The TSA and AU units are synthesized via the Synopsys Design Compiler based on their VHDL descriptions and with the utilization of the 45nm Nangate Open Cell [202] technology library for typical operating conditions (DSENT uses similar technology models [89]). The TRM master-tile (TRM-MT) values contain the sum over the SNoC, TSA and AU results. The TRM slave-tile (TRM-ST) values contain the sum over the SNoC and TSA results. The NoC component values contain the combined results for the router as well as the links per tile and were obtained via DSENT [89]. The DNoC power values are given for the low and the high traffic intensity scenarios as already applied in Appendix for the preliminary evaluations (see pp. 167-175). The SNoC values represent the mean power dissipation per tile over a full variation of the allowed TRM/TPM parameter configurations inside the 4x4 and the 8x8 cluster scenarios.

Table 4.7: Power and area evaluation results of the proposed traffic monitoring per tile at N_{CSmax} of 64 tiles

COMPONENT	2D-MESH - 1 VC			2D-MESH - 2 VC			QMESH		
	P_{dyn}	P_{stat}	Area	P_{dyn}	P_{stat}	Area	P_{dyn}	P_{stat}	Area
DNoC – HT	1,69E-02	9,96E-03	3,54E-08	1,97E-02	1,68E-02	5,87E-08	1,45E-02	1,75E-02	6,33E-08
DNoC – LT	5,98E-03	9,96E-03		6,91E-03	1,68E-02		5,33E-03	1,75E-02	
SNoC	2,80E-04	1,08E-03	3,44E-09	2,80E-04	1,08E-03	3,44E-09	3,30E-04	2,65E-03	8,57E-09
TSA	9,07E-05	1,83E-04	9,93E-09	9,07E-05	1,83E-04	9,93E-09	9,93E-05	2,00E-04	1,08E-08
AU	1,16E-03	5,02E-03	2,85E-07	1,16E-03	5,02E-03	2,85E-07	1,47E-03	5,70E-03	3,22E-07
TRM-MT	**1,53E-03**	**6,28E-03**	**2,99E-07**	**1,53E-03**	**6,28E-03**	**2,99E-07**	**1,89E-03**	**8,55E-03**	**3,42E-07**
TRM-ST	**3,71E-04**	**1,26E-03**	**1,34E-08**	**3,71E-04**	**1,26E-03**	**1,34E-08**	**4,29E-04**	**2,85E-03**	**1,94E-08**
Units	W	W	m^2	W	W	m^2	W	W	m^2

The power dissipation values per component from Table 4.7 are depicted in the stacked bar-chart of Figure 4.13 (a). The bar-chart in Figure 4.13 (b) depicts the relative area footprint of the total area results from Table 4.7 as share of the foreseen 3 mm x 3 mm tile. The total power dissipation of the SNoC per tile is dominated by the static power, despite the minimum input port buffer sizing to the depth of one flit. Furthermore, the evaluated variations of the TRM/TPM parameter configurations results in minimal differences in the total power dissipation. The 2D-Mesh-based SNoC consumes on average 1.36 mW per tile for its router and four links, while the QMesh-based SNoC requires 2.98 mW on average. In comparison to the DNoC, the relative power overhead due to the SNoC is around 10% or less. The power and area costs of the TRM are dominated by the sensor/counter arrays inside the TSA/AU. As expected, the power/area costs for the TRM in a QMesh topology are higher due to the increased number of traffic sensors and load counter. Thereby, the share of the AU per TRM-MT

is obvious in charts of Figure 4.13 (a) and (b), while the power impact of the TRM-ST is lower by a factor of 3 up to 5 and its area overhead is at least one magnitude lower if compared to the TRM-MT. Furthermore, it is expected that most tiles inside the CMP are slave-tiles and only a share of around 25% are potential master-tiles (see Figure 4.8 at p. 114). Hence, the cost overhead of the TRM-MT is relativized in the global CMP context.



(a) Total power (b) Total area

Figure 4.13: Overview for power and area cost of the proposed traffic monitoring at N_{CSmax} of 64 tiles

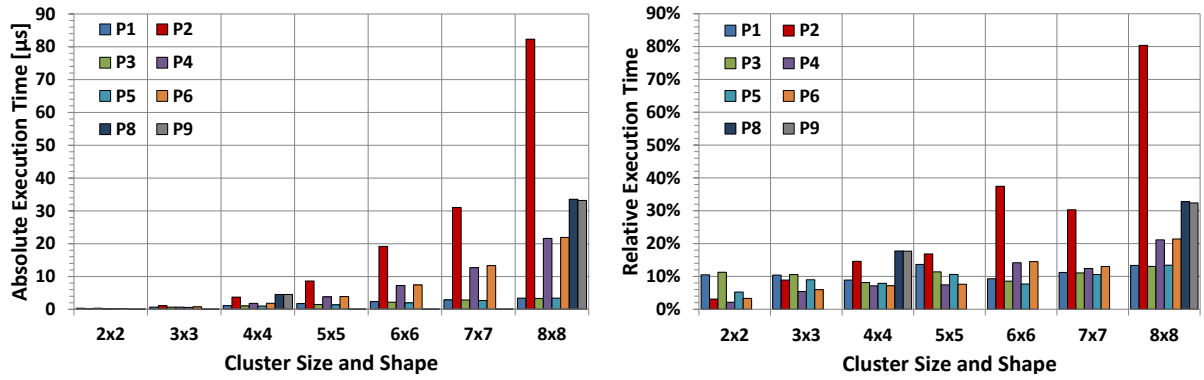
4.4.2 SOFTWARE MODULE PERFORMANCE

In addition to the hardware evaluation, the performance of the TRM cluster agent software module on the master-tile processor is profiled. Therefore, nine vital procedures (P1 up to P9) for the monitoring data evaluation are profiled via Sniper on a single processor core of the 64-tile CMP specified in Appendix (see setup tabe at p. 165). The functional context, the evaluated worst case topology case and the required TRM coverage configuration can be obtained from Table 4.8. The procedures are programmed in C and compiled with the Gnu compiler without any optimization flags. In general, the processing of minimum/mean/maximum utilization values in P1 to P3 represents basic functions performed for a light-weighted traffic monitoring and traffic classification. The procedures P4 to P6 represents the necessary processing to evaluate the additional internal/external communication states indirectly from the monitored observations as outlined in Section 4.3.4 (see pp. 119-123). Procedure P7 is a typical calculation for active traffic management with detailed path evaluations. P8 and P9 implement the approximation method for the rentian exponent as mentioned in Section 4.3.4 (see pp. 119-123). The timing results for each procedure represents the average of 1000 runs. They are provided by Sniper through cycle-approximate instruction level instrumentation for defined regions-of-interest inside the source-code. Hence, the preliminary data generation and setup is excluded and the pure procedure runs are profiled.

Table 4.8: Test procedures for the performance evaluation of the traffic monitoring software module

Procedure Descriptions		WC-SNoC	Coverage
P1	Calculate minimum/mean/maximum link utilization for the cluster → Parse $LLOAD[N_P][N_{CS}]$	QMesh	'10'
P2	Calculate minimum/ mean /maximum path utilization for the cluster → Parse $PLOAD[N_{CS}][N_{CS}]$	Equal	'11'
P3	Calculate minimum/ mean /maximum global tile output for the cluster → Parse $TLOAD[4][N_{CS}]$	QMesh	'01'
P4	Calculate minimum/ mean /maximum cluster-internal tile output for the cluster → Parse $PLOAD[N_{CS}][N_{CS}]$	Equal	'11'
P5	Calculate minimum/ mean /maximum global tile input for the cluster → Parse $LLOAD[N_P][N_{CS}]$	QMesh	'10'
P6	Calculate minimum/ mean /maximum cluster-external tile input/output for the cluster → Parse $PLOAD[N_{CS}][N_{CS}]$, $LLOAD[N_P][N_{CS}]$, $TLOAD[4][N_{CS}]$	QMesh	'11'
P7	Calculate minimum/ mean /maximum link utilization along xy-paths inside the cluster → Parse $LLOAD[N_P][N_{CS}]$	2D-Mesh	'10'
P8	Rentian exponent approximation inside the 2D-Mesh clusters with 4x4 and 8x8 shape → Parse $PLOAD[N_{CS}][N_{CS}]$		'11'
P9	Rentian exponent approximation inside the QMesh clusters with 4x4 and 8x8 shape → Parse $PLOAD[N_{CS}][N_{CS}]$		'11'

The bar-chart in Figure 4.14 (a) illustrates the absolute execution times per procedure call over the variation of the cluster size for P1 to P6, P8 and P9.



(a) Absolute Procedure Execution Times

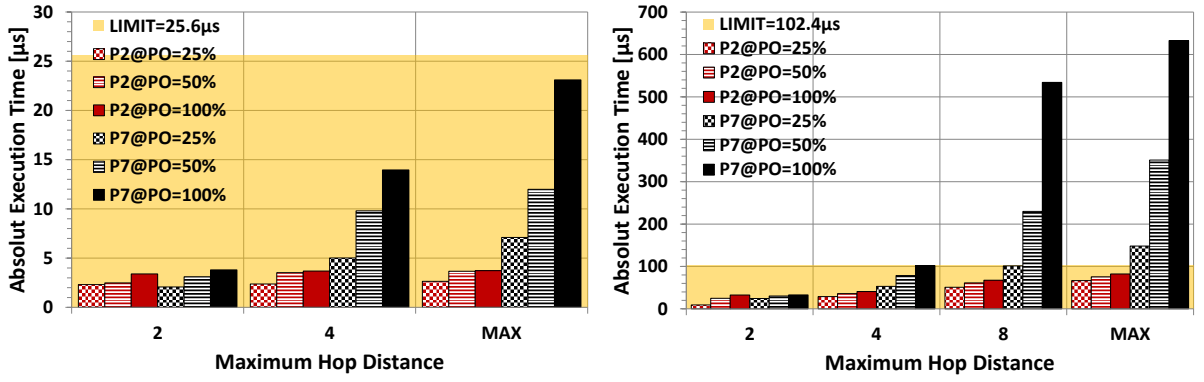
(b) Relative Procedure Execution Times (k_S=1)

Figure 4.14: Measured test procedure execution times (P1 to P6, P8 and P9) as absolute values and relative to the deadlines defined by the adjusted monitoring cycle over a variation of the cluster size

Additionally, the bar-chart in Figure 4.14 (b) depicts the relative fraction of these executions times regarding the periodic deadlines defined by the specific monitoring cycle duration T_{MC} at the configured component coverage and a scale resolution of $k_s = 1$. As Figure 4.14 (b) shows, all of the evaluated procedures would meet their deadlines until the next monitoring data is available at all cluster sizes. Furthermore, the vast majority consume less than 20% of the available CPU time and would be applicable for increased scale resolutions ($k_s = 2\ and\ 4$) with decreased monitoring cycles. Only procedures with more complex functionality, such as the parsing of all paths in P2 or the parsing of dedicated hop regions per tile in P8/P9, would not support the deadline reductions at scale resolution of $k_s = 4$ for cluster sizes above 25 tiles. But this won't be critical at all, because the evaluated scenarios consider a full path occupation as given in a random uniform communications

scenario. More realistic traffic scenarios are expected to have lower fractions of occupied paths with few dominating connections [19].

A more detailed evaluation of the impact of path occupation and maximum hop distances of the communication is performed for the procedures P2 and P7 due to the increased dependency of their computational efforts on these parameters. The path utilizations in P2 as well as the link utilizations along xy-paths in P7 must be considered solely if those ETE paths carry non-zero loads. The bar-charts in Figure 4.15 illustrate the resulting absolute executions times as well as the required deadlines (LIMIT) for the evaluated parameter variations in a 4x4 and an 8x8 cluster. Thereby, the path occupation (PO) is varied over 25%, 50% and 100%. The maximum hop distance is set to 2, 4 and unconstrained (MAX) in the 4x4 cluster scenario, while the 8x8 cluster is further evaluated for the additional maximum distance of 8 hops. As Figure 4.15 (a) shows, P2 and P7 would meet their deadlines for all parameter variations.



(a) 4x4 cluster – v_{thresh} = 256 (k_S=1) (b) 8x8 cluster – v_{thresh} = 1024 (k_S=1)

Figure 4.15: Measured test procedure execution times (P2 and P7) as absolute values over a variation of the maximum hop distance (2, 4, 8, MAX) and communication activities along these paths (PO=25%, 50%, and 100%) in a 2D-Mesh for the 4x4 and the 8x8 cluster setup

The results of the 8x8 cluster cases depicted in Figure 4.15 (b), show that at least P2 would meet the deadlines for all evaluated scenarios, while P7 would not meet them for all scenarios with maximum distances of at least 8 hops and a path occupation equal or higher to 50%. But these critical parameter variations are extreme scenarios that might not occur in realistic operating conditions. The configuration with a maximum hop distance of 8 and a path occupation of 50% considers the communication of each tile with at least 18 up to 32 other tiles inside the cluster without major locality restrictions. The other critical configurations are even more intense regarding the number of active communication partners. But a look on the distribution of more realistic traffic pattern, such as the rentian CDF in Appendix (see p. 162), reveals that such extreme situations are unlikely. With a rentian exponent of R=0.7, the main fraction of communication occurs within the hop range of 4 inside the 2D-Mesh and within the hop range of 3 inside the QMesh. This region is responsible for about 89% of the traffic of a tile. With a rentian exponent of R=0.3, this fraction increases up to 98%.

But if such situations occur, they could be handled through different optional workarounds that consider the adjustment of the timing higher cycle durations such as 204.8 µs or 409.6 µs, the increase of the utilization threshold level for relevant paths to values higher than zero, the reduction of the evaluation to a defined hop region, the reduction of the evaluation to a defined/relevant subset of tiles in each monitoring cycle, and the split-up of the original monitoring cluster to multiple smaller ones if such traffic scenarios are expectable.

Finally, it has to be mentioned that such critical traffic scenarios might not consider path adaptations at all. As the gathered simulation results in Sections 5.2.2 (see pp. 142-149) on such uniform distributed communication scenarios show, such mechanisms would be effective for path occupations below 40% inside an 8x8 cluster and the number of resulting path updates are far lower. Furthermore, the performance of P7 is dominated by the division operations for the calculation of the mean link utilization per path. The simulated core architecture requires one CPU cycle per addition or comparison, but the final division consumes 18 cycles per instruction for every evaluated path and this more than doubles the computational costs for the average case. Hence, the software procedures should avoid such costly operations in their path evaluation metrics.

4.5 SUMMARY AND OUTLOOK

The proposed traffic monitoring represents a vital run-time feature for the CMP's NoC infrastructure and serves itself as intermediate mechanisms to provide self-awareness regarding the current traffic situation. It could be shown that traffic monitoring can be designed to provide sufficient spatio-temporal adaptivity, capabilities and flexibility to capture the communication behavior of modern workloads with adequate accuracy for different use cases. The main properties of the proposed solution can be summarized as follows:

- The **spatial scope** of the monitoring can be adjusted at run-time to rectangular shaped regions with **up to 64 tiles** and it is shown that this sizing limit is sufficient for parallelized workload applications. Multiple regions can be observed independently with individual configurations adaptable during run-time. Fixed spatial regimes at design-time can be avoided and more flexible support for mapping adaptations is obtained.
- The **timing scope** of the traffic monitoring is adjustable **in the range of 10^3 up to 10^5 clock cycles** per monitoring cycle for a cluster. At the end of each monitoring cycle, the provided monitoring data consists of ready-to-use utilization values of the observed NoC components. Thereby, costly computations as well was timestamp synchronizations are avoided, finer resolutions can be achived, and the timing range is suitable for dynamic workload behavior.

- The monitoring in **each cluster has a central organization** and a single entity has a global view on its assigned region. The monitoring **data evaluation itself is performed online in software** and provides full adaptivity to support different use cases. The data sensing and aggregation is realized in hardware and distributed via dedicated stages along the tiles. Furthermore, specialized out-of-band processing with single point if failure ressources are avoided.
- **For a cluster the timing scope can be adjusted locally by the central master-tile as well as globally by the reconfiguration of all slave-tiles.** Furthermore, the classes of components covered by the monitoring can be configured at run-time and therefore adjusted for the desired use case. The dependencies between cluster sizing, component coverage, timing, and monitoring sensitivity were discussed and a monitoring configuration flow is proposed.
- **The monitoring data communication is performed over the fully customizable and independent NoC.** Therefore, it could be shown that this kind of communication offers good reusability and can be extended to a general infrastructure for the data communication relevant for the system control.
- **The evaluated hardware costs are in a feasible range of lower than 5% of the anticipated tile area** and the performance profiling of the software module shows that the monitoring can keep up with the required deadlines in all realistic scenarios. Thereby, the resulting monitoring errors did not exceed the double of the adjusted scale resolution in maximum and were lower than the half of this resolution in the average case.
- **The evaluated power consumption of the additional TRM efforts** (SNoC, Traffic Sensors, Aggregation Units) **reside in a suitable range below 20 mW at 45nm CMOS technology**. Considering the assumptions of [43] with a power budget per computational core in the range of 0.5 to 1 Watt at 45nm, these additional efforts would range in a **power budget fraction of 4 to 2 percent**.

The focus of the next investigations in the work at hand is on the combination of this traffic monitoring with the QMesh to apply software-controlled run-time traffic management through a centralized path adaptation for the clusters. For the outlook regarding future investigations, the combination of the monitoring with dedicated performance counter strategies [107], [228], [229] to improve the coverage of the workload behavior with additional observations on the computational episodes, seem most promising. Furthermore, full integration of the traffic monitoring as service instance at the level of an operating system should encompass this next step, to evaluate all side effects and implications in the interplay with other management services.

5 REGIONAL ADAPTIVE ROUTING

In this Chapter the proposed QMesh topology of Chapter 3 (see pp. 55-85) and the traffic monitoring of Chapter 4 (see pp. 85-135) are combined to a software-based and regionally-oriented path adaptation that works independent for each created cluster inside the NoC. The resulting solution covers the complete functional framework introduced in Figure 4.1 (a) (see p. 95). Hence, a spatially restricted run-time traffic management service is introduced that works with a global view on its assigned region and workload fraction. The expected features of this combination are as follows:

- The QMesh already provides all actors for a dual-path adaptation strategy through its programmable path tables and applies deterministic xy-routing without any demand for virtual channels. The intended adjustment of the dual paths operates upon a deadlock- and livelock-free communication infrastructure (see Section 2.2 at pp. 46-55 for background information). The selectable paths are truly independent, locality-optimized, deterministic and known by one central instance per region. The last circumstance can be very helpful for the coordinated resolving of critical traffic situations, fault-detection and general traffic optimization between several services (such as mapping and path adaptation). The paths remain stable on the message-level and the packet per message will be delivered in-order.
- The applied traffic monitoring covers the complete path from the traffic utilization at the corresponding NIs, over all traversed links, down to the ejection port at the destination router with a scale resolution of 1% for each gathered path/component value. Furthermore, this monitoring data includes global contention information at the link-level and raw cluster-internal data traversals at the path level. The additional computational steps for internal and external cluster traffic (see at pp. 119-123) would allow more complex evaluations for each cluster in the global system-level context.
- The traffic monitoring supports flexible spatial and temporal run-time configurability. Furthermore, the data is processed by a software-module, which can be migrated and whose evaluation policy/functionality can be changed during run-time. An independent QMesh network (SNoC) is responsible for the monitoring and control data transfers.

The corresponding path adaptation flow is introduced and discussed in the next Section 5.1. Afterwards, Section 5.2 presents the results of the experimental evaluation considering the software module performance, synthetic traffic pattern, and real application workloads. Finally, Section 5.3 provides a summarizing conclusion and a partial outlook for future investigations regarding the proposed traffic management.

5.1 PATH ADAPTATION FLOW

Prior to the description of the path adaptation flow, it has to be mentioned that this solution is intended to work upon the clustering of the traffic monitoring and its software module operates at the same master-tile. When a monitoring cycle is finished, the path adaptation can start immediately after the utilization data is written into the monitoring memory. Thus, no further preprocessing is required. Furthermore, the communicational part of the path evaluation contains a simple data transfer of the updates path tables as packets over the existing SNoC. Hence, the following sub-sections are concentrated on the path comparison metric and the path evaluation inside the software module.

5.1.1 PATH COMPARISON MEASURE

The initial step towards the intended path adaptation is the formulation of a metric/measure to compare the two available path options for each source-destination pair inside the clusters of the QMesh. Thereby, major differences to the proposals for 2D-Mesh-based NoCs exist due to the spatially fully independent paths inside the QMesh and the full ETE evaluation of these paths. In example, the ATDOR solution of *Manevich et. al.* [60], [125] for the 2D-Mesh NoC with xy/yx-routing compares the most congested router-to-router links along each path option between the source and the destination based on the mean port input buffer utilization. This is an indirect way to compare the maximum available bandwidth along these paths. But the independent injection/ejection terminals for each path options in the QMesh require the consideration of additional load information at these path elements for a full path comparison. The applied traffic monitoring provides these data at the full component coverage mode (COVERAGE='11' see Table 4.4 at p. 120). Normally, the congestion states at the NI of the source inside the 2D-Mesh with integrated VCs should be considered as well, because each VC has an own transmission buffer at the tile and the mean utilization of this buffer determines the potential waiting time before a packet can be injected into the network. Furthermore, each link along a path is a potential source of congestions and the monitored utilizations by the TRM in the work at hand can be interpreted as probability that these links will be blocked by other traffic flows inside the NoC.

Accordingly, the required metric must consider the traffic situation along the full ETE path inside the QMesh clusters. The chart of the mean packet header delay in Figure 2.8 (see p. 39) and discussed in the context of Equation 2.16 (see p. 40) underscores this. It becomes obvious that the head-of-line blocking at the transmission buffers due to backpressure from the network has a significant impact on the packet delay at higher traffic intensities. Hence, the metric should cover the pre-NI decision of the right buffer injection in addition to the post-NI network path delay as well as well. Furthermore,

the performance profiling of the traffic monitoring software module regarding the path evaluation test procedure P7 in Section 4.4.2 (see pp. 130-133) revealed the constraint to avoid computational steps that include division or multiplication (like required for the computation of mean values).

As results, the path evaluation in the work at hand uses a simple summation to $\boldsymbol{SUM_{A/B}}$ as described by the Equation set 5.1 below. The metric sums up all monitored utilization values beginning at the correct tile output quadrant at the source SRC via $\boldsymbol{LOAD_{IN-A/B}}$, proceeds over all router-to-router link utilizations (thus DIR={UP,RIGHT,DOWN,LEFT}) along the xy-paths $\boldsymbol{LOAD_{XY-A/B}}$, and finally adds up the utilization of the router-to-NI port at the destination through $\boldsymbol{LOAD_{OUT-A/B}}$. Therefore, the $TLOAD[][]$ and the $LLOAD[][]$ maps from the traffic monitoring are required. For the comparison of the two path options (A/B), this metric has to be calculated for both ones under consideration of the correct input/output terminals at the source/destination. Figure 5.1 depicts an exemplary case for such a dual path situation inside a 4x4 cluster of the QMesh and the utilization compositions for each considered component along the path B in form as a bar-chart.

$$\boldsymbol{LOAD_{IN-A/B}} = TLOAD[SRC]\left[Q_{IN-A/B}\right]$$

$$\boldsymbol{LOAD_{XY-A/B}} = \sum_{x=x_{SRC-A/B}}^{x_{DST-A/B}\pm 1} LLOAD\left[P_{x-A/B}\right][DIR] + \sum_{y=y_{SRC-A/B}}^{y_{DST-A/B}\pm 1} LLOAD\left[P_{y-A/B}\right][DIR]$$

$$\boldsymbol{LOAD_{OUT-A/B}} = +LLOAD\left[DST_{A/B}\right]\left[Q_{OUT-A/B}\right]$$

$$\boldsymbol{SUM_{A/B}} = \boldsymbol{LOAD_{IN-A/B}} + \boldsymbol{LOAD_{XY-A/B}} + \boldsymbol{LOAD_{OUT-A/B}} \quad (5.1)$$

The illustrated path situation reveals a QMesh-specific benefit of the applied summation over the calculation of mean values.

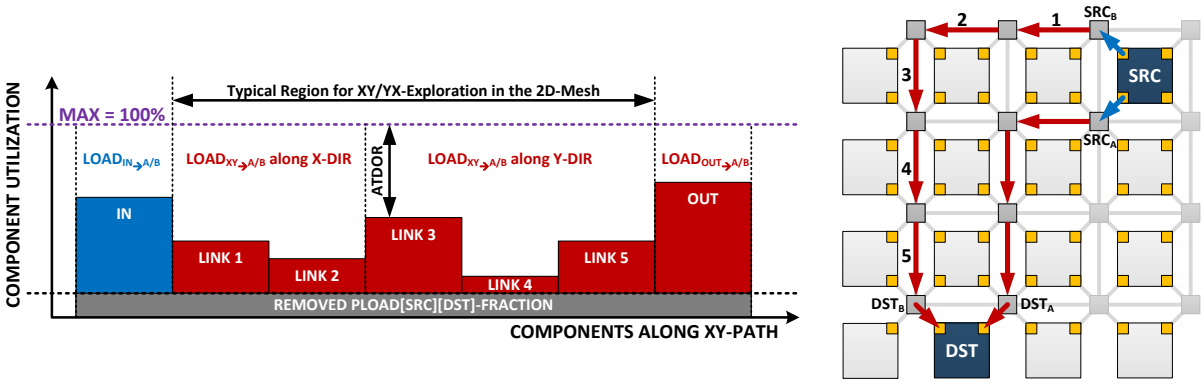


Figure 5.1: Exemplary illustration of the dual path situation and metric content of path B

If the destination is located inside one of the four QMesh quadrants, the two path options have different hop distances that differ by 2 (see Table 3.3 at p. 73) and the mean value calculation would hide this detail. This would further require the introduction of an additional threshold for path A to

make it comparable to the longer path B. But the path comparison based on the sums of component utilizations implicitly favors the shorter path option A and path B is only chosen if it could save more load than resulting from the two additional hops. Hence, this threshold is already integrated and can save undesired path updates. The proposed path evaluation metric covers the complete ETE path and therefore should serve as good measure to perform global comparisons inside each cluster at lowered computational efforts for the software module. One critical aspect could be the consideration of the tile output utilization via raw flit injections, because a congested output would be monitored with low utilization rates as well and therefore could provide misleading data inputs. But due to the monitoring over 10^3 up to 10^5 clock cycles such effects are expected to fade out. Alternatively, a workaround through the analysis of the link utilizations at the relevant routers could be performed or the tile output could be simply removed from the metric calculation.

5.1.2 PATH UPDATE EVALUATION

At the end of each monitoring cycle, after the data takeover into the monitoring memory from the AU at the master-tile of a cluster, the software module is activated and performs the path update evaluation for the complete cluster. The required computation for each path consists of five basic steps as illustrated by the flow in Figure 5.2 with all major details. The processing order of the cluster tiles is considered as static, beginning at the LLC and proceeding row-by-row up to the URC. For each source tile along this order (SRC), the software module evaluates each path to cluster-internal destinations (DST) with the same scheme. This static fashion is applied to achieve a convergence in the traffic optimization over several monitoring cycles. The path evaluation operates on copies of the tile output and link utilization maps ($TLOAD^*[][]$ and $LLOAD^*[][]$), because modifications with the monitored path utilizations of $PLOAD[][]$ are performed during the update evaluations. Furthermore, the considered path table entries in $PTABLE_{SRC}[]$ do only represent the relevant subset of tiles inside the specified cluster. The update evaluation is performed for valid paths, which requires at least the availability of a second path option (in the QMesh some path from and to tiles at the left and lower edge do not meet this condition (see Section 3.2.2 pp. 70-73) and a monitored path utilization that is greater than a defined threshold ($PLOAD[SRC][DST] > 0$ here). The integrated functionality of the five path evaluation steps is as follows:

1. Prepare the $TLOAD^*[][]$ and $LLOAD^*[][]$ maps along the selected xy-path $SP_{OLD} = A\ or\ B$ by removing the monitored utilization resulting from the raw flit transfers given in $PLOAD[][]$ for the current evaluated source-destination pair. This is necessary for a fair comparison of both path options. The required computational steps can be merged with the calculation of the comparison metric and contains simple subtraction instructions.

2. Calculate the comparison metric $SUM_{A/B}$ for both path options according to Equation set 5.1 from the prepared monitoring data in the $TLOAD^*[][]$ and $LLOAD^*[][]$ maps. The required computational steps contain simple addition instructions, while parsing through the corresponding maps along the path coordinates defined by the xy-paths ($P_{x/y-A/B}$).

3. Compare the resulting values for $SUM_{A/B}$ and select the new path option $SP_{NEW} = A\ or\ B$ that offers the minimal utilization sum $MIN(SUM_A; SUM_B)$. Furthermore, if this chosen new path differs from the old one $SP_{NEW} != SP_{OLD}$ then register this change by incrementing the number of occurred updates $\#UPDATES + 1$ for the path evaluations of the current source.

4. After the new path is chosen, the $TLOAD^*[][]$ and $LLOAD^*[][]$ maps have to be updated along the selected xy-path $SP_{NEW} = A\ or\ B$ by adding the monitored utilization resulting from raw flit transfers given in $PLOAD[][]$ for the current evaluated source-destination pair. This ensures that following path evaluations operate on the adapted traffic situation that considers the prior performed path adjustments. Furthermore, the path table entry of the corresponding source-destination $PTABLE_{SRC}[DST]$ pair has to be updated to the new path.

5. After all path options of the current source SRC are evaluated along all destinations DST inside the cluster region ($DST = N_{CS} - 1$ is reached), check if the path setup for this source has changed ($\#UPDATES > 0$). If the paths were adapted, transmit the updated path table $PTABLE_{SRC}[]$ to this source as packet over the SNoC. Afterwards, proceed with the next source if available.

At the end of this path update evaluation, other service operations can be performed on the monitoring data, such as light-weighted monitoring evaluation, further traffic classification or parameter calculations as feedback for the application mapping. In general, the integrated computation of the path update evaluation is implementation-friendly, because it uses simple addition as well as comparison instructions and operates on integer-based data values. Thus, the requirements for the CPUs at the master-tiles are basic. For more powerful processors, a multi-threading version might be favorable to utilize the full capabilities of a core and speed-up computations.

The path filter in the proposed flow is basic and excludes source-destination pairs that are not valid at all for path updates. But in general, more sophisticated path selection strategies for dedicated traffic scenarios and path prioritizations (re-ordering of the cluster walkthrough) are realizable. Furthermore, such changes are promising as well if the total computational efforts during a monitoring cycle would exceed the deadline constraints and the increase of the monitoring cycle

would be prohibited due to coverage of workload dynamics or hardware limits (v_{thresh} = max). Exemplary modifications of the path selections would be:

- Streching the path update evaluation over multiple monitoring cycles.
- A path selection based on the restriction to a hop region.
- Increasing the path threshold to handle only the most dominant traffic flows

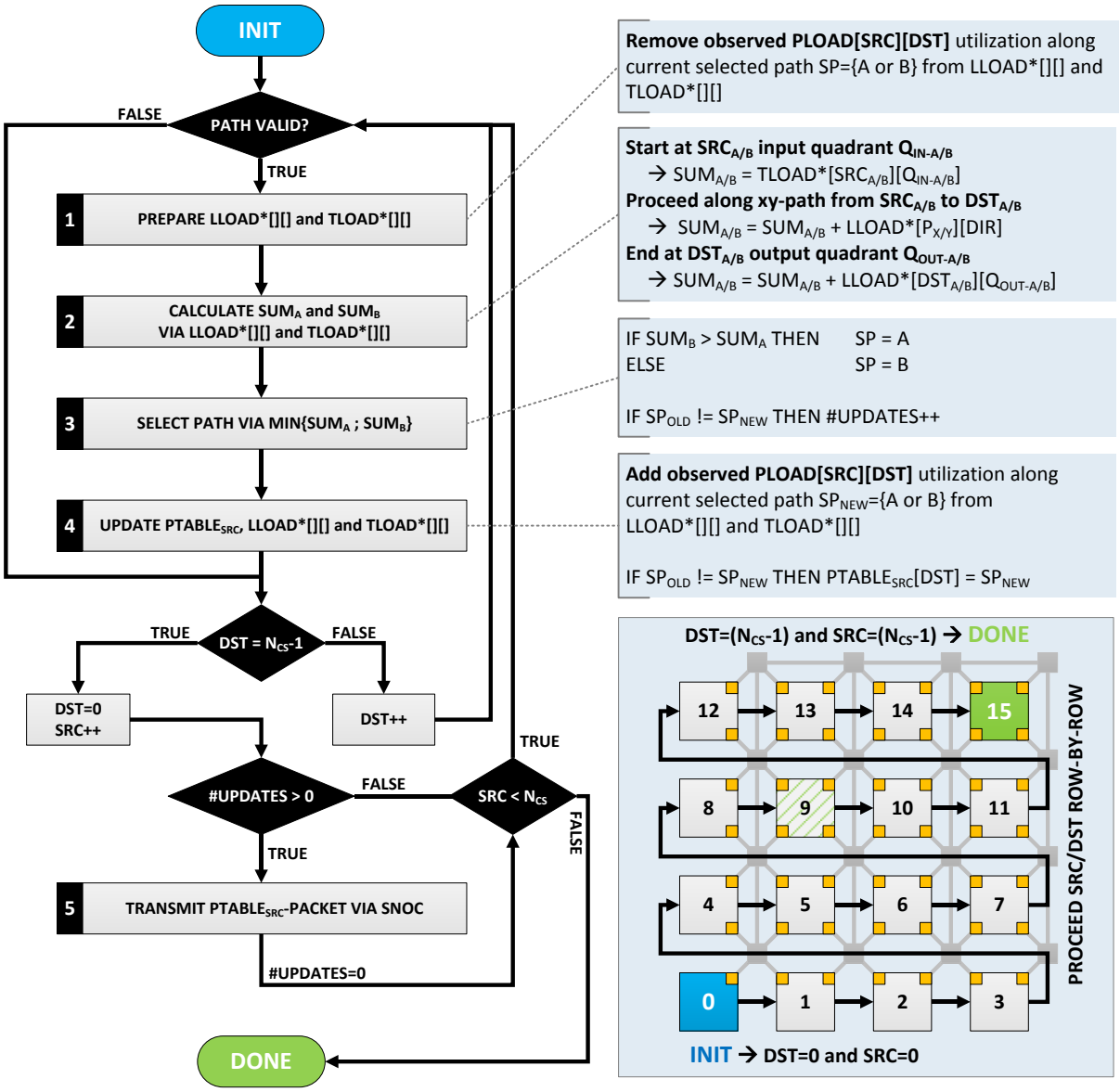


Figure 5.2: Path adaptation flow and functionality in the software module on the master-tile of each cluster

In summary, there are many potential variations of the path update strategy like and beyond these simple proposals, but therefore a dedicated classification of the current traffic situations should be performed prior to this. Another option would be the provisioning of hints or recommendations for a specific selection from the application mapping. The work at hand concentrates on the general evaluation of the proposed traffic management.

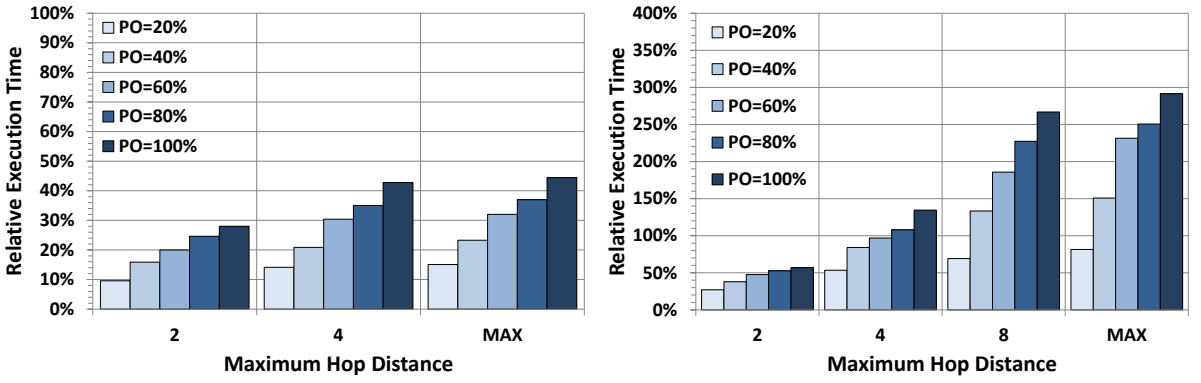
5.2 EXPERIMENTAL RESULTS

This part provides the experimental evaluation of the prior proposed software-based spatially-restricted traffic management. Therefore, the first Section 5.2.1 summarizes the profiling results of the path evaluation as introduced in Section 5.1. The next Section 5.2.2 provides the results of the experimental evaluation. Therein, the results from the SystemC-based simulations of synthetic traffic pattern setups are summarized and discussed regarding the major aspects. Finally, the Section 5.2.3 contains the results of the real application workloads.

5.2.1 SOFTWARE MODULE PERFORMANCE

The performance profiling of the software module for the path update evaluation is realized via Sniper on a single processor core of the 64-tile CMP specified in Appendix (see setup table at p.165). Similar to the TRM profiling in Section 4.4.2 (see pp. 130-133), the path update evaluation is simulated for the 4x4 and the 8x8 cluster scenarios as corner cases. The TRM timing is set to $v_{thresh} = 256$ in the 4x4 cluster and to $v_{thresh} = 1024$ in the 8x8 cluster. Furthermore, full component coverage is required (COVERAGE='11') and the configured scale resolution is $k_S = 1$. For the consideration of different traffic scenarios, the path occupation and the maximum hop distance are varied. Thereby, the path occupation (PO) is varied over 20%, 40%, 60%, 80% and 100%. The maximum hop distance is set to 2, 4 and unconstrained (MAX) in the 4x4 cluster scenario, while the 8x8 cluster is further evaluated for the additional maximum distance of 8 hops. The timing results represent the mean over 1000 path evaluation runs. They are provided by Sniper through cycle-approximate instruction level instrumentation for defined regions-of-interest inside the source-code. Hence, the preliminary data generation and setup is excluded and the pure procedure runs are profiled.

The bar-charts in Figure 5.3 (a) and (b) contain the relative execution times as fraction of the processing period defined by the monitoring cycle for each parameter configuration (LIMIT=25.6 µs and 102.4 µs). Similar to the profiling results of the TRM test procedures P1 and P7 in Section 4.4.2, the path update evaluation inside the 4x4 cluster does not violate the deadline and less than 45% of the available CPU time is utilized in the worst case of all parameter variations. In the 8x8 cluster, the path update evaluation would miss its deadlines in several parameter settings with increased PO above 20% at allowed path lengths with more than 4 hops. Furthermore, a PO of 100% at maximum path lengths would be critical. But the prior Section 5.1.2 already introduced several options for a workaround and the expectable traffic scenarios were discussed in Section 4.4.2 (see pp. 130-133). If the sensor period is increased to $v_{thresh} = 4096$, the processing for all parameter variations would meet the required deadlines.



(a) Relative → 4x4 cluster – v_{thresh} = 256 (k_S=1) (b) Relative → 8x8 cluster – v_{thresh} = 1024 (k_S=1)

Figure 5.3: Measured path adaptation execution times as absolute values and relative to the deadlines defined by the adjusted monitoring cycle for the 4x4 and the 8x8 cluster case

Another interesting aspect is the impact of the selected path evaluation metric on the computational efforts, if compared to the profiling results of test procedure P7 for the 2D-Mesh in Figure 4.15 (see p. 132). P7 calculates the mean link utilizations along each xy-path inside the cluster and therefore applies division operations. The summation based evaluation of two paths per source-destination pair inside the QMesh performs faster. In addition to the avoidance of the division for the calculation of the mean value, the QMesh comes along with reduced hop distance for the most path options if compared to the 2D-Mesh and therewith reduces the number off required additions per metric calculation for each path.

5.2.2 SYNTHETIC WORKLOADS

This section provides the result of the SystemC-based simulations of the NoCs under synthetic BP and CDF traffic patterns. The managed QMesh operates with the cluster sizes of 4x4 and 8x8 as follows:

1. The traffic pattern and TRM are restricted to all source-destination pairs inside a 4x4 cluster (N_{CS}=16). The path update evaluation covers the full traffic situation.
2. The traffic pattern and TRM are restricted to all source-destination pairs inside an 8x8 cluster (N_{CS}=64). The path update evaluation covers the full traffic situation.

These clustering scenarios are illustrated in Figure 4.9 (see p. 125) for the 2D-Mesh and equally applied in the QMesh. The TRM timing is set to $v_{thresh} = 256$ in the 4x4 cluster and to $v_{thresh} = 1024$ in the 8x8 cluster. Furthermore, full component coverage is required (COVERAGE='11') and the configured scale resolution is $k_S = 1$. The complete path update evaluation is integrated into the SystemC-based simulator (see Appendix at p. 166), computational delays are considered and realistic monitoring as well as path table update packet transfers are handled over the SNoC. The NoC setups are the same as used for the experiments in Section 4.4 for the TRM (see Table 4.6 at p. 124). The default configuration of the path tables in the basic QMesh uses the path option A for all Q0 to Q3

destinations. The paths to UP/RIGHT/DOWN/LEFT destinations are initiated with an alternating strategy that depends on the hop distance. Even hop distances get the assignment of path option A, while odd hop distances utilize path option B (if constraints are met, otherwise A). The managed QMesh starts with the same path table setup, which represents the optimized minimum distance version of the path table configurations to exploit the maximum static benefits of the QMesh by default. The setups are the same as used in Section 3.3.1, but with all TRM extensions.

The synthetic BP traffic patterns represent typical cases of workloads with few dominating paths. Each source has only one fixed destination assigned by the specific selection rule of the applied BP pattern (see Appendix at p. 161). The achievable improvements due to the proposed path update evaluation are significant and the managed QMesh outperforms the basic QMesh in all simulated patterns and clustering scenarios. The individual differences between the simulated BP patterns were discussed in Section 3.3.1 (see pp. 74-78). For a more detailed discussion regarding the effect of the traffic management and the reason for the resulting order of improvements along the evaluated BP patterns, the mean hop distance and the number of paths with required link traversal can be used as additional indicators. The bar-charts in Figure 3.8 (a) and (b) depict these features for all patterns in both cluster sizes (see p. 75). The main effect of the QMesh on the mean hop distance is a reduction by 1 up to 2 hops. The concrete value depends on the locational relations between the source destination pairs. The bit complement pattern has the highest value and the shuffle pattern the lowest. The mean hop distance for the transpose and the bit reverse pattern are equal, but the symmetry inside the transpose pattern makes it more disturbing. The other effect of the QMesh is that paths between tiles with NI terminal to a joint router do not traverse any router-to-router link and this minimizes the concurrency as well as the number of paths which are most likely affected by the path update evaluation. The number of remaining link traversing paths is depicted in Figure 3.8 (b) and explains why the 4x4 reaches such high packet injection rates per tile. Therein, up to 57% of the 2D-Mesh paths become one hop router traversal and “disappear” (shuffle pattern). The remaining paths can be successful rebalanced along the different options by the path update evaluation. In the 8x8 cluster, the resulting reductions are far lower and the number of total paths increases for all pattern.

Contrary to the BP-based synthetic traffic patterns, the CDF patterns result in traffic situations that describe mixed workloads with all source-destination constellations and will not be dominated by single connections. This makes it more complex for the path update evaluation to find optimized solutions as long as there are no dominant flows at reduced disturbance. Many paths with small fractions of traffic are hard to handle and the inject/eject traffic is balanced along all of the four available NIs by default. In those cases, the managed QMesh does not find better solutions than

provided by the basic QMesh with its initial shortest path configuration. Thereby, the nearest-neighbor pattern suffers from its remaining random uniform fraction that will not be transmitted/received to/from the proximity tiles if it comes to rebalancing of traffic loads. The improvements of the managed QMesh over the basic QMesh are negligible at 20% and only moderate near to the saturation region at 80%. Thereby, an applied hop restriction to 3 as well as full path evaluation at longer monitoring cycles ($v_{thresh} = 4096$) showed no major differences. The hotspot pattern results in the managed QMesh are similar to those of the nearest-neighbor pattern at 20% due to the issues regarding the traffic rebalancing. The general problem in here is the traffic dominance of few tiles with no major options to circumvent their overloaded NIs. Contrary to the nearest-neighbor and the hotspot pattern, the application of the managed QMesh to the rentian traffic offers improvements even at path occupations of 100%. For the 8x8 cluster case the path update evaluation has been restricted to the region of 5 hops. This range covers nearly all of the traffic that is observable by the traffic monitoring for the simulated rentian exponents of 0.3 and 0.7. The rentian traffic pattern does not contain a random uniform fraction, which is also one reason for the smaller spreads of the achievable packet injection rates per tile between the 4x4 and the 8x8 cluster case. A scale-up with random uniform fraction would result in increased concurrency due to the higher mean hop distance for each path. The improvement of the managed QMesh over the basic QMesh increases with the degree of traffic locality. The path adaptivity provides higher saturation level as well as lower packet delays than both reference cases. But as the 8x8 cluster results on the rentian pattern show, a lowered locality (R=0.7) in the greater cluster size does shrink the potential improvements due to the managed QMesh as well. Only if high locality can be maintained, the path updates will have a positive effect.

The following bar-charts contain a summarized picture on the impact of the managed QMesh on the NoCs run-time costs as well as performance in comparison to the standard 2D-Mesh at the evaluated 4x4 and 8x8 clusters with synthetic traffic pattern scenarios. Furthermore, the results of the basic QMesh are integrated to provide a comparison at which cases the path updates are necessary or not. The data evaluation procedures to calculate the relative improvements of the managed QMesh over the 2D-Mesh regarding network saturation, power dissipation and packet delay are the equal to those already applied in Section 3.3.1 (see pp. 74-78 and Appendix at pp. 175-182). Regarding the evaluation of the power dissipation for the managed QMesh statistics, it has to be mentioned that the corresponding results of the TRM from Section 4.4.1 (see pp. 124-130) has been integrated to cover the full overhead of the QMesh based SNoC, the TSAs and the AU.

The bar-chart in Figure 5.4 contains the relative saturation improvement ($\Delta_{saturation}$) of both QMesh variants over the 2D-Mesh for all BP and CDF patterns. Thereby, it becomes obvious that the BP

patterns at both cluster sizes profit most regarding the NoC saturation. In the 4x4 cluster the saturation gains over all simulated BP patterns range from 286% to 436% and the 8x8 cluster achieves 306% to 503% with applied traffic management. The CDF patterns in Figure 5.4 show a more moderate growth of the NoC saturation limits. Only if the communication locality is high enough and the uniform fraction is low, the managed QMesh provide non-negligible gains over the basic QMesh. In summary, the mean saturation gain of the managed QMesh over the reference 2D-Mesh is 170% in the simulated 4x4 scenarios and 121% in the 8x8 cases.

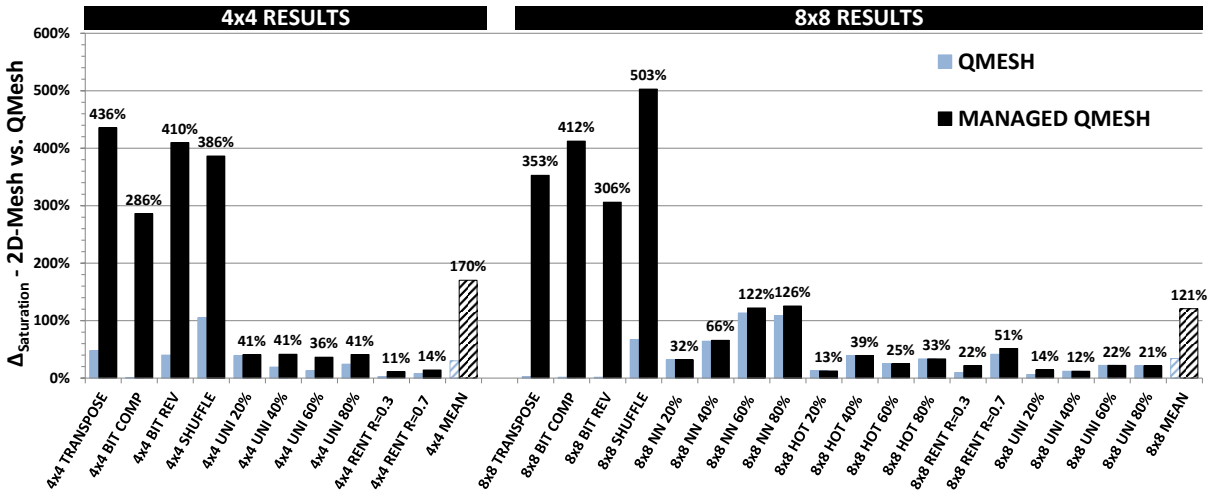


Figure 5.4: Comparison of network saturation for simulated 2D-Mesh and QMesh with as well as without applied path adaptation under synthetic traffic pattern

The bar-chart in Figure 5.5 contrast the relative impact of both QMesh variants on the mean total power dissipation and the mean packet header delay in comparison to the 2D-Mesh (Δ_{Delay} and Δ_{Power}). Similar to the saturation improvements, the plots in Figure 5.5 shows that the managed QMesh provides better results for the BP patterns. Therein, the power overhead of the TRM can be mitigated due to better traffic balancing along more routers and therefore the activity-driven temperature increase is more moderate. Furthermore, the path adaptivity achieves lower packet delay levels. The CDF pattern plots in Figure 5.5 show, that the power overhead is higher than for the BP patterns. In parallel, the packet delay is reduced in a similar range compared to the results obtained with the basic QMesh. Only at CDF patterns with high communication locality (NN 60%, NN 80% and RENT) the packet delay reductions outweigh over the power increase inside the comparable regions to the 2D-Mesh. Overall, the managed QMesh introduced a mean power overhead of 60% and a mean packet delay reduction of 59% for the simulated 4x4 scenarios. For the 8x8 cases, the mean power overhead rises to 71%, while the mean packet delay is reduced by 53%. Thereby, the progress of Δ_{Delay} and Δ_{Power} from low to high packet injection rates in the comparable region must be considered (see Appendix at pp. 179-182). Thus, the higher the packet injection rates per tile become, the better the power increase to packet delay reduction ratio becomes. Furthermore, these

values are restricted to those traffic ranges where the 2D-Mesh is able to operate at, while both QMesh variations provide capacities beyond.

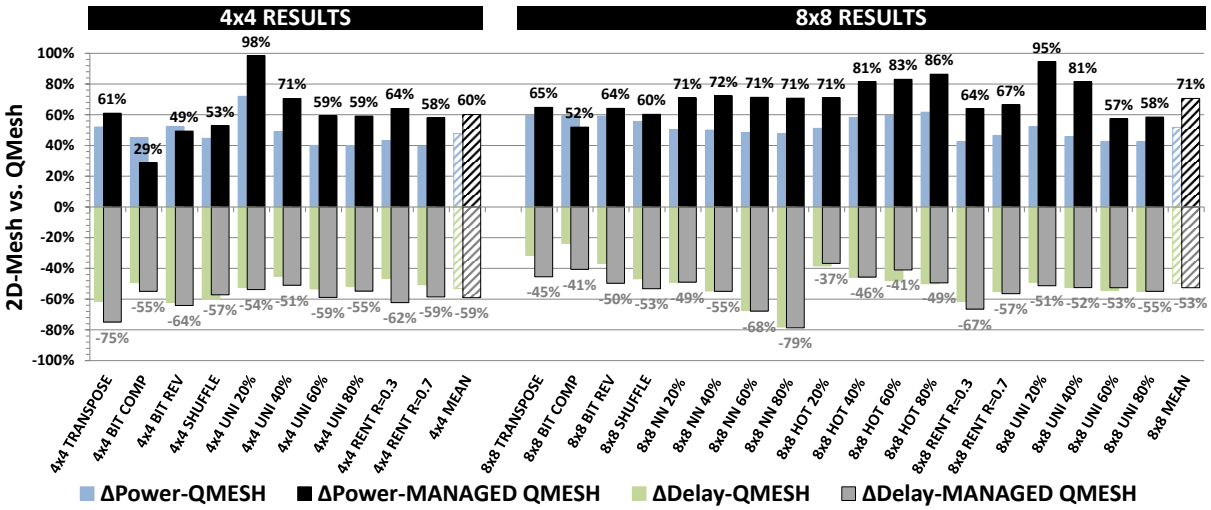
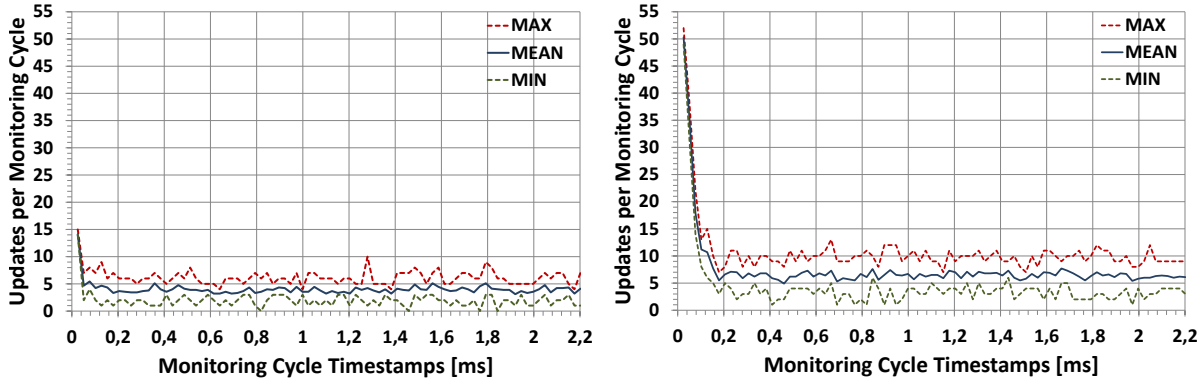


Figure 5.5: Comparison of network power/delay for simulated 2D-Mesh and QMesh with as well as without applied path adaptation under synthetic traffic pattern

For the evaluation of the path update characteristics, the 4x4 and 8x8 cluster scenarios are simulated with the random uniform traffic pattern (UNI) as worst-case regarding path diversity and mean hop distance. Thereby, the path occupation (PO) is varied in the range of 20%, 40%, 60%, and 80% to study the impact on the resulting optimization capabilities for the managed QMesh. A change of the path occupation results in the increase/decrease of the source-destination pairs that inject and receive traffic. Furthermore, the utilization per path changes as well, because the same amount of injected traffic as done for (N_{CS}-1) different destinations and sources in the PO=100% setup is now carried by a smaller number of paths (80%, 60%. 40%, 20%) with fewer participating tiles. In addition to the normal traffic statistics, the total number of applied path updates inside the complete cluster for each expired monitoring cycle is logged. The path update results presented below are obtained for the simulation runs at packet injection rates in the high traffic region before the managed QMesh starts to saturate (0.017 at P=20% up to 0.022 at PO=80%). This shows the overall effort and gives insight into the convergence behavior of the proposed path update evaluation flow/computation.

The corresponding path update statistics for two of the 4x4 cases are shown in the line-charts of Figure 5.6 (a) and (b). Therein, the minimum, mean, and maximum number of total path updates per cluster is plotted over the progress of monitoring cycles. The results are obtained from 10 simulation runs for each parameter variation. In the 4x4 cluster, the maximum amount of touched paths is $N_{CS}\cdot(N_{CS}-1)=240$ as worst case. As Figure 5.6 (a) to (b) show, the expectable number is far below that limit. In the PO=80% the maximum observed number of path updates per monitoring cycle along the cluster is 53 and thus only around 22% of all available paths are affected by adjustments. Additionally, the curves indicate a consistent convergence behavior of the path update evaluations.

The simulation variation of the PO from 20% to 80% further shows a steady increase of the start level from 15 maximum path updates at 20% to 53 maximum path updates at 80%, but the following curve progress remains similar. After the first monitoring cycle, the maximum amount of applied path updates is reached and followed by a steep decline to a low number of toggling paths. Moreover, this allows the conclusion that major improvements can be achieved by the applied updates during the first 10 monitoring cycle (4x4 → 10·25.6 µs = 0.256 ms).



(a) Uniform and PO=20% (b) Uniform and PO=80%

Figure 5.6: Path update statistics (MIN/MEAN/MAX) for the uniform CDF pattern in a 4x4 cluster inside the QMesh with applied path adaptation at path occupations of 20% and 80%

Contrary to the 4x4 cluster setup, the only case of saturation improvements by the managed QMesh at the 8x8 clustering is detectable at PO=20%. Therein, the path diversity and traffic fraction per path is already small enough in comparison to the 4x4 setup. The remaining cases above this limit behave similar to the results already seen for the 8x8 clusters under nearest-neighbor traffic. Especially, the collected path update statistics indicate the problems of the path update evaluation to rebalance the traffic loads along the dual path options. The number of applied path updates is very low and in the same range as obtained for the 4x4 clustering. At POs of 40% to 80% the number of applied path updates remains zero until the managed QMesh starts to saturate.

In summary, the simulation results for the synthetic traffic patterns provide an adequate view on the capabilities of the proposed traffic management solution in the work at hand. Thereby, the experimental evaluation offers two general statements:

- Single application workloads are represented by the BP patterns and experience the full performance benefits of the managed QMesh at all evaluated cluster sizes. Thereby, the discussion of the results revealed that the absolute values for the achievable packet injection rates as well as packet delays depend on the mean hop distance, on the communication locality and on the number of remaining paths with link traversal along the QMesh.
- Multi-application workloads are represented by the CDF patterns and provide a mixed picture regarding the achievable performance gains through applied traffic management in

the QMesh. Small clusters provide better relative as well as absolute results. Thus, if such situations are inevitable smaller clusters should be selected by default. Furthermore, the results reveal the importance of communication locality as well as the presence of sufficient spread of traffic intensities along the paths.

The common aspect of communication locality along both workload scenarios indicates the necessity of the coordinated interplay between application mapping and traffic management, which is an argument for the centralized software-based approach itself regarding its flexibility and integration capabilities.

Finally, an exemplary overview for the reported results from different publications [60]–[62], [125], [163], [168]–[171] on adaptive traffic management in mesh-based NoCs is provided to rate the results of the proposed solution in the work at hand in a wider context.

Table 5.1: Overview on reported saturation gains in diverse publications on applied path adaptation in mesh-based NoC with a size of 4x4 or 8x8

REFERENCE PROPOSALS		TRAFFIC PATTERN - $\Delta_{Saturation}$ in [%]				
WORK	MECHANISM	UNI	TRANSPOSE	SHUFFLE	BIT COMP	BIT REV
[60], [125] 8x8 NoC	LOCAL ADAPTIVE	-13.2	-22.7	Not covered	Not covered	Not covered
	O1TURN	+3	+9	Not covered	Not covered	Not covered
	RCA	-1.5	+5.7	Not covered	Not covered	Not covered
	ATDOR	+7.3	+18.2	Not covered	Not covered	Not covered
[61] 8x8 NoC	LOCAL ADAPTIVE	-7	+128	Not covered	-21.7	Not covered
	RCA	0	+157	Not covered	-4.3	Not covered
[62] 8x8 NoC	LOCAL ADAPTIVE	Not covered	+126	+50	-26.9	+120
	NoP	Not covered	+120	+58	-30.7	+113
	RCA	Not covered	+140	+67.7	-7.7	+133
	DBAR	Not covered	+153	+75	-11.5	+153
[168]–[171] 4x4 NoC	2D-MESH ADAPTIVE	-14.9	Not covered	Not covered	-4.5	-3.6
	CMESH DOR	-45.5	Not covered	Not covered	-40.9	+5.5
	NR-MESH DOR	+29.8	Not covered	Not covered	+120	+58.2
	NR-MESH ADAPTIVE	+10.4	Not covered	Not covered	+70.5	+34.5

The selected publications were already discussed as part of the related work in Section 2.1.2 (see pp. 40-44), Section 2.2.3 (see at p. 49) and Section 3.1 (see pp. 63-65). Furthermore, the different solutions are partially compared to each other in independent works. All simulated NoCs presented in these publications have a similar input port buffer depth to packet size ratio (see Section 2.1.1 at pp. 31-40 and Figure I.2 at p. 167), a standard router pipeline with minor modifications and covered partially redundant traffic patterns as used in the work at hand. For the comparison, the relative saturation improvement represents an effective measure, because it fades out the differences between absolute values, which results from modifications at the main operational NoC parameters. The extracted results are depicted in Table 5.1. Additionally, Table 5.2 contains the number of virtual channels the simulated NoCs were configured with.

In comparison of the results in Figure 5.4, it can be stated that the managed QMesh would outperform nearly all referenced solutions by a factor of 2 or more. Thereby, the QMesh operates without any virtual channels. Furthermore, a review of the works of *Kim* in [119] on the FBFly topology and *Grot* in [153] on the Express Cube topologies shows, that the managed QMesh is able to compete regarding the achievable delay improvements over the 2D-Mesh case. In [119], *Kim* reports 20% up to 30% mean packet delay improvement over a mix of BP and CDF pattern. In [153], *Grot* obtains up to 50% lower mean packet delay for different BP and CDF pattern.

Table 5.2: Reported utilization of virtual channels in the presented result overview on applied path adaptation of diverse publications in mesh-based NoC with a size of 4x4 or 8x8

N_V	ATDOR	O1TURN	RCA	LOCAL ADAPTIVE	NoP	DBAR	NR-MESH	CMESH
[60], [125]	2	2	8	1 [140]				
[61]			8	8				
[62]			8	8	8	8		
[168]–[171]				2			2	1

5.2.3 APPLICATION WORKLOADS

This last part of the experimental evaluation contains the results of the full CMP simulations as outlined in Appendix (see setup table at p. 165). The Sniper-based simulation flow and the issues/implications of the presented results are the same as already discussed in Section 3.3.3 (see pp. 81-83). For the experimental evaluation of the applied path adaptation with Sniper, the simulator is extended with the required TRM mechanism. Unfortunately, the utilized Sniper simulator does not provide multi-program workloads as well as the required program-to-simulator interfaces for the implementation of the TRM and path update evaluations as fully independent software module in co-existence with the application workload. Hence, the required functionality must be integrated as part of the simulator itself.

The simulated 64-tile CMPs apply an 8x8 TRM cluster that includes all of their tiles and works with the minimum allowed sensor period of v_{thresh} = 1024 at a scale resolution of $k_S = 1$. Thereby, the packet delay of the 2D-Mesh, the basic QMesh as well as the managed QMesh is measured over 10 simulation runs for each application. Afterwards, the mean packet header delay is calculated from these statistics and normalized to the results of the 2D-Mesh. The bar-chart in Figure 5.7 depicts these normalized values for the selected benchmark applications. The QMesh results are equal to those discussed in Section 3.3.3 and depicted in Figure 3.14 (see p. 82). The results show that despite the low traffic injection rates of the simulated workloads (see discussion on that issue in Section 3.3.3 at pp. 81-83), the managed QMesh provides packet delay reductions of 17% in minimum (blackscholes) up to 47% in maximum (fmm) over the 2D-Mesh. While the basic QMesh comes along with 9% delay improvement on average, the managed QMesh achieves 24% and outperforms the

basic QMesh by 15%. These results are promising and future investigations should focus on the integration of the proposed traffic management into a more suitable simulator as well as the evaluation of benchmark test cases with higher traffic rates.

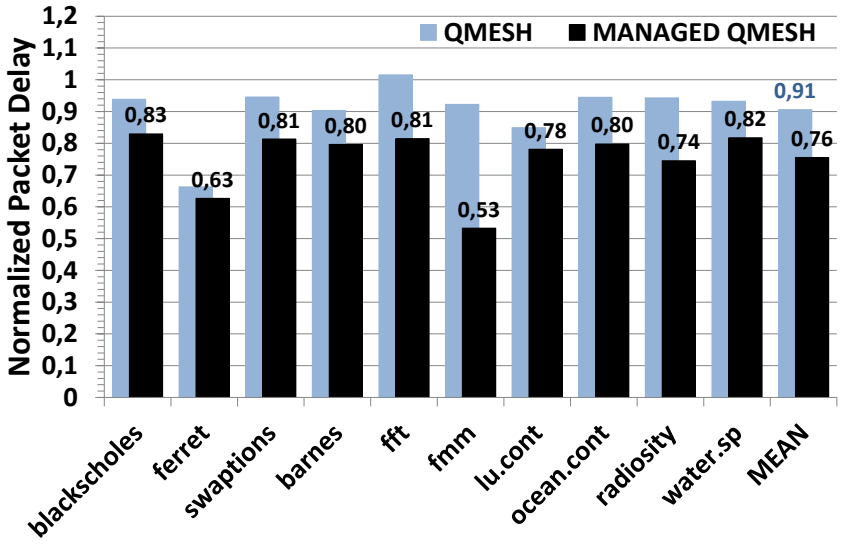


Figure 5.7: Normalized packet header delay for selected benchmark applications in an 8x8 QMesh cluster with and without applied path adaptation

5.3 SUMMARY AND OUTLOOK

This chapter proposes a concrete use case of the combination of the QMesh and the spatio-temporal run-time TRM, which provides remarkable potential performance gains over the 2D-Mesh for a wide range of traffic patterns by improving the saturation up to factors of 5, reducing the mean packet delay by around 40% up to 80% and thereby imposing power overheads of 29% to 98%. For a better evaluation of the power consumption overheads in the CMP context, the assumptions of [43] with a power budget per computational core in the range of 0.5 to 1 Watt at 45nm CMOS will be used. For an 8x8 tile CMP this would result in a total power budget of 32 up to 64 Watt. The observed mean power consumption for the majority of the simulated NoC scenarios in this section ranged between 1.2 and 4.2 Watt, depending on the traffic pattern and injected load per tile. This 4.2 Watt in maximum for **the majority of scenarios would consume between 6.5 and 13.2 percent of the CMP's power budget.** At peak saturations in the managed QMesh up to 6.2 Watt are reached at the NN=80% pattern before the QMesh becomes saturated (at loads 126% higher than those of the 2D-Mesh). In such peak load scenarios, the NoC would consume between 10 and 20 percent of the CMP's power budget. Hence, regarding the improvement of the QMesh on saturation as well as delay, the power power overheads are suitable on a global CMP level.

It has been further shown, that software-directed and regionally-centralized traffic management is a valid option for modern CMPs despite the generalized context of the applied TRM. The

introduced path update evaluation flow and the corresponding path comparison metric proved as efficient regarding the achieved improvements as well as the computational efforts. **The presented simulation results for the synthetic traffic patterns allow the narrowing of application scenarios in that the application of the proposed run-time traffic is profitable.** The major corner cases of balanced and unbalanced traffic loads were evaluated. **Fine-grained and distributed communication makes the applied traffic management less effective, while scenarios with coarse-grained communication dominant over dense paths will be boosted maximally.**

Additionally, it becomes obvious that great cluster sizes and applied centralized path adaptation with knowledge of the global traffic situation inside the spanned region do not necessary result in the better choice. **The simulated cases with applied 4x4 clustering offered better absolute results and show that the support of 16 tile clustering with better workload mapping/consolidation into this smaller region is a suitable design alternative if it comes to run-time traffic management.** Furthermore, the power and area overhead of the TRM can be reduced noticeably. Table 5.3 provides the hardware synthesis results for this reduced configuration of N_{CSmax}=16 with QMesh parameterization and compares them to the results of the N_{CSmax}=64 configuration from Table 4.7 presented in Section 4.4.1 about the experimental TRM evaluation. Especially the costly TRM components at master-tiles can save around 60% up to 70% power overhead and around 86% of the additional area footprint can be removed by the maximum cluster size reduction. In addition to the cost aspect, the performance of the TRM is improved as well due to the smaller amounts of monitoring data and MDR packet sizes (see Section 4.3.4 at pp. 119-123).

Table 5.3: Power and area evaluation results of the proposed traffic monitoring per tile at N_{CSmax} of 16 tiles

COMPONENT	QMESH at N_{CSmax}=64 tiles			QMESH at N_{CSmax}=16 tiles			RELATIVE REDUCTION		
	P_{dyn}	P_{stat}	Area	P_{dyn}	P_{stat}	Area	P_{dyn}	P_{stat}	Area
DNoC – HT	1,45E-02	1,75E-02	6,33E-08	1,45E-02	1,75E-02	6,33E-08			
DNoC – LT	5,33E-03	1,75E-02		5,33E-03	1,75E-02				
SNoC	3,30E-04	2,65E-03	8,57E-09	3,30E-04	2,65E-03	8,57E-09			
TSA	9,93E-05	2,00E-04	1,08E-08	3,98E-05	7,94E-05	4,29E-09	-59,9	-60,3	-60,3
AU	1,47E-03	5,70E-03	3,22E-07	1,64E-04	5,91E-04	3,44E-08	-88,8	-89,6	-89,3
TRM-MT	**1,89E-03**	**8,55E-03**	**3,42E-07**	**5,34E-04**	**3,32E-03**	**4,72E-08**	**-71,8**	**-61,1**	**-86,2**
TRM-ST	**4,29E-04**	**2,85E-03**	**1,94E-08**	**3,69E-04**	**2,73E-03**	**1,29E-08**	**-13,9**	**-4,2**	**-33,7**
Units	W	W	m²	W	W	m²	%	%	%

The proposed focus of the future investigations should be on further improvements regarding the availability of additional path options, the additional coverage of fault-tolerance/wear-out mechanisms, the integration as full operating system service, and the combination with services for the run-time application mapping.

While the last two topics are already proposed for the TRM outlook, the investigations on the first and the second point should begin as follows:

- The number of path options can be enhanced by two different solutions. First, the application of virtual channels would enable the integration of xy/yx-routing and the number of source-destination paths would increase up to four. But this comes at the costs of an additional pipeline stage inside the router and the area/power costs would nearly double. Alternatively, the odd-even turn model could be applied to the xy/yx-routing without virtual channels (see Section 2.2.3 at pp. 49-55).
- The preferred scope of fault-tolerance improvements should be the ETE path level for fast reactions, because a centralized approach might be too slow. But the knowledge of the NoC vulnerabilities has to be integrated into the path update evaluations. Therefore, measured ETE failure rates seem a promising input parameter. As wear-out indicator, the mean traffic utilizations can be used and provided by the TRM.

6 CONCLUSION

Since the introduction of CMPs nearly a decade ago, the number of integrated cores/tiles has been steady growing and workload applications have been adapted to exploit the increasing degrees of core-level parallelism. This changed the importance of efficient on-chip communication significantly and the corresponding infrastructure has to keep step with these new requirements of massively parallel transactions. As the parallelization proceeds, the ratio of computation to communication is shifted to a higher weighting of the communication performance. Hence, new infrastructures are build as on-chip networks and outperform conventional paradigms regarding the anticipated trade-off between hardware costs, bandwidth capacities, delay, and power. But with the same progress the technology advances to smaller feature sizes, the influence of physical imperfections on CMP operations grows, and the consequences propagates through all design- as well as run-time aspects. Therein, the communication infrastructure is the key to optimized CMP operations and the work at hand makes significant contributions to the state-of-the-art of Networks-on-Chip (NoCs).

As a start, the main challenges of the NoC in its operational context as part of the CMP and under consideration of the expected workload behavior are introduced. Furthermore, the leading guidelines for the investigations are outlined and a discussion of the NoC background is provided. This preliminary work is discussed in the context of existing state-of-the-art proposals, major side-effects/interrelations are identified, and the basic cost/performance/reliability dependencies are formulated. As result, simple mesh-based NoCs are chosen as baseline, because they provide sufficient benefits regarding the upcoming technology challenges. Thereby, the provided regularity and simplicity should be maintained, while the suggested improvements address the connectivity of the computational resources.

The first major contribution addresses the joint improvement of the performance, reliability and traffic management capabilities of the 2D-Mesh via the proposed QMesh topology. The integration of multiple spatially independent terminals per computational resource offers a valid alternative to optimization strategies that rely on the integration of more router-to-router links for improved network connectivity. Under consideration of deterministic xy-routing and static path table setups, it is shown:

- That the QMesh clearly outperforms the reference 2D-Mesh regarding the achievable network saturation levels and packet delays by around 50% on average for a wide range of traffic pattern at moderate increases of around 50% regarding total power dissipation and area footprint.

- Furthermore, the wear-out is reduced significantly by the inherent features of a reduced mean hop distance and the lowered traffic intensities per router, while the integrated dual path capabilities provides efficient workarounds for permanent failures.
- Finally, the QMesh offers enough degrees of freedom to establish adaptive traffic management upon this structure that supports the combination of online and offline policies by reprogramming the path tables.

As next step, a flexible run-time configurable traffic monitoring solution for mesh-based NoCs is introduced that represents a vital run-time feature such CMP infrastructures and serves itself as intermediate mechanisms to provide self-awareness over the current traffic situation. It has been shown that traffic monitoring can be designed to provide sufficient spatio-temporal adaptivity, component coverage and flexibility to capture the communication behavior of modern workloads with adequate accuracy for different use cases. The main properties of the proposed solution can be summarized as follows:

- The spatial scope of the monitoring can be adjusted at run-time to rectangular shaped regions-of-interest with up to 64 tiles and it was shown that this sizing limit is sufficient for a wide range of parallelized workload applications. Furthermore, a clustering is introduced and thus multiple regions can be observed independently with individual configurations required by their assigned workload fractions.
- The timing scope of the traffic monitoring is adjustable in the range of 10^3 up to 10^5 clock cycles per full monitoring cycle inside each cluster. At the end of each monitoring cycle, the provided monitoring data consists of ready-to-use utilization values of the observed NoC components. Furthermore, it was discussed that these ranges are sufficient to capture traffic statistics that keeps up with the expectable dynamic communication behavior of workloads.
- The monitoring in each cluster has a central organization and thus a single entity has a global view on its assigned region. The monitoring data evaluation itself is performed in software and provides full adaptivity to support different use cases and data sharing capabilities. The data sensing and aggregation is realized in hardware and distributed between the observed slave-tiles as well as the central master-tile.
- For each cluster the timing scope can be adjusted locally by the central master-tile as well as globally by the reconfiguration of all slave-tiles. Furthermore, the classes of components covered by the monitoring can be configured at run-time and therefore adjusted for the desired use case.

- The monitoring data communication is performed over the fully customizable and independent SNoC. Therefore, it could be shown that this kind of communication offers reusability and can be extended to a general infrastructure for the data communication relevant for the system control.
- Contrary to the majority of the presented solutions in the state-of-the-art, the traffic monitoring of the work at hand is not maximally optimized for a single use case. Instead, the implementation costs are addressed by the increased reusability of data and infrastructure, which improves the cost-to-benefit ratio of the necessary investments and allows different management services to profit.

Furthermore, the proposed traffic monitoring was discussed regarding its application to different use case scenarios. The evaluated hardware costs are in a feasible range of lower than 5% of the anticipated realistic tile area. The resulting monitoring errors did not exceed the double of the adjusted scale resolution in maximum and were lower than the half of this resolution on average.

Finally, a concrete use case for the combination of the QMesh and the run-time traffic monitoring is proposed, which provides remarkable potential performance gains for a wide range of traffic patterns by improving the saturation up to factors of 5, reducing the mean packet delay by around 40% up to 80% and thereby imposing suitable power overheads in comparison to the 2D-Mesh reference case. It has been further shown, that software-directed and regionally-centralized traffic management is a valid option for modern CMPs despite the generalized context of the applied traffic monitoring. The introduced path update evaluation flow and the corresponding path comparison metric proved as efficient regarding the achieved improvements as well as the computational efforts. The presented simulation results for the synthetic traffic patterns allow the narrowing of application scenarios in that the application of the proposed run-time traffic is profitable. Additionally, the performed simulations showed that the evaluated use case of run-time traffic management for the proposed QMesh might not require supported cluster sizes of 64 tiles and a reduction to 16 tiles in maximum is possible. Hence, the costs of the traffic monitoring regarding area footprint and power dissipation decrease significantly.

At the end of this thesis, it has to be mentioned that each of these major contributions offers further potential for optimizations and the work at hand provides a promising proof-of-concept evaluation. Therefore, the outlook at the end of each chapter in the work at hand formulates dedicated optimizations and further modifications for future investigations.

I APPENDIX

I.A MODELING, TOOLS AND SYSTEM SETUPS

Tool	Version	Web Resource
CACTI	6.5	→ www.hpl.hp.com/research/cacti/
McPAT	0.8	→ www.hpl.hp.com/research/mcpat/
Sniper	5.1	→ www.snipersim.org
DSENT	0.91	→ https://sites.google.com/site/mitdsent/
Pin tool	2.12-58423	→ https://software.intel.com/en-us/articles/pin-a-dynamic-binary-instrumentation-tool
SystemC	2.2.0	→ http://www.accellera.org/downloads/standards/systemc
SystemC-TLM	2.01	

This part introduces the relevant information on the software tools and methodologies to simulate the CMPs as well as the standalone NoCs. Afterwards, the evaluated synthetic and realistic workloads are discussed. Finally, the basic setups for the simulated systems are introduced.

I.A.1 MODELING AND SIMULATION

The main simulation results of the work at hand were generated via hardware/software-models implemented in SystemC [233]. It is a C++ library with included event-based simulation kernel, which allows writing software as well as describing the underlying hardware in the same programming language and simulating the implemented model after successful compilation. Furthermore, an extension for transaction level modeling (SystemC-TLM) represents an excellent mechanism to abstract complex signal flows and protocols in digital hardware-software systems. Such C++- or SystemC-based simulation models are the standard technique in the many-core research community [30], [75], [234]–[238]. As preliminary work for the exploration of NoCs in the work at hand, we developed a parameterized SystemC-TLM simulator [234] to support standalone evaluation of the infrastructure in detail and with cycle approximate timing coverage. In every IP core as well as inside the routers traffic statistic are logged. For the generation of power or energy statistics external tools must be used with these gathered traffic statistics as inputs. The simulated workload scenarios are mainly the synthetic traffic as introduced in the paragraphs below.

For the simulation of a complete CMP, different freely available simulation platforms exist. Thereby, typical simulation platforms used in the many-core research domain are Sniper [239], Graphite [240], Multi2Sim [241], GEM5 [242], MARSS [243] or ArchC [244]. In the work at hand, the CMPs were simulated via a customized version of Sniper [239], which is a branch of the Graphite simulator [240]. Therefore, the proposed adaptations regarding topology, routing and traffic management needed to be integrated as part of the preliminary work. Furthermore, the integration of Pin tools [229] for the simulation of Intel's x86 and x86-64 processors setups allows deep introspection at the instruction

set level for realistic workload benchmark collections like SPLASH-2 and PARSEC [76]. For the exploration of run-time based management strategies, the simulator offers an interface to external scripts programmed in the Python language to control its behavior and components. Regarding the covered details of communication and NoC-based infrastructure, Sniper imposes some limitations compared to the pure NoC simulator written in SystemC, which are as follows:

- The NoC is defined as hop-to-hop structure with unlimited input-buffer resources and analytical contention model. The missing buffer size constraints neglect critical side effects from blocking situations inside the NoC regarding high saturated traffic regions. In the contention model links are interpreted as shared resources whose free time slots are organized in a history tree. The total queuing delay of a packet along the defined path is evaluated via successive parsing of those trees for free slots and afterwards updating them as well as the own delay sum. Thus, concurrency is not modelled explicitly as part of the packet traversal and the timing is not as accurate as in the SystemC-based simulator.
- The simulated traffic is reduced on memory transactions, because they represent the main sources of data transfers [239]. Therefore, the main hotspots inside the CMP are memory controllers and shared memory locations.

Those abstractions are necessary to simulate complex CMP containing memory, computational resources as well as communication infrastructure in a feasible amount of time. For the simulated CMP setups in the work at hand, the simulator was configured to operate with maximum accuracy according to the results presented by *Carlson et. al.* in [239].

In addition to the simulators discussed above, external tools for the evaluation of preliminary parameter settings, statistics and visualizations are used. For technology specific evaluations of simulation statistics of the NoC infrastructure regarding area, power, timing and energy the DSENT [89] tool is applied. It is an improved successor of the widely used Orion tool [92], [99], [100] and offers standardized hardware component models (routers and links) as well as CMOS technologies. For the evaluations in the work at hand, those characteristics of DSENT offer the best option to produce comparable results, which are not affected by specific side effects due to variations in HDL coding of hardware components, synthesis tool chain flows or technology libraries. Technology specific simulation statistics evaluations for the CMPs are realized via McPAT [245] to produce specific data regarding area, power, timing and energy. Thereby, McPAT was constrained to the computational and memory resources. The communication infrastructure evaluation is done via DSENT as well as generated post-processing models from it for the evaluation of the final statistics. Preliminary parameter settings for cache memories in specific technologies were produced with the CACTI [174] tool. The supported CMOS technology settings of DSENT, McPAT and CACTI are based on

the combination of predictive technology models (PTM) [246] with the projections of the ITRS [2]. Furthermore, the post-processing of traffic statistics regarding the simulated infrastructures considers a simplified model for the approximation of the operating temperature, which was developed as part of the work at hand. The fundamentals of this model and its interplay with estimation of the NoC power dissipation are discussed in Appendix I.C (see p. 175). The applied methods for the evaluation and aggregation of the NoC traffic statistics are discussed in Appendix I.D (p. 179). The software versions and web-resources of all tools are referred in the initial table. The complete tool-chain flow for the performed simulations in the work at hand is as depicted in Figure I.1 below.

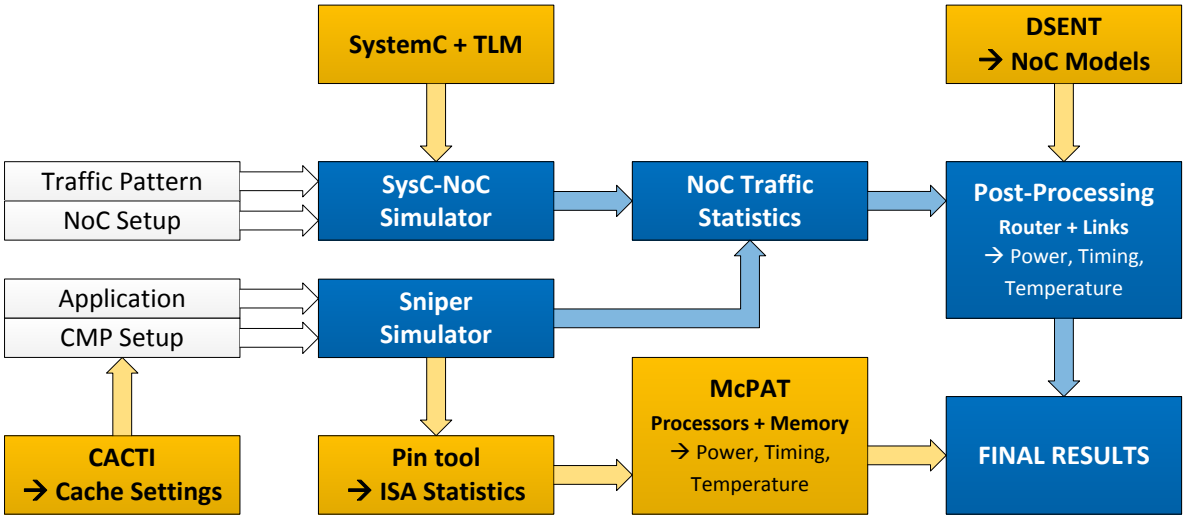


Figure I.1: Overview on modeling and simulation tools

I.A.2 SYNTHETIC TRAFFIC PATTERN

Traffic pattern are essential to explore the characteristics of communication infrastructures. Thereby, the capturing of realistic traces requires the complex modeling of memory, computation and communication of a CMP with all necessary features to perform real application workloads, which prohibits a wide ranged design space exploration regarding the necessary simulation times and insufficient abstraction of implementation details. Therefore, those workloads are most commonly reduced to synthetic traffic pattern [12], [56], [175]–[184] which reflect the average communication behavior of single- or multi-application scenarios running on the CMP. They abstract the details of data flow generation as well as processing via statistic approximations and support simplified scale-up evaluations. The complexity of the traffic generation can be further implemented with full coverage of the transactional behavior, such as conditional acknowledges and replies, or unresponsive receptions at the destination. The base for the traffic generation process is most commonly the global identifier (ID) or address of each source inside the NoC, which is interpreted as bit vector (ID_{BV}), dimension-based position (ID_{XY}) or logical decimal address value (ID_{LOGIC}) to select packet destinations after a specific pattern function. Thereby, bit vector and logical address are related as follows:

$$\boldsymbol{ID_{BV}} = a_{w_{ID}-1} \dots a_1 a_0 \quad \Longrightarrow \quad \boldsymbol{ID_{LOGIC}} = \sum_{k=0}^{w_{ID}-1} a_k 2^k \quad with \quad a_k \in \{0,1\}$$

$$0 \leq ID_{LOGIC} < N_X \cdot N_Y$$

In mesh-based two-dimensional floorplans, the conversion between logical address and dimension-based positions is realized via the NoC sizing parameter (N_X and N_Y) as follows:

$$\boldsymbol{ID_{LOGIC}} = f(x,y) = y \cdot N_X + x \quad \Longrightarrow \quad x = ID_{LOGIC} \bmod N_X \quad ; \quad y = \frac{ID_{LOGIC}}{N_X}$$

$$\boldsymbol{ID_{XY}} = \{x,y\} \; with \;\; 0 \leq x < N_X \;\; and \;\; 0 \leq y < N_Y$$

For the selection function of destinations at the traffic generating sources, three major approaches exist, where the destination selection is implemented via ***bit permutation*** *(BP)*, ***digit permutation*** *(DP)* or as spatial ***communication distribution function*** *(CDF)*. In a *bit permutation pattern* the logical resource address ID_{LOGIC} (or concatenated ID_{XY}) of the packet source is interpreted as bit vector ID_{BV} and manipulated following a specific pattern-function (f_{BP}) to generate a single destination. The *digit permutation patterns* work similar, but typically operate (f_{DP}) on the x and y digits of the position identifier ID_{XY}.

$$\boldsymbol{a_l^{DST}} = f_{BP}\left(a_k^{SRC}\right) \;\; with \;\; k,l \in \{0,1,\dots,w_{ID}\} \;\; or \;\; \{x_{DST}; y_{DST}\} = f_{DP}(x_{SRC}, y_{SRC})$$

Both kinds of address permutations (bit or digit) create traffic flows with one destination per source and reflect static workload scenarios of singular application domains with dominant flows. Thereby, the bit permutations like the transpose, shuffle, bit complement, bit reverse, or bit rotations are widely used in the NoC research domain [25], [61], [143], [163], [175], [179], [183], [247]. The destination selection functions of the BP pattern used in the work at hand (transpose, shuffle, bit complement and bit reverse) are as follows:

TRANSPOSE	SHUFFLE
$a_l^{DST} = a^{SRC}_{\left(l+\frac{w_{ID}}{2}\right) \bmod w_{ID}}$	$a_l^{DST} = a^{SRC}_{(l-1) \bmod w_{ID}}$
BIT COMPLEMENT (BIT COMP)	**BIT REVERSE (BIT REV)**
$a_l^{DST} = \overline{a_l^{SRC}}$	$a_l^{DST} = a^{SRC}_{(w_{ID}-l-1)}$

The simulation of mixed application workloads with increased dynamic behavior and multi-destination flows is typically abstracted via patterns formulated as spatial CDF that assigns selection probabilities ($\lambda_{i\to j}$) to source-destination pairs ($i \to j$) following specific spatial parameter functions, such as the hop distance ($n_{hop,i\to j}$). For the generation of each packet at the sources this function is called to serve the destination.

The most prominent CDF implements the random uniform probability assignment policy. Therein, each possible destination is weighted with the same selection probability. In the work at hand, this ***random uniform*** CDF (UNI) is only applied for general or worst-case evaluations, because the resulting traffic patterns do not provide a sufficient reflection of realistic system workloads. Furthermore, it is modelled via the additional modification of a variable path occupation (PO=20/40/60/80%) to cover scenarios with fewer destinations per source $(N_X \cdot N_Y\text{-}1) \cdot S_{PO}$ with S_{PO}={0.2, 0.4, 0.6, 0.8}. Thereby, the remaining path fraction covers the same packet injection rate as it would be for the unmodified uniform random pattern.

RANDOM UNIFORM CDF (UNI)

$$\lambda_{i\to j} = \frac{1}{(N_X \cdot N_Y - 1)} \quad \Rightarrow \quad \sum_{j=0}^{N_X \cdot N_Y - 1} \lambda_{i\to j} = 1 \;\; with \;\; i \neq j \;\; and \;\; i,j \in \{0, 1, \dots, (N_X \cdot N_Y - 1)\}$$

Alternative CDF-based synthetic traffic pattern provide better coverage of the realistic communication behavior by the consideration of spatial locality characteristics, which result from optimized application mapping strategies and traffic concentration (hotspots) to dedicated resources such as memory controller. Therein, the key parameters for probability assignments are the hop distance $n_{hop,i\to j}$ or a specific subset of resources. The simplest spatial constrained CDF is an extended version of the random uniform assignment policy that implements a ***nearest-neighbor*** (NN) preference. The set of $N_{min\to hop,i}$ nearest neighbors of a source i inside the range of the minimum possible hop distance have a defined total selection probability of λ_{NN}.The remaining set

of $(N_X \cdot N_Y - N_{min \to hop,i} - 1)$ resources outside this minimum distance region has a total selection probability of $(1 - \lambda_{NN})$. Inside each of those two sets the destination selection is random uniform for each generated packet.

NEAREST NEIGHBOR CDF (NN)

$$\lambda_{i \to j} = f(n_{hop,i \to j}) = \begin{cases} \frac{\lambda_{NN}}{N_{min \to hop,i}} & if\ (n_{hop,i \to j} = \min) \\ \frac{1-\lambda_{NN}}{N_X \cdot N_Y - N_{min \to hop,i} - 1} & else \end{cases} \quad with\ \ i \neq j\ \ and\ \ i,j \in \{0, 1, \dots, (N_X \cdot N_Y - 1)\}$$

The nearest neighbor CDF applies a discrete separation of communication intensities under consideration of two different probability levels.

RENTIAN CDF (RENT)

$$\lambda_{i \to j} = \frac{\lambda_{hop,i \to j}}{N_{hop,i \to j}} \quad with\ \ i \neq j\ \ and\ \ i,j \in \{0, 1, \dots, (N_X \cdot N_Y - 1)\}$$

$$\lambda_{hop,i \to j} = \left[1 + 2 \cdot n_{hop,i \to j} \cdot (n_{hop,i \to j} - 1)\right]^R + \left[2 \cdot n_{hop,i \to j} \cdot (n_{hop,i \to j} - 1) + 4 \cdot n_{hop,i \to j}\right]^R - \left[2 \cdot n_{hop,i \to j} \cdot (n_{hop,i \to j} - 1)\right]^R - \left[1 + 2 \cdot n_{hop,i \to j} \cdot (n_{hop,i \to j} - 1) + 4 \cdot n_{hop,i \to j}\right]^R$$

But the findings of *Heirman et. al.* in [67], *Greenfield* in [11] as well as *Bezerra* in [12] show that the distribution of communication probabilities as function of the hop distance in a wide range of realistic workloads follows a power law shape as given by Rent's rule. Hence, the ***rentian*** (RENT) pattern seems another good candidate for synthetic traffic generation. The work at hand uses this rentian traffic pattern similar to this one formulated by *Manevich et. al.* in [14]. Thereby, each reachable subset of $N_{hop,i \to j}$ resources around a source i within a defined hop distance $n_{hop,i \to j}$ gets assigned a specific selection probability $\lambda_{hop,i \to j}$, which is approximated after a formula as given in [231]. The selection policy inside each subset applies a random uniform distribution. The rentian exponent R defines the shape of the probability distribution and was varied in the interval of 0.3 up to 0.7.

Contrary to this distance-based communication probability distribution, the ***hotspot*** (HOT) pattern assigns a defined communication probability λ_{HOT} to a free selectable subset $\mathbb{A}_{HOT}$ with N_{HOT} resources. Those dedicated floorplan-positions typically receive more traffic and represent hotspots as well as the potential bottlenecks of the communication infrastructure.

HOTSPOT CDF (HOT)

$$\lambda_{i \to j} = f(j, \mathbb{A}_{HOT}) = \begin{cases} \frac{\lambda_{HOT}}{N_{HOT}} & if\ (j \in \mathbb{A}_{HOT}) \\ \frac{1-\lambda_{HOT}}{N_X \cdot N_Y - N_{HOT} - 1} & else \end{cases} \quad with\ \ \mathbb{A} = \{0, 1, \dots, (N_X \cdot N_Y - 1)\}\ \ and\ \ \mathbb{A}_{HOT} \subset \mathbb{A}$$

These spatial distribution functions above are applied for the destination selection at each IP core for each generated network packet. In addition, discrete distributions must be specified for the size of the network packets and the activation periods of the traffic generation.

I.A.3 REAL APPLICATION-BENCHMARKS

For the full system simulations of a CMP in Sniper a collection of benchmark applications from SPLASH-2 [248] and PARSEC 2.0 [249], [250] are utilized (see table below). This collection represents a wide range of application domains.

Table I.1: Selected benchmark applications for the Sniper-based CMP simulations

Benchmark Application	Application Domain	Problem Size	Collection
Blackscholes	Financial Analysis	simsmall	PARSEC 2.0
Ferret	Similarity Search	simsmall	PARSEC 2.0
Swaptions	Financial Analysis	simsmall	PARSEC 2.0
Barnes	High Performance Computing	small	SPLASH-2
FFT	Signal Processing	small	SPLASH-2
FMM	High Performance Computing	small	SPLASH-2
LU.cont	High Performance Computing	small	SPLASH-2
Ocean.cont	High Performance Computing	small	SPLASH-2
Radiosity	Graphics	small	SPLASH-2
Water.sp	High Performance Computing	small	SPLASH-2

The simulated workloads are single applications with a parallelization based on POSIX-threads. Thereby, the number of parallel threads is configured to fit the number of available processor resources (N_{tile}) in the CMP. Multi-application workloads are prohibited in the utilized version of Sniper due to the absence of necessary profiling capabilities for such use cases. For detailed evaluations computational characteristics for the utilized application the author refers to [11], [12], [67], [251], [252]. The default thread mapping of Sniper is a static direct assignment strategy without rescheduling, where the index of the thread (0, 1, …, N_{tile}-1) matches with the logical NoC address of the assigned resource (0, 1, …, N_{tile}-1). For the logging of statistics, the application code contains special region-of-interest (ROI) hook-commands to mark the beginning and end of relevant sections in the code. This ensures that non-relevant data pre-processing/generation/post-processing sections do not affect or falsify the results. The input problem is set to small/simsmall sizes due to simulation time issues. The simulation host was a virtual machine with eight AMD Opteron 4184 cores at 2.8 GHz operating frequency and 32 Gbyte of assigned RAM. The simulation time for a single simulation run of the benchmark simulations varies between 1 up to 10 hours. Each application workload was simulated 10 times.

I.A.4 SYSTEM SETUPS

SystemC-based NoC Simulation Setup: The full system configuration setup with all parameter variations for the SystemC-based NoC simulation can be obtained from the table below. Detailed explanations on design-/run-time parameters are presented in the Sections 2.1 (pp. 26-46) and 2.2 (pp. 46-55), which contain the NoC fundamentals.

Table I.2: System and parameter setups for SystemC-based NoC simulations

Parameter	Value
Topology	2D-Mesh, QMesh
Technology setup	45nm (see Table I.5 p. 168 and Table I.6 p. 169)
NoC size – $N_X \times N_Y$	4x4, 8x8 tiles
# of ports – N_P	5, 8 ports
Routing algorithm	Dimension-ordered XY
Switching	Wormhole-switching (WHS)
Allocation SA	Matrix Arbiter
Buffering	FF-based FIFO at input port
Pipeline stages	RL, SA, ST, LT
Handshaking	REQ/ACK
Link's data width – w_{data}	64 bit / 8 byte
Buffer depth @ Input ports – d_{buf}	9 flit / 72 byte
Buffer depth @ NI – s_{buf}	4096 flit / 32kbyte
# of VC – N_V	No VCs
Clock cycle – t_{NoC}	1 ns
RL delay – t_{RL}	1 ns / 1 clock cycle
SA delay – t_{SA}	1 ns / 1 clock cycle
ST delay – t_{ST}	1 ns / 1 clock cycle
LT delay – t_{LT}	1 ns / 1 clock cycle
Handshaking	REQ/ACK
Workload setup	
Simulated operation time per run	2 up to 5 ms
# of simulation runs per setting	10
Simulated workloads / Synthetic traffic pattern	- Transpose, Shuffle, Bit complement, Bit reverse
	- Nearest Neighbor → λ_{NN} = {0.2, 0.4, 0.6, 0.8} - Rentian → R= {0.3, 0.7} - Random Uniform → PO={0.2, 0.4, 0.6, 0.8} - Hotspot → λ_{HOT} = {0.2, 0.4, 0.6, 0.8} and $\mathbb{A}_{HOT}$ = {8, 15, 16, 23, 40, 47, 48, 55}@8x8
Packet size probabilities	- $s_{packet} = 9\ flit$ → 80% - $s_{packet} = 2\ flit$ → 20% - $\bar{s}_{packet} = 7.6\ flit$

Sniper-based NoC Simulation Setup: The configuration setting of the Sniper-based simulation runs of a full CMP are listed in the table below. The setup for this system is oriented on state-of-the-art experiments and platforms as given by [16], [31], [43], [116].

Table I.3: System and parameter setups for Sniper-based CMP simulations

Parameter	Value			
Topology	2D-Mesh, QMesh			
Technology setup	45nm (see Table I.5 p. 168 and Table I.6 p. 169)			
NoC size – $N_X \times N_Y$	8x8 tiles			
# of ports – N_P	5, 8 ports			
Routing algorithm	Dimension-ordered XY			
Switching	Wormhole-switching (WHS)			
# of VC – N_V	No VCs			
Link's data width – w_{data}	64 bit / 8 byte			
Core Frequency / Clock rate	2 GHz / 0.5 ns			
Static hop delay	2 ns / 4 clock cycles			
IP core architecture	Intel Nehalem + Pentium M Branch Predictor (Sniper defaults) - Misspredict penalty = 4 ns / 8 clock cycles - Dispatch width = 4 - Window size = 128 - # of outstanding load/stores = 10 - Reschedule costs = 500 ns / 1000 clock cycles			
Cache coherency protocol	Modified Shared Invalid (MSI)			
Cache levels	2			
Cache algorithm	Least Recently Used (LRU)			
Cache setup per tile	**Parameter**	**L1-I**	**L1-D**	**L2**
	Block size	64 Byte	64 Byte	64 Byte
	Cache Size	32 kbyte	32 kbyte	256 kbyte
	Associativity	4	4	4
	Sharing	Private	Private	Private
	Data access time	0.5 ns / 1 clock cycle	0.5 ns / 1 clock cycle	2 ns / 4 clock cycle
	Tag access time	0.5 ns / 1 clock cycle	0.5 ns / 1 clock cycle	2 ns / 4 clock cycle
	Access mode	Parallel	Parallel	Parallel
	Write through	No	No	No
TLB	**Parameter**	**ITLB**	**DTLB**	**STLB**
	TLB size	64	64	128
	Associativity	8	8	8
	Penalty	15 ns / 30 clock cycles		
Memory controller	- # of controller = 8 - Read-/Write-Latency = 22.5 ns - Bandwidth = 5.25 GB/s per controller			
Memory controller positions	ID_{LOGIC} ={8, 15, 16, 23, 40, 47, 48, 55}			
Workload setup				
# of simulation runs per setting	10			
Simulated workloads / Application benchmarks	See Table I.1 at p. 163			

SystemC-based TRM Simulation Setup: For the explored System-NoC (SNoC) configurations in the work at hand, the link setup $w_{data \rightarrow SNoC} = 16\ bits$ has been chosen. This would guarantee the application of this SNoC for CMP sizes of up to 256x256 tiles with a 2D-Mesh and 128x128 tiles with a QMesh topology. Furthermore, the selected data word size is in the range of related work proposals such as RCA with ≥ 16 bits [61], DBAR with ≥ 8 bits [62], HPRA with ≥ 8 bits [142], MNoC with ≥ 24 bits [195], [200], and *Guang et. al.* with ≥ 8 bits [204], [205] for monitoring links.

Table I.4: Applied parameterization and feature composition of the System-NoC

SNoC-PARAMETER	2D-MESH	QMESH
Technology setup	45nm (see Table I.5 p. 168 and Table I.6 p. 169)	
NoC size – $N_X \times N_Y$	See NoC in setup table at p. 164	
# of ports – N_P	5 ports	8 ports
Routing algorithm	Dimension-ordered XY	Dual-path XY
Switching	Wormhole-Switching (WHS)	
Allocation SA	Matrix Arbiter	
Buffering	FF-based FIFO at input ports	
Pipeline stages	RL, SA, ST, LT	
Handshaking	REQ/ACK	
Link's data width – $w_{data \rightarrow SNoC}$	16 bit / 2 bytes	
Buffer depth @ Input ports – d_{buf}	1 flit / 2 bytes	
# of VC – N_V	No VCs	
Clock cycle – t_{NoC}	1 ns	
Handshaking	REQ/ACK	

The selected buffer depth at the input ports is set to the minimum of one flit. As the preliminary work on the router component showed (see pp. 170-172), the port buffers dominate the area footprint as well as the static power consumption. Thus, minimum sized buffers come along with minimum additional design- and run-time costs for the SNoC. From the performance perspective, the addressed centralized organization results in a many-to-one traffic pattern with a single hotspot at the assigned master-tile. Thus, deeper buffers will have minor effect on possible performance improvements, because the packet reception rate of the master-tile specifies the throughput and therefore an increase of the link's data width or the clock frequency of the SNoC would have higher impacts. The frequency strategy for multi-networks was already proposed and explored by in [53].

I.B BASIC NOC COMPONENT PROFILES

According to the description of the basic Network-on-Chip router architecture and links in Chapter 2, the following design exploration provides further insights regarding timing, area cost, and power dissipation. These investigations at the component level are indispensable to evaluate the impact of modifications at the NoC infrastructure according to the design-time as well as run-time metrics (timing/performance, area footprint, power dissipation). Furthermore, the collected data represents a consistent reference based on homogeneous technological assumptions for the work at hand. It was performed via the DSENT [89] software tool that integrates design analysis as well as implicit timing optimizations at the standard cell layer using a parameterized model of the input-buffered reference router.

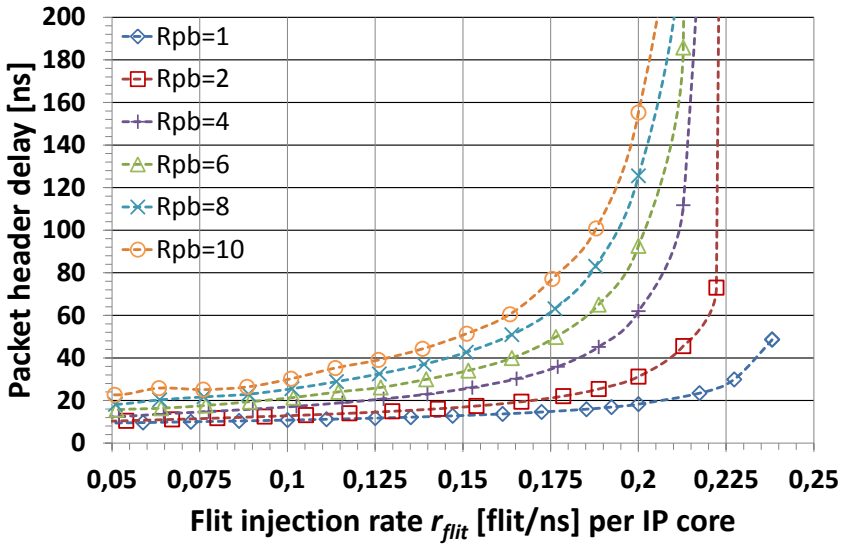


Figure I.2: Impact of packet- to buffer-size ratio (R_{pb}=1, 2, 4, 8, 10) for the exemplary case of a wormhole-switching two-dimensional 4x4 tiled 2D-Mesh in a uniform random traffic scenario and applied dimension ordered xy-routing

Thereby, the utilized CMOS technology models for 45nm, 32nm and 22nm are based on the projections of the ITRS [2] for low threshold-voltage (LVT) circuits. The number of router ports N_P was varied from 5 to 8 and evaluated with two different VC configurations (none/1 and 2 VCs). These router degrees cover most of the NoC topologies discussed and compared in this work. Furthermore, for the implementation of some adaptive routing algorithms or traffic separation in NoCs at least two VC will be needed [25] and their inpact is investigated as well. Thus, the maximum number of 2 VCs is needed to study the cost impact of these algorithms at the router level. The link's data width w_{data} was set to 64 bit and the buffer depth d_{buf} was fixed at 9 flit. Both parameters were hold constant and sized for a typical CMP traffic scenario, where most of the packets carry complete cache lines with 64 bytes of data [21]. The selected data width of the link corresponds to the typical size of data words in 32 and 64 bit processors. Thus, each packet contains 8 payload flits plus 1 header flit and will fit completely into the buffer at the input port. This is in general a beneficial entry-point

regarding the avoidance of interferences in wormhole-switching NoC, where packets greater than the port buffer stretch over multiple router along the path and thus may lead to an increase in ETE-latency as well as a reduction of throughput in concurrent data flows. The line-chart in Figure I.2 illustrates the resulting effect of those interferences on the mean packet delay for the exemplary case of a simulated 4x4 tiled 2D-Mesh NoC. Thereby, the ratio of packet- to buffer-size ($R_{pb} = s_{packet} / d_{buf}$) was varied and the communication pattern of cores was random uniform. It becomes obvious that the adaptation of input buffers to expected packet sizes ($R_{pb} = 1$) offers the best results regarding ETE latencies, which makes the selected parameter setup reasonable.

For the classification of area overheads as well as the link's length, a square-shaped tile with an initial size of 3.0mm×3.0mm at 45nm was derived from Intel's SCC dual-core tile layout [16], [29] and adapted to a single core tile. Afterwards, the dimensions were scaled down with a factor of $1/S \approx 1/\sqrt{2} \approx 0.7$ for the 32nm and 22 nm technology. Regarding the timing, area and power analysis are based on routers clocked at 1 ns. The following Table I.5 summarizes the defined parameter setup.

Table I.5: Parameter setup for the generic NoC router evaluation

PARAMETER	VALUE		
CMOS TECHNOLOGY	**45nm LVT**	**32nm LVT**	**22nm LVT**
Supply voltage – V_{DD}	1.0 V	0.9 V	0.8 V
Tile size – a_{tile}×b_{tile}	3.0 mm×3.0 mm	2.1 mm×2.1 mm	1.47 mm×1.47 mm
Tile area – A_{tile}	9.0 mm²	4.41 mm²	2.16 mm²
NoC size – N_X×N_Y	8×8 tiles		
# of ports – N_P	5, 6, 8 ports		
# of VC – N_V	1, 2 VC		
Routing algorithm	Dimension-ordered XY		
Link's data width – w_{data}	64 bit		
Buffer depth – d_{buf}	9 flit		
Operating temperature	340 K / 40°C		
Allocation VCA & SA	Matrix Arbiter		
Switching	Wormhole-switching (WHS)		
Pipeline stages	RL, VCA, SA, ST		
Buffering	FF-based FIFO at input port		
Clock cycle – t_{NoC}	1 ns		

In addition to the evaluation of the router architecture above, the following preliminary exploration of different repeated link configurations (LT stages), as defined in Table I.6, was performed. Thereby, the optimal segmentation was calculated automatically based on the distributed RC wire model as in Equation 2.13 (see p. 37). The analysis was performed for short (SL) and long link (LL) configurations, whereas the length of the SL was equal to the side length of a tile and the length of LL was set to the doubled side length of a tile. Additionally, the links were evaluated under the assumption that rebalanced timing of the router pipeline could be applied with a relative shortening of the link lengths by 33 percent. As data load for the link configurations, a low traffic (LT) scenario and a high

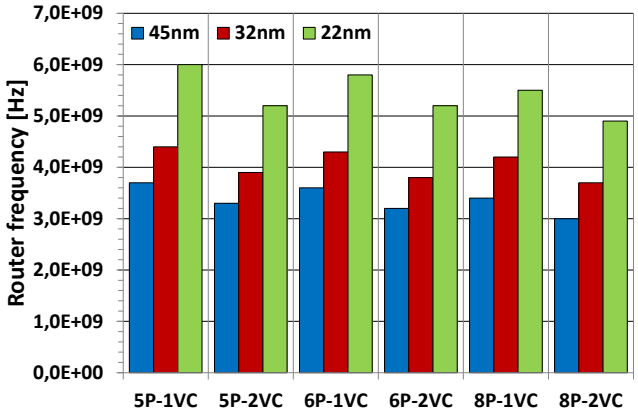
traffic (HT) scenario were defined with a flit-to-flit bit-toggle rate of 0.5. Thereby, the data transmissions over the link are assumed to be free of blocking situations.

Table I.6: Additional sizing parameter for the evaluation of Network-on-Chip links

PARAMETER	VALUE		
CMOS TECHNOLOGY	**45nm LVT**	**32nm LVT**	**22nm LVT**
Long link (LL) / Rebalanced LL – l_{link} [mm]	6.0 / 4.0	4.2 / 2.8	2.94 / 1.96
Short link (SL) / Rebalanced SL – l_{link} [mm]	3.0 / 2.0	2.1 / 1.4	1.47 / 0.98
Wire width – W_{wire} [nm]	110	55	32
Wire spacing – S_{wire} [nm]	110	55	32
Wire thickness – H_{wire} [nm]	140	100	60
Wiring-/Metal-layer	Semiglobal / Intermediate [2]		
Injection rate – r_{flit} [flit / clock cycle]	0.1 (LT) / 0.3 (HT)		

I.A.5 ROUTER TIMING CHARACTERSITICS

Figure I.3 illustrates the scaling of the maximum reachable frequencies at the internal pipeline (no LT stage) of the reference router for the defined parameter configurations in Table I.5.



ROUTER FREQUENCY [Hz]			
CONFIG	**45nm**	**32nm**	**22nm**
5P-1VC	3,7E+09	4,4E+09	6,0E+09
5P-2VC	**3,3E+09**	**3,9E+09**	**5,2E+09**
6P-1VC	3,6E+09	4,3E+09	5,8E+09
6P-2VC	3,2E+09	3,8E+09	5,2E+09
8P-1VC	**3,4E+09**	**4,2E+09**	**5,5E+09**
8P-2VC	3,0E+09	3,7E+09	4,9E+09

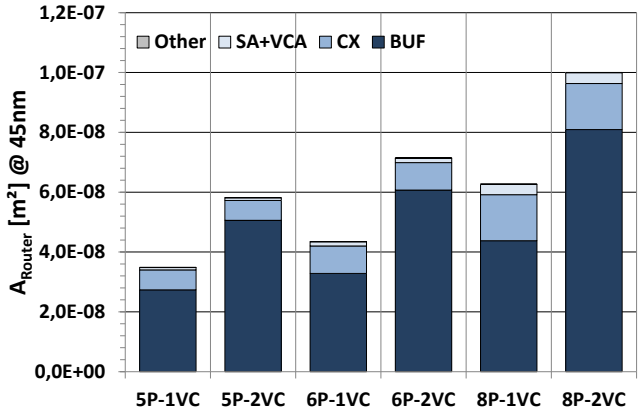
Figure I.3: Maximum frequency of the router architecture (without links) over variation of process technology (45, 32, 22nm), port number (5P, 6P, 8P) and virtual channel count (1VC, 2VC)

The data shows that the addition of VCs has a higher impact on the achievable frequency than an increase in the number of ports. According to the delay model of *Peh et. al.* in [90], this results from the logarithmic scaling of the critical path delay inside the router control logic with the number of inputs, which grow proportionally to $N_P \times N_V$ in the combination of VCA+SA and only proportionally to N_{PI} in the SA if the router is VC-free. Furthermore, the integration of VCs will intensify the imbalances in the timing of the pipeline stages. According to the model in [90] as well as the evaluation in [253], the VCA+SA unit suffers most with an increase in its critical path timing compared to a VC-free solution. The achievable frequency of the router internal pipeline ranges from 3.7 to 3.0 GHz in 45nm, 4.4 to 3.7 GHz in 32nm and 6 to 4.9 GHz in 22nm. Depending on the anticipated layout of the router, this leaves the option to rebalance the timing of the pipeline stages by stretching RL, SA, VCA and ST to shorten LT and thereby decrease t_{wire}. Those concepts were evaluated by

Cornelius in [84] as well as by *Silla* in [88]. The above presented results do not consider any timing variations induced by PVT effects, which cause additional derating and/or increased fault probabilities of the router [43], [102], [254]–[256]. Especially the comprehensive fault-model and fine-grained results of *Aisopos* in [43] demonstrate the PVT impact on a 45nm CMOS technology. It is shown that the operation of the reference router at the adjusted maximum synthesis frequency may lead to timing violation probabilities in all critical paths of up to 80% and the reliable operational range is at 75% up to 80% of the maximum frequency. Furthermore, an increasing complexity of the pipeline stages (i.e. more VCs) will intensify this relation and may widen the frequency gap, because the occurrence probability of faults becomes too high. In combination with activity-driven temperature variations [43] or self-heating [102], [254] the fluctuations on the critical path delays get worse. The results in [43] for real application benchmarks on a normal 8x8 2D-Mesh shows that even in the anticipated 75% to 80% frequency range the activity driven increase of on-chip temperatures results in mean fault occurrence probabilities of 0.1% up to 1% per router during benchmark execution, which is unacceptably high and comes along with a performance degradation due to packet retransmissions or wrong data processing. The summarized impact of those effects grows with the technology scaling and highlights the demand of simplified resilient hardware with relaxed critical path conditions.

I.A.6 ROUTER AREA CHARACTERISTICS

The area costs per router for the analyzed parameter configurations (see Table I.5 at p. 168) are illustrated in the stacked bar-chart of Figure I.4 below for the 45nm technology.



	A_{Router} [m²]			A_{Router}/A_{Tile}
CONFIG	45nm	32nm	22nm	45nm
5P-1VC	**3,5E-08**	2,1E-08	8,4E-09	**0,39%**
5P-2VC	5,8E-08	3,4E-08	1,4E-08	0,65%
6P-1VC	4,4E-08	2,7E-08	1,1E-08	0,48%
6P-2VC	7,1E-08	4,3E-08	1,7E-08	0,79%
8P-1VC	**6,3E-08**	4,0E-08	1,5E-08	**0,70%**
8P-2VC	1,0E-07	6,1E-08	2,4E-08	1,11%

Figure I.4: Area cost of the router architecture (without links) over variation of process technology (45, 32, 22nm), port number (5P, 6P, 8P) and virtual channel count (1VC, 2VC) at targeted clock rate of 1 ns

For each of the processed configurations the combined area fraction of buffers and crossbar demand at least more than 94% of the total router area and the buffers alone consumes at least 60% of it. To rate the evaluated router configurations in the global context of the CMP, the relative fractions of the resulting router areas as part of the tile areas were calculated. Based on the premise of a regular layout, these calculated area ratios represent the expected total overhead. The obtained results

show that the tile area utilization of the router does not exceed 1.5% in maximum and in most of the evaluated cases is even below 1%.

I.A.7 ROUTER POWER CHARACTERISTICS

To evaluate the power consumption of the specified router configurations, an equalized data throughput scenario was selected to provide a basic comparison and power scaling estimation.

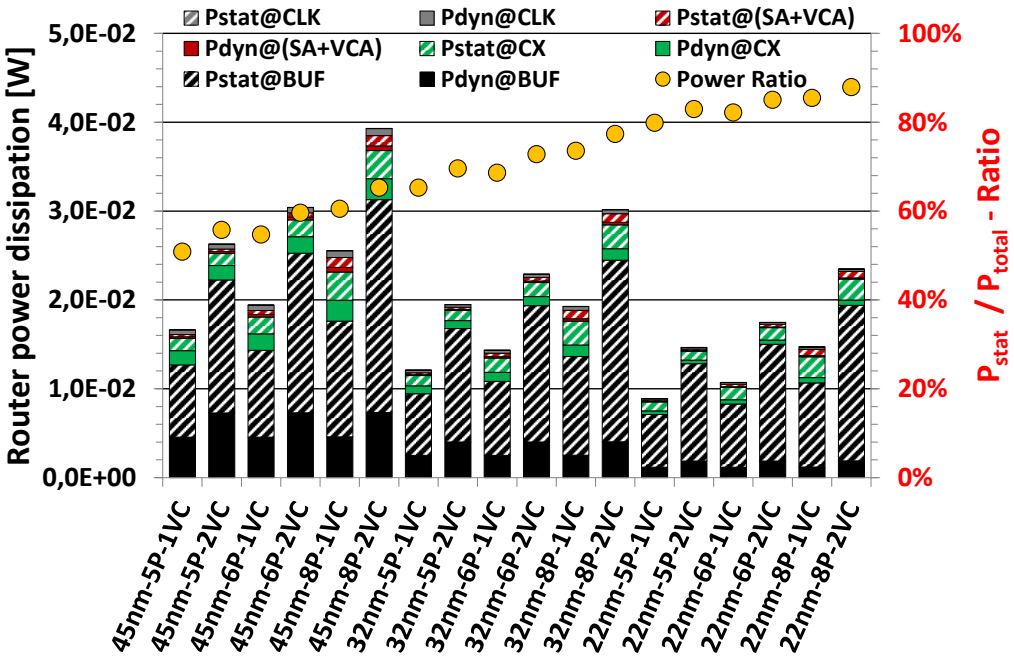
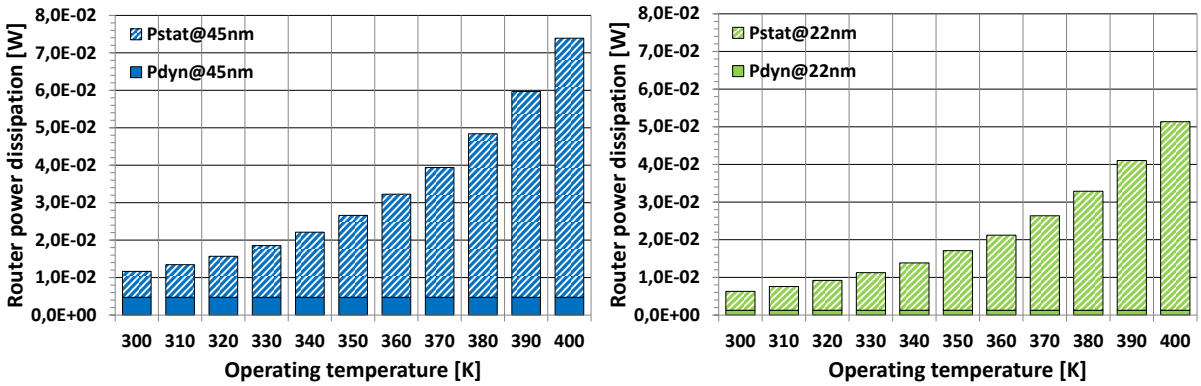


Figure I.5: Dynamic (Pdyn) and static (Pstat) power dissipation of the router (without links) over variation of process technology (45, 32, 22nm), port number (5P, 6P, 8P) and virtual channel count (1VC, 2VC) at targeted clock rate of 1 ns, operating temperature of 340 K and equalized data throughput per port.

Thereby, the reference mean flit injection rate $\bar{r}_{flit\rightarrow in}^{5P}$ per input port was set 0.3 flit/ns for the base configuration of a VC-free 2D-Mesh router with five ports ($N_P = 5$) with a random flit-to-flit bit-toggle rate of 0.5. To maintain the same data throughput per router the remaining configurations with higher port counts, the rate was adjusted via $\bar{r}_{flit\rightarrow in}^{N_P} = (5/N_P) \cdot \bar{r}_{flit\rightarrow in}^{5P}$ to 0.25 flit/ns at the six-ported and to 0.1875 flit/ns at the eight-ported router. For configuration with two VCs, it is assumed that the incoming traffic per input port splits up uniformly distributed with a VC selection probability of $1/N_V$. A similar distribution was set for the targeted output port of the incoming traffic with the output port selection probability of $1/N_P$. The stacked bar-chart in Figure I.5 illustrates the results, which are broken down to dedicated parts of the router's architecture. Buffers (BUF) and crossbar (CX) are responsible for at least 90% up to 97% of the total power dissipation in all processed configurations. The remaining fractions can be ascribed to the control logic (SA+VCA) and the clock distribution network (CLK). The evaluated total power dissipation per router varies from 16mW to 40mW at 45nm, 12mW to 30mW at 32nm, and 8mW to 24mW in 22nm CMOS. Furthermore, the power dilemma becomes obvious for the comparison of the same router

configuration as the technology is scaled down. While dynamic power decreases, the static power remains high and becomes the dominating source of power dissipation. The calculated power ratio (P_{stat}/P_{total}) indicates this ongoing trend from activity driven to leakage driven power consumption, which will be dominated by the sizing of the buffers. The assumed injection rate of 0.3 flit/ns is already in a region of higher NoC communication activity with an increased fraction of dynamic power. Additionally, it has to be mentioned that the low threshold-voltage (LVT) setup was used for the evaluation, which comes along with optimized switching delays at the cost of increased static power consumption. Furthermore, no special techniques like mixed threshold-voltage (MVT) designs [257] or fine grained power gating [258], [259] were applied, which can help to reduce the static power dissipation.



(a) 45nm (b) 22nm

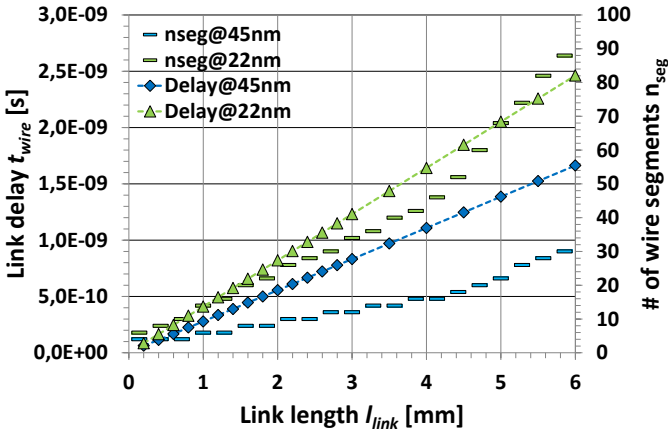
Figure I.6: Thermal impact on total power dissipation related to the increase of static power consumption (Pstat) for eight-ported VC-free router architecture at (a) 45nm and (b) 22nm process technology in the operating temperature range of 300K to 400K at an injection rate per input port of 0.1 flit/ns.

In addition to the design driven dependencies, the operating temperature has a high impact on the static power dissipation. Figure I.6 illustrates the exponential ($\gamma > 1$) increase of the leakage power for the typical temperature range of 300K up to 400K [43], [102], [254] of an eight-ported VC-free router at input port injection rates of 0.1 flit/ns. The observed relative increase between the total power dissipation at 300K to 400K accelerates with shrinking technology from a factor of 6 at 45m to 8 at 22nm and may reach one full magnitude for smaller feature sizes.

I.A.8 LINK TIMING CHARACTERISTICS

A scaling analysis of the link's minimum delay over the variation of the wire length and CMOS technology is performed. The line-chart in Figure I.7 below illustrate the expected growth of the minimum delay proportional the link's length ($t_{wire} \propto l_{link}$) at constant technologies and optimized repeater insertion. Furthermore, the drawbacks of technology scaling on the delay of interconnects become obvious by the comparison of the results for 45nm and 22nm CMOS. The linear slope of the link's delay in 22nm is higher than in 45nm and furthermore the segmentation becomes more fine-

grained with shrinking feature sizes. The minimum delay results for the specified link configurations can be obtained from in Figure I.7 as well. The data show that repeated links under consideration of the fault improved timing with 75% to 80% [43] of maximum possible frequency and the defined layout constraints only support NoC operating frequencies in the 1 GHz up to 2 GHz range if implemented in SL or rebalanced SL-R configuration. Thus, if modifications of the topology come at the price of increased wire length for the links the achievable operating frequency of the NoC is affected negatively. This impacts the throughput or enforces the designer to increase the link's width to maintain the required performance. To consider the influence of the LT stage on the possible timing of the NoC, a comparison between the achievable minimum delays of the link and the internal router pipeline can be performed by calculating the ratio (t_{wire}/t_{router}) for all defined configurations. The results show that even for the combination of the slowest router configuration (8P-2VC) with rebalanced SL the link's delay is around 1.7 times higher in 45nm and this spread increases to around 2 in 22nm. Thereby the timing results do not consider any PVT variations, which will have an additional impact on the timing degradation of the links [40], [80], [81].



t_{wire} [ps]			
Config	**45nm**	**32nm**	**22nm**
LL	1660	1364	1206
LL-R	1107	910	**805**
SL	**831**	**686**	**604**
SL-R	**555**	**456**	**403**

Figure I.7: Scaling of delay and segment count for a 64-bit wide repeated link at 45nm, 32nm and 22nm CMOS technology over a variation of the link length

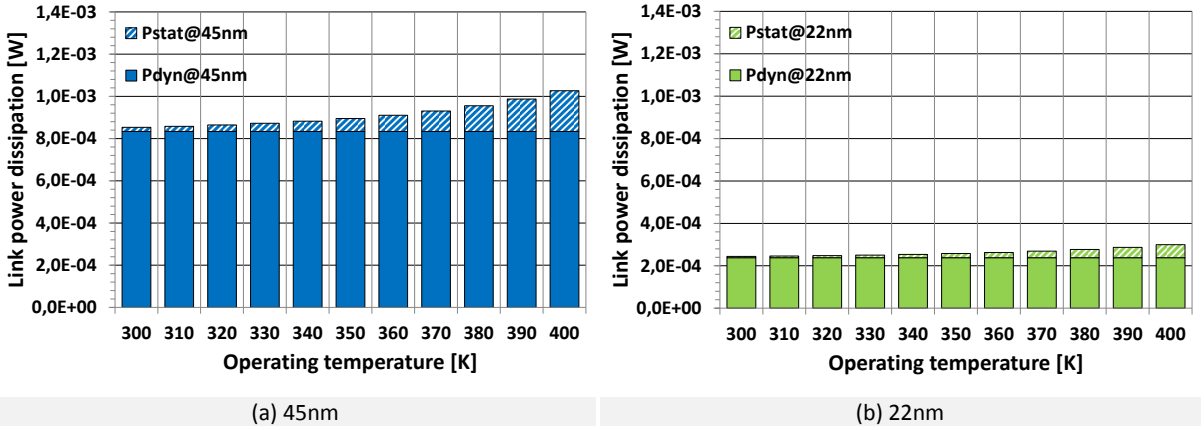
I.A.9 LINK POWER CHARACTERISTICS

The results of the power evaluation for the specified link configurations (see Table I.6) that support an operating frequency of 1 GHz (only SL and SL-Rebalanced) are presented in Table I.7. It becomes obvious, that the total power dissipation of these links will be dominated by the dynamic fraction and rebalancing can effectively help to reduce the power consumption by nearly the same portion the link's length is trimmed (around 1/3 here). Furthermore, a comparison with the HT scenario power consumption of the analyzed router configurations (see Figure I.5 at p. 171) shows, that links contribute to the total power dissipation of the NoC with the same magnitude like the routers themselves.

Table I.7: Power evaluation results for the specified 64 bit wide SL and SL-Rebalanced link configurations operating at 1GHz at LT as well as HT traffic scenario

TECHNOLOGY		45nm		32nm		22nm	
CONFIG	**POWER**	**LT**	**HT**	**LT**	**HT**	**LT**	**HT**
SL	P_{dyn} [mW]	0,83	2,50	0,47	1,41	0,24	0,72
	P_{stat} [mW]	0,05	0,05	0,02	0,02	0,02	0,02
	$\boldsymbol{P_{total}}$ [mW]	**0,88**	**2,55**	**0,49**	**1,43**	**0,25**	**0,73**
SL-Rebalanced	P_{dyn} [mW]	0,56	1,67	0,32	0,95	0,16	0,48
	P_{stat} [mW]	0,03	0,03	0,02	0,02	0,02	0,02
	$\boldsymbol{P_{total}}$ [mW]	**0,58**	**1,70**	**0,34**	**0,97**	**0,17**	**0,49**

Thereby, it has to be considered, that the total power values for single links in Table I.7 have to be multiplied, because each router in the analyzed configurations incorporates $(N_P - 1)$ incoming/outgoing links from/to adjacent routers.



(a) 45nm (b) 22nm

Figure I.8: Thermal impact on the link's power dissipation related to the increase of static power (Pstat) for a 64 bit wide link in SL configuration at (a) 45nm and (b) 22nm process technology in the operating temperature range of 300K to 400K at the injection rate per input port of 0.1 flit/ns with an operating frequency of 1 GHz

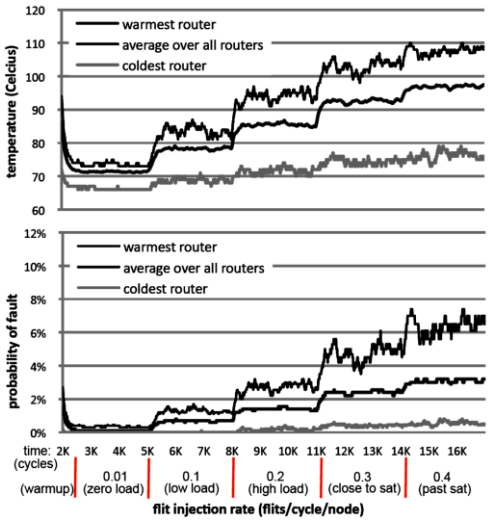
The results for the exploration of the thermal dependency regarding the static power are illustrated in Figure I.8. Similar to the evaluated router scenario (see Figure I.6 at p. 172), the static power shows an exponential increase over the linear variation of the temperature.

I.C NOC TEMPERATURE AND POWER APPROXIMATION

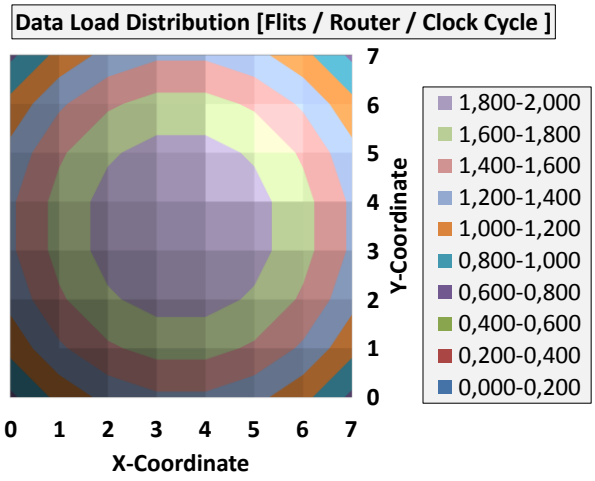
For the consideration of the operating temperature (T_{router}) of the routers and links in the simulated NoCs, a post-processing of the captured simulation statistics is performed. The basic model for the approximation of the operating-temperature and its impact on the power dissipation are discussed below. The entry-point for the temperature modelling is a relation of the flit injection rate per IPcore and the resulting mean router temperature presented by *Aisopos* in [43] for the case of an 8x8 2D-Mesh NoC with applied wormhole-switching, xy-routing and 64-bit wide links. For the analyzed injection rates the resulting router load distributions in the NoC and the mean load per router ($r_{flit \rightarrow router}$) for each distribution were calculated.

Basic assumption for relation of temperature and data load

Figure from [43] (Figure 3.8 at page 62)



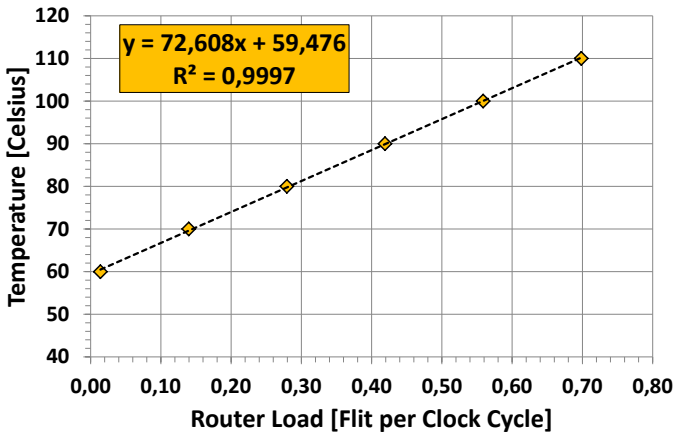
Router Load in NoC at $r_{flit} = 0.3$ [Flit / IPcore / Clock Cycle]



Afterwards, these mean router loads are projected on an operating temperature range of 60°C up to 110°C and approximated by a linear function. The utilized data and the resulting linear curve fit are as follows:

Injection Rate	Avrg. Router Load	Temperature
[Flit / IPcore / Clock Cycle]	[Flit / Router / Clock Cycle]	[Celsius]
0,01	**0,01**	**60**
0,1	**0,14**	**70**
0,2	**0,28**	**80**
0,3	**0,42**	**90**
0,4	**0,56**	**100**
0,5	**0,70**	**110**

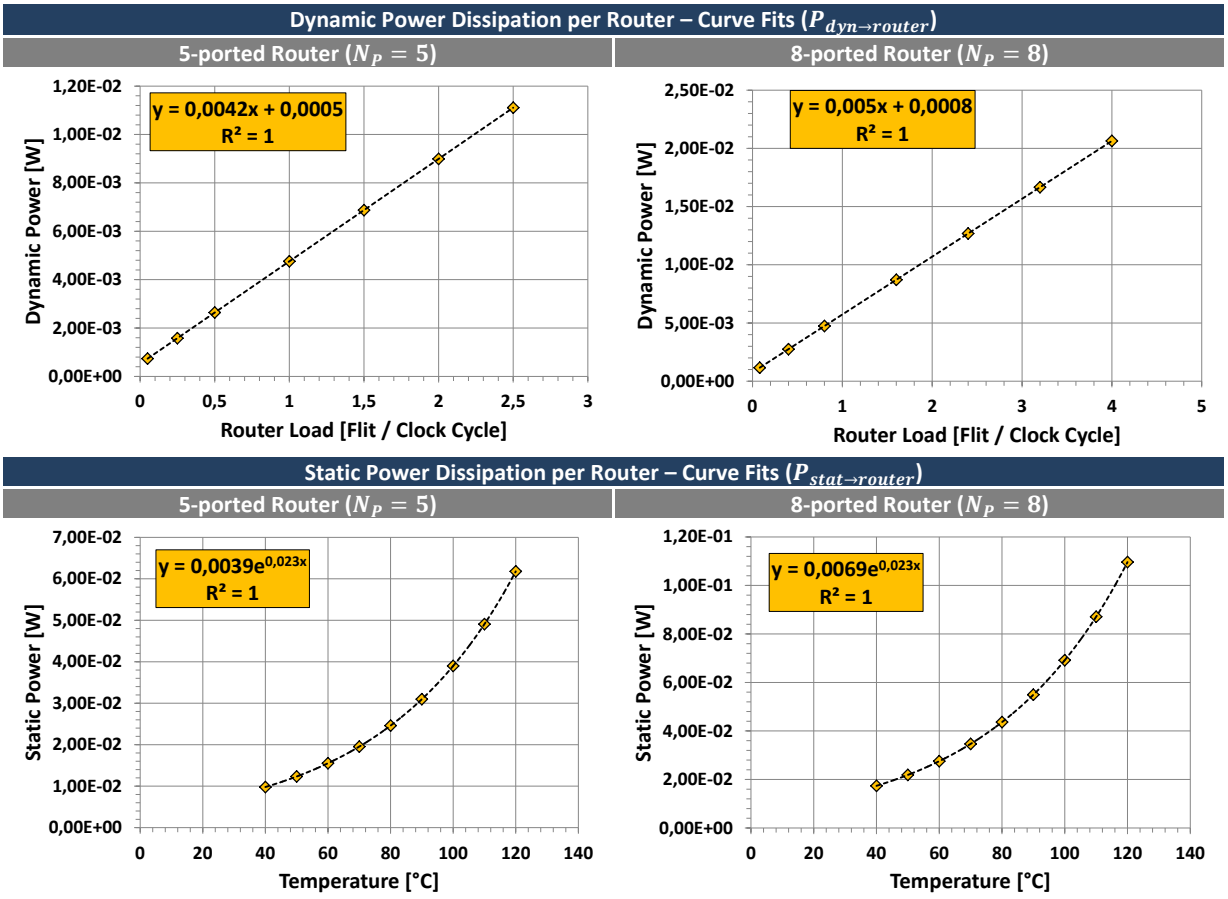
Curve Fit



Thus, the utilized linear equation for the operating temperature (T_{router}) of a router as function of its data load ($r_{flit \rightarrow router}$) is:

$$T_{router} = f\left(r_{flit \rightarrow router}\right) = 72{,}608 \cdot r_{flit \rightarrow router} + 59{,}476 \quad in\ [°C]$$

This equation is applied to the configurations of the 5-ported as well as the 8-ported router, which are used for the simulations in this work (see Table I.2 at p. 164). Additionally, the functional dependencies of the dynamic/static power dissipation on the router load/temperature are evaluated via the DSENT [89] tool. The resulting curve fits for both configurations are as follows:



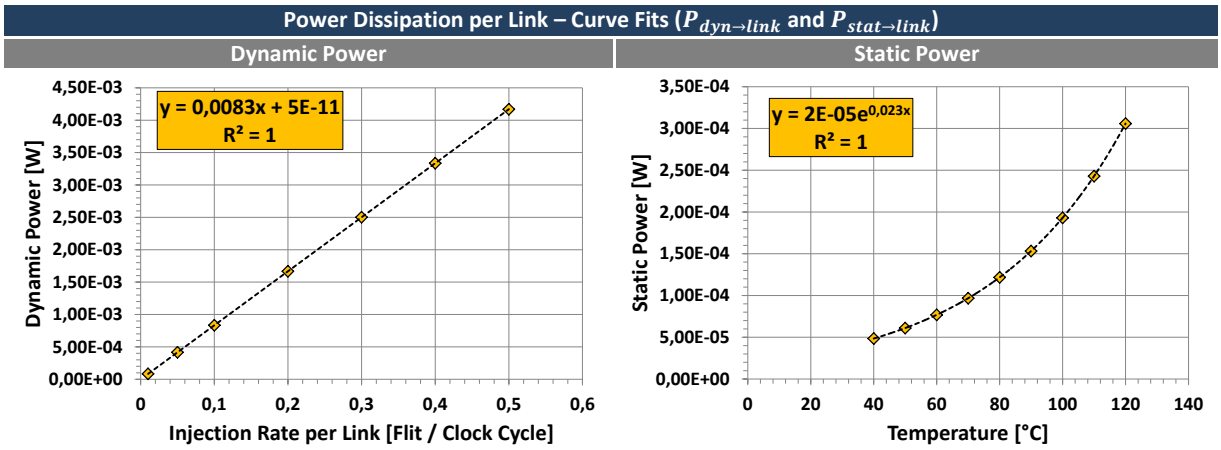
The linear function for the dynamic power dissipations of both router configurations is independent of the operating-temperature and as follows:

$$P_{dyn \rightarrow router} = f\left(r_{flit \rightarrow router}\right) = \begin{cases} 0{,}0042 \cdot r_{flit \rightarrow router} + 0{,}0005 & N_P = 5 \\ 0{,}005 \cdot r_{flit \rightarrow router} + 0{,}0008 & N_P = 8 \end{cases} \quad in\ [W]$$

The static power dissipation is affected by the data load due to its exponential dependency on the operating-temperature. Thus, as preliminary step the operating temperature must be calculated and afterwards the static power dissipation for both router configurations can be evaluated via the following equation:

$$P_{stat \rightarrow router} = f\left(T_{router}\right) = \begin{cases} 0{,}0039 \cdot e^{0{,}023 \cdot T_{router}} & N_P = 5 \\ 0{,}0069 \cdot e^{0{,}023 \cdot T_{router}} & N_P = 8 \end{cases} \quad in\ [W]$$

The functions for the dynamic ($P_{dyn\rightarrow link}$) and static ($P_{stat\rightarrow link}$) power dissipation of a link (for specifications see Section 2.1.1 and setup table at p. 164) are obtained via the same flow. Thereby, link itself has no dedicated temperature model, because of its heat conductor properties. It is assumed that links have the same operating temperature as the connected router. The resulting curve fits and equations are as follows:



$$P_{dyn\rightarrow link} = f\left(r_{flit\rightarrow link}\right) = 0{,}0083 \cdot r_{flit\rightarrow router} + 5 \cdot 10^{-11}$$

$$P_{stat\rightarrow link} = f(T_{router}) = (2 \cdot 10^{-5}) \cdot e^{0{,}023 \cdot T_{router}}$$

The simulators, used in the work at hand, produce traffic statistics for each router and link inside the simulated system setups. Hence, the post-processing for the power evaluation parses through these statistic files and performs the evaluation on each router i ($1 \leq i \leq N_{router}$) and its connected links $N_{links}(i)$. The results are summed up as follows:

$$P_{total\rightarrow router} = \sum_{i=1}^{N_{router}} \left[P_{dyn\rightarrow router}(i) + P_{stat\rightarrow router}(i) \right]$$

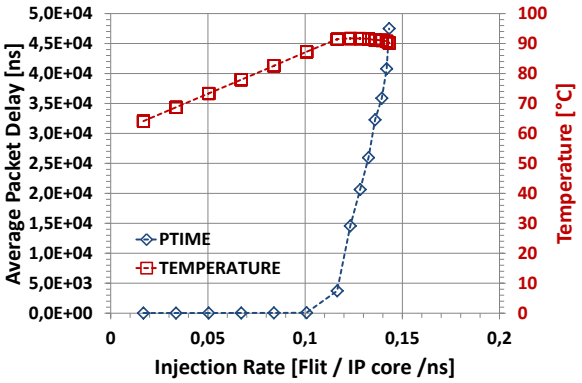
$$P_{total\rightarrow links} = \sum_{i=1}^{N_{router}} \left[N_{links}(i) \cdot P_{dyn\rightarrow link}(i) + N_{links}(i) \cdot P_{stat\rightarrow link}(i) \right]$$

$$P_{total\rightarrow NoC} = P_{total\rightarrow router} + P_{total\rightarrow links}$$

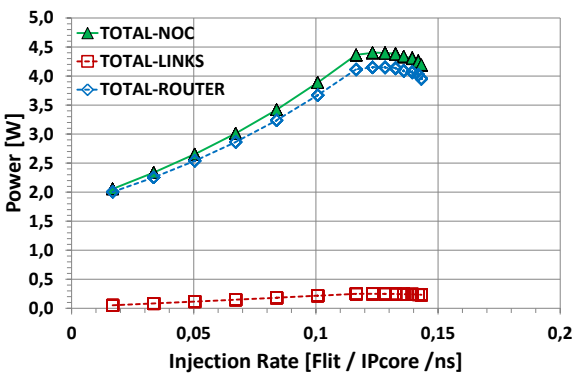
Below the results of the post-processed simulation statistics for an exemplary system setup are illustrated. The left plot shows the curve of the average operating-temperature in combination with the average packet header delay (PTIME). The right plot illustrates the corresponding curve of the power dissipation inside the NoC and its fractions resulting from the links/routers.

Exemplary case of an 8x8 NoC with Nearest Neighbor Pattern (λ_{NN} = 0.2)

Avrg. Router Termperature vs. Packet Delay



NoC Power Composition



I.D TRAFFIC STATISTICS EVALUATION METHOD

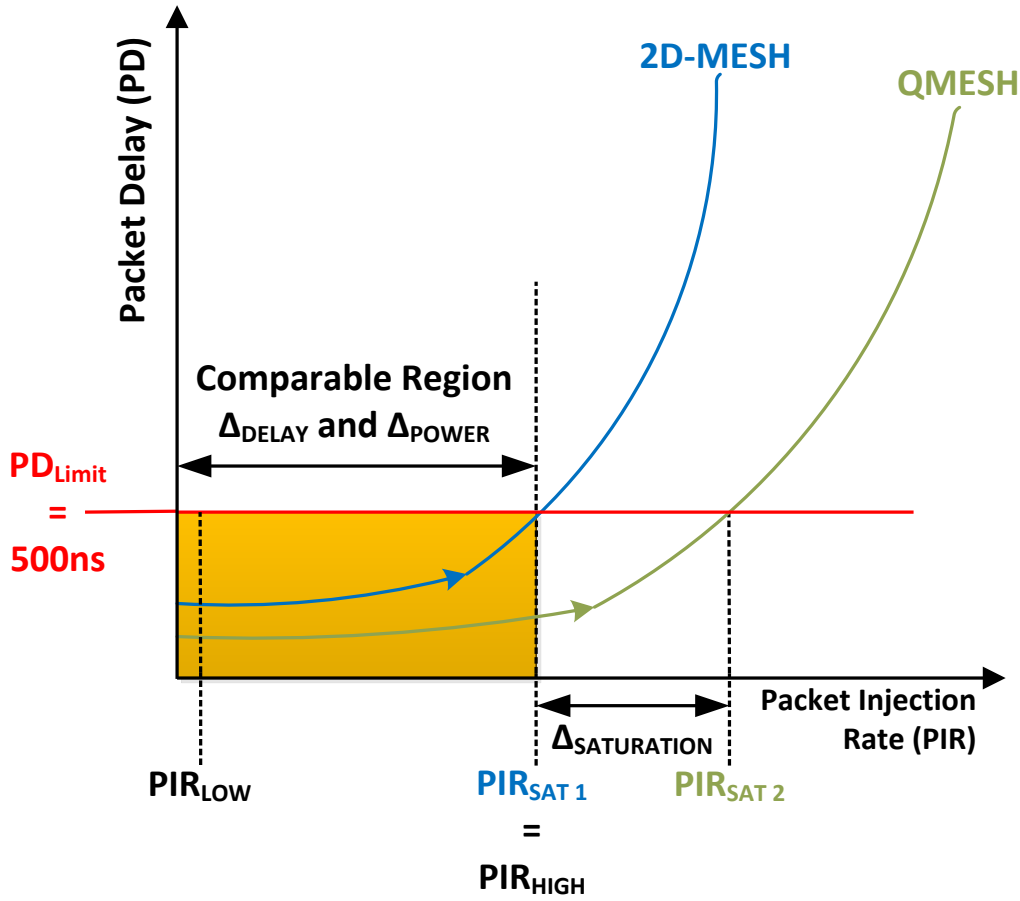
The evaluation of the gathered traffic statistics during the performed simulation runs in the work at hand is realized via the methodology as described below. During the simulation runs the ETE packet header delay ($t_{path \rightarrow header,p}$) for all packets that traversed the NoC and the total number of packets ($N_{packets}$) are logged. Thus, the mean packet header delay ($\bar{t}_{packet \rightarrow NoC}$) for each performed simulation run is computed as follows:

$$\bar{t}_{packet \rightarrow NoC} = \frac{1}{N_{packets}} \cdot \sum_{p=1}^{N_{packets}} t_{path \rightarrow header,p}$$

Furthermore, the mean packet injection rate per resource tile ($\bar{r}_{packet \rightarrow NoC}$) is calculated for each performed simulation run (using the basic Equation 2.16 from the introduction of Section 2.1 at p. 40). Therefore, in addition to the total number of packets ($N_{packets}$) the total simulation time (t_{sim}) and the number of resource tiles (N_{tile}) are used as follows:

$$\bar{r}_{packet \rightarrow NoC} = \frac{N_{packets}}{t_{sim} \cdot N_{tile}}$$

Traffic statistics evaluation of SystemC-based NoC simulations



The collection of these statistic results from all simulations runs for specific NoC configurations are used to generate the characteristic delay plots that depict the mean packet header delay (PD=$\bar{t}_{packet \rightarrow NoC}$) as function of the mean packet injection rate per resource tile (PIR = $\bar{r}_{packet \rightarrow NoC}$). Additionally, these packet delay curves are utilized for the comparison of different NoC approaches. First, a packet header delay limit (PD_{Limit}) is specified as equally defined entry point for the NoCs into the saturated traffic region (see Figure above). For the evaluation in the work at hand this delay limit is fixed at 500 ns. This is illustrated above for the comparison of two curves (2D-Mesh and QMesh). For both curves the break-even point with the PD_{Limit} is determined via linear interpolation between the nearest lower/higher values of each curve. Afterwards, the corresponding packet injection rates ($r_{packet,1}^{PD_{Limit}}$ at PIR_{SAT1} and $r_{packet,2}^{PD_{Limit}}$ at PIR_{SAT2}) are calculated to determine the relative change of the saturation level ($\Delta_{Saturation}$) for both NoC approaches as follows:

$$\Delta_{\boldsymbol{Saturation}} = \frac{r_{packet,2}^{PD_{Limit}} - r_{packet,1}^{PD_{Limit}}}{r_{packet,1}^{PD_{Limit}}} \cdot 100 \quad in \quad [\%]$$

Second, the power dissipation as well as the packet delay characteristics have to be compared between different NoC approaches. Therefore, the curve with the lowest saturation level constraints the comparable region for the NoCs with its break-even point to the PD_{Limit}. Inside, this region the packet delay curve segments and the power curve segments are approximated via the application of curve fitting ($f_{delayfit\left(r_{packet}^{p}\right)}^{1,2}$ and $f_{power\left(r_{packet}^{p}\right)}^{1,2}$). Afterwards, the mean value of the relative changes along these segment fits are calculated for both parameter (Δ_{Delay} and Δ_{Power}) over a set of equidistant packet injection rates inside the comparable region as defined below:

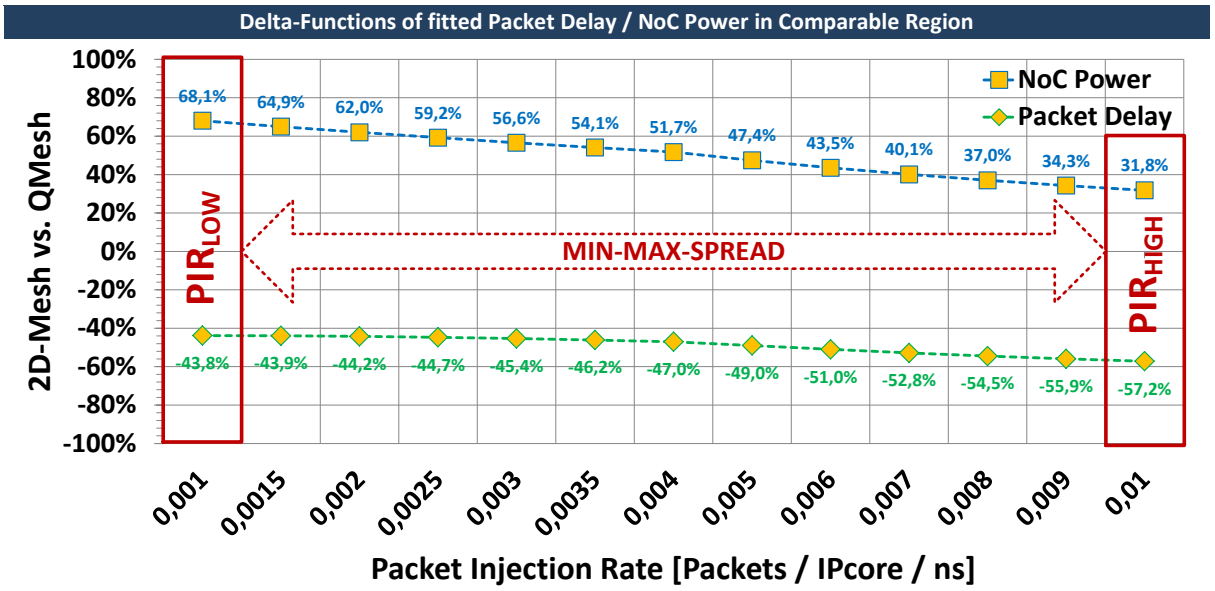
$$\Delta_{\boldsymbol{Delay}} = \frac{100}{N_{rv}} \cdot \sum_{p=1}^{N_{rv}} \left(\frac{f_{delayfit\left(r_{packet}^{p}\right)}^{2} - f_{delayfit\left(r_{packet}^{p}\right)}^{1}}{f_{delayfit\left(r_{packet}^{p}\right)}^{1}} \right) \quad in \quad [\%]$$

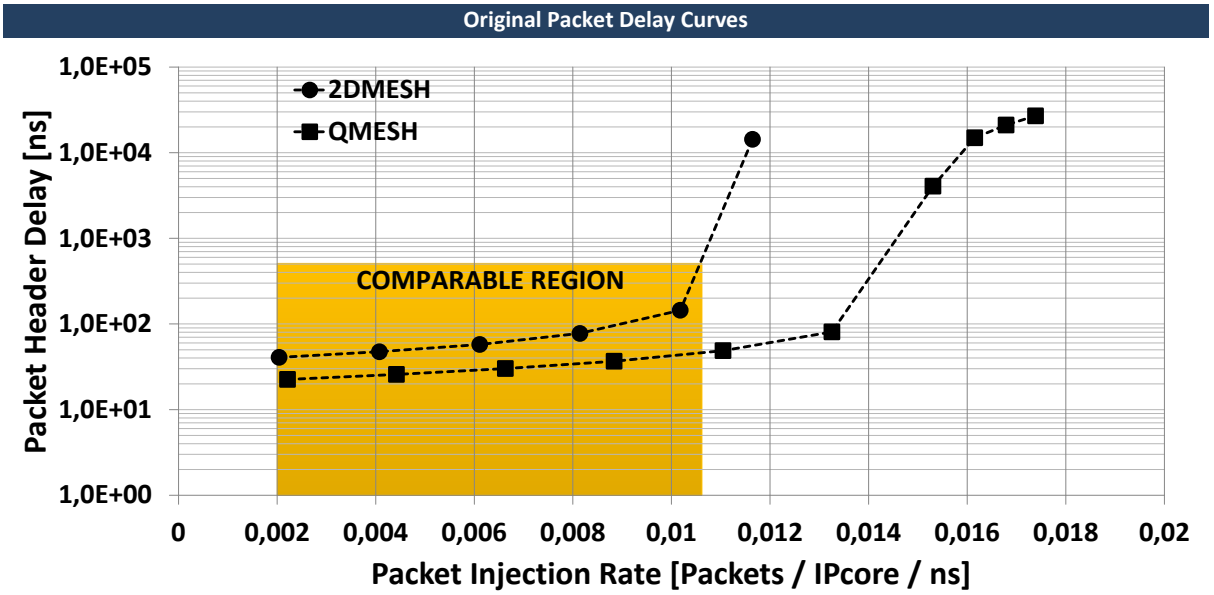
$$\Delta_{\boldsymbol{Power}} = \frac{100}{N_{rv}} \cdot \sum_{p=1}^{N_{rv}} \left(\frac{f_{powerfit\left(r_{packet}^{p}\right)}^{2} - f_{powerfit\left(r_{packet}^{p}\right)}^{1}}{f_{powerfit\left(r_{packet}^{p}\right)}^{1}} \right) \quad in \quad [\%]$$

$$with \quad \Delta r = \frac{r_{compare}^{max}}{N_{rv}} \quad \Longrightarrow \quad r_{packet}^{p} = p \cdot \Delta r \quad and \quad r_{packet}^{p} \leq r_{compare}^{max}$$

The progress of the delta-functions along the comparable region is given in the plot below. The results are obtained from a SystemC-based simulation of an 8x8 NoC with 2D-Mesh as well as unmanaged QMesh under nearest-neighbor traffic with a NN-fraction of 20%. Along these curves the mean value computation for Δ_{Delay} and Δ_{Power}, as introduced above, is performed. Positive changes represent additional efforts/costs/delay introduced by the QMesh, while negative numbers

are savings. As the example shows, the NoC power is increased and the packet delay reduced by the QMesh. The mean value computation along the comparable region considers the case that the NoC utilizes the full packet injection range with a random uniform distribution. With increasing packet injection rates the relative power overhead of the QMesh shrinks, while the relative packet delay savings increase (→ compare PIR_{LOW} to PIR_{HIGH} values).

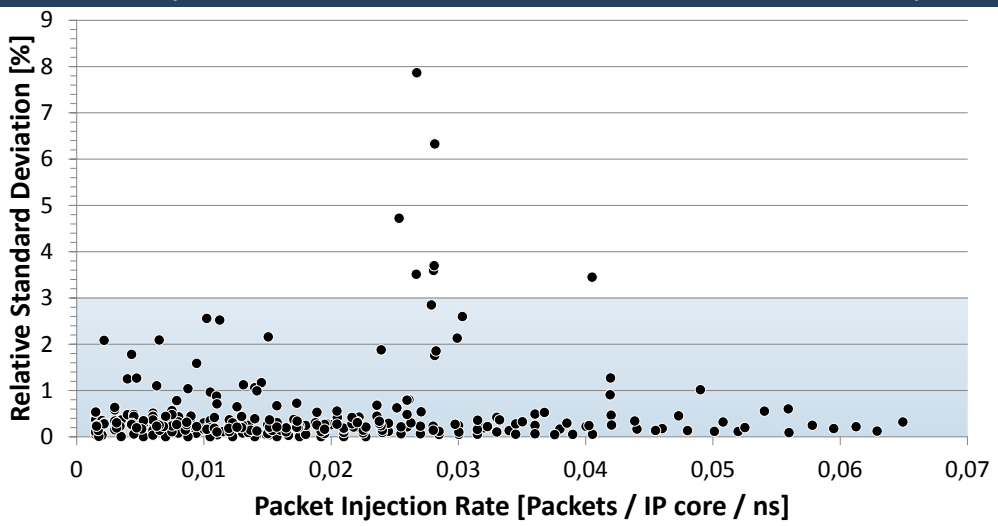




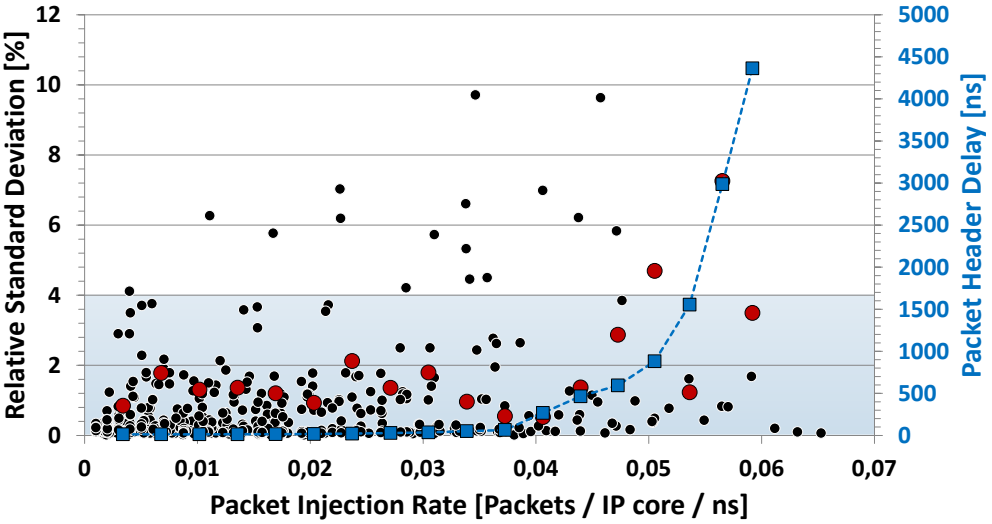
I.E ADDITIONAL PLOTS

This sub-section of the contains all additional plots from the experimental evaluation of the approaches presented in the work at hand. They depict the resulting standard deviation over the 10 performed simulation runs relative to the calculated mean packet delay for each parameter setup and each adjusted packet injection rate. It becomes obvious that the resulting variation of the results is low and below 10% at all. Thereby, spikes up to 4-10% occur when the simulated NoCs reach the network saturation (see additional packet delay plot in 8x8 with red marked standard deviation values as examples). But these regions are not considered for the aggregated results along the comparable regions at all.

Standard Deviation of SystemC-based Simulation Results relative to the Mean Packet Header Delays → 4x4 NoCs



Standard Deviation of SystemC-based Simulation Results relative to the Mean Packet Header Delays → 8x8 NoCs



II THESIS BIBLIOGRAPHY

[1] S. Borkar and A. A. Chien, "The future of microprocessors," *Commun. ACM*, vol. 54, no. 5, p. 67, May 2011.

[2] "International Roadmap for Semiconductors (ITRS)," 2012. [Online]. Available: http://www.itrs.net/Links/2012ITRS/Home2012.htm.

[3] T. Zidenberg, I. Keslassy, and U. Weiser, "Multi-Amdahl: Optimal Resource Sharing with Multiple Program Execution Segments," Haifa, Israel, 2011.

[4] S. Borkar, "Thousand core chips: a technology perspective," in *DAC 2007: Proceedings of the 44th annual Design Automation Conference*, 2007, pp. 746–749.

[5] S. Williams, A. Waterman, and D. Patterson, "Roofline: An Insightful Visual Performance Model for Multicore Architectures," *Commun. ACM*, vol. 52, no. 4, p. 65, Apr. 2009.

[6] E. Bolotin, I. Cidon, R. Ginosar, and A. Kolodny, "Cost considerations in network on chip," *Integr. VLSI J.*, vol. 38, no. 1, pp. 19–42, 2004.

[7] W. J. Dally and B. P. Towles, *Principles and Practices of Interconnection Networks*. Morgan Kaufmann Publishers, 2004.

[8] W. J. Dally and B. Towles, "Route packets, not wires: on-chip interconnection networks," in *Proceedings of the 38th Design Automation Conference*, 2001, pp. 684–689.

[9] G. De Micheli and D. Bertozzi, *Networks on Chips: Technology and Tools*. Morgan Kaufmann Publishers, 2006.

[10] J. Duato, S. Yalamanchili, and L. Ni, *Interconnection Networks: An Engineering Approach*. Morgan Kaufmann Publishers, 2003.

[11] D. L. Greenfield, "Communication Locality in Computation : Software, Chip Multiprocessors and Brains - Dissertation," University of Cambridge, 2010.

[12] G. B. P. Bezerra, "Energy Consumption in Networks on Chip : Efficiency and Scaling - Dissertation," University of New Mexico, 2012.

[13] D. Sanchez, "An analysis of on-chip interconnection networks for large-scale chip multiprocessors," *ACM Trans. Archit. Code Optim.*, vol. 7, no. 1, pp. 1–28, 2010.

[14] R. Manevich, I. Cidon, and A. Kolodny, "Handling global traffic in future CMP NoCs," in *Proceedings of the International Workshop on System Level Interconnect Prediction - SLIP '12*, 2012, pp. 40–48.

[15] M. Taylor, J. Kim, J. Miller, D. Wentzlaff, F. Ghodrat, B. Greenwald, H. Hoffman, P. Johnson, W. Lee, A. Ma, A. Saraf, M. Seneski, N. Shnidman, V. Strumpen, M. Frank, S. Amarasinghe, and A. Agarwal, "The Raw microprocessor: a computational fabric for software circuits and general-purpose programs," *IEEE Micro*, vol. 22, no. 2, pp. 25–35, Mar. 2002.

[16] P. Salihundam, S. Jain, and T. Jacob, "A 2 Tb/s 6× 4 mesh network for a single-chip cloud computer with DVFS in 45 nm CMOS," *IEEE J. Solid-State Circuits*, vol. 46, no. 4, pp. 757–766, 2011.

[17] A. K. Singh, M. Shafique, A. Kumar, and J. Henkel, "Mapping on multi/many-core systems," in *Proceedings of the 50th Annual Design Automation Conference on - DAC '13*, 2013, p. 1.

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

[19] Y. Jin, E. J. Kim, and T. M. Pinkston, "Communication-Aware Globally-Coordinated On-Chip Networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 23, no. 2, pp. 242–254, Feb. 2012.

[20] J. Balfour and W. J. Dally, "Design tradeoffs for tiled CMP on-chip networks," in *Proceedings of the 20th annual international conference on Supercomputing - ICS '06*, 2006, p. 187.

[21] D. Sanchez, G. Michelogiannakis, and C. Kozyrakis, "An analysis of on-chip interconnection networks for large-scale chip multiprocessors," *ACM Trans. Archit. Code Optim.*, vol. 7, no. 1, pp. 1–28, 2010.

[22] P. K. Sahu and S. Chattopadhyay, "A survey on application mapping strategies for Network-on-Chip design," *J. Syst. Archit.*, vol. 59, no. 1, pp. 60–76, Jan. 2013.

[23] S. R. Vangal, J. Howard, G. Ruhl, S. Dighe, H. Wilson, J. Tschanz, D. Finan, A. Singh, T. Jacob, S. Jain, V. Erraguntla, C. Roberts, Y. Hoskote, N. Borkar, and S. Borkar, "An 80-Tile Sub-100-W TeraFLOPS Processor in 65-nm CMOS," *IEEE J. Solid-State Circuits*, vol. 43, no. 1, pp. 29–41, Jan. 2008.

[24] G. P. Nychis, C. Fallin, T. Moscibroda, O. Mutlu, and S. Seshan, "On-chip networks from a networking

perspective," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 42, no. 4, pp. 407–418, Sep. 2012.

[25] E. Salminen, "On Design and Comparison of On-Chip Networks - Dissertation," Tampere University of Technology, 2010.

[26] K. Sankaralingam, R. Nagarajan, H. Liu, C. Kim, J. Huh, D. Burger, S. W. Keckler, and C. R. Moore, "Exploiting ILP, TLP, and DLP with the polymorphous TRIPS architecture," *Proc. 30th Annu. Int. Symp. Comput. Archit. - ISCA '03*, p. 422, 2003.

[27] Y. Nishikawa, "A Study of Interconnection Network for Many-Core Processors Based on Traffic Analysis - Dissertation," Keio University, 2011.

[28] A. Agarwal, C. Iskander, H. T. Multisystems, and R. Shankar, "Survey of Network on Chip (NoC) Architectures and Contributions," *J. Eng. Comput. Archit.*, vol. 3, no. 1, pp. 1–15, 2009.

[29] J. Howard, S. Dighe, S. R. 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. K. 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," *IEEE J. Solid-State Circuits*, vol. 46, no. 1, pp. 173–183, Jan. 2011.

[30] T. Bjerregaard and S. Mahadevan, "A survey of research and practices of network-on-chip," *ACM Comput. Surv.*, vol. 38, no. 1, p. 1, 2006.

[31] 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*, vol. 27, no. 5, pp. 15–31, Sep. 2007.

[32] N. Weste and D. Harris, *CMOS VLSI design: a circuits and systems perspective*, 4th ed. Boston, MA, USA: Addison-Wesley, 2011.

[33] M. B. Taylor, "A Landscape of the New Dark Silicon Design Regime," *IEEE Micro*, vol. 33, no. 5, pp. 8–19, Sep. 2013.

[34] H. Esmaeilzadeh, E. Blem, R. St. Amant, K. Sankaralingam, and D. Burger, "Power challenges may end the multicore era," *Commun. ACM*, vol. 56, no. 2, pp. 93–102, Feb. 2013.

[35] H. Esmaeilzadeh, T. Cao, X. Yang, S. M. Blackburn, and K. S. McKinley, "Looking back and looking forward: Power, Performance, and Upheaval," *Commun. ACM*, vol. 55, no. 7, p. 105, Jul. 2012.

[36] M. White, "A Study of Nanometer Semiconductor Scaling Effects on Microelectronics Reliability - Dissertation," University of Maryland, 2009.

[37] W. Huang, "HotSpot — A Chip and Package Compact Thermal Modeling Methodology for VLSI Design - Dissertation," University of Virginia, 2007.

[38] H. Esmaeilzadeh, E. Blem, R. St. Amant, K. Sankaralingam, and D. Burger, "Dark Silicon and the End of Multicore Scaling," *IEEE Micro*, vol. 32, no. 3, pp. 122–134, May 2012.

[39] M. Radetzki, C. Feng, X. Zhao, and A. Jantsch, "Methods for Fault Tolerance in Networks on Chip," *it.kth.se*, vol. V, no. January, pp. 1–36, 2013.

[40] S. Murali, *Designing Reliable and Efficient Networks on Chips*, 1st ed., vol. 34. Dordrecht: Springer Netherlands, 2009.

[41] S. Mitra, K. Brelsford, Y. M. Kim, H.-H. K. Lee, and Y. Li, "Robust System Design to Overcome CMOS Reliability Challenges," *IEEE J. Emerg. Sel. Top. Circuits Syst.*, vol. 1, no. 1, pp. 30–41, Mar. 2011.

[42] S. Mitra, "Robust System Design," in *23rd International Conference on VLSI Design*, 2010, pp. 434–439.

[43] K. Aisopos, "Fault Tolerant Architectures for On-Chip Networks - Dissertation," Princeton University, 2012.

[44] J. Choi and R. Vuduc, "A roofline model of energy," in *Technical Report # GT-CSE-2012-01*, 2012, pp. 1–53.

[45] M. D. Hill and M. R. Marty, "Amdahl's Law in the Multicore Era," *Computer (Long. Beach. Calif).*, vol. 41, no. 7, pp. 33–38, Jul. 2008.

[46] J. L. Gustafson, "Reevaluating Amdahl's law," *Commun. ACM*, vol. 31, no. 5, pp. 532–533, May 1988.

[47] A. S. Cassidy and A. G. Andreou, "Beyond Amdahl's Law: An Objective Function That Links Multiprocessor Performance Gains to Delay and Energy," *IEEE Trans. Comput.*, vol. 61, no. 8, pp. 1110–1126, Aug. 2012.

[48] S. Eyerman and L. Eeckhout, "Modeling critical sections in Amdahl's law and its implications for multicore design," *ACM SIGARCH Comput. Archit. News*, vol. 38, no. 3, p. 362, Jun. 2010.

[49] L. Yavits, A. Morad, and R. Ginosar, "The Effect of Communication and Synchronization on Amdahl Law in Multicore Systems," Haifa, Israel, 2012.

[50] A. Hartstein, V. Srinivasan, T. Puzak, and P. Emma, “On the nature of cache miss behavior: Is it√ 2 ?,” *J. Instr. Parallelism*, vol. 10, pp. 1–22, 2008.

[51] B. M. Rogers, A. Krishna, G. B. Bell, K. Vu, X. Jiang, and Y. Solihin, “Scaling the bandwidth wall,” *ACM SIGARCH Comput. Archit. News*, vol. 37, no. 3, pp. 371–382, Jun. 2009.

[52] H. Attiya, R. Guerraoui, D. Hendler, P. Kuznetsov, M. M. Michael, and M. Vechev, “Laws of order: Expensive Synchronization in Concurrent Algorithms Cannot be Eliminated,” *ACM SIGPLAN Not.*, vol. 46, no. 1, pp. 487–498, Jan. 2011.

[53] A. K. Mishra, O. Mutlu, and C. R. Das, “A heterogeneous multiple network-on-chip design: an application-aware approach,” in *Proceedings of the 50th Annual Design Automation Conference on - DAC ’13*, 2013, pp. 1–10.

[54] L. Chen, K. Hwang, and T. Pinkston, “RAIR: Interference Reduction in Regionalized Networks on Chip,” in *27th IEEE International Parallel & Distributed Processing Symposium*, 2013, pp. 1–12.

[55] U. Y. Ogras, J. Hu, and R. Marculescu, “Key research problems in NoC design: a holistic perspective,” *Proc. 3rd IEEE/ACM/IFIP Int. Conf. Hardware/software codesign Syst. Synth.*, pp. 69–74, 2005.

[56] R. Marculescu, U. Y. Ogras, L.-S. Peh, N. E. Jerger, and Y. Hoskote, “Outstanding Research Problems in NoC Design: System, Microarchitecture, and Circuit Perspectives,” *IEEE Trans. Comput. Des. Integr. Circuits Syst.*, vol. 28, no. 1, pp. 3–21, Jan. 2009.

[57] S. Garg and D. Marculescu, “Technology-driven limits on DVFS controllability of multiple voltage-frequency island designs: a system-level perspective,” in *Proceedings of 46th ACM/IEEE Design Automation Conference (DAC’09)*, 2009, pp. 818 – 821.

[58] W. Jang, D. Ding, and D. Z. Pan, “Voltage and frequency island optimizations for many-core/networks-on-chip designs,” in *The 2010 International Conference on Green Circuits and Systems*, 2010, pp. 217–220.

[59] G. Kornaros and D. Pnevmatikatos, “A survey and taxonomy of on-chip monitoring of multicore systems-on-chip,” *ACM Trans. Des. Autom. Electron. Syst.*, vol. 18, no. 2, pp. 1–38, Mar. 2013.

[60] R. Manevich, I. Cidon, A. Kolodny, I. Walter, and S. Wimer, “A Cost Effective Centralized Adaptive Routing for Networks-on-Chip,” *2011 14th Euromicro Conf. Digit. Syst. Des.*, vol. 9, no. 2, pp. 39–46, Aug. 2011.

[61] P. Gratz, B. Grot, and S. W. Keckler, “Regional congestion awareness for load balance in networks-on-chip,” in *2008 IEEE 14th International Symposium on High Performance Computer Architecture*, 2008, pp. 203–214.

[62] S. Ma, N. Enright Jerger, and Z. Wang, “DBAR: An Efficient Routing Algorithm to Support Multiple Concurrent Applications in Networks-on-Chip,” in *Proceeding of the 38th annual international symposium on Computer architecture - ISCA ’11*, 2011, p. 413.

[63] M. Ebrahimi, M. Daneshtalab, P. Liljeberg, and J. Plosila, “CATRA-Congestion Aware Trapezoid-based Routing Algorithm for On-Chip Networks,” in *Design, Automation & Test in Europe Conference & Exhibition (DATE’12)*, 2012, pp. 320 – 325.

[64] M. Ebrahimi, M. Daneshtalab, P. Liljeberg, J. Plosila, and H. Tenhunen, “LEAR -- A Low-Weight and Highly Adaptive Routing Method for Distributing Congestions in On-chip Networks,” in *2012 20th Euromicro International Conference on Parallel, Distributed and Network-based Processing*, 2012, vol. 1, pp. 520–524.

[65] W. Kim, M. S. Gupta, G.-Y. Wei, and D. Brooks, “System level analysis of fast, per-core DVFS using on-chip switching regulators,” in *2008 IEEE 14th International Symposium on High Performance Computer Architecture*, 2008, pp. 123–134.

[66] S. Eyerman and L. Eeckhout, “Fine-grained DVFS using on-chip regulators,” *ACM Trans. Archit. Code Optim.*, vol. 8, no. 1, pp. 1–24, Apr. 2011.

[67] W. Heirman, J. Dambre, D. Stroobandt, and J. Van Campenhout, “Rent’s Rule and Parallel Programs: Characterizing Network Traffic Behavior,” in *Proceedings of the tenth international workshop on System level interconnect prediction - SLIP ’08*, 2008, p. 87.

[68] P. Gorski, C. Cornelius, D. Timmermann, and V. Kühn, “RedNoCs: A Runtime Configurable Solution for Cluster-based and Multi-objective System Management in Networks-on-Chip,” in *Eighth International Conference on Systems (ICONS’13)*, 2013, no. c, pp. 192–201.

[69] P. Gorski, T. Wegner, and D. Timmermann, “Evaluation of a software-based centralized Traffic Management inside run-time reconfigurable regions-of-interest of a mesh-based Network-on-Chip topology,” in *18th Workshop on Methoden und Beschreibungssprachen zur Modellierung und*

Verifikation von Schaltungen und Systemen (MBMV 2015), 2015, pp. 1–10.

[70] P. Gorski, T. Wegner, and D. Timmermann, "Centralized and software-based run-time traffic management inside configurable regions of interest in mesh-based Networks-on-Chip," in *11th International Symposium on Applied Reconfigurable Computing (ARC 2015)*, 2015, pp. 1–12.

[71] P. Gorski, T. Wegner, and D. Timmermann, "Joint consideration of performance, reliability and fault tolerance in regular Networkson-Chip via multiple spatially-independent interface terminals," in *27th Workshop on Testmethoden und Zuverlässigkeit von Schaltungen und Systemen (TuZ 2015)*, 2015, pp. 1–4.

[72] P. Gorski, C. Cornelius, D. Timmermann, and V. Kühn, "Centralized Adaptive Source-Routing for Networks-on-Chip as HW/SW-Solution with Cluster-based Workload Isolation," in *8th International Conference on Systems (ICONS'13)*, 2013, no. c, pp. 207–215.

[73] P. Gorski and D. Timmermann, "Centralized traffic monitoring for online-resizable clusters in Networks-on-Chip," in *2013 8th International Workshop on Reconfigurable and Communication-Centric Systems-on-Chip (ReCoSoC)*, 2013, pp. 1–8.

[74] E. Salminen, A. Kulmala, and T. D. Hamalainen, "On network-on-chip comparison," in *10th Euromicro Conference on Digital System Design Architectures, Methods and Tools (DSD 2007)*, 2007, pp. 503–510.

[75] E. Salminen, A. Kulmala, and T. D. Hämäläinen, "Survey of Network-on-chip Proposals," *WHITE Pap. OCP-IP, MARCH*, no. March, pp. 1–12, 2008.

[76] N. Barrow-Williams, C. Fensch, and S. Moore, "A communication characterisation of Splash-2 and Parsec," in *2009 IEEE International Symposium on Workload Characterization (IISWC)*, 2009, pp. 86–97.

[77] A. Danalis, G. Marin, C. McCurdy, J. S. Meredith, P. C. Roth, K. Spafford, V. Tipparaju, and J. S. Vetter, "The Scalable Heterogeneous Computing (SHOC) benchmark suite," in *Proceedings of the 3rd Workshop on General-Purpose Computation on Graphics Processing Units - GPGPU '10*, 2010, p. 63.

[78] S. Che, M. Boyer, J. Meng, D. Tarjan, J. W. Sheaffer, S. H. Lee, and K. Skadron, "Rodinia: A benchmark suite for heterogeneous computing," in *Proceedings of the IEEE*, 2009, vol. 2009, no. c.

[79] U. Y. Ogras, R. Marculescu, D. Marculescu, and E. G. Jung, "Design and Management of Voltage-Frequency Island Partitioned Networks-on-Chip," *IEEE Trans. Very Large Scale Integr. Syst.*, vol. 17, no. 3, pp. 330–341, Mar. 2009.

[80] C. H. Luz, "Addressing Manufacturing Challenges in NoC-based ULSI Designs - Dissertation," Universitat Politecnica de Valencia, 2012.

[81] D. Ludovici, "Technology Aware Network-on-Chip Connectivity and Synchronization Design - Dissertation," Delft University of Technology, 2011.

[82] F. G. Villamon, "Design Space Exploration for Networks On-chip - Dissertation," Universitat Politècnica de València, 2011.

[83] P. Lotfi-Kamran, E. Ozer, B. Falsafi, B. Grot, M. Ferdman, S. Volos, O. Kocberber, J. Picorel, A. Adileh, D. Jevdjic, and S. Idgunji, "Scale-out processors," *ACM SIGARCH Comput. Archit. News*, vol. 40, no. 3, p. 500, Sep. 2012.

[84] C. Cornelius, "Design of complex integrated systems based on networks-on-chip - Trading off performance, power and reliability - Dissertation," University of Rostock, 2010.

[85] S. S. Majzoub, R. A. Saleh, S. J. E. Wilton, and R. K. Ward, "Energy Optimization for Many-Core Platforms: Communication and PVT Aware Voltage-Island Formation and Voltage Selection Algorithm," *IEEE Trans. Comput. Des. Integr. Circuits Syst.*, vol. 29, no. 5, pp. 816–829, May 2010.

[86] A.-M. Rahmani-Sane, "Exploration and Design of Power-Efficient Networked Many-Core Systems - Dissertation," University of Turku, 2012.

[87] A. Das, "Microarchitectural Approaches for Optimizing Power and Profitability in Multi-core Processors - Dissertation," Northwestern University, Illinois, 2010.

[88] A. R. Perez, "Floorplan-Aware High Performance NoC Design - Dissertation," Polytechnic University of Valencia, 2012.

[89] C. Sun, C.-H. O. Chen, G. Kurian, L. Wei, J. Miller, A. Agarwal, L.-S. Peh, and V. Stojanovic, "DSENT - A Tool Connecting Emerging Photonics with Electronics for Opto-Electronic Networks-on-Chip Modeling," in *2012 IEEE/ACM Sixth International Symposium on Networks-on-Chip*, 2012, pp. 201–210.

[90] L.-S. Peh and W. J. Dally, "A delay model and speculative architecture for pipelined routers," in *Proceedings HPCA Seventh International Symposium on High-Performance Computer Architecture*, 2001, pp. 255–266.

[91] W. J. Dally, "A delay model for router microarchitectures," *IEEE Micro*, vol. 21, no. 1, pp. 26–34, 2001.

[92] A. Kahng, B. Li, L. 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 (DATE'09)*, 2009, pp. 423–428.

[93] D. U. Becker and W. J. Dally, "Allocator implementations for network-on-chip routers," in *Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis - SC '09*, 2009, p. 1.

[94] Z. Guz, I. Walter, E. Bolotin, I. Cidon, R. Ginosar, and A. Kolodny, "Network Delays and Link Capacities in Application-Specific Wormhole NoCs," *VLSI Des.*, vol. 2007, pp. 1–15, 2007.

[95] Y. Ben-Itzhak, I. Cidon, and A. Kolodny, "Delay Analysis of Wormhole Based Heterogeneous NoC," in *5th IEEE/ACM International Symposium on Networks on Chip (NoCS'11)*, 2011, pp. 161–168.

[96] E. S. Shin, V. J. Mooney, and G. F. Riley, "Round-robin arbiter design and generation," *Proc. 15th Int. Symp. Syst. Synth. - ISSS '02*, p. 243, 2002.

[97] T. Mak, P. Y. K. Cheung, K.-P. Lam, and W. Luk, "Adaptive Routing in Network-on-Chips Using a Dynamic-Programming Network," *IEEE Trans. Ind. Electron.*, vol. 58, no. 8, pp. 3701–3716, Aug. 2011.

[98] M. Palesi, R. Holsmark, S. Kumar, and V. Catania, "Application Specific Routing Algorithms for Networks on Chip," *IEEE Trans. Parallel Distrib. Syst.*, vol. 20, no. 3, pp. 316–330, Mar. 2009.

[99] A. Kahng, B. Lin, and S. Nath, "Comprehensive modeling methodologies for NoC router estimation," *IEEE Trans. Comput. Des. Integr. Circuits Syst.*, pp. 1–20, 2012.

[100] A. B. Kahng, B. Lin, and S. Nath, "Explicit modeling of control and data for improved NoC router estimation," in *Proceedings of the 49th Annual Design Automation Conference on - DAC '12*, 2012, p. 392.

[101] O. Semenov, A. Vassighi, M. Sachdev, A. Keshavarzi, and C. F. Hawkins, "Burn-in temperature projections for deep sub-micron technologies," in *International Test Conference, 2003. Proceedings. ITC 2003.*, 2003, vol. 1, pp. 95–104.

[102] O. Semenov, A. Vassighi, and M. Sachdev, "Impact of Self-Heating Effect on Long-Term Reliability and Performance Degradation in CMOS Circuits," *IEEE Trans. Device Mater. Reliab.*, vol. 6, no. 1, pp. 17–27, Mar. 2006.

[103] E. E. Nigussie, "Exploration and Design of High Performance Variation Tolerant On-Chip Interconnects - Dissertation," University of Turku, 2010.

[104] R. Manevich, L. Polishuk, I. Cidon, and A. Kolodny, "Design Tradeoffs of Long Links in Hierarchical Tiled Networks-on-Chip," in *2013 Euromicro Conference on Digital System Design*, 2013, no. February, pp. 769–776.

[105] T. Kanamoto, T. Okumura, K. Furukawa, H. Takafuji, A. Kurokawa, K. Hachiya, T. Sakata, M. Tanaka, H. Nakashima, H. Masuda, T. Sato, and M. Hashimoto, "Impact of Self-Heating in Wire Interconnection on Timing," *IEICE Trans. Electron.*, vol. E93–C, no. 3, pp. 388–392, 2010.

[106] L. Peh, A. Kumar, and N. K. Jha, "Thermal Modeling, Characterization and Management of On-Chip Networks," in *37th International Symposium on Microarchitecture (MICRO-37'04)*, 2004, pp. 67–78.

[107] C. Hankendi and a. K. Coskun, "Reducing the energy cost of computing through efficient co-scheduling of parallel workloads," in *2012 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2012, pp. 994–999.

[108] M. Dashti, A. Fedorova, J. Funston, F. Gaud, R. Lachaize, B. Lepers, V. Quema, and M. Roth, "Traffic Management: A Holistic Approach to Memory Placement on NUMA Systems," in *Proceedings of the eighteenth international conference on Architectural support for programming languages and operating systems - ASPLOS '13*, 2013, p. 381.

[109] D. Tam, R. Azimi, and M. Stumm, "Thread clustering," *ACM SIGOPS Oper. Syst. Rev.*, vol. 41, no. 3, p. 47, Jun. 2007.

[110] R. Marculescu, "Contention-aware application mapping for Network-on-Chip communication architectures," in *2008 IEEE International Conference on Computer Design*, 2008, pp. 164–169.

[111] M. Kandemir, Y. Zhang, J. Liu, and T. Yemliha, "Neighborhood-aware data locality optimization for NoC-based multicores," in *International Symposium on Code Generation and Optimization (CGO 2011)*, 2011, pp. 191–200.

[112] C.-H. O. Chen, N. Agarwal, T. Krishna, K.-H. Koo, L.-S. Peh, and K. C. Saraswat, "Physical vs. Virtual Express Topologies with Low-Swing Links for Future Many-Core NoCs," in *2010 Fourth ACM/IEEE*

International Symposium on Networks-on-Chip, 2010, pp. 173–180.

[113] R. Das, S. Eachempati, A. K. Mishra, V. Narayanan, and C. R. Das, "Design and evaluation of a hierarchical on-chip interconnect for next-generation CMPs," in *2009 IEEE 15th International Symposium on High Performance Computer Architecture*, 2009, pp. 175–186.

[114] N. Abeyratne, R. Das, Q. Li, K. Sewell, B. Giridhar, R. G. Dreslinski, D. Blaauw, and T. Mudge, "Scaling towards kilo-core processors with asymmetric high-radix topologies," in *2013 IEEE 19th International Symposium on High Performance Computer Architecture (HPCA)*, 2013, pp. 496–507.

[115] G. Passas, "VLSI Micro-Architectures for High-Radix Crossbars - Dissertation," University of Crete, Crete, 2012.

[116] K. L. Sewell, "Scaling High-Performance Interconnect Architectures to Many-Core Systems - Dissertation," University of Michigan, 2012.

[117] A. T. Tran, "On-Chip Network Designs for Many-Core Computational Platforms - Dissertation," UNIVERSITY OF CALIFORNIA, 2009.

[118] Y.-H. Kao, M. Yang, N. S. Artan, and H. J. Chao, "CNoC: High-Radix Clos Network-on-Chip," *IEEE Trans. Comput. Des. Integr. Circuits Syst.*, vol. 30, no. 12, pp. 1897–1910, Dec. 2011.

[119] J. Kim, "High-radix interconnection networks - Dissertation," Stanford University, 2008.

[120] C. Neeb and N. Wehn, "Designing efficient irregular networks for heterogeneous systems-on-chip," *J. Syst. Archit.*, vol. 54, no. 3–4, pp. 384–396, Mar. 2008.

[121] S. Herbert, S. Garg, and D. Marculescu, "Exploiting Process Variability in Voltage/Frequency Control," *IEEE Trans. Very Large Scale Integr. Syst.*, vol. 20, no. 8, pp. 1392–1404, Aug. 2012.

[122] A. Sharifi and M. Kandemir, "Feedback control for providing QoS in NoC based multicores," in *2010 Design, Automation & Test in Europe Conference & Exhibition (DATE 2010)*, 2010, pp. 1384–1389.

[123] K. S. Shim, M. H. Cho, M. Kinsy, T. Wen, M. Lis, G. E. Suh, and S. Devadas, "Static virtual channel allocation in oblivious routing," in *2009 3rd ACM/IEEE International Symposium on Networks-on-Chip*, 2009, pp. 38–43.

[124] P. Kumar, Y. Pan, J. Kim, G. Memik, and A. Choudhary, "Exploring concentration and channel slicing in on-chip network router," in *2009 3rd ACM/IEEE International Symposium on Networks-on-Chip*, 2009, pp. 276–285.

[125] R. Manevich, I. Cidon, A. Kolodny, and I. Walter, "Centralized Adaptive Routing for NoCs," *IEEE Comput. Archit. Lett.*, vol. 9, no. 2, pp. 57–60, Feb. 2010.

[126] Y. J. Yoon, N. Concer, M. Petracca, and L. Carloni, "Virtual channels vs. multiple physical networks: a comparative analysis," in *Proceedings of the 47th Design Automation Conference on - DAC '10*, 2010, p. 162.

[127] Y. J. Yoon, N. Concer, M. Petracca, and L. P. Carloni, "Virtual Channels and Multiple Physical Networks: Two Alternatives to Improve NoC Performance," *IEEE Trans. Comput. Des. Integr. Circuits Syst.*, vol. 32, no. 12, pp. 1906–1919, Dec. 2013.

[128] R. Das, S. Narayanasamy, S. Satpathy, and R. Dreslinski, "Catnap: Energy Proportional Multiple Network-on-Chip," in *40th International Symposium on Computer Architecture (ISCA, 2013)*, 2013, pp. 1–12.

[129] S. Volos, C. Seiculescu, B. Grot, N. K. Pour, B. Falsafi, and G. De Micheli, "CCNoC: Specializing On-Chip Interconnects for Energy Efficiency in Cache-Coherent Servers," in *2012 IEEE/ACM Sixth International Symposium on Networks-on-Chip*, 2012, no. Nocs, pp. 67–74.

[130] C. Chou and R. Marculescu, "User-Aware Dynamic Task Allocation in Networks-on-Chip," in *2008 Design, Automation and Test in Europe*, 2008, pp. 1232–1237.

[131] T. Moscibroda and O. Mutlu, "A case for bufferless routing in on-chip networks," *ACM SIGARCH Comput. Archit. News*, vol. 37, no. 3, p. 196, Jun. 2009.

[132] A. Bakhoda, J. Kim, and T. M. Aamodt, "Throughput-Effective On-Chip Networks for Manycore Accelerators," in *2010 43rd Annual IEEE/ACM International Symposium on Microarchitecture*, 2010, pp. 421–432.

[133] M. Hayenga, N. E. Jerger, and M. Lipasti, "SCARAB: a single cycle adaptive routing and bufferless network," in *42nd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO'09)*, 2009, pp. 1–11.

[134] C. J. Glass and L. M. Ni, "The turn model for adaptive routing," *ACM SIGARCH Comput. Archit. News*, vol. 20, no. 2, pp. 278–287, Jun. 1992.

[135] R. Ramanujam and B. Lin, "Destination-based adaptive routing on 2D mesh networks," in *ACM/IEEE Symposium on Architectures for Networking and Communications Systems (ANCS)*, 2010, pp. 1 – 12.

[136] H. Zhu, P. Pande, and C. Grecu, "Performance evaluation of adaptive routing algorithms for achieving fault tolerance in NoC fabrics," in *International Conference on Application -specific Systems, Architectures and Processors (ASAP'07)*, 2007, pp. 38–41.

[137] M. Mirza-Aghatabar, S. Koohi, S. Hessabi, and M. Pedram, "An Empirical Investigation of Mesh and Torus NoC Topologies Under Different Routing Algorithms and Traffic Models," in *10th Euromicro Conference on Digital System Design Architectures, Methods and Tools (DSD 2007)*, 2007, no. 4, pp. 19–26.

[138] G. Chiu and I. C. Society, "The odd-even turn model for adaptive routing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 11, no. 7, pp. 729–738, Jul. 2000.

[139] J. Hu and R. Marculescu, "DyAD: smart routing for networks-on-chip," in *Proceedings of the 41st annual Design Automation Conference*, 2004, vol. 04, p. 263.

[140] M. Li, Q. Zeng, and W. Jone, "DyXY - a proximity congestion-aware deadlock-free dynamic routing method for network on chip," *2006 43rd ACM/IEEE Des. Autom. Conf.*, pp. 849–852, 2006.

[141] P. Lotfi-Kamran, a. M. Rahmani, M. Daneshtalab, a. Afzali-Kusha, and Z. Navabi, "EDXY – A low cost congestion-aware routing algorithm for network-on-chips," *J. Syst. Archit.*, vol. 56, no. 7, pp. 256–264, Jul. 2010.

[142] E. Kakoulli, V. Soteriou, and T. Theocharides, "HPRA: A pro-active Hotspot-Preventive high-performance routing algorithm for Networks-on-Chips," in *2012 IEEE 30th International Conference on Computer Design (ICCD)*, 2012, pp. 249–255.

[143] A. Ali, N. Rafique, and M. Thottethodi, "Near-Optimal Worst-Case Throughput Routing for Two-Dimensional Mesh Networks," in *32nd International Symposium on Computer Architecture (ISCA'05)*, 2005, pp. 432–443.

[144] G. Sun, C. Chang, B. Lin, and L. Zeng, "Oblivious routing design for mesh networks to achieve a new worst-case throughput bound," in *2012 IEEE 30th International Conference on Computer Design (ICCD)*, 2012, pp. 427–432.

[145] S. Mubeen and S. Kumar, "Designing Efficient Source Routing for Mesh Topology Network on Chip Platforms," *2010 13th Euromicro Conf. Digit. Syst. Des. Archit. Methods Tools*, pp. 181–188, Sep. 2010.

[146] Y. B. Kim and Y.-B. Kim, "Fault Tolerant Source Routing for Network-on-chip," in *22nd IEEE International Symposium on Defect and Fault-Tolerance in VLSI Systems (DFT 2007)*, 2007, pp. 12–20.

[147] M. Palesi, G. Longo, S. Signorino, R. Holsmark, S. Kumar, and V. Catania, "Design of Bandwidth Aware and Congestion Avoiding Efficient Routing Algorithms for Networks-on-Chip Platforms," in *Second ACM/IEEE International Symposium on Networks-on-Chip (nocs 2008)*, 2008, pp. 97–106.

[148] E. A. Carara and F. G. Moraes, "Deadlock-Free Multicast Routing Algorithm for Wormhole-Switched Mesh," in *IEEE Computer Society Annual Symposium on VLSI, 2008. Proceedings.*, 2008, pp. 341–346.

[149] F. A. Samman, T. Hollstein, and M. Glesner, "Adaptive and Deadlock-Free Tree-Based Multicast Routing for Networks-on-Chip," *IEEE Trans. Very Large Scale Integr. Syst.*, vol. 18, no. 7, pp. 1067–1080, Jul. 2010.

[150] F. A. Samman, "Microarchitecture and Implementation of Networks-on-Chip with a Flexible Concept for Communication Media Sharing - Dissertation," Technical University Darmstadt, 2010.

[151] C. Cornelius, P. Gorski, S. Kubisch, and D. Timmermann, "Trading Hardware Overhead for Communication Performance in Mesh-Type Topologies," *2010 13th Euromicro Conf. Digit. Syst. Des. Archit. Methods Tools*, pp. 173–180, Sep. 2010.

[152] B. Grot, J. Hestness, S. W. Keckler, and O. Mutlu, "Kilo-NOC," *ACM SIGARCH Comput. Archit. News*, vol. 39, no. 3, p. 401, Jul. 2011.

[153] B. Grot, "Network-on-Chip Architectures for Scalability and Service Guarantees - Dissertation," University of Texas at Austin, 2011.

[154] D. Ludovici, G. N. Gaydadjiev, F. Gilabert, M. E. Gomez, and D. Bertozzi, "Contrasting topologies for regular interconnection networks under the constraints of nanoscale silicon technology," *Proc. Third Int. Work. Netw. Chip Archit. - NoCArc '10*, p. 37, 2010.

[155] A. Strano, D. Bertozzi, F. Angiolini, L. Di Gregorio, F. O. Sem-Jacobsen, V. Todorov, J. Flich, F. Silla, and T. Bjerregaard, "Quest for the ultimate network-on-chip," in *Proceedings of the 2012 Interconnection Network Architecture on On-Chip, Multi-Chip Workshop - INA-OCMC '12*, 2012, pp. 43–46.

[156] D. Atienza, F. Angiolini, S. Murali, A. Pullini, L. Benini, and G. De Micheli, "Network-on-Chip design and synthesis outlook," *Integr. VLSI J.*, vol. 41, no. 3, pp. 340–359, May 2008.

[157] F. Gilabert, D. Ludovici, S. Medardoni, D. Bertozzi, L. Benini, and G. N. Gaydadjiev, "Designing Regular Network-on-Chip Topologies under Technology, Architecture and Software Constraints," *2009 Int. Conf. Complex, Intell. Softw. Intensive Syst.*, pp. 681–687, Mar. 2009.

[158] Y. He, H. Sasaki, S. Miwa, and H. Nakamura, "Predict-More Router: A Low Latency NoC Router with More Route Predictions," in *2013 IEEE International Symposium on Parallel & Distributed Processing, Workshops and Phd Forum*, 2013, pp. 842–850.

[159] A. Kumary, P. Kunduz, A. P. Singhx, L.-S. Pehy, and N. K. Jhay, "A 4.6Tbits/s 3.6GHz single-cycle NoC router with a novel switch allocator in 65nm CMOS," in *2007 25th International Conference on Computer Design*, 2007, pp. 63–70.

[160] T. Krishna, C.-H. O. Chen, W. C. Kwon, and L.-S. Peh, "Breaking the on-chip latency barrier using SMART," *2013 IEEE 19th Int. Symp. High Perform. Comput. Archit.*, vol. 3, no. 2, pp. 378–389, Feb. 2013.

[161] S. T. Nguyen and S. Oyanagi, "The Design of On-the-Fly Virtual Channel Allocation for Low Cost High Performance On-Chip Routers," in *2010 First International Conference on Networking and Computing*, 2010, no. Vc, pp. 88–94.

[162] L. Xin and C. Choy, "A low-latency NoC router with lookahead bypass," in *Proceedings of 2010 IEEE International Symposium on Circuits and Systems*, 2010, pp. 3981–3984.

[163] B. Grot, J. Hestness, S. W. Keckler, and O. Mutlu, "Express Cube Topologies for on-Chip Interconnects," in *2009 IEEE 15th International Symposium on High Performance Computer Architecture*, 2009, pp. 163–174.

[164] I. Koren and C. Krishna, *Fault-tolerant systems*. San Francisco, CA, USA: Morgan Kaufmann Publisher, 2007.

[165] P. Zarkesh-Ha, G. B. P. Bezerra, S. Forrest, and M. Moses, "Hybrid Network on Chip (HNoC): Local Buses with a Global Mesh Architecture," in *Proceedings of the 12th ACM/IEEE international workshop on System level interconnect prediction - SLIP '10*, 2010, no. c, pp. 9–14.

[166] C. Wang, W.-H. Hu, S. E. Lee, and N. Bagherzadeh, "Area and Power-efficient Innovative Network-on-Chip Architecture," in *2010 18th Euromicro Conference on Parallel, Distributed and Network-based Processing*, 2010, pp. 533–539.

[167] W. Hu, S. Lee, and N. Bagherzadeh, "DMesh: a diagonally-linked mesh network-on-chip architecture," in *First International Workshop on Network on Chip Architectures (NoCArc'08)*, 2008, pp. 1–7.

[168] J. Camacho, J. Flich, J. Duato, H. Eberle, and W. Olesinski, "Towards an Efficient NoC Topology through Multiple Injection Ports," in *2011 14th Euromicro Conference on Digital System Design*, 2011, pp. 165–172.

[169] J. Camacho, J. Flich, J. Duato, H. Eberle, and W. Olesinski, "A power-efficient network on-chip topology," in *Proceedings of the Fifth International Workshop on Interconnection Network Architecture On-Chip, Multi-Chip - INA-OCMC '11*, 2011, pp. 23–26.

[170] J. Camacho, J. Flich, A. Roca, and J. Duato, "PC-Mesh: A Dynamic Parallel Concentrated Mesh," in *2011 International Conference on Parallel Processing*, 2011, pp. 642–651.

[171] J. Camacho and J. Flich, "HPC-Mesh: A Homogeneous Parallel Concentrated Mesh for Fault-Tolerance and Energy Savings," in *2011 ACM/IEEE Seventh Symposium on Architectures for Networking and Communications Systems*, 2011, pp. 69–80.

[172] A. E. Zonouz, M. Seyrafi, A. Asad, M. Soryani, M. Fathy, and R. Berangi, "A Fault Tolerant NoC Architecture for Reliability Improvement and Latency Reduction," in *2009 12th Euromicro Conference on Digital System Design, Architectures, Methods and Tools*, 2009, pp. 473–480.

[173] D. Park, C. Nicopoulos, J. Kim, N. Vijaykrishnan, and C. R. Das, "A Distributed Multi-Point Network Interface for Low-Latency, Deadlock-Free On-Chip Interconnects," in *2006 1st International Conference on Nano-Networks and Workshops*, 2006, pp. 1–6.

[174] N. Muralimanohar, R. Balasubramonian, and N. Jouppi, "Optimizing NUCA Organizations and Wiring Alternatives for Large Caches with CACTI 6.0," in *40th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO 2007)*, 2007, pp. 3–14.

[175] J. Bahn and N. Bagherzadeh, "A generic traffic model for on-chip interconnection networks," in *1st International Workshop on Network on Chip Architectures to be held in conjunction with MICRO-41 (NoCArc'08)*, 2008, pp. 1–8.

[176] K. Hoste and L. Eeckhout, "Microarchitecture-Independent Workload Characterization," *IEEE Micro*, vol. 27, no. 3, pp. 63–72, May 2007.

[177] C. Grecu, A. Ivanov, P. Pande, A. Jantsch, E. Salminen, U. Ogras, and R. Marculescu, "Towards Open Network-on-Chip Benchmarks," in *First International Symposium on Networks-on-Chip (NOCS'07)*, 2007, pp. 205–205.

[178] A. Scherrer, A. Fraboulet, and T. Risset, "Long-range dependence and on-chip processor traffic," *Microprocess. Microsyst.*, vol. 33, no. 1, pp. 72–80, Feb. 2009.

[179] L. Tedesco, A. Mello, D. Garibotti, N. Calazans, and F. Moraes, "Traffic Generation and Performance Evaluation for Mesh-based NoCs," in *2005 18th Symposium on Integrated Circuits and Systems Design*, 2005, pp. 184–189.

[180] P. Bogdan, M. Kas, R. Marculescu, and O. Mutlu, "QuaLe: A Quantum-Leap Inspired Model for Non-stationary Analysis of NoC Traffic in Chip Multi-processors," in *2010 Fourth ACM/IEEE International Symposium on Networks-on-Chip*, 2010, pp. 241–248.

[181] P. Bogdan and R. Marculescu, "Non-Stationary Traffic Analysis and Its Implications on Multicore Platform Design," *IEEE Trans. Comput. Des. Integr. Circuits Syst.*, vol. 30, no. 4, pp. 508–519, Apr. 2011.

[182] P. Bogdan and R. Marculescu, "Workload characterization and its impact on multicore platform design," in *Proceedings of the eighth IEEE/ACM/IFIP international conference on Hardware/software codesign and system synthesis - CODES/ISSS '10*, 2010, p. 231.

[183] V. Soteriou and L. Peh, "A Statistical Traffic Model for On-Chip Interconnection Networks," in *14th IEEE International Symposium on Modeling, Analysis, and Simulation*, 2006, pp. 104–116.

[184] H. Cota de Freitas, L. M. Schnorr, M. A. Z. Alves, and P. O. A. Navaux, "Impact of Parallel Workloads on NoC Architecture Design," in *2010 18th Euromicro Conference on Parallel, Distributed and Network-based Processing*, 2010, no. c, pp. 551–555.

[185] H. F. Tatenguem, D. Ludovici, A. Strano, D. Bertozzi, and H. Reinig, "Contrasting multi-synchronous MPSoC design styles for fine-grained clock domain partitioning," in *Proceedings of the 4th International Workshop on Network on Chip Architectures - NoCArc '11*, 2011, pp. 37 – 42.

[186] I. M. Panades and A. Greiner, "Bi-synchronous FIFO for synchronous circuit communication well suited for network-on-chip in GALS architectures," *… -on-Chip, 2007. NOCS 2007. First …*, pp. 83–94, May 2007.

[187] V. Rantala, "On Dynamic Monitoring Methods for Networks-on-Chip - Dissertation," University of Turku, 2012.

[188] D. Abts, N. D. Enright Jerger, J. Kim, D. Gibson, and M. H. Lipasti, "Achieving predictable performance through better memory controller placement in many-core CMPs," *ACM SIGARCH Comput. Archit. News*, vol. 37, no. 3, p. 451, Jun. 2009.

[189] L. Tedesco, T. Rosa, and F. Moraes, "A message-level monitoring protocol for QoS flows in NoCs," *2010 Int. Symp. Syst. Chip*, pp. 84–88, Sep. 2010.

[190] L. Tedesco, F. Clermidy, and F. Moraes, "A monitoring and adaptive routing mechanism for QoS traffic on mesh NoC architectures," in *Proceedings of the 7th IEEE/ACM international conference on Hardware/software codesign and system synthesis - CODES+ISSS '09*, 2009, pp. 109–118.

[191] M. J. Irwin, "Optimizing power and performance for reliable on-chip networks," in *2010 15th Asia and South Pacific Design Automation Conference (ASP-DAC)*, 2010, pp. 431–436.

[192] H. Zhao, M. Kandemir, and M. J. Irwin, "Exploring performance-power tradeoffs in providing reliability for NoC-based MPSoCs," in *2011 12th International Symposium on Quality Electronic Design*, 2011, pp. 1–7.

[193] M. A. Al Faruque, T. Ebi, and J. Henkel, "ROAdNoC: Runtime observability for an adaptive network on chip architecture," in *2008 IEEE/ACM International Conference on Computer-Aided Design*, 2008, pp. 543–548.

[194] M. A. Al Faruque, R. Krist, and J. Henkel, "ADAM: Run-time Agent-based Distributed Application Mapping for on-chip Communication," in *Proceedings of the 45th annual conference on Design automation - DAC '08*, 2008, pp. 760–765.

[195] J. Zhao, S. Madduri, R. Vadlamani, W. Burleson, and R. Tessier, "A Dedicated Monitoring Infrastructure for Multicore Processors," *IEEE Trans. Very Large Scale Integr. Syst.*, vol. 19, no. 6, pp. 1011–1022, Jun. 2011.

[196] R. Abdel-Khalek and V. Bertacco, "Functional post-silicon diagnosis and debug for networks-on-chip,"

Proc. Int. Conf. Comput. Des. - ICCAD '12, p. 557, 2012.

[197] A. Alhonen, E. Salminen, J. Nieminen, and T. D. Hamalainen, "A scalable, non-interfering, synthesizable Network-on-chip monitor," in *NORCHIP 2010*, 2010, pp. 1–6.

[198] J. W. van den Brand, C. Ciordas, K. Goossens, and T. Basten, "Congestion-Controlled Best-Effort Communication for Networks-on-Chip," in *2007 Design, Automation & Test in Europe Conference & Exhibition*, 2007, pp. 1–6.

[199] C. Ciordas, T. Basten, A. Radulescu, K. Goossens, and J. Meerbergen, "An event-based network-on-chip monitoring service," in *Proceedings. Ninth IEEE International High-Level Design Validation and Test Workshop (IEEE Cat. No.04EX940)*, 2004, pp. 149–154.

[200] S. Madduri, R. Vadlamani, W. Burleson, and R. Tessier, "A monitor interconnect and support subsystem for multicore processors," in *2009 Design, Automation & Test in Europe Conference & Exhibition*, 2009, pp. 761–766.

[201] K. R. Vaddina, L. Guang, E. Nigussie, P. Liljeberg, and J. Plosila, "On-line Distributed Thermal Sensing and Monitoring of Multicore Systems," *2008 Norchip*, pp. 89–93, Nov. 2008.

[202] J. Zhao, R. Tessier, and W. Burleson, "Distributed sensor data processing for many-cores," in *Proceedings of the great lakes symposium on VLSI - GLSVLSI '12*, 2012, p. 159.

[203] C. Ciordas, K. Goossens, T. Basten, A. Radulescu, and A. Boon, "Transaction Monitoring in Networks on Chip: The On-Chip Run-Time Perspective," in *2006 International Symposium on Industrial Embedded Systems*, 2006, pp. 1–10.

[204] L. Guang, P. Rantala, E. Nigussie, J. Isoaho, and H. Tenhunen, "Low-latency and Energy-efficient Monitoring Interconnect for Hierarchical-agent-monitored NoCs," *2008 Norchip*, pp. 227–232, Nov. 2008.

[205] L. Guang, E. Nigussie, J. Isoaho, P. Rantala, and H. Tenhunen, "Interconnection alternatives for hierarchical monitoring communication in parallel SoCs," *Microprocess. Microsyst.*, vol. 34, no. 5, pp. 118–128, Aug. 2010.

[206] C. Ciordas, K. Goossens, A. Radulescu, and T. Basten, "NoC Monitoring: Impact on the Design Flow," in *2006 IEEE International Symposium on Circuits and Systems*, 2006, pp. 1981–1984.

[207] C. Ciordas, A. Hansson, K. Goossens, and T. Basten, "A Monitoring-Aware Network-on-Chip Design Flow," in *9th EUROMICRO Conference on Digital System Design (DSD'06)*, 2006, pp. 97–106.

[208] M. Fattah, M. Daneshtalab, P. Liljeberg, and J. Plosila, "Exploration of MPSoC monitoring and management systems," in *6th International Workshop on Reconfigurable Communication-Centric Systems-on-Chip (ReCoSoC)*, 2011, pp. 1–3.

[209] L. Guang, J. Plosila, and H. Tenhunen, "Hierarchical Agent Monitoring Services on Reconfigurable NoC Platform: A Formal Approach," in *Workshop Digest of DSNOC 09 (Diagnostic Services in Network-on-Chips), in conjunction with DATE 2009*, 2009, pp. 98–101.

[210] L. Guang, A. W. Yin, P. Rantala, P. Liljeberg, J. Isoaho, and H. Tenhunen, "Hierarchical Power Monitoring for On-chip Networks," Turku, Finland, 2009.

[211] L. Guang, B. Yang, J. Plosila, K. Latif, and H. Tenhunen, "Hierarchical power monitoring on NoC - a case study for hierarchical agent monitoring design approach," in *NORCHIP 2010*, 2010, pp. 1–6.

[212] A. W. Yin, P. Rantala, E. Nigussie, J. Isoaho, and H. Tenhunen, "Hierarchical agent based NoC with dynamic online services," in *2009 4th IEEE Conference on Industrial Electronics and Applications*, 2009, pp. 434–439.

[213] A. Yin, L. Guang, and P. Liljeberg, "Hierarchical agent architecture for scalable NoC design with online monitoring services," in *1st International Workshop on Network on Chip Architectures (NoCArc'08)*, 2008, no. Section III, pp. 58–63.

[214] S. Zhuravlev, J. C. Saez, S. Blagodurov, A. Fedorova, and M. Prieto, "Survey of scheduling techniques for addressing shared resources in multicore processors," *ACM Comput. Surv.*, vol. 45, no. 1, pp. 1–28, Nov. 2012.

[215] A. Bhattacharjee and M. Martonosi, "Thread criticality predictors for dynamic performance, power, and resource management in chip multiprocessors," *ACM SIGARCH Comput. Archit. News*, vol. 37, no. 3, pp. 290–301, Jun. 2009.

[216] H. Kim, P. Ghoshal, B. Grot, P. V. Gratz, and D. a. Jiménez, "Reducing Network-on-Chip energy consumption through spatial locality speculation," in *Proceedings of the Fifth ACM/IEEE International Symposium on Networks-on-Chip - NOCS '11*, 2011, p. 233.

[217] E. Yao, Y. Bao, G. Tan, and M. Chen, "Extending Amdahl's law in the multicore era," *ACM SIGMETRICS Perform. Eval. Rev.*, vol. 37, no. 2, p. 24, Oct. 2009.

[218] X.-H. Sun and Y. Chen, "Reevaluating Amdahl's law in the multicore era," *J. Parallel Distrib. Comput.*, vol. 70, no. 2, pp. 183–188, Feb. 2010.

[219] N. Gunther, "Unification of Amdahl's Law, LogP and Other Performance Models for Message-Passing Architectures.," in *17th IASTED International Conference on Parallel and Distributed Computing and Systems*, 2005, no. L, pp. 569–576.

[220] T. Yano and T. Hayashida, "Importance of Single-Core Performance in the Multicore Era," in *Proceedings of the Thirty-Fifth Australasian Computer Science Conference (ACSC'12)*, 2012, no. Acsc, pp. 107–114.

[221] X. Xiang, C. Ding, H. Luo, and B. Bao, "HOTL: a higher order theory of locality," in *Proceedings of the eighteenth international conference on Architectural support for programming languages and operating systems - ASPLOS '13*, 2013, pp. 343–356.

[222] J. R. Srinivasan, "Improving cache utilisation," in *Technical Report*, 2011, no. 800, pp. 1–184.

[223] C.-J. Wu, "Dynamic Techniques for Mitigating Inter- and Intra-Application Cache Interference - Dissertation," Princeton University, 2012.

[224] R. Balasubramonian, N. P. Jouppi, and N. Muralimanohar, "Multi-Core Cache Hierarchies," *Synth. Lect. Comput. Archit.*, vol. 6, no. 3, pp. 1–153, May 2011.

[225] L. Leem, J. Bau, Q. A. Jacobson, and S. Mitra, "ERSA: Error Resilient System Architecture for probabilistic applications," in *2010 Design, Automation & Test in Europe Conference & Exhibition (DATE 2010)*, 2010, vol. 31, no. 4, pp. 1560–1565.

[226] L. Leem, H. Cho, H. Lee, Y. M. Kim, Y. Li, and S. Mitra, "Cross-layer error resilience for robust systems," in *2010 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2010, pp. 177–180.

[227] R. Ayoub, U. Ogras, E. Gorbatov, Y. Jin, T. Kam, P. Diefenbaugh, and T. Rosing, "OS-level power minimization under tight performance constraints in general purpose systems," *IEEE/ACM Int. Symp. Low Power Electron. Des.*, pp. 321–326, Aug. 2011.

[228] R. Zamani and A. Afsahi, "A study of hardware performance monitoring counter selection in power modeling of computing systems," in *2012 International Green Computing Conference (IGCC)*, 2012, pp. 1–10.

[229] C.-K. Luk, R. Cohn, R. Muth, H. Patil, A. Klauser, G. Lowney, S. Wallace, V. J. Reddi, and K. Hazelwood, "Pin: Building Customized Program Analysis Tools with Dynamic Instrumentation," *ACM SIGPLAN Not.*, vol. 40, no. 6, pp. 190–201, Jun. 2005.

[230] W. Heirman and J. Dambre, "Runtime variability in scientific parallel applications," in *Fourth Annual Workshop on Modeling, Benchmarking and Simulation*, 2008, pp. 1–10.

[231] P. Christie and D. Stroobandt, "The interpretation and application of Rent's rule," *IEEE Trans. Very Large Scale Integr. Syst.*, vol. 8, no. 6, pp. 639–648, Dec. 2000.

[232] P. Christie, "A differential equation for placement analysis," *IEEE Trans. Very Large Scale Integr. Syst.*, vol. 9, no. 6, pp. 913–921, Dec. 2001.

[233] T. Grötker, S. Liao, G. Martin, and S. Swan, *System design with SystemC*, 1st ed. Hingham, MA, USA: Kluwer Academic Pub, 2002.

[234] M. Gag, T. Wegner, and P. Gorski, "System level modeling of Networks-on-Chip for power estimation and design space exploration," in *Methoden und Beschreibungssprachen zur Modellierung und Verifikation von Schaltungen und Systemen (MBMV 2013)*, 2013, pp. 25–34.

[235] M. Lis, K. Shim, and M. Cho, "DARSIM: a parallel cycle-level NoC simulator," in *Sixth Annual Workshop on Modeling, Benchmarking and Simulation (MoBS'10)*, 2010, pp. 1–9.

[236] 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 *2013 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, 2013, pp. 86–96.

[237] M. Lis, P. Ren, M. H. Cho, K. S. Shim, C. W. Fletcher, O. Khan, and S. Devadas, "Scalable, accurate multicore simulation in the 1000-core era," in *(IEEE ISPASS) IEEE INTERNATIONAL SYMPOSIUM ON PERFORMANCE ANALYSIS OF SYSTEMS AND SOFTWARE*, 2011, pp. 175–185.

[238] A. Ben and S. Ben, "A Survey of Network-On-Chip Tools," *Int. J. Adv. Comput. Sci. Appl.*, vol. 4, no. 9, pp. 61–67, 2013.

[239] T. E. Carlson, W. Heirman, and L. Eeckhout, "Sniper: Exploring the Level of Abstraction for Scalable and

Accurate Parallel Multi-Core Simulation," in *Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis on - SC '11*, 2011, pp. 1–12.

[240] J. E. Miller, H. Kasture, G. Kurian, C. Gruenwald, N. Beckmann, C. Celio, J. Eastep, and A. Agarwal, "Graphite: A distributed parallel simulator for multicores," in *HPCA - 16 2010 The Sixteenth International Symposium on High-Performance Computer Architecture*, 2010, pp. 1–12.

[241] R. Ubal, J. Sahuquillo, S. Petit, and P. Lopez, "Multi2Sim: A Simulation Framework to Evaluate Multicore-Multithreaded Processors," in *19th International Symposium on Computer Architecture and High Performance Computing (SBAC-PAD'07)*, 2007, pp. 62–68.

[242] N. Binkert, S. Sardashti, R. Sen, K. Sewell, M. Shoaib, N. Vaish, M. D. Hill, D. a. Wood, B. Beckmann, G. Black, S. K. Reinhardt, A. Saidi, A. Basu, J. Hestness, D. R. Hower, and T. Krishna, "The gem5 simulator," *ACM SIGARCH Comput. Archit. News*, vol. 39, no. 2, pp. 1–7, Aug. 2011.

[243] A. Patel, F. Afram, S. Chen, and K. Ghose, "MARSS: a full system simulator for multicore x86 CPUs," in *48th ACM/EDAC/IEEE Design Automation Conference (DAC'11)*, 2011, pp. 1050–1055.

[244] S. Rigo, G. Araujo, M. Bartholomeu, and R. Azevedo, "ArchC: A SystemC-Based Architecture Description Language," in *16th Symposium on Computer Architecture and High Performance Computing*, 2004, pp. 66–73.

[245] S. Li, J. Ahn, and R. Strong, "McPAT: an integrated power, area, and timing modeling framework for multicore and manycore architectures," in *42nd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO-42)*, 2009, no. c, pp. 469–480.

[246] Y. Cao and W. Zhao, "Predictive Technology Model for Nano-CMOS Design Exploration," in *2006 1st International Conference on Nano-Networks and Workshops*, 2006, no. Vdd, pp. 1–5.

[247] A. Banerjee, "Communication flows in power-efficient Networks-on-Chips - Dissertation," University of Cambridge, 2010.

[248] S. C. Woo, M. Ohara, E. Torrie, J. P. Singh, and A. Gupta, "The SPLASH-2 programs," *ACM SIGARCH Comput. Archit. News*, vol. 23, no. 2, pp. 24–36, May 1995.

[249] M. Bhadauria, V. M. Weaver, and S. a. McKee, "Understanding PARSEC performance on contemporary CMPs," in *2009 IEEE International Symposium on Workload Characterization (IISWC)*, 2009, pp. 98–107.

[250] C. Bienia and K. Li, "Parsec 2.0: A new benchmark suite for chip-multiprocessors," in *5th Annual Workshop on Modeling, Benchmarking and Simulation*, 2009, pp. 1–9.

[251] C. Bienia and S. Kumar, "PARSEC vs. SPLASH-2: A quantitative comparison of two multithreaded benchmark suites on Chip-Multiprocessors," in *2008 IEEE International Symposium on Workload Characterization*, 2008, pp. 47–56.

[252] N. Barrow-Williams, C. Fensch, and S. Moore, "A communication characterisation of Splash-2 and Parsec," in *2009 IEEE International Symposium on Workload Characterization (IISWC)*, 2009, pp. 86–97.

[253] C. Nicopoulos, S. Srinivasan, A. Yanamandra, D. Park, V. Narayanan, C. R. Das, and M. J. Irwin, "On the Effects of Process Variation in Network-on-Chip Architectures," *IEEE Trans. Dependable Secur. Comput.*, vol. 7, no. 3, pp. 240–254, Jul. 2010.

[254] R. Kumar and V. Kursun, "Impact of temperature fluctuations on circuit characteristics in 180nm and 65nm CMOS technologies," in *IEEE International Symposium on Circuits and Systems (ISCAS 2006)*, 2006, no. V, p. 4.

[255] R. Kumar and V. Kursun, "Voltage optimization for temperature variation insensitive CMOS circuits," in *48th Midwest Symposium on Circuits and Systems, 2005.*, 2005, pp. 476–479.

[256] B. Li, L.-S. Peh, and P. Patra, "Impact of Process and Temperature Variations on Network-on-Chip Design Exploration," in *Second ACM/IEEE International Symposium on Networks-on-Chip (nocs 2008)*, 2008, pp. 117–126.

[257] A. Pullini, F. Angiolini, S. Murali, D. Atienza, G. De Micheli, and L. Benini, "Bringing NoCs to 65 nm," *IEEE Micro*, vol. 27, no. 5, pp. 75–85, Sep. 2007.

[258] L. Chen and T. M. Pinkston, "NoRD: Node-Router Decoupling for Effective Power-gating of On-Chip Routers," in *2012 45th Annual IEEE/ACM International Symposium on Microarchitecture*, 2012, pp. 270–281.

[259] K. C. Hale, B. Grot, and S. W. Keckler, "Segment gating for static energy reduction in Networks-on-Chip," in *Proceedings of the 2nd International Workshop on Network on Chip Architectures - NoCArc '09*, 2009, pp. 57–62.

[260] R. S. Ramanujam, V. Soteriou, B. Lin, and L.-S. Peh, "Design of a High-Throughput Distributed Shared-

Buffer NoC Router," in *2010 Fourth ACM/IEEE International Symposium on Networks-on-Chip*, 2010, pp. 69–78.

[261] R. Ramanujam and V. Soteriou, "Extending the effective throughput of NoCs with distributed shared-buffer routers," *Comput. Des. …*, vol. 30, no. 4, pp. 548–561, 2011.

[262] C. Nicopoulos, D. Park, J. Kim, N. Vijaykrishnan, M. Yousif, and C. Das, "ViChaR: A Dynamic Virtual Channel Regulator for Network-on-Chip Routers," in *2006 39th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO'06)*, 2006, pp. 333–346.

[263] J. H. Collet, A. Louri, V. T. Bhat, and P. Poluri, "ROBUST: a new self-healing fault-tolerant NoC router," in *Proceedings of the 4th International Workshop on Network on Chip Architectures - NoCArc '11*, 2011, pp. 11–16.

[264] K. Constantinides, S. Plaza, J. Blome, V. Bertacco, S. Mahlke, T. Austin, and M. Orshansky, "BulletProof: A Defect~Tolerant CMP Switch Architecture," in *The Twelfth International Symposium on High-Performance Computer Architecture, 2006.*, 2006, pp. 3–14.

[265] D. Fick, A. DeOrio, J. Hu, V. Bertacco, D. Blaauw, and D. Sylvester, "Vicis: A Reliable Network for Unreliable Silicon," in *Proceedings of the 46th Annual Design Automation Conference on ZZZ - DAC '09*, 2009, p. 812.

[266] P. Poluri and A. Louri, "Tackling Permanent Faults in the Network-on-Chip Router Pipeline," in *2013 25th International Symposium on Computer Architecture and High Performance Computing*, 2013, pp. 49–56.

[267] M. H. Neishaburi and Z. Zilic, "ERAVC: Enhanced reliability aware NoC router," in *2011 12th International Symposium on Quality Electronic Design*, 2011, pp. 1–6.

[268] A. P. Frantz, M. Cassel, F. L. Kastensmidt, É. Cota, and L. Carro, "Crosstalk-and SEU-aware networks on chips," *IEEE Des. TEST*, pp. 340–350, 2007.

[269] C. Duan, B. J. LaMeres, and S. P. Khatri, *On and Off-Chip Crosstalk Avoidance in VLSI Design*. Boston, MA: Springer US, 2010.

[270] A. Ganguly, P. P. Pande, B. Belzer, and C. Grecu, "Design of Low Power & Reliable Networks on Chip Through Joint Crosstalk Avoidance and Multiple Error Correction Coding," *J. Electron. Test.*, vol. 24, no. 1–3, pp. 67–81, Jan. 2008.

[271] A. Ganguly, P. Pande, and B. Belzer, "Crosstalk-Aware Channel Coding Schemes for Energy Efficient and Reliable NOC Interconnects," *IEEE Trans. Very Large Scale Integr. Syst.*, vol. 17, no. 11, pp. 1626–1639, Nov. 2009.

[272] P. Pande, A. Ganguly, B. Feero, B. Belzer, and C. Grecu, "Design of Low power & Reliable Networks on Chip through joint crosstalk avoidance and forward error correction coding," in *2006 21st IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems*, 2006, pp. 466–476.

[273] B. Wang, J. Xie, and Q. Wang, "Crosstalk-aware channel transmitting scheme for error resilience NoC interconnects," in *2011 International Conference on Electronics, Communications and Control (ICECC)*, 2011, pp. 190–193.

[274] B. Wang, J. Xie, Z. Mao, and Q. Wang, "Multiple continuous error correct code for high performance network-on-chip," *2011 Asia Pacific Conf. Postgrad. Res. Microelectron. Electron.*, pp. 98–101, Oct. 2011.

[275] B. Fu, "Crosstalk-Aware Multiple Error Control for Reliable On-Chip Interconnects - Dissertation," University of Rochester, New York, 2010.

[276] B. Fu and P. Ampadu, "An Energy-Efficient Multiwire Error Control Scheme for Reliable On-Chip Interconnects Using Hamming Product Codes," *VLSI Des.*, vol. 2008, no. c, pp. 1–15, 2008.

[277] Q. Yu, B. Zhang, Y. Li, and P. Ampadu, "Error control integration scheme for reliable NoC," *Proc. 2010 IEEE Int. Symp. Circuits Syst.*, vol. 1, pp. 3893–3896, May 2010.

[278] Q. Yu and P. Ampadu, "Adaptive error control for nanometer scale network-on-chip links," *IET Comput. Digit. Tech.*, vol. 3, no. 6, p. 643, 2009.

[279] Q. Yu and P. Ampadu, "Transient and Permanent Error Co-management Method for Reliable Networks-on-Chip," *2010 Fourth ACM/IEEE Int. Symp. Networks-on-Chip*, pp. 145–154, May 2010.

[280] D. Rossi, C. Metra, a. K. Nieuwland, and a. Katoch, "Exploiting ECC redundancy to minimize crosstalk impact," *IEEE Des. Test Comput.*, vol. 22, no. 1, pp. 59–70, Jan. 2005.

[281] D. Rossi, P. Angelini, and C. Metra, "Configurable Error Control Scheme for NoC Signal Integrity," in *13th IEEE International On-Line Testing Symposium (IOLTS 2007)*, 2007, no. Iolts, pp. 43–48.

[282] D. Rossi, P. Angelini, and C. Metra, “Configurable Error Control Scheme for NoC Signal Integrity,” in *13th IEEE International On-Line Testing Symposium (IOLTS 2007)*, 2007, no. Iolts, pp. 43–48.

[283] D. Bertozzi, L. Benini, and G. De Micheli, “Error Control Schemes for On-Chip Communication Links: The Energy-Reliability Tradeoff,” *IEEE Trans. Comput. Des. Integr. Circuits Syst*, vol. 24, no. 6, pp. 818–831, 2005.

[284] G. Ascia, V. Catania, F. Fazzino, and M. Palesi, “An encoding scheme to reduce power consumption in Networks-on-Chip,” in *2009 International Conference on Computer Engineering & Systems*, 2009, pp. 15–20.

[285] M. Palesi, G. Ascia, F. Fazzino, and V. Catania, “Data Encoding Schemes in Networks on Chip,” *IEEE Trans. Comput. Des. Integr. Circuits Syst.*, vol. 30, no. 5, pp. 774–786, May 2011.

[286] J. C. S. Palma, F. G. Moraes, A. G. Ortiz, M. Glesner, and R. A. L. Reis, “Inserting Data Encoding Techniques into NoC-Based Systems,” in *IEEE Computer Society Annual Symposium on VLSI (ISVLSI ’07)*, 2007, pp. 299–304.

[287] P. T. Huang, W. L. Fang, Y. L. Wang, and W. Hwang, “Low Power and Reliable Interconnection with Self-Corrected Green Coding Scheme for Network-on-Chip,” in *Proceedings of the Second ACM/IEEE International Symposium on Networks-on-Chip*, 2008, pp. 77–83.

[288] A. Vitkovski, R. Haukilahti, A. Jantsch, and E. Nilsson, “Low-power and error coding for network-on-chip traffic,” in *Proceedings Norchip Conference, 2004.*, 2004, pp. 20–23.

[289] C. Raghunandan, K. Sainarayanan, and M. Srinivas, “Encoding with Repeater Insertion for Minimizing Delay in VLSI Interconnects,” *2006 6th Int. Work. Syst. Chip Real Time Appl.*, pp. 205–210, Dec. 2006.

[290] S. E. Lee and N. Bagherzadeh, “A variable frequency link for a power-aware network-on-chip (NoC),” *Integr. VLSI J.*, vol. 42, no. 4, pp. 479–485, Sep. 2009.

[291] B. Kim and V. Stojanović, “Characterization of Equalized and Repeated Interconnects for NoC Applications,” *IEEE Des. Test Comput.*, vol. 25, no. 5, pp. 430–439, Sep. 2008.

[292] B. Kim and V. Stojanovic, “Equalized interconnects for on-chip networks: modeling and optimization framework,” in *2007 IEEE/ACM International Conference on Computer-Aided Design*, 2007, pp. 552–559.

[293] A. Joshi, B. Kim, and V. Stojanovic, “Designing Energy-Efficient Low-Diameter On-Chip Networks with Equalized Interconnects,” in *2009 17th IEEE Symposium on High Performance Interconnects*, 2009, pp. 3–12.

[294] A. Mineo, M. Palesi, G. Ascia, and V. Catania, “NoC links energy reduction through link voltage scaling,” *2013 Int. Conf. Embed. Comput. Syst. Archit. Model. Simul.*, pp. 113–120, Jul. 2013.

III ABSTRACT

Since the introduction of chip-multiprocessor systems (CMPs) nearly a decade ago, the number of integrated cores/tiles has been steady growing and workload applications have been adapted to exploit the increasing degrees of core-level parallelism. This changed the importance of efficient on-chip communication significantly and the corresponding infrastructure has to keep step with these new requirements of massively parallel transactions. As the parallelization proceeds, the ratio of computation to communication regarding system optimizations is shifted to a higher weighting of the communication performance at each new technology generation that allows the integration of more cores on a single die. Hence, new infrastructures are build as on-chip networks and outperform conventional paradigms such as bus-based or point-to-point solutions regarding the anticipated trade-off in hardware costs, bandwidth capacities, delay, and power. But with the same progress the technology advances to smaller feature sizes, the influence of physical imperfections on CMP operations grows and the consequences propagates through all design- as well as run-time aspects. Each new system generation provides more computational capacity, but at the same time it becomes more vulnerable, the wire-to-logic delay becomes more unbalanced, and active run-time management is required to exploit the enhanced capacities as trade-off between performance, power and reliability. Therein, the communication infrastructure is the key to optimized CMP operations and the work at hand makes significant contributions to the state-of-the-art of the latest generation of such solutions, calles Networks-on-Chip (NoCs).

As a start, the main challenges of the NoC in its operational context as part of the CMP and under consideration of the expected workload behavior are introduced. Furthermore, the leading guidelines for the investigations are outlined and a holistic discussion of the NoC background is provided. Next, the impact of basic design decisions at the component level, such as router and link, on these objectives is shown by the evaluation of their physical implementation characteristics over varios technologies. This preliminary work is discussed in the context of existing state-of-the-art proposals, major side-effects/interrelations are identified, and the basic cost/performance/reliability dependencies are formulated. As result, it could be shown that simple mesh-based NoCs provide sufficient benefits regarding the upcoming challenges of physical imperfection.

Following, the first major contribution addresses the joint improvement of the performance, reliability and traffic management capabilities of the 2D-Mesh via the proposed QMesh topology. The integration of multiple spatially independent terminals per computational resource offers a valid alternative to optimization strategies that rely on the integration of more router-to-router links for

improved network connectivity. Thereby, regularity and simplicity of the 2D-Mesh topology remain untouched.

As next step, a flexible run-time configurable traffic monitoring solution for mesh-based NoCs is introduced that represents a vital run-time feature such CMP infrastructures and serves itself as intermediate mechanisms to provide self-awareness over the current traffic situation. It could be shown that traffic monitoring can be designed as combined hardware/software-solution to provide sufficient spatio-temporal adaptivity, capabilities and flexibility to capture the communication behavior of modern workloads with adequate accuracy for different use cases.

Finally, a concrete use case of the combination of the QMesh and the spatio-temporal run-time traffic monitoring is proposed, which provides remarkable potential performance gains for a wide range of traffic patterns by improving the saturation up to factors of 5, reducing the average packet delay by around 40% up to 80% and thereby imposing suitable power overheads in comparison the the 2D-Mesh reference case. It could be further shown, that software-directed and regionally-centralized traffic management is a valid option for modern CMPs despite the generalized context of the applied traffic monitoring. The introduced path update evaluation flow and the corresponding path comparison metric proved as efficient regarding the achieved improvements as well as the computational efforts.

IV KURZREFERAT

Seit der Einführung der Chip-Multiprozessor-Systeme (CMP) vor fast zehn Jahren hat die Zahl der integrierten Kerne stetig zugenommen. Zudem wurden Anwendungen entsprechend angepasst, um den steigenden Grad von Parallelität optimal ausnutzen zu können. Dies veränderte die Gewichtung der on-chip Kommunikation signifikant und die eingesetzten Infrastrukturen mussten angepasst werden, um mit diesen neuen Anforderungen der massiv parallelen Transaktionen Schritt zu halten. Mit einem Fortschreiten dieser Parallelisierung steigt die Bedeutung der Kommunikation gegenüber der Datenverarbeitung, sodass die Optimierung der benötigten Infrastruktur über die Technologiegenerationen hinweg zu einem zentralen Punkt bei der Entwicklung von CMPs geworden ist. Dabei stellen die sogenannten Networks-on-Chip (NoCs) einen wichtigen Meilenstein dar und übertreffen herkömmliche Paradigmen wie Bus-basierte oder Punkt-zu-Punkt-basierte Lösungen im Bezug auf die zu erwartenden Hardwarekosten, Kommunikationsbandbreiten, Verzögerungen, und den Leistungsverbrauch. Zugleich steigt mit der Nutzung von Technolgien mit immer kleiner werdenden Strukturgrößen der Einfluss der physischen Unvollkommenheiten auf den zuverlässigen Betrieb der CMPs. Die Konsequenzen betreffen dabei alle Abstraktionsschichten eines solchen Systems. Jede neue Systemgeneration bietet mehr Rechenkapazität, aber zur gleichen Zeit erhöht sich die Fehleranfälligkeit, die Leitungsverzögerungen sind steigend und dominieren die Leistungsfähigkeit, und ein aktives Laufzeitmanagement ist erforderlich, damit die erweiterten Kapazitäten unter der Berücksichtigung des Leistungsverbrauchs sowie der Zuverlässigkeit nutzbar gemacht werden. Dabei ist die Kommunikationsinfrastruktur der Schlüssel zu einer solchen Laufzeitoptimierung und die vorliegende Arbeit stellt bedeutende Beiträge gezielten Verbessungen von NoCs vor.

Initial werden die wichtigsten Herausforderungen an das NoC in seinem Kontext als integraler Bestandteil eines CMPs und unter Berücksichtigung der zu erwartenden Arbeitslasten eingeführt. Darüber hinaus werden die führenden Leitlinien für die Exploration skizziert und ganzheitlich im Kontext der ausgearbeiteten Grundlagen von NoCs diskutiert. Als nächstes wird der Einfluss grundlegender Entwurfsentscheidungen auf der Komponentenebene (z.B. Router und Link) bezüglich ihres Optimierungspotentials anhand der resultierenden physikalischen Eigenschaften für verschieden Technologien diskutiert. Diese Vorarbeiten werden im Rahmen bestehender Lösungen diskutiert, wichtige Nebenwirkungen sowie Wechselwirkungen identifiziert, und die Anbhängigkeiten zwischen Kosten, Leistungsfähigkeit, und Zuverlässigkeit formuliert. Als Ergebnis kann gezeigt werden, dass einfache gitter-basierte NoCs grundlegend einen sehr guten Einstiegspunkt für weitere Optimierungen bilden.

Die erste vorgestellte Lösung addressiert anschließend die übergreifende Verbesserung der Leistungsfähigkeit, Zuverlässigkeit und Regulierbarkeit von Verkehrslasten eines bestehenden 2D-Meshes. Dafür wird eine neue Topologie namens QMesh eingeführt. Diese integriert mehrere räumlich unabhängige Anbindungen and das NoC für jede Rechenressource und bietet eine vielversprechende alternative Optimierungsstrategie zu bestehenden Ansätzen, welche auf die Integration von mehr Router-zu-Router Verbindungen setzen. Dennoch bleiben die strukturelle Regelmäßigkeit und sowie die Einfachheit der 2D-Mesh-Topologie vollwertig erhalten.

Im nächsten Schritt wird eine flexible sowie konfiguriebare Lösung zur Laufzeitüberwachung von Verkehrsflüssen im gitter-basierten NoC eingeführt. Diese erlaubt dem CMP die globale Sicht auf das aktuelle Kommunikationsverhalten der ausgeführten Anwendungen zu gewinnen. Gleichzeitig kann gezeigt werden, dass diese Verkehrsüberwachung als kombinierte Hardware/Software-Lösung integrierbar ist und über ausreichend Adaptivität, Funktionalität sowie Flexibilität verfügt um das zu erwartende Kommunikationsverhalten in modernen CMPs mit ausreichender Genauigkeit für verschiedene Anwendungsfälle erfassen zu können.

Abschließend wird ein konkreter Anwendungsfall für die Kombination des QMesh mit der ausgearbeiteten Laufzeitüberwachung vorgestellt, welche bemerkenswerte Leistungssteigerungen für eine Vielzahl von explorierten Verkehrsmustern erlaubt. Dabei konnten Verbesserungen der Netzwerksättigung bis zu Faktoren von 5 sowie eine durchschnittliche Reduktion der Paketverzögerungen von 40% bis zu 80% gegenüber dem 2D-Mesh nachgewiesen werden. Gleichzeitig stiegen sowohl der Leistungsverbrauch als auch der Flächenbedarf für die Implementierung um moderate 50%. Es konnte weiterhin gezeigt werden, dass das software-basierte und zentral organisierte Verkehrsmanagement den Anforderungen moderner CMPs standhält.

V DECLARATION

I declare that this thesis was composed independently and that it does not incorporate, without any acknowledgement, any material previously submitted for a degree in any university. To the best of my knowledge this doctoral thesis does not contain any materials previously published or written by any other person except where due reference is made.

_____ _____

Place, Date Authors signature