
Networks-on-Chip for heterogeneous 3D Systems-on-Chip

Dissertation

zur Erlangung des akademischen Grades

Doktoringenieur (Dr.-Ing.)

von M. Sc. Jan Moritz Joseph

geb. am 27. März 1990 in Berlin, genehmigt durch die

Fakultät für Elektrotechnik
und Informationstechnik der

O t t o – v o n – G u e r i c k e
U n i v e r s i t ä t M a g d e b u r g

Gutachter:

Prof. Dr.-Ing. Thilo Pionteck

Prof. Dr.-Ing. Alberto García-Oritz

Prof. Dr. Gert Jervan

Promotionskolloquium am 20. August 2019

ABSTRACT

Recently proposed manufacturing methods enable the production of heterogeneous 3D System-on-Chips (3D SoCs), in which dies manufactured in different technology nodes are stacked and vertically interconnected. This allows for the combination of components with different electrical requirements on a single chip. One example of such systems are “Vision System-on-Chips” that combine analog image sensors, analog-digital converters and digital signal processing.

Communication architectures using the advantages of heterogeneity have not been considered prior to this thesis. We propose *Asymmetric 3D Networks-on-chips* (A-3D NoCs) for this purpose. A-3D NoCs are NoCs that target heterogeneous 3D SoCs and further exploit the specific properties of silicon dies in disparate technologies. Asymmetry for 3D NoCs is a novel design paradigm, offering advantages in performance, power consumption and area. It further unleashes the full potential of heterogeneous integration for the network itself.

The approach of this thesis is twofold: First, we consider A-3D NoCs on a system level to take the advantages of heterogeneous integration for NoC planning including, optimized topology and placement. Second, we improve routers on an architectural and micro-architectural level to tackle technological challenges emerging from heterogeneity. This thesis provides the following specific contributions:

The design space of A-3D NoCs is modeled. Technology-specific features are taken into account, in contrast to models for conventional on-chip networks. This results in a deeper understanding of the design space and leads to a systematic approach for its exploration. Next, we propose an analytical approach to system-level optimizations by means of modeling via linear programs for exact solutions and heuristics for efficient solutions. For the first time, models account for technology-specific properties of routers and components. Furthermore, routers and components are placed simultaneously. This combination of models and methods is necessary because properties vary between layers. Area reductions of up to 26.3% over related approaches are possible while maintaining performance of the network. Plus, we achieve white space reductions of up to 21.6% over traditional linear models for placement of components. Thereafter, we propose a comprehensive design and simulation framework including processes for design space exploration and analysis. As a novel feature, it accounts for the described structure and proposed hierarchical order of the design space and it includes technology-specific properties. Furthermore, it considers architectures that are only reasonable due to heterogeneity. This allows for rapid prototyping using

parameter sets. Therefore, we propose well-reasoned models. These are implemented in a simulator which is embedded in a design and simulation process, including tools for benchmarking, reporting and analysis. The performance of the simulator is close to state-of-the-art, despite extended features, and it allows for faster design space exploration using parameter sets. Going further, we improve the router architecture by means of better buffer depth and buffer distributions. As a novel feature, memory of routers is divided over heterogeneous layers to reduce area and power in those that are more expensive. This allows for cost reductions. We achieve up to 8.3% area savings and 5.4% power reductions at a minor performance loss of 2.1%; Area reduction of 28% and power savings of 15% are possible at 4.6% performance loss. Hereafter, we optimize routing in A-3D NoCs. For the first time, general principles and models are proposed, which measure the impact of heterogeneity on router area, speed, packet latency and router throughput for any set of commercial technologies. Based on these models, concrete implementations of routing algorithms for heterogeneous 3D SoCs are proposed. Latency reductions of between $1.5\times$ and $6.5\times$ for packets between layers in different technologies and about $1.6\times$ for packets within slow technologies are achieved for a given case study. After that, network throughput reductions are analyzed with the aforementioned models, which are a result of non-purely synchronous communication between routers due to heterogeneity. A co-design of a router architecture with the proposed routing algorithms allows for up to $4\times$ throughput increase with negligible hardware overhead.

This thesis comprehensively introduces asymmetry as a novel paradigm for NoCs targeting heterogeneous 3D SoCs. Further, it contributes methods and tools for their design, optimizations on their system level and novel router architectures and microarchitectures. The contributions tackle the most important challenges for implementation of communication networks in heterogeneous 3D systems. Thereby, the design of diverse heterogeneous 3D SoCs is made possible for many new application fields.

KURZFASSUNG

Durch Fortschritte in der Fertigung ist es mittlerweile möglich, heterogene 3D System-on-Chips (3D SoCs) zu erstellen, in denen Chips unterschiedlicher Technologie gestapelt und vertikal verbunden sind. Dieses erlaubt es, Komponenten mit verschiedenen elektrischen Anforderungen in einem einzigen Chip zusammenzuführen. Ein Beispiel dafür sind sogenannte „Vision System on Chips“, die analoge Bildsensoren, Analog-Digital-Wandler und digitale Signalverarbeitung kombinieren.

In dieser Arbeit werden erstmals Kommunikationsnetze erforscht, die Vorteile aus heterogener Integration ziehen. Dafür wird Asymmetrie als ein neues Entwurfparadigma für Networks-on-chips (NoCs) eingeführt. Asymmetrische 3D NoCs, kurz A-3D NoCs, sind solche NoCs, die eigens für heterogene 3D Chips entworfen werden und die dabei spezifische Charakteristika einzelner Chipebenen ausnutzen. Dieses bietet viele Vorteile, da Flächenkosten, Energieverbrauch und Leistungsfähigkeit verbessert werden. So kann das große Potential heterogener Integration auch für Kommunikationsnetze ausgeschöpft werden.

Wir verwenden einen zweistufigen Ansatz: Zuerst werden Optimierungen von A-3D NoCs auf Systemebene erforscht. Bei der Netzwerkplanung werden Eigenschaften der Netzwerktopologie und der Platzierung von Routern und Komponenten verbessert. Danach berücksichtigen wir die einzelnen Router und schlagen Optimierungen derer Architektur und Mikroarchitektur vor, sodass Größe, Leistungsaufnahme und Geschwindigkeit verbessert werden.

Diese Arbeit leistet die folgenden wissenschaftlichen Beiträge: Wir modellieren den Entwurfsraum von A-3D NoCs und berücksichtigen dabei, im Gegensatz zu bisherigen Modellen, spezifische Eigenschaften der Fertigungstechnologien. Durch dieses Verständnis ist eine systematische Erkundung des Entwurfsraums möglich. Als Nächstes optimieren wir die Systemebene von A-3D NoCs und stellen dafür ein analytisches Modell in Form eines ganzzahlig linearen Programms vor. Darauf aufbauend wird ein heuristischer Algorithmus präsentiert, der effizient eine Lösung für große Eingaben findet. Dabei berücksichtigen wir, dass sich die Eigenschaften von Routern und Komponenten zwischen Chipebenen ändern. Damit verbessern wir die Flächenausnutzung um bis zu 26,3% bei gleicher Leistungsfähigkeit des Netzwerks für Anwendungsdatenströme gegenüber vergleichbaren Ansätzen. Außerdem verringern wir die ungenutzte Chipfläche um bis zu 21,6% im Vergleich zu typischerweise verwendeten linearen Modellen bei der Platzierung von Komponenten. Weitergehend stellen wir Methoden und Werkzeuge zur empirischen Exploration

des Entwurfsraums vor, um Architektur und Mikroarchitektur von Routern zu verbessern. Wir berücksichtigen dabei sowohl die Struktur des Entwurfsraums als auch die Eigenschaften der zu modellierenden Systeme und erstellen ein wohlbegründetes Simulationsmodell. Eine schnelle und effiziente Erkundung des Entwurfsraums, im Sinne eines Rapid Prototyping, ist möglich, da das Simulationsmodell mittels einstellbaren Parametern flexibel an sinnvolle Architekturen angepasst werden kann, ohne dass diese einzeln implementiert werden müssen. Der Simulator bietet, trotz erweiterter Funktionen, eine ähnliche Simulationsgeschwindigkeit wie Konkurrenzprodukte. Des Weiteren optimieren wir Routerarchitekturen. Durch eine innovative Herangehensweise, in der Router über mehrere, heterogene Chipebenen aufgeteilt werden, können wir die Größe und Leistungsaufnahme des Routers minimieren: Durch eine verbesserte Speicherstruktur wird der Router bis zu 8,3% kleiner und benötigt bis zu 5,4% weniger Leistung. Dieses wird durch eine kleine Verminderung der Latenz von 2,1% erkauft. Es ist sogar möglich, bis zu 28% Routerfläche und bis zu 15% Leistungsaufnahme zu sparen, wenn eine 4,6%-ige Reduktion der Latenz im Netzwerk akzeptiert wird. Als nächsten Punkt optimieren wir Routing in A-3D-NoCs mit dem Ziel einer besseren Übertragungslatenz. Wir stellen dafür erstmals allgemeingültige Modelle für beliebige Kombinationen von Technologien vor, die den Einfluss von Heterogenität auf Größe und Taktrate von einzelnen Routern sowie die Latenz und den Durchsatz des gesamten Netzwerks erfassen. Daraus leiten wir allgemeine Prinzipien für Routingverfahren ab. Außerdem zeigen wir, wie heterogene Integration bei konventionellen Routerarchitekturen den Netzwerkdurchsatz beschränkt. Durch neue Routingverfahren und eine verbesserte Mikroarchitektur können wir die Übertragungslatenz um bis zu Faktor 6,5 zwischen Ebenen in unterschiedlichen Technologien reduzieren. Die Latenz bei Übertragung zwischen Routern in einer langsameren Technologie wird um etwa Faktor 1,6 verbessert. Beide Ergebnisse werden für einen realistischen Testfall mit kommerziellen Fertigungstechnologien ermittelt. Ein integrierter Entwurf von Routingverfahren und Routerarchitektur ermöglicht weiterhin einen höheren Durchsatz um Faktor 4 bei zu vernachlässigendem Flächenmehrbedarf.

Damit stellt diese Arbeit nicht nur ein neues Entwurfparadigma für NoCs in heterogene 3D Systeme vor, sondern präsentiert auch Methoden und Werkzeuge für deren Entwurf, deren Optimierung auf Systemebene und neue Routerarchitekturen und -mikroarchitekturen. Es werden die wichtigsten Hürden für die Implementierung von Kommunikationsnetzen in heterogenen 3D Systemen beseitigt. Somit ebnet diese Arbeit den Weg für den Entwurf heterogener 3D Chips für neue Anwendungsgebiete.

PUBLICATIONS

PUBLICATIONS AS FIRST AUTHOR

Original works in peer-reviewed international journals

- [JM 1] **Joseph, J. M.**, L. Bamberg, D. Ermel, B. R. Perjikolaie, T. Drewes, A. García-Ortiz, and T. Pionteck. "Routing in Network-on-Chips for heterogeneous 3D System-on-Chips." In: *under revision in IEEE Access* (2019).
- [JM 2] **J. M. Joseph** with Bamberg, L. (authors contributed equally), I. Hajar, R. Schmidt, T. Pionteck, and A. García-Ortiz. "Simulation Environment for Link Energy Estimation in Networks-on-Chip with Virtual Channels." In: *Integration* 68 (2019), pp. 147–156. ISSN: 0167-9260. DOI: 10.1016/j.vlsi.2019.05.005.
- [JM 3] **Joseph, J. M.**, C. Blochwitz, A. García-Ortiz, and T. Pionteck. "Area and power savings via asymmetric organization of buffers in 3D-NoCs for heterogeneous 3D-SoCs." In: *Microprocessors and Microsystems* 48 (2017), pp. 36–47. ISSN: 0141-9331. DOI: 10.1016/j.micpro.2016.09.011.

Original works in peer-reviewed international conferences

- [JM 4] **J. M. Joseph**, D. Ermel, T. Drewes, L. Bamberg, A. García-Ortiz, and T. Pionteck. "Area Optimization with Non-linear Models in Core Mapping for System-on-Chips." In: *International Conference on Modern Circuits and Systems Technologies* (2019), pp. 1–4. DOI: 10.1109/MOCAS.2019.8742035.
- [JM 5] **Joseph, J. M.**, L. Bamberg, G. Krell, I. Hajar, A. García-Ortiz, and T. Pionteck. "Specification of Simulation Models for NoCs in Heterogeneous 3D SoCs." In: *International Symposium on Reconfigurable Communication-centric Systems-on-Chip*. IEEE, 2018, pp. 1–8. DOI: 10.1109/ReCoSoC.2018.8449387.
- [JM 6] **Joseph, J. M.**, L. Bamberg, S. Wrieden, D. Ermel, A. García-Ortiz, and T. Pionteck. "Design method for asymmetric 3D interconnect architectures with high level models." In: *International Symposium on Reconfigurable Communication-centric Systems-on-Chip*. IEEE, 2017, pp. 1–8. ISBN: 978-1-5386-3344-1. DOI: 10.1109/ReCoSoC.2017.8016143.
- [JM 7] **Joseph, J. M.**, M. Mey, K. Ehlers, C. Blochwitz, T. Winker, and T. Pionteck. "Design space exploration for a hardware-accelerated embedded real-time pose estimation using vivado HLS." In: *International Conference on ReConfigurable Computing and FPGAs*. IEEE, 2017, pp. 1–8. ISBN: 978-1-5386-3797-5. DOI: 10.1109/RECONFIG.2017.8279785.

- [JM 8] **Joseph, J. M.**, C. Blochwitz, and T. Pionteck. "Adaptive allocation of default router paths in Network-on-Chips for latency reduction." In: *International Conference on High Performance Computing & Simulation*. IEEE, 2016, pp. 140–147. ISBN: 978-1-5090-2088-1. DOI: 10.1109/HPCSim.2016.7568328.
- [JM 9] **Joseph, J. M.**, T. Winker, K. Ehlers, C. Blochwitz, and T. Pionteck. "Hardware-Accelerated Pose Estimation for Embedded Systems using Vivado HLS." In: *International Conference on ReConfigurable Computing and FPGAs*. IEEE, 2016, pp. 1–7. DOI: 10.1109/ReConFig.2016.7857173.
- [JM 10] **Joseph, J. M.**, S. Wrieden, C. Blochwitz, A. García-Oritz, and T. Pionteck. "A simulation environment for design space exploration for asymmetric 3D-Network-on-Chip." In: *International Symposium on Reconfigurable Communication-centric Systems-on-Chip*. IEEE, 2016, pp. 1–8. ISBN: 978-1-5090-2520-6. DOI: 10.1109/ReCoSoC.2016.7533908.
- [JM 11] **Joseph, J. M.**, C. Blochwitz, A. García-Oritz, and T. Pionteck. "Area and power savings via buffer reorganization in asymmetric 3D-NoCs for heterogeneous 3D-SoCs." In: *Nordic Circuits and Systems Conference*. IEEE, 2015, pp. 1–4. ISBN: 978-1-4673-6576-5. DOI: 10.1109/NOR-CHIP.2015.7364370.
- [JM 12] **Joseph, J. M.** and T. Pionteck. "A cycle-accurate Network-on-Chip simulator with support for abstract task graph modeling." In: *International Symposium on System-on-Chip*. IEEE, 2014, pp. 1–6. ISBN: 978-1-4799-6890-9. DOI: 10.1109/ISSOC.2014.6972440.

PUBLICATIONS AS COAUTHOR

Original works in peer-reviewed international journals

- [JM 13] L. Bamberg, **J. M. Joseph**, T. Pionteck, and A. Garcia-Ortiz. "Crosstalk optimization for through-silicon and through-interposer vias by exploiting temporal signal misalignment." In: *Integration 67* (2018), pp. 60–72. ISSN: 0167-9260. DOI: 10.1016/j.vlsi.2019.04.009.

Original works in peer-reviewed international conferences

- [JM 14] L. Bamberg, **J. M. Joseph**, R. Schmidt, T. Pionteck, and A. García-Oritz. "Coding-aware Link Energy Estimation for 2D and 3D Networks-on-Chip with Virtual Channels." In: *International Symposium on Power and Timing Modeling, Optimization and Simulation* (2018), pp. 222–228. DOI: 10.1109/PATMOS.2018.8464171.
- [JM 15] C. Blochwitz, J. Wolff, M. Berekovic, D. Heinrich, S. Groppe, **Joseph, J. M.**, and T. Pionteck. "Hardware-Triplestore – a Hardware-centric Database for Semantic Web." In: *International Conference on Field-Programmable Technology*. IEEE, 2018.
- [JM 16] T. Drewes, B. Gurumurthy, **Joseph, J. M.**, D. Broneske, G. Saake, and T. Pionteck. "Efficient Inter-Kernel Communication for OpenCL

- Database Operators on FPGAs." In: *International Conference on Field-Programmable Technology*. IEEE, 2018.
- [JM 17] C. Blochwitz, R. Klink, **Joseph, J. M.**, and T. Pionteck. "Continuous live-tracing as debugging approach on FPGAs." In: *International Conference on ReConfigurable Computing and FPGAs*. IEEE, 2017, pp. 1–8. DOI: 10.1109/RECONFIG.2017.8279783.
- [JM 18] C. Blochwitz, J. Wolff, **Joseph, J. M.**, S. Werner, D. Heinrich, S. Groppe, and T. Pionteck. "Hardware-Accelerated Radix-Tree Based String Sorting for Big Data Applications." In: *Architecture of Computing Systems*. Springer, 2017, pp. 47–58. ISBN: 978-3-319-54999-6.
- [JM 19] T. Drewes, **Joseph, J. M.**, and T. Pionteck. "An FPGA-based prototyping framework for Networks-on-Chip." In: *International Conference on ReConfigurable and FPGAs*. IEEE, 2017, pp. 1–7. DOI: 10.1109/RECONFIG.2017.8279775.
- [JM 20] C. Blochwitz, **Joseph, J. M.**, R. Backasch, S. Werner, D. Heinrich, S. Groppe, and T. Pionteck. "An optimized radix-tree for hardware-accelerated dictionary generation for semantic web databases." In: *International Conference on ReConfigurable Computing and FPGAs*. IEEE, 2015, pp. 1–7. DOI: 10.1109/ReConFig.2015.7393291.

ACKNOWLEDGMENTS

This work was funded by Deutsche Forschungsgesellschaft's grant PI 447/8.

CONTENTS

I	PRELUDE	
1	NEXT DECADE'S COMMUNICATION ARCHITECTURES	3
2	CONTRIBUTION	5
2.1	Objectives	5
2.2	Working hypothesis	5
2.3	Outcomes	6
2.4	Outline	6
II	BACKGROUND	
3	3D TECHNOLOGIES	11
3.1	Potentials	12
3.2	Challenges	13
3.3	Fabrication	16
3.4	Applications (heterogeneous 3D chips)	21
4	NETWORKS-ON-CHIP	25
4.1	Packet transmission	26
4.2	Router architecture	28
4.3	Timing of routers	30
4.4	Flow control	30
4.5	Virtual channels	30
4.6	Network topology	31
4.7	Routing algorithm	32
4.8	Application mapping	35
4.9	Evaluation	35
4.10	3D NoCs	40
III	INNOVATION	
5	SPECIFICATION AND DESIGN SPACE	45
5.1	Definition	45
5.2	Limitations of today's approaches	45
5.3	Potentials	46
5.4	Challenges solely present in A-3D NoCs	47
5.5	Typical example for NoCs in heterogeneous 3D SoCs	49
5.6	Design space of A-3D NoCs	49
6	SYSTEM-LEVEL OPTIMIZATION	57
6.1	Introduction	57
6.2	Problem formulation and technology model	60
6.3	Mixed integer linear program model	64
6.4	Heuristic algorithm	75
6.5	Performance and computational complexity	82
6.6	Results	84
6.7	Discussion	93

6.8	Conclusion	99
7	TOOLS AND METHODS FOR SIMULATION	101
7.1	Models	103
7.2	Tools	109
7.3	Exploration process	117
7.4	Analysis	119
7.5	Results and discussion	121
7.6	Conclusion	125
8	OPTIMIZATION OF ROUTER MEMORY	127
8.1	Buffer distributions and buffer depths	127
8.2	Routers with optimized buffer distribution	129
8.3	Routers with optimized buffer depths	131
8.4	Results	132
8.5	Discussion	138
8.6	Conclusion	140
9	OPTIMIZATION OF ROUTING AND ARCHITECTURES	141
9.1	Influence of heterogeneity on routing	141
9.2	Modeling technology heterogeneity	142
9.3	Modeling communication	145
9.4	Limitations of routing due to heterogeneity	147
9.5	Tackling latency: Routing algorithms	150
9.6	Tackling throughput: Router architectures	162
9.7	Results	165
9.8	Discussion	170
9.9	Conclusion	172
IV	FINALE	
10	SUMMARY AND OUTLOOK	175
10.1	Asymmetry – a novel design paradigm	175
10.2	Impact of future technologies	176
V	APPENDIX	
A	SYSTEM-LEVEL OPTIMIZATION	181
A.1	Overview of symbols	181
A.2	Component, router and tile count	182
A.3	Definitions, notations and prerequisites	183
A.4	Cost function	184
A.5	Constraints	185
A.6	Auxiliary variables	211
A.7	Heuristic algorithm	222
B	SIMULATION MODELS	225
B.1	Application model	225
	GLOSSARY	229
	BIBLIOGRAPHY	235

Part I

PRELUDE

COMMUNICATION FOR THE NEXT DECADE'S CHIPS

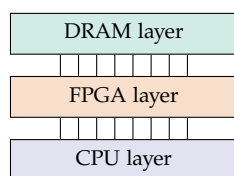


Figure 1: 3D CPU-FPGA-DRAM [1].

It is an everlasting challenge to increase the performance of computer systems with tight area and power constraints. One promising approach is 3D integration, in which Through-Silicon Vias (TSVs) directly connect the stacked dies. The promises of 3D integration are manifold: smaller area footprint, less power consumption and increased performance [2]. Be-

side these incremental advantages, 3D manufacturing provides a unique feature which is currently impossible with 2D production: It allows for heterogeneous integration. In traditional chip design, chips can only be manufactured in a single technology node, since it is produced using a single die. If different components such as memory (digital), logic (digital), mixed-signal components, or sensors (analog) are implemented on the same chip, these will present contradicting electronic requirements: For instance, mixed-signal components cannot be implemented in small technology nodes; memory and processing profits from small technology nodes. In addition, technology nodes can be optimized either for memory or logic. These contradicting requirements limit the integration of components on single dies. 3D technology can overcome this, since dies manufactured in disparate technologies can be stacked. Each die is optimized for electronic requirements of its components. Furthermore, tight integration is possible, since 3D technology removes the need for long and slow interconnects between different dies. One example of such chips is shown in Figure 1: Dies optimized for memory, logic and even field programmable gate arrays (FPGAs) are stacked to build high performance processors [1]. There is a clear trend towards a higher degree of heterogeneity for 3D technology. The EU has recently founded a project, FlexTiles [3], targeting the integration of a heterogeneous 3D platform including a manycore layer and reconfigurable layer, with an investment of 5,336,349 €. As reported in the project, “the interconnection is a tremendous issue for using reconfigurable technologies with manycore. The solution proposed by the FlexTiles consortium focuses on the interfaces between the layers of the 3D stacked chip to ensure an efficient access to the reconfigurable layer by the manycore layer” [3].

One everlasting challenge in chip design poses the integration of even larger component counts. This results in more sophisticated

requirements to communication architectures, which connect these components. Efficient and scalable communication architectures have become one major concern. Bus systems, which are widely used today, are limited in scalability. Furthermore, globally asynchronous locally synchronous (GALS) clock frequencies are the most common design principle for heterogeneous designs. Bus systems are challenged because these would implement global wires spanning multiple clock domains. Networks-on-Chip (NoCs) [4] promise scalability and non-purely synchronous communication. NoCs implement a micro-network on the chip to diminish the limitations of bus systems.

3D technology and heterogeneous integration provide novel challenges and perspectives for interconnect architectures: The unique electrical characteristics of individual dies due to heterogeneity result in a larger number of influential factors. In addition, properties of 2D and 3D links are different and, partially, even complementary. Controlling technological heterogeneity and communication complexity are essential factors for successful application of 3D System-on-Chips (SoCs). The relevance of heterogeneous 3D systems and their complex communication requirements are widely recognized [5–9].

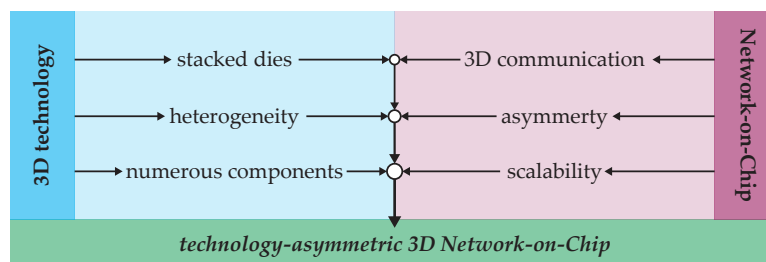


Figure 2: Heterogeneous 3D integration and packet based on chip transmission yield asymmetric 3D NoCs.

To summarize, there are three major trends today: First, 3D technology allows for stacked dies. Second, the dies can be manufactured in different nodes, i. e. are heterogeneous. Third, 3D systems will possibly consist of many components. These trends, as shown in Figure 2, require a new concept for on-chip communication and its central component should be a on-chip network: It has the capacity for 3D communication. It can be asymmetric to exploit the advantages of heterogeneity. It offers scalability to connect many different components. In this thesis, *technology-asymmetric 3D NoCs* are introduced, these being the most promising solution for such networks. Thereby, this work lays the foundations to build communication architectures for the next decade's chips.

CONTRIBUTION

2.1 OBJECTIVES

New production methods enable the design of heterogeneous 3D SoCs, which consist of stacked silicon dies manufactured with disparate technologies. In contrast to homogeneous 3D SoCs, this allows for adjusting the technological characteristics of each die to specific requirements of components. Heterogeneous 3D SoCs are the most promising concept to combine sensing and processing in a single chip to, for instance, build SoCs targeting vision applications. Powerful, flexible and scalable communication infrastructures are required to fully exploit their potential. Current interconnect networks tacitly assume homogeneous 3D SoCs and do not consider the influence of different technology parameters on the communication architectures.

This thesis proposes technology-asymmetric 3D Network-on-chips (A-3D NoCs) targeting communication in heterogeneous 3D SoCs. These networks span multiple layers manufactured in disparate technologies and incorporate distributed interconnect components among layers, specialized communication infrastructure per layer, as well as specialized inter-layer links using TSVs. A-3D NoCs are orthogonal to related works, since most of them consider NoCs or dedicated links in combination with homogeneous 3D SoCs.

Two main innovations are targeted. First, we exploit the specific technology characteristics at system level. This requires re-evaluation and extension of existing approaches. Second, new interaction mechanics between components are invented, at architectural and micro-architectural level. The components may be spatially distributed among layers. We innovate by means of systematic design methodologies, using a novel design framework and providing automatic optimization and custom architectures.

2.2 WORKING HYPOTHESIS

The working hypothesis is that existing 3D NoCs do not fully exploit the intrinsic asymmetry present in heterogeneous 3D SoCs. Limitations posed by heterogeneity as well as additional degrees of freedom provided by the individual characteristics of layers have not been considered so far. This thesis demonstrates how existing approaches must be extended, since they neither consider the influence of disparate technologies nor the potentials arising from spatial distribution of components among layers.

2.3 OUTCOMES

The main outcome of this thesis is a deeper understanding of 3D NoCs targeting heterogeneous 3D SoCs; in particular, a deeper understanding of limitations and potentials of heterogeneity in the context of communication networks. This thesis provides the following novelty: It is the first work that exploits advantages and tackles limitations of heterogeneous 3D integration for NoCs by introduction of the novel paradigm of *asymmetric 3D NoCs*. Due to the special characteristics of heterogeneity, the design of A-3D NoCs cannot rely on existing methods. Asymmetry in 3D NoCs is not considered, in terms of research, as up to date. This thesis contributes novel methods, tools and architectures to fill the gap. A-3D NoCs can only be designed by a holistic approach that considers aspects from system level down to microarchitectural level.

This thesis provides the following concrete contributions:

1. Systematic design methodologies are provided to optimize A-3D NoCs. The approach exploits the additional degrees of freedom arising when considering the specific technological characteristics of each individual die.
2. A full-fledged simulation and evaluation framework is provided for the analysis of A-3D NoCs. Its open source simulator supports cycle-accurate models as well as high level models for fast evaluation. It is capable of accounting for technology-specific parameters in components, which are possibly distributed on neighboring dies.
3. Design optimizations are proposed; novel architectural and microarchitectural optimizations for routers are introduced under exploitations of heterogeneity to improve performance, power and area.

2.4 OUTLINE

In the first part, the technological background is explained: In Chapter 3, manufacturing of heterogeneous 3D chips is introduced¹. The work specially focuses on potentials and challenges of 3D technology, fabrication of vertical interconnects and application examples for heterogeneous 3D systems. In Chapter 4, NoCs are introduced. After defining the basic building blocks, their functionality is presented. Finally, 3D NoCs are discussed and we highlight related approaches to this thesis.

The second part presents the innovation provided by this thesis: In Chapter 5, we introduce the 3D NoCs targeting heterogeneous 3D SoCs: *Asymmetric 3D NoCs*. The chapter comprises potentials, challenges, open research issues, a detailed model of the design space

¹ The terms 3D chips and 3D integrated circuits (3D ICs) are synonymous, here.

and approaches to design space exploration. Here, we explain the differentiation of system-level, architectural and microarchitectural optimization of these networks. In Chapter 6 NoCs are optimized on system level. We describe the optimization problem and formulate it as a mixed-integer linear program. Based hereupon, we contribute a heuristic algorithm for an efficient solution. It is applied to heterogeneous 3D chip example sets and case studies to highlight properties that are not present in a homogeneous 3D scenario. In Chapter 7, we focus on simulation of asymmetric 3D NoCs. We introduce models, tools, processes and analysis and thus are covering the full range of empirical design space exploration with simulations. To conclude the section on innovation, concrete (micro-) architectural optimizations for routers are presented, which exploit potentials of asymmetry. In Chapter 8, novel router architectures with asymmetric buffer distributions and buffer depths are proposed. In Chapter 9, we focus on routing. Novel routing algorithms are presented. Further, a mesochronous router architecture is presented to handle the clock differences present due to heterogeneity. In summary, this part regards all aspects of asymmetric 3D NoCs.

The third part brings us to the conclusion of the thesis. In Chapter 10, we summarize the findings of the thesis. We highlight their practical applicability by explaining how they are useful when designing A-3D NoCs based on communication requirements for a heterogeneous 3D chip. Finally, we explain the impact of future technologies, which are expected to become available in the next decade, on the thesis' results.

Part II
BACKGROUND

Throughout the last decades, there has been a clear trend towards requirements for lower power consumption, higher performance and smaller area footprint of computing systems. Therefore, difficulty and complexity of chip design are steadily increasing. The demands cannot always be fulfilled by traditional 2D structures, due to limitations such as global interconnect delays and restricted functionality on single substrates. Stacking silicon dies to 3D structures is one solution; different manufacturing processes for package, die and wafer level were proposed in 2005 [10]. The promises of the 3D technology are significant and therefore many architectures have been proposed. Among them are 3D Vision Systems-on-Chip (3D VSoCs) with stacked sensors [11] and 3D FPGAs [12]. Only recently, 3D technology reached market maturity with the foundation of the hybrid memory cube (HMC) consortium in 2012 and the release of first products containing stacked high bandwidth memory (HBM) in June 2015, such as the AMD Radeon R9 graphic cards. Obstacles of 3D technologies were mainly their high production and development costs. Their potentials however are worth the effort.

Stacked architectures use either a *2.5D approach*, where a silicon interposer connects wire-bonded or micro-bump-bonded dies, or a *true 3D approach*, where TSVs directly connect the stacked dies. 3D chips distinguish themselves from 2.5D systems by more dense and shorter vertical interconnects. Hence, communication within and in between dies has similar performance. TSVs are the most common and most promising technique for vertical connections. 3D chips are better than 2.5D chips, in terms of price, power consumption and long (i. e. slow) interconnects. Thus, 3D chips unleash the full potential of 3D systems but are more difficult to produce. 3D chips are a truly innovative design paradigm: They do not only offer higher integration density, but also allow for heterogeneous chip design: Dies are manufactured in different technologies, such as analog, mixed-signal or (purely) digital, and then are stacked and are connected by TSVs [13, 14]. 3D technology is gaining maturity with promising application areas such as *high-performance processors* with DRAM stacking [15, 16], *wireless sensors* [17], and *3D Vision Systems-on-Chip (3D VSoCs)* [11, 18].

3.1 POTENTIALS

3.1.1 *Electrical performance*

The maximum achievable clock frequency of a chip is dependent on the length of the longest wire. In general, the worst case approximation for the longest interconnect is the summed length of the die edges. For a squared 2D chip with a total area of A , the length of a single die edge is \sqrt{A} estimating the wire length as $L_{2D} = 2\sqrt{A}$ (Figure 3a). For 3D designs, this length is reduced; we do not consider the length of vertical links due to their relatively small length. For a 3D chip with two layers and a total area of A , a single edge is $\sqrt{A/2}$ long (Figure 3b). Hence, the maximum wire length is estimated with $L_{3D, 2 \text{ dies}} = \sqrt{2}\sqrt{A}$, which is considerably smaller than L_{2D} . A 3D chip with N die layers and a total area of A reduces the edge length to $\sqrt{A/N}$, resulting in a maximum wire length estimation of $L_{3D} = \sqrt{N}\sqrt{A}$ (Figure 3c). In a statistical analysis with a variable separation of die connections, the link length between layers was considered [19], which reveals a similar trend for two and three stacked dies despite our simplification. In summary, the potential of 3D integration lies in reduced wire length, which is generally beneficial and results, for instance, in higher clock frequency and hence performance.

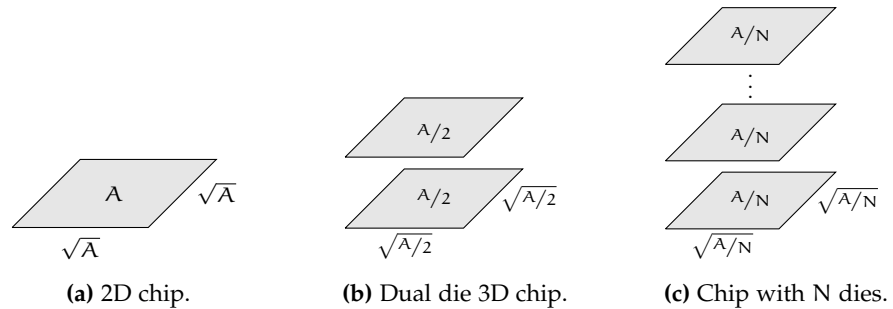


Figure 3: Wire length estimation for chips, with a total area of A .

3.1.2 *Power consumption and noise*

Due to reduced interconnect wire length, the average load capacitance, the resistance, and the number of repeaters in long links are decreased. Furthermore, in 3D systems less total wiring is required. Therefore, the energy consumption of the interconnect is reduced, which has a significant share of the overall chips energy consumption. In addition, the reduction of load capacitance reduces noise from simultaneous switching events. Plus, shorter wires have less coupling noise due to lower wire-to-wire capacitance. [20]

3.1.3 *Form factor*

3D integration can provide important contributions to build systems with a reduced area footprint. For instance, stacking memory on processing layers reduces the access times along with providing improved packaging efficiency. Typically, reducing the area footprint of memory relies on utilization of the advancement in lithography, which increases the memory density. Stacking provides similar advantages without using next generation lithography. 3D stacked memory is an example of form factor benefits [21]. Another example is complementary metal-oxide-semiconductor (CMOS) sensors, in which 3D links directly connect from the sensor to its back side [11].

3.1.4 *Heterogeneous integration*

Layers in 3D chips can be manufactured in disparate technologies which yield heterogeneous systems combining multiple functions on a single SoC. Components may require different, or even contradicting, electrical characteristics and the technology of individual dies can be aligned. Due to aligned technology properties and component requirements this leads to potentially higher performance or lower power consumption. This allows integration of digital logic, memory, analog, and mixed-signal components in a single SoC by stacking them into different die layers. Heterogeneity can provide advantages in many SoC applications, as introduced in Section 3.4.

3.2 CHALLENGES

3.2.1 *Technologies and fabrication*

One important issue lies in the fabrication process, during which multiple layers are bonded. The lamination process must fulfill two requirements: First, the performance of each individual layer must not be degraded. Second, the lamination must ensure bonding of the layers for the lifetime of the system. Furthermore, packaging solutions must be developed for heterogeneous 3D systems that are considerably more complicated in comparison to 2D chips. Plus, vertical interconnects for both signal and power transmission become one key factor for the fabrication of 3D systems, as they must provide high density and quality, otherwise the potentials in performance of 3D chips are diminished or even reversed [22]: Too low density reduces performance by up to 10% in comparison to 2D; up to 20% increased performance is possible for the published setting in comparison with 2D chips.

3.2.2 *Yield*

In comparison to 2D systems, 3D chips of the same size have a reduced yield, since the particle contamination probability is proportional to the layer area and the number of mask levels and each additional layer adds area. Thus, yield is proportional to the layer count. The yield of a bonding process effecting the interconnect's reliability further decreases the overall yield. However, there are two advantages for increasing 3D systems' yield: First, the process complexity of single layers can be significantly reduced, since only a single type of component is implemented per layer. For instance, memory processing has significantly less mask levels than logic ones. Second, the area of each individual layer may be reduced and layers can be tested individually before bonding. Currently, the yield of 3D chips is rather low due to the very low yield of TSVs. [20]

3.2.3 *Testing*

New testing methods must be developed, which test both the bonding process and the functionality of layers. With traditional methods, bonding is not tested and dies are only considered individually. This results in three challenges [23]: First, methods must be developed to generate useful input per individual layer to test the overall function of the system. Second, minimizing the additional circuitry per layer for testing is beneficial for system's costs. Third, bonding test methods are sought after. Solutions have been proposed for TSVs [24].

3.2.4 *Thermal issues*

The advantage of shorter interconnects in 3D systems is a double-edged sword: Shorter interconnects allow for a larger component count in equal area. As a consequence, the power density in these areas is higher, yielding higher heat loss. Furthermore, some layers are not adjacent to a heat sink, thus thermal hot-spots in the inner of 3D chips emerge. Both factors result in performance degradation and faster wear-out effects. To tackle thermal issues of 3D systems, design methods must be developed considering thermal effects. Better heat sinks must be found, as well. Many works on thermal issues have been published. For instance, [25] optimizes the location of TSVs based on thermal considerations. Thermal driven floor planning is proposed in [26] and thermal driven standard cell placement in [27] for 3D chips. Planning of vias is conducted considering their thermal properties in [28, 29]. Furthermore, analytical models are introduced for the thermal performance of 3D chips [30].

3.2.5 Interconnect architectures

3D systems result in novel research questions for interconnect architectures: First, clock and power distribution networks are more relevant. Second, well-known 2D noise mitigation techniques must be reconsidered due to inductive and capacitive coupling between adjacent dies [31]. This can be easily exemplified considering a dual die chip, in which the metal die of the digital layer is bonded front-to-front to the analog die. As a consequence, switching located in the digital die result in noise spikes in the analog die. Third, 3D links have different or even complementary characteristics in comparison to 2D links. For full 3D integration the properties of TSVs are not fully understood and currently under investigation [32]. Heterogeneous integration provides further challenges, as tackled in this thesis.

3.2.6 Physical design

Physical design of chips is a difficult problem since it comprises NP-hard subproblems. Deterministic algorithms do not allow finding optimal solutions in acceptable, i. e. polynomial, computation time. Therefore, heuristic algorithms, that allow for efficient yet approximate solutions, are researched. The physical design of 3D systems itself comprises similar subproblems as 2D physical design. While its computation complexity is not higher, it poses two important new challenges. First, there are more feasible points to consider in the solution space. Second, traditional algorithms cannot be applied directly and are modified due to interdependencies between steps of the layout synthesis, which are handled interdependently for 2D.

The physical design is composed of the following steps: The subject of *floor planning* is to place blocks, i. e. parts of the circuit, while minimizing the chip area used and the wire length of the interconnects. This is shown in Figure 4. Many approaches have been proposed tackling the computational complexity via different partitioning schemes, e. g. [33, 34]. Algorithms considering power, temperature, and noise, beside area and wirelength, propose different objective functions [35].

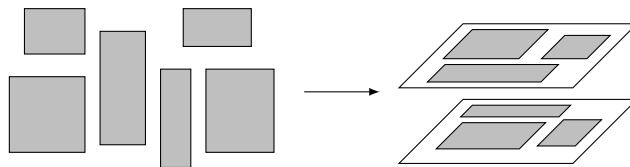


Figure 4: Floorplanning for 3D chips.

Next, in *placement*, area and wire length are minimized. For 3D systems new trade-offs emerge, which do not exist for 2D systems, since vertical and horizontal links have different properties (cf. Section 3.3.4). Moving from 2D to 3D placement, the components must

be located in a volume rather than on a die. Therefore, diverse approaches are proposed, e. g. [36]. The problem is rather hard, partially due to discrete dies [37, 38], i. e. integer variables in optimizations. More recently, with the introduction of via-middle processes, placement of components and vertical links must be considered simultaneously ([39], area requirements of TSVs, Section 3.3.4). One approach is to exploit white spaces (area without any other placed cells) [40]. The placement of TSVs can be done prior, after or along with components. The impact of the order is evaluated in [41].

As a final step, *routing* determines the actual position of nets in the interconnect. If the position of vertical links is known, *routing* will be reduced to traditional, efficient 2D routing algorithms [42]. Otherwise, techniques are required to route vertical links as well. *Enbody et al.* demonstrated the complexity of the problem in 1991 [43] for a simple example; in fact, it is NP-hard [44]. Therefore, the research focuses on finding efficient heuristic algorithms [45].

Few commercial tools are available for fully-integrated 3D chip design. Cadence advertises a 3D chip tool flow [46], yet it is currently, to the best of the author's knowledge, not all-embracing. Features such as TSV positioning are still missing out on good solutions. Only recently, academia adopted the Cadence 2D flow for 3D. Components are placed in 3D via size shrinking, partitioning and size restoring for two and three layers [47]. There hardly exist any tools targeting heterogeneous 3D chips.

3.3 FABRICATION

In general, full 3D integration can be separated into *monolithic* and *stacked* 3D chips. For monolithic fabrication a sequential process is set up, in which the whole chip is produced on a single wafer. Monolithic fabrication is still experimental. For stacked 3D chips, individual layers are manufactured on different wafers and bonded afterwards. Stacking does not require changes in the conventional fabrication process per layer and is the more practical approach. For full 3D integration, high vertical interconnect density is essential.

3D chips with TSVs are the most promising and most advanced approach to 3D integration. Face-to-back bonding is possible by stacking die layers with vertical interconnections traversing the silicon substrate of each layer. The fabrication of TSVs is well-controlled and can be realized at acceptable costs. High vertical interconnection density can be achieved as well. The potential of 3D chips with TSVs is impressively demonstrated by the application of TSVs for vertical interconnection, like in commercial applications such as Micron's Hybird Memory cube [48].

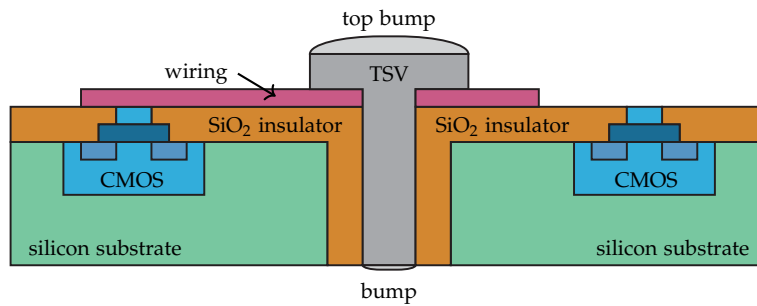
3.3.1 *Through-silicon viae*

Figure 5: Sectional drawing of a TSV.

A TSV is a *vertical interconnect passing completely through silicon*. This is shown in Figure 5. As depicted, there are CMOS components, e. g. transistors, located on the silicon substrate. In the middle of the Figure, a TSV is located, which is insulated against the substrate with an SiO_2 insulator. This TSV passes through the complete silicon. On the top of the insulator, wiring connects the CMOS component on the left-hand side with the TSV. The TSV itself has two bumps, on the top and the bottom, which can be used for vertical interconnection between multiple dies of a 3D chip. This is depicted in Figure 6. TSVs can be used for vertical interconnection between more than two layers and offer high interconnect density. This thesis relies extensively on diverse characteristics of TSVs. Therefore, these are discussed in detail in this section.

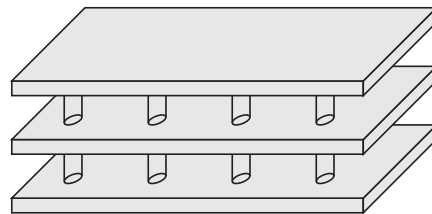


Figure 6: Three-layer 3D chip with TSVs for vertical interconnection.

The TSV was originally invented by William Shockley and granted as US Patent “Semiconductive wafer and method of making the same” in 1962 [49]. A process was patented, in which holes are etched into silicon. These can be filled with a conductive material. Etching holes into silicon is used for TSV production to this day. In the 1990s a significant advancement in the production of TSVs was the invention of the BOSCH process, patented in [50]. The BOSCH process is a pulsed or time-multiplexed etching process to produce vertical holes in silicon. It is shown in Figure 7. The process iteratively conducts two steps until a hole is etched:

ETCHING For vertical ionic plasma etching the silicon is charged with a bias voltage for directional bombardment with plasma. Sulfur hexafluoride (SF_6) is used. The etching is shown in Figure 7b and Figure 7d.

DEPOSITION The lateral walls of the TSV must be passivated, which prevents (or minimizes) further, lateral etching. Different deposition materials can be used; octafluorocarbon (C_4F_8) is the most common. The deposition step is shown in Figure 7c.

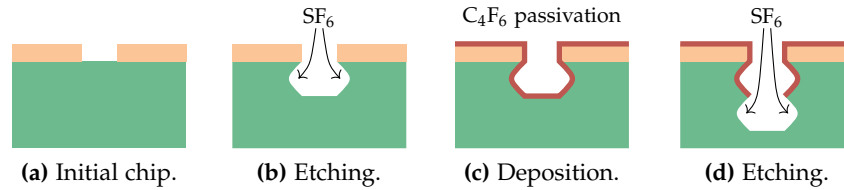


Figure 7: BOSCH process.

After the BOSCH process etches a hole through the silicon, it can be filled with a conductive material. Therefore, the edges of the hole must be covered with a barrier to prevent diffusion of the conductive material into the silicon. Typically, TiN or TaN is used. The coating process is done at approx. 623.15 K.¹ As filling material, copper, tungsten, or poly-Si is filled into the hole at high temperatures of approx. 725 K. The selection of the material depends on its compatibility with the rest of the manufacturing process, most commonly copper.

There are three major issues during the production of TSVs, which are shown in Figure 8. First, the shape and the tapering of the TSV must be controlled. In general, slight tapering, as shown in Figure 8a, is favorable due to better deposition. However, a uniform width of a TSV as shown in Figure 8b results in better bonding at the bottom. Second, sufficient adhesion and deposition of both barrier and filling (seed) is required. During the BOSCH process scallops can emerge from the iterative etching and deposition, as shown in Figure 8c. This irregular surface limits uniformly-distributed adhesion of barrier and filling. Therefore, as an alternative to the BOSCH process, laser drilling through the silicon was proposed [51]. This, however, is currently not realizable for state-of-the-art technology nodes. The laser drilling process is, in contrast to the BOSCH process, not sequential, and therefore, is less manageable. Third, void-free filling is important for a high quality signal paths, which requires well-controlled Cu-seed injection.

¹ Reduced temperatures to 443.15 K with highly concentrated NH_3 in the seed stream are possible [20].

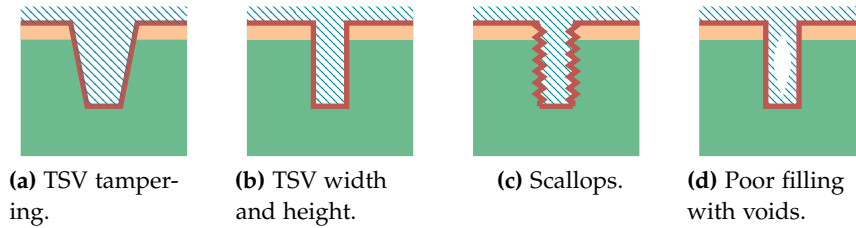


Figure 8: Common issues in TSV production.

3.3.2 TSV production methods

There are three methods of TSV production, *via first*, *via last* and *via middle*. The current rule of technology is *via middle* process flow, in which the TSV is produced between device placement and interconnection. The process flow is shown in Figure 9. At the beginning, during front-end-of-line (FEOL), devices are patterned into the semiconductor (cf. Figure 9a). The functionality of the chip emerges from these devices. Next, using BOSCH-process, a via is etched into the silicon (Figure 9b). In the following step, an insulator material (SiO_2) is placed on the complete chip and the edgings of the TSV via oxidation of the silicon (cf. Figure 9c). Due to this barrier separating the TSV from the silicon, conductive material can be filled into the TSV without interfering with the devices or the silicon. This filling with a seed is shown in Figure 9d. Next, there is a step of chemical mechanical polishing and planarization (CMP). This is conducted to remove insulator and filling material from the surface and to smooth the surface. It is realized via hybrid chemical etching and free abrasive mechanical polishing (cf. Figure 9e). Following thereafter, the components from FEOL are interconnected via wiring of the wafer within the metalization layer. This is called back-end-of-line (BEOL). During this step, the TSV is connected as well as depicted in Figure 9f. At this point, a single layer chip is produced with a TSV. For face to back bonding, CMP can be applied for TSV backside reveal, in which the bump of the TSV is exposed (cf. Figure 9g). Then, a second layer is bond as shown in Figure 9h. This process flow is currently used by many companies during their 3D chip production such as UMC, TSMC and Global Foundries [52].

3.3.3 Keep-out zone

TSVs induct stress to components in the area around the TSV. A conservative method to tackle this issue is the introduction of keep-out-zones (KOZs) around the TSV landing zone, in which no other components are placed. If the zone is large enough, the stress will not impact any components. Tensile mechanical stress is introduced around the TSV during production [53]. The coefficients of thermal expansion

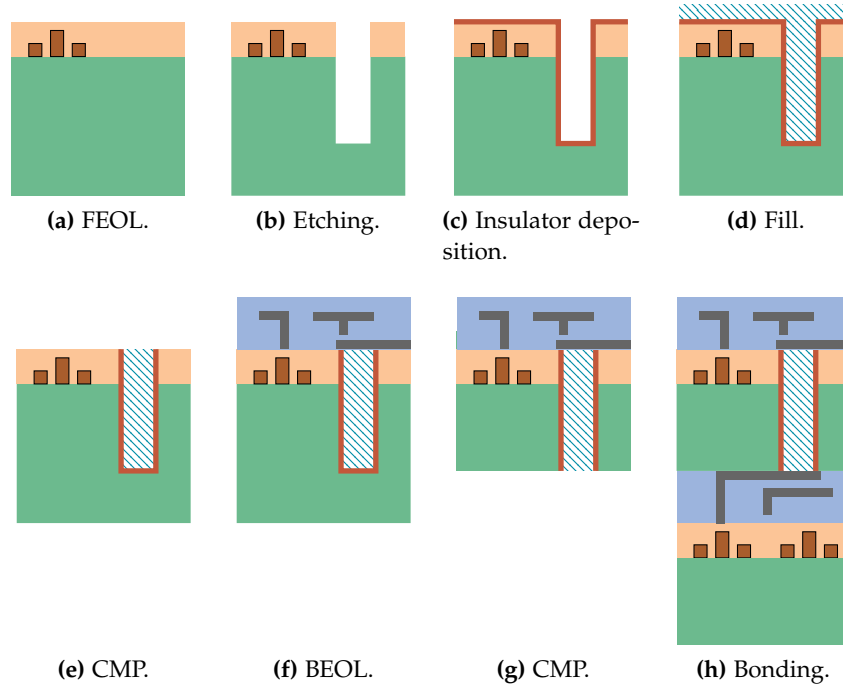


Figure 9: Via middle process flow.

of copper ($17 \times 10^{-6} \text{ K}^{-1}$) and silicon ($3 \times 10^{-6} \text{ K}^{-1}$) mismatch. Copper is filled into the etched hole in the silicon at high temperature, typically at 573.15 K for TSV fabrication. This high temperature influences the electron mobility and therefore has a negative impact on timing and reliability of components [54]. Furthermore, during cooling to room temperature, copper contracts faster than silicon. Thus, the surface of the silicon around the TSV is pulled [55] and components, which are located in the near surroundings of a TSV, could potentially be damaged [56]. The size of the KOZ is, hence, a critical design parameter, which was for instance studied in [57].

The KOZs have two important impacts on chip design. First, the *influence of TSVs on the overall chip area is large*, since the KOZs are large. Second, *TSVs do only induce additional KOZ area costs if the TSV connects downwards*, i. e. connect through the silicon. Many works, especially those considering chips on a high level of abstraction, are not aware of both design factors and therefore, introduce too many TSVs for realistic chip design. For instance *Xu et al.* [58] optimizes the number and location of TSVs for symmetric 3D NoCs, yet does not consider any area costs of TSV. In another example, in [59], a NoC router is split up over multiple layers in a 3D chip and the components of the router communicate via TSVs.

3.3.4 Characteristics and properties of TSVs

The electrical properties and characteristics of TSVs are not fully understood [32]. Many current works are based on incomplete models; for instance, energy models are incomplete [JM 6]. Some TSV characteristics, however, are already known and show that 3D links realized via TSVs have different, even contradicting properties in comparison to 2D links within each layer of a 3D chip. This allows for a large, unprecedented optimization potential.

The following properties of TSVs are currently well-understood. TSVs are short in comparison to 2D links, since the distance between two layers in a 3D chip is approx. $50\ \mu\text{m}$. The pitch of TSVs could be, in theory, even smaller, but is currently limited by the stability of each chip layer to avoid cracking and bending. TSVs show large capacitive coupling because of their conductive substrate and the increased number of aggressors despite their length. The coupling is comparable to the values of 2D links. Inductivities play a minor role [60, 61]. TSVs are rather wide, approx. $1\text{-}2\ \mu\text{m}$. In combination with the high conductivity of the seed material and the short length of each TSV, the resistance of TSVs is low ($R = \rho A^{-1}$). 2D links, in comparison, have a larger resistance by a factor $100\times\text{-}1000\times$, mainly due to their larger length.

In summary, the delay of 3D links via TSVs is smaller than those of 2D links. However, the energy consumption, and thus the thermal temperature rise, of TSVs under load is high. 2D links can be used to connect to TSV arrays.

3.4 APPLICATIONS (HETEROGENEOUS 3D CHIPS)

Heterogeneous integration will be advantageous if and only if components of a SoCs have different requirements to the technology. Three main application areas provide this requirement: High performance processors, combining digital circuits and memory; wireless sensors, combining analog or mixed-signal components for the sensors and digital circuits for the data processing; and 3D vision system on chips (3D VSoc), combining digital circuits for the data processing with analog co-processors. The following examples review state-of-the-art.

For *high performance processors*, heterogeneity increases energy efficiency of multi-core processors [62]. Dynamic Random Access Memory (DRAM) stacking [63] allows to mitigate the memory wall in processor based systems. Interleaving of dedicated dies with either memory or processing increases performance in comparison to dies with both [64]. Examples of these kinds of 3D designs are [65–68]. These designs comprise two different silicon technologies and at least two silicon layers, one optimized for logic and the other for memory. However, the dies are most often designed as individual components

and thus the holistic system aspect is limited to the adjustment of the number and locations of TSVs. 3D technology for memory is steadily gaining in maturity; even commercial applications are available, such as the HMC. In the HMC four DRAM dies on a logic (SerDES) are stacked on top of an interface chip [69] to increase memory bandwidth. A more advanced technique, which does not only combine RAM blocks per die but uses several dies to form a RAM block, is DiRAM (dis-integrated memory) [70]. Here, memory cells and access transistors of one RAM block are spread among dies. An integrated 3D stacked multi-core processor with stacked memory layers is presented by [71]. In contrast to the previously mentioned work, four processor layers are stacked on top of each other in addition to two RAM layers. [1] advocates to increase the degree of heterogeneity with an intermediate FPGA layer between central processing unit (CPU) and DRAM layers as shown in Figure 1; this reduces the power consumption by 47.5% versus a baseline system, but requires a richer and a more complex communication architecture than the traditional ones found in multiprocessor systems.

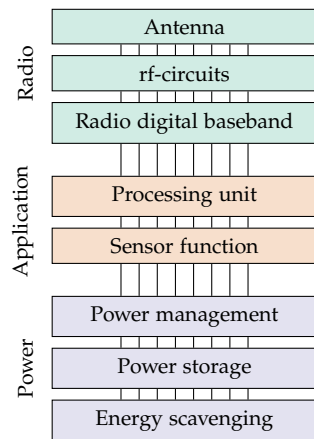


Figure 10: “e-Cube” [17].

In terms of 3D VSoCs [11], high-speed digital technologies are connected with more conservative mixed-signal and analog technologies. The layers are optimized for their functions: There are layers for (analog) photo sensor arrays, mixed-signal layers for signal conversion, memory layers with frame buffers, and digital layers for a foveal processor array. In recent research, a SoC is proposed for self-driving cars that realizes up to 10,000 frame per second integrating analog sensors, mixed-signal conversion, digital image processing and memory on a single die [72]. Going further, the mixed-signal layers can be

For *wireless sensors*, a current approach [17] proposes to use 3D stacking of more than eight heterogeneous layers. The work targets ambient intelligence applications. Small chips with broad functionality are required for sensing and interaction with the environment. This so-called “eCube” is a wireless, autonomous module concept which consists of multiple stacked layers for power scavenging and saving, application, sensing, processing and radio communication (see Figure 10). The requirements of bandwidth, latency and flexibility of each part differs.

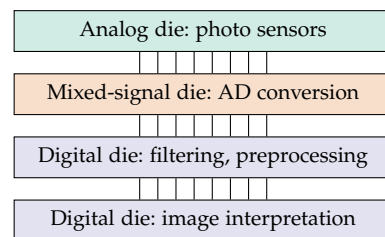


Figure 11: 3D VSoC: exemplary design [11].

used to implement analog accelerators. For instance, these calculate a cellular neural network [73]. Implementations of such accelerators have been proposed for 180 nm [74] and 130 nm [75] CMOS technology. An example for VSoC is shown in Figure 11. Data flow from CMOS sensors on top to the analog digital conversion below. Then, the data are transmitted to the digital processing. Complex communication patterns emerge both between and within digital and mixed-signal components: Calculations rely on results of other components due to the complex image processing tool flow.

Beyond these design paradigms, the VSoC “Viscube” [11] is a 3D SoC design composed of multiple layers with stacked sensors, mixed-signal components and processors. As a defining feature, this chip design is composed of multiple, heterogeneous, and stacked layers and thus a truly integrated 3D design. An entire 3D sensor-processor circuit design is presented within the Viscube project [11]. This 3D sensor-processor does not only readout the sensor data but offers image processing as well as basic feature extraction capabilities on-chip. A 3D layout was chosen to meet the strict performance requirements and thus allowing for a frameless sensor plate and high frame rates. The design is intrinsically heterogeneous as it comprises four different layers. The top layer is the sensor layer that is connected to the subjacent layer using a bump-bonding interface. The second is a mixed-signal processing layer that is connected to the frame buffer layer via TSVs. TSVs are also used to connect the digital processor layer that is the lowest layer of the 3D stack. The communication infrastructure is adapted to the expected communication patterns inside such a system, e.g. only local links between processors. An even more disruptive architecture which combines the sensing and data-processing as in [11] with the radio part as in [17] has been proposed in [18].

The number of components in SoCs is steadily increasing, since this allows for higher performance and more diverse functions. This brings traditional interconnects architectures to limits: Dedicated point-to-point connections are not flexible enough for the complex communication patterns. Fully connected crossbars yield unbearable costs since their area grows quadratically with the number of components. Bus systems are limited in arbitration for many components. In contrast, *Networks-on-Chip* (NoCs) [4] offer better scalability. NoCs are interconnection architectures, in which components are connected via a network of routers and data is transmitted via packets. The number of routers can be adjusted to the number of components. One distinctive feature between NoCs and the other interconnect architectures is that NoCs implement packet based transmission, in contrast to wire-based transmission of data, as in the case of direct links, crossbars and buses.¹ This follows the principle “route packets, not wire” [76]. Therefore, the payload data are split up into packets and configuration data are attached with information, for instance on the packet’s destination address.

There is a wide variety of NoC architectures and possible implementations. An example is shown in Figure 12: There are different processing elements, each of which is connected to a network interface. This is the entry point for data into the network. Data is

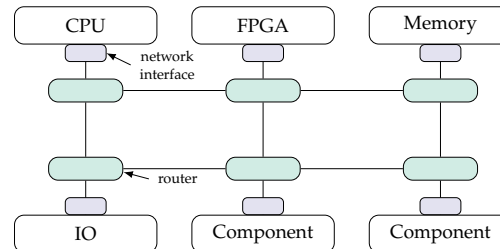


Figure 12: NoC with different components.

transmitted via the routers. These three parts are the typical building blocks of NoCs; these have the following functions:

PROCESSING ELEMENT: Processing elements (PEs) connect via a network interface to the interconnection network. Each PE represents one component of the SoC. Addresses are associated with PEs, which are used for the calculation of packet paths. The set of a router, a network interface and a PE at the same location (i. e. with the same address) is called tile.

NETWORK INTERFACE: Network interfaces (NIs) connect PEs and routers. Within the NI the data from the PE are serialized, split up

¹ The nature of buses is also wire-based in that a sender and a receiver are directly connected by a wire, shared among resources, by arbitration.

into packets, and vice versa. The NI connects to the local port of the router.

ROUTER: Routers transmit data through the network. Routers are connected among each other following a given topology. A very simple mesh network is shown in Figure 12; routers are connected via their four output ports, which are called north, east, south and west to other routers. Routers may have different number of ports in other network topologies (connection schemes between routers). Routers receive and send packets to transmit data. The correct output port is calculated in each router per packet following a routing algorithm. If the output port is not in use, i.e. not blocked, the router sends the data via an internal small crossbar to the next downstream router. Otherwise, depending on the router architecture, the data are held back or stored until a path is available.

There are still many challenges for NoC design. For instance, NoCs are ultimately limited in scalability, as well, since every new router increases the maximum communication latency due to an additional hop. Although less severe in comparison to other interconnect paradigms, it leads to large performance degradations for some applications, e.g. [77]. Therefore, optimized architectures are proposed, which for instance accelerate data streams in network loads [JM 8]. Another challenge lies in router costs. This is tackled for example by small and efficient routers which are frugal in terms of implementation area and power consumption [78]. Furthermore, the incorporation of today's emerging technologies, such as 3D production chips, is a relevant research topic, which is also within the scope of this thesis.

4.1 PACKET TRANSMISSION

The switching method determines how data are transmitted through the network. Switching can be either packet based, in which data are transmitted using packets without path reservation, or circuit switching, in which data packets are transmitted along a reserved path.

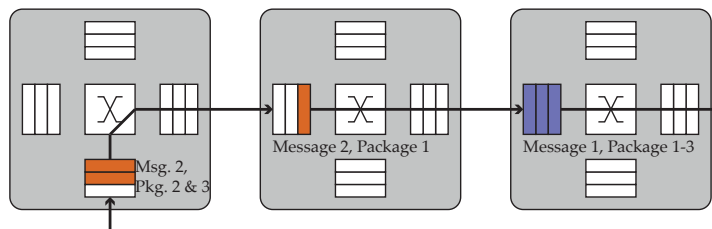


Figure 13: Store-and-forward switching: Message transmission is started after prior transmission is completed.

Store-and-forward switching is a packet based method, in which packets are completely stored in a router before they are forwarded

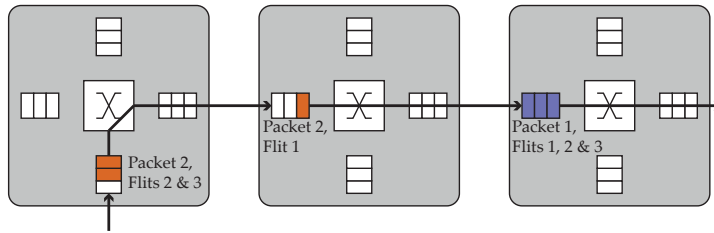


Figure 14: Virtual-cut-through switching with packets spanning multiple routers with three flits. Filled buffers are marked in gray.

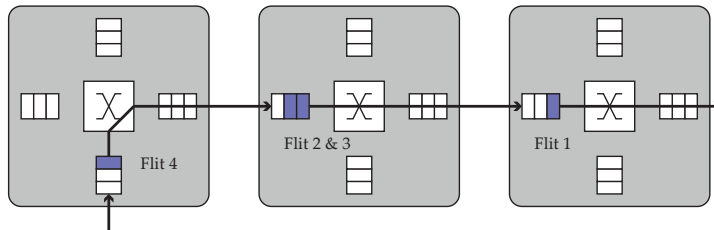


Figure 15: Wormhole switching with a packet spanning multiple routers with four flits. Buffers with flits are gray.

to the next downstream router as shown in Figure 13. Since each message is completely stored in routers along the path, large buffer space is required. Packet switching was the first transmission scheme used in NoCs (e.g. [79]). Today it is obsolete due to its large area costs. In virtual cut through switching, a packet header is allowed to leave the router before receiving the entire packet. The header allocates crossbar connections and this allocation is kept until the packet is completely transmitted. If the header is blocked, the buffer space for the remaining packet part will be allocated and the part of the packet which is still in transmission can be stored at the blocking router. This is shown in Figure 14. Wormhole switching is the most common packet based switching method. It was proposed by *Dally et al.* [80]. Packets are split up into flits (short for flow control digits or flow control unit). The head flit contains routing information, the body flits contain payload and tail flits finish the transmission of the packet.² Flow control is applied at flit level rather than on packet level: Each flit must secure its buffer space downstream separately. This allows reducing buffer sizes required in routers, since packets must not be stored completely within each router, as shown in Figure 15. If a packet is stalled in a wormhole based network, it can block many links. This is called head of line blocking and results in congestion, which is propagated through the network. Virtual-cut-through and wormhole switching have the same latency in a zero load model i.e. without congestion. Virtual-cut-through switching has reduced latency and higher acceptance rates for light traffic and equal buffer sizes [81]. In any case, wormhole switching is widely used in NoCs due to its acceptable area costs. It can be found in industry-near re-

² Depending on the implementation, tail flits can be the last body flit.

search such as Intel’s Teraflops [82], the Tile64 [83] or the TRIPS [84] chips.

When circuit switching is used, a virtual circuit from the source to the destination is established in the network. The data are transmitted through this connection. After successful transmission, the connection is terminated. One possible application is quality of service (QoS) in the network, since circuit connections allow for service guarantees of bandwidth and throughput. For instance, the *Æ*ternal NoC [85] consists of separated subrouters, of which one is optimized for guaranteed throughput traffic using circuit switching and the other subrouter targets best-effort traffic by wormhole switching. The QoS is managed by service guarantees, which determine the chosen subnetwork. Routers can combine multiple switching methods to add advantages.

4.2 ROUTER ARCHITECTURE

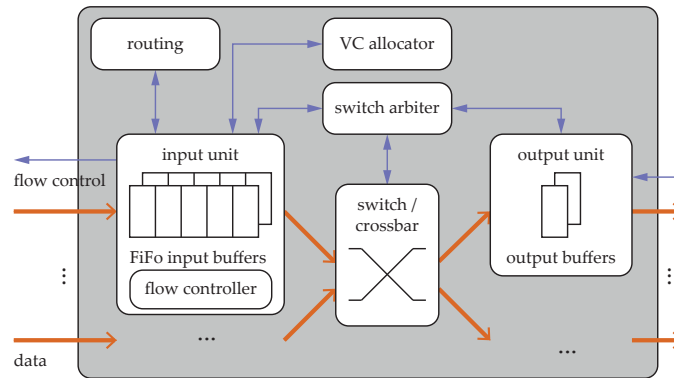


Figure 16: Schematics of an exemplary router design.

The router architecture is the hardware structure in which a router is build. An exemplary, basic router architecture is shown in Figure 16. It is not optimized and is used to exemplify the functionality of NoC routers. It is based on [86]. Data are transmitted from the input ports (left) to the output ports (right). Flits are stored in the input unit’s buffers. Based on the routing information in the head flits, the routing unit calculates a path for the packet. The switch arbiter allocates the crossbar using the state of the output unit and the results of routing calculation and virtual channel (VC) allocation (cf. Section 4.5). Data transmission can, for instance, be done using status fields in the input and output units (see [86]). The microarchitecture of such a router consists of the following components:

BUFFER: There are two buffer locations in the exemplary router design. In the input unit, First-in-first-out (FiFo) buffers are used. Each buffer has the same width as a single flit and is some flits deep. Buffers are replicated per VC and per port. There are also buffers in

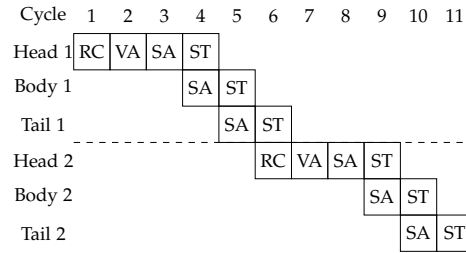


Figure 17: Exemplary router pipeline in a zero load model.

the output unit in the exemplary design, which reduces the length of the critical path but increases area costs. Since memory adds significant area overhead to the router design, many works reduce the buffer area, for instance by buffer sharing [87, 88] or by (buffer-less) deflection routing schemes [78].

CROSSBAR: In general, crossbars consist of multiplexers and demultiplexers, which pairwise connects input and output ports. These are configured by the arbiter. Optimized crossbar designs to save costs are proposed in [89].

ROUTING UNIT: The routing decision is computed here for incoming packets. There are different routing algorithms; in general, those with smaller area and power overhead are preferred.

VC ALLOCATOR: The VC controller allocates a virtual channel for incoming packets. Therefore, the VC controller reads the status of each input port and VC. It selects the next free VC available at the output port based on the state of the downstream router. In general, VCs enable deadlock-free routing methods, a better router utilization and QoS. VCs are explained in Section 4.5.³

ARBITRATION UNIT: The arbitration unit selects which pairs of input and output ports are allowed to transmit data via the crossbar in each clock cycle. It reads the status of input channels and finds a cover between the requests. An optimal solution of the coverage problem is impossible, so heuristics are implemented. The most common is round robin due to low costs, but priority-based, congestion-aware or fixed orders are also published.

FLOW AND LINK CONTROLLER: The flow and link controller manages the data transmission between adjacent routers or routers and PEs. It ensures that data are not duplicated or lost during transmission and it prevents buffer overflow by flow control. Flow control is explained in Section 4.4.

4.3 TIMING OF ROUTERS

The timing behavior of an exemplary, basic router architecture is shown in Figure 17 under zero load, in which only two subsequent packets traverse the router without congestion or collision. After the head flit of the first packet was received, in the first cycle the routing is calculated (RC). In the second clock cycle, the virtual channel is allocated for the packet (VC). Next, the switch is arbitrated (SA). Finally, the first head flit traverses the switch (ST) in the fourth clock cycle. In parallel, the body flit wins switch arbitration. This is repeated for all body flits and, finally, the tail flit, which resets the routing calculation and the virtual channel allocation for the next packet. Then, the subsequent packet is processed. Other router architectures require less clock cycles to set up a connection.

4.4 FLOW CONTROL

Flow control assures that packets are neither lost nor duplicated. It is on packet level or flit level, depending on the switching method. There are two basic schemes for flow control. First, *ready-valid method* or *on-off method* uses two binary signals: Valid indicates that the sender is providing data. Ready indicates that the receiver can accept data. Thus, if both signals are true, a flit will be transmitted. Second, *credit-based method* has a counter in the sender which stores the available space in the receiver. The counter is decremented for each data unit sent and incremented for each data unit removed from the receiver's buffers. Both methods provide the required functionality. A credit-based method is more costly in terms of area. However, it has the advantage of better buffer utilization because the equivalent ready signal is generated locally in the sender and therefore must not be transmitted with a delay.

4.5 VIRTUAL CHANNELS

Virtual channels (VCs) are a time multiplexing of a physical channel between senders and receivers: If a packet is blocked, another packet on a different VC will be allowed to pass the channel in the otherwise idle time. This reduces the impact of head-of-line blocking in wormhole switching [91]. VC-based wormhole and virtual cut through switching are compared in [92]. The latency of both methods is comparable, yet a much higher throughput can be achieved using VCs for equal buffer capacity. Therefore, wormhole switching with VCs is preferable compared to virtual-cut-through switching. Routers with VCs have higher energy consumption and yield a larger area,

³ Please note that allocators may also be separated between inputs and output. We do not explain this architecture here for the sake of brevity and kindly refer to [90].

since buffers are replicated per VC in every physical channel. In [77] it is found that more than four VCs per channel only add a marginal performance advantage. There are different selection strategies for VCs. Dynamic VC allocation in each router allows for time-efficient multiplexing of the physical channel, which can be realized for instance using a round-robin arbiter. VCs that are assigned to packets during payload generation and do not change during transmission can be used for QoS by prioritization of channels.

4.6 NETWORK TOPOLOGY

The network topology is the interconnection scheme between routers. It can be defined as the topology graph of the network.

Definition 4.1 (Topology graph). The *topology graph* of a NoC is given by the digraph $N = (R, E_N)$, in which the set R of vertices in the network consists of all routers r_i , with $i \in \{1, \dots, |R|\}$, and the set of directed edges $e_{i,j} \in E_N$ models the connections between the routers r_i and $r_j \in R$.

The graph is not directed, since links are bidirectional, usually.⁴ Routers are not self-connected, i.e. $E_N = \{(i,j) \mid i,j \in R, i \neq j\}$. Network topologies are characterized as follows:

NETWORK DIAMETER: The average diameter is defined as the maximum shortest distance in counted hops in the network. In a network with large diameter, packets travel many hops to reach their destination in the worst case. In general, small network diameters are preferable [93].

AVERAGE DISTANCE: The average distance, counted in hops, is calculated by the average of the shortest distance between all pairs of routers. Large average distances have a negative influence on performance, since it increases the average transmission latency.

NODE DEGREE: The node degree denotes the number of ports in routers. Routers with fewer ports generally have reduced area costs but yield a possibly worse network performance.

BISECTION WIDTH: The bisection width is defined as the number of edges that can be removed before the network bisects. For a large number the network is more prone to errors due to link failures.

NUMBER OF LINKS: The amount of links in a network should be large, since it increases the network's bandwidth.

There are many works on the influence of different topologies. Most common is 2D mesh, in which routers are located in a grid and neighbored routers connect as shown in Figure 18a. It has an obvious

⁴ Please note that link are unidirectional from an architectural point of view. However, from a network perspective they are bidirectional, because in nearly all proposed practical implementations connected pairs of routers have links in both directions. Hence, we prefer modeling via an undirected graph.

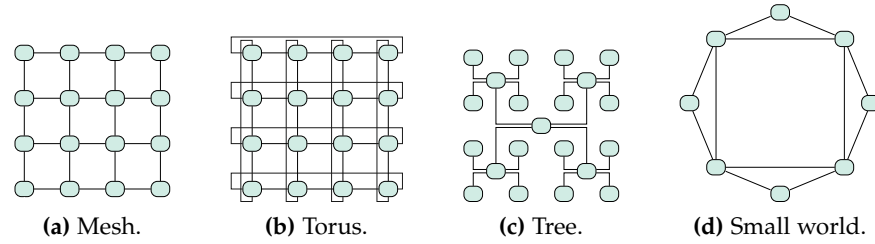


Figure 18: NoC topologies.

structure and allows for lightweight routing algorithms. This topology is also used in industry-near research. For instance, in 2007 Intel implemented a prototype, the Teraflops, with 80 core and a NoC as interconnect following mesh topology [94]. A 2D mesh topology with n and m routers per dimension has a rather large diameter of $m + n - 2$ [95]. The bisection width is also rather large with $\min(n, m)$ and the number of links is $2(m(n - 1) + n(m - 1))$ [95]. The node degree is between 3 and 5 depending on the router's position. The average distance is $(m + n) / 3$ [95]. A mesh topology can be extended to a torus by connecting the outer routers to peers at the opposing side of the network (Figure 18b). Torus topology reduces the average hop distance, but is difficult to implement in production chips due to wire length restrictions and layout complications. It is gaining attention again with optical NoCs [96] and 3D technologies [97] due to reduced layout constraints. Another topological type, which is rather popular, are tree-based networks. One example is shown in Figure 18c. Sending data via the network requires moving them up and down in the tree, depending on the number of components, their location and the tree structure. Routing algorithms are rather simple, in particular for binary trees. Finally, the advantages of small world graphs such as a small diameter can be used in NoCs as well, [98] (Figure 18d).

4.7 ROUTING ALGORITHM

The routing algorithm, or routing function, calculates the path of packets from source to destination. The calculation depends on the topology of the network and might as well consider the state of the network. The routing function is defined as ([99]): $S : N \times N \rightarrow P(E_N)$, in which $P(E_N)$ denotes the power set of E_N . Routing functions possess the following properties:

UNIQUE ADDRESSES OF NODES: A unique address is be assigned to each node. Otherwise addressing is ambiguous.

DEADLOCK FREEDOM: Deadlocks stall the network and thus prevent packet delivery. Therefore, deadlock freedom must be guaranteed by the routing algorithm. Deadlocks are explained in detail in Section 4.7.1.

LIVELOCK FREEDOM: Packets travel without ever reaching their destination in a livelock. This prevents packet delivery. Livelock freedom must be proven. Livelocks are explained in Section 4.7.2 in further detail.

NoCs can implement two options for the locality of the calculation of the routing algorithm. First, in *source routing*, the path of each packet is calculated in the source and attached to the packet. During transmission, this information is read and routing is executed without further calculations. Second, in *distributed routing* each node calculates the route with the header information in the packet from the destination address. Both techniques can be either implemented using look-up tables, in which for the destination address a routing decision is stored or via dynamic calculation using a routing function. The latter is preferable because look-up tables requires large area overhead for larger networks.

Routing function can be classified in two categories. In the first category, routing algorithms can be *minimal*, in which packets follow the shortest path between source and destination, or *non-minimal*, where longer routes can be selected. The second category distinguishes the information sources, which the routing function takes as input. If the path between source and destination is known before the transmission starts, this is called *deterministic routing*. The path does not depend on network configurations such as the link loads, congestion or faulty links. In *oblivious routing* the selected path of the packet does not depend on the state of the network, but it is selected randomly or cyclically from a given set of alternatives. If the decision about the path taken depends on the network status during run time, *adaptive routing* will be applied. Adaptive routing algorithms can adapt to influences on the network from external sources, such as faulty links and from internal sources, such as congestion or high link loads. However, adaptive routing algorithms have larger area overhead and might even further increase the total network load due to detours. Many adaptive routing algorithms have been proposed for NoCs [100–102]. It is required to differentiate between routing algorithm and selection for deterministic routing algorithms. While the routing algorithm defines a set of possible paths, the selection selects one element in this set.

Popular algorithms are turn-based models [103], in which one of the turns in the network is forbidden to avoid cyclic dependencies and deadlocks. The most common routing algorithm, which implements a turn-based model, is dimension order routing, short DOR. The packet travels along one dimension of the network until the difference between packet location and the destination is zero in this dimension. Then, this is repeated in the next dimensions until the destination is reached. In a 2D mesh NoC, this routing algorithm is

called “XY routing”, which describes that the X dimension is zeroed before the Y dimension.

4.7.1 Deadlocks

Deadlocks occur due to cyclic dependencies between packets in a network. This is depicted in Figure 19. Starting in the upper left router, the gray flits cannot be transmitted since the northern input ports of the router on the lower right-hand side is blocked by green flits. The green flits themselves cannot be transmitted, as well, since their route is blocked by red flits. Those, in turn, are blocked by blue flits. These, closing the cyclic dependency, are blocked by the gray flits again. The network stalls since none of the flits can be transmitted. Deadlocks can be identified using the channel dependency graph [104]. The vertices of this graph are channels in the NoC. For each dependency between channel pairs in the network, an edge is drawn. Each cycle in this graph represents a deadlock configuration.

In the example in Figure 19, the four channels between the four routers have a cyclic dependency. Therefore, a cycle can be found in the corresponding channel dependency graph. Deadlock freedom can be proved using the channel dependency graph with Duato’s Theorem [99]. A simple way to avoid deadlocks is the usage of routing algorithms, in which one of the turn directions is prohibited. Another solution is the introduction of VCs: The dependency between the blocked packets can be resolved by switching VCs for one of the packets, which then can be transmitted, thus resolving the cyclic dependency.

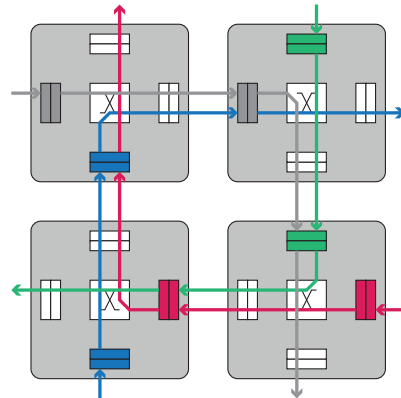


Figure 19: Deadlock configuration in the network due to cyclic dependency between four packets. The network stalls, since none of the packets can be transmitted, although subsequent routes are not blocked.

4.7.2 Livelocks

Livelocks will occur if packets infinitely travel in the network but never reach the destination. As a solution, packets can be only allowed to take minimal paths: Packets reduce the distance to their destination in each step and therefore ultimately will reach their destination. If the routing algorithm is non-minimal, nonexistence of livelocks must either be proven or mechanisms must be implemented for livelock detection. The latter can, for instance, be realized via coun-

ters, how often a packet traverses nodes. If a threshold is surpassed, a livelock will be detected and the packet will be treated specially.

4.8 APPLICATION MAPPING

The problem of assigning cores to PEs in a chip is called application mapping. The aim is reduction of communication latency for a given network and application. It is a rather important problem because it allows for efficient application execution on chips connected by a NoC. The mapping consists of two steps: First, the task graph of the application is used to allocate tasks to cores. The resulting *core graph* has cores as vertices and the bandwidth requirements are edge weights:

Definition 4.2. Core Graph: The core graph is defined as the digraph $G = (C, E)$. Vertexes $c_i \in C$ represents cores. Edges $e_{(i,j)} \in E$ model communication between cores c_i and c_j . The bandwidth requirement between the two cores is given by the edge weight $u_{(i,j)}$.

Second, during the actual mapping, cores from the core graph are mapped onto PEs under consideration of the topology graph. The objective is to reduce the communication delay of the application. The mapping problem is NP hard [105]. Mapping is divided into two classes: In *static* mapping, the core graph is assigned to the topology graph during design time. For a *dynamic* mapping, this is allowed to change during run time, which may yield performance benefits by avoiding congestion for other task communicating [106]. There are exact methods to solve the problem via formulation as a mixed-integer linear program (MILP) [107]. This allows for an exact solution, yet yields large or even unbearable compute times, so that heuristics are developed. An overview on those can be found in the survey [106].

4.9 EVALUATION

Network performance, area costs and power consumption are the most important design metrics for NoCs. This set is called Quality of Results (QoR) or PPA (performance, power, area). Other, less relevant metrics are defined as well: For instance, reusability allows to reduce time-to-market. Reliability is relevant for special applications such as space. Scalability measures how many components can be connected. Good PPA naturally increases scalability, why it is seldom considered as a separate measure.

4.9.1 Performance

The performance of a NoC measures its capability to transmit data. Different design targets can be relevant: If transmission is time-critical,

the latency must be small. If large amounts of data are transmitted, the throughput must be large.

PACKET LATENCY: Packet latency is the time for a transmission of a packet via the NoC. If a packet p is injected at time t_i and received at time t_r , the packet's latency will be $l_p = |t_i - t_r|$. Latency can be measured in simulations for every packet and usual statistical measures can be applied. It is denoted in clock cycles, independent of the implementation, or in seconds, if the clock frequency is available.

THROUGHPUT: Throughput measures available bandwidth for communication. It is defined as the number of packets n_{packets} that are accepted per period of time Δt : $\text{Throughput} = \frac{n_{\text{packets}}}{\Delta t}$. This metric is denoted in packets or flit per cycle or per second. The throughput can be converted to [Mb/s] using link widths and packet size.

There are three options to evaluate the metrics: First, analytical models are fast but demanding to formulate. Second, emulations can be conducted on transaction level (TLM) or cycle-accurate level (CA). These offer less speed but can rather effortlessly cover many effects. Finally, simulations on register transfer level (RTL) or via FPGA prototyping are possible, in which the actual circuit is modeled and therefore offers the highest precision.

Since the network load depends on the traffic patterns within the network, it is essential to compare different NoCs under the same load. Therefore, different traffic patterns, so called benchmarks, are defined. Using a clear definition of benchmarks, a fair comparison and, therefore, a valid evaluation is possible.

4.9.1.1 *Synthetic traffic patterns*

Synthetic traffic patterns follow mathematically defined spacial and temporal distributions, which do not necessarily reflect properties of a real system. Synthetic traffic patterns offer high comparability between different designs but limited modeling of realistic traffic properties. The spacial distribution can be modeled via a relationship between source and destination addresses. Assuming a n -bit long binary address b_1, b_2, \dots, b_n per component, the source destination relation is given by an address permutation $f_p : \{1, \dots, 2^n - 1\} \rightarrow \{1, \dots, 2^n - 1\}$. The most common spacial distributions are the following:

UNIFORM RANDOM: In uniform random traffic, a new destination address is drawn per packet prior to transmission following a uniform random function. It is a very common traffic pattern, which (nearly) all simulators offer. The traffic pattern is shown in Figure 20a.

HOTSPOT: In the hotspot traffic scenario, each component sends data to the same destination with the address d_1, d_2, \dots, d_n . It follows the function $f_{\text{hotspot}}(b_1, b_2, \dots, b_n) = d_1, d_2, \dots, d_n$. The spacial distribution is not a permutation since it is not bijective. Hotspot traffic is used to test the worst case communication in a network since the router at the global destination is used as much as possible. The traffic pattern is shown in Figure 20b.

TRANSPOSE: Here, data are sent to the diagonal opposite side of the network, which is equivalent to a matrix transposition permutation. This traffic pattern put high stress on both vertical and horizontal links. The traffic pattern is shown in Figure 20c.

BIT COMPLEMENT: Here, data are sent to the opposing side of the network. The permutation function of the bit complement spacial distribution is $f_{\text{bit complement}}(b_1, b_2, \dots, b_n) = \neg b_n, \neg b_{n-1}, \dots, \neg b_1$. This traffic pattern stresses vertical and horizontal links (from diagonally opposing nodes) similar to transpose traffic pattern; its distribution is more homogeneous, because some links are not loaded at all in transpose traffic pattern. The traffic pattern is shown in Figure 20d.

BIT REVERSAL: Data are sent between nodes with reversed bit addresses. The permutation function for the spacial distribution is $f_{\text{bit reversal}}(b_1, b_2, \dots, b_n) = b_n, b_{n-1}, \dots, b_1$. The pattern stresses horizontal and vertical links, locally, from locally vertically opposing nodes. The traffic pattern is shown in Figure 20e.

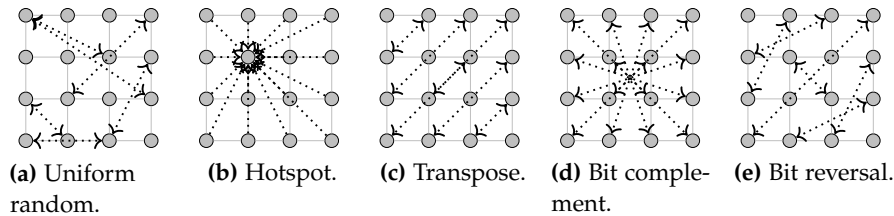


Figure 20: Synthetic traffic patterns.

In terms of temporal distribution, the injection rate is modulated from low to high to find the saturation point of the network. Injection rates are measured in flits or packets per cycle. Therefore, the average time between two injected packets/flits must leave enough idle time in between that their ratio gives the desired injection rate. This is shown in Figure 21: The PEs inject at a random time in each time slot; the length of the time slots is proportional to the injection rate. Please note, that the injection rate can be measured per component or for the whole network. If measured per component, the instances of time for injection must necessarily be different between components: Otherwise bursts in the traffic result in (virtually) higher injection rates and disproportionate traffic properties.

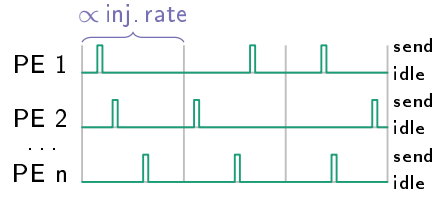


Figure 21: Modeling a random injection rate. Time-slot length (gray) is given by the injection rate and PEs send data at a random instance of time in the slots.

4.9.1.2 Task graph based traffic patterns

To increase the realism of traffic patterns without the burden of full-system simulation, task-graph-based traffic patterns have been proposed. Combining simulation of application-based data streams and router simulation on a low abstraction level is conducted in the Noc-Bench project [108]. Applications are modeled using Kahn process networks [109], a common software model. There are different benchmarks such as universal mobile telecommunication system (UMTS) modems or video encoders and decoders [110]. *Liu et al.* propose traffic patterns for NoCs based on real-world traffic patterns using a task graph in the MCSL suite [111]. The suite comprises eight benchmarks covering applications such as a sample rate converter, media encoders and decoders or robot control.

4.9.1.3 Real-world based traffic patterns

The most precise method to benchmark NoCs with real-world-based applications is a full-system simulation in parallel to the simulation of the NoC. This method suffers from very bad performance. For full-system simulation of multi-core processors, simulators such as Gem5 [112] can be used. To benchmark modern multi-core processors, the “Princeton Application Repository for Shared-Memory Computers” (PARSEC) benchmark suite [113] is proposed with focus on workloads that are emerging as representative of the design of next generation multi-core processors. It is not focused on SoC workloads.

To tackle limited performance of full system simulation, trace-driven simulation is possible. A trace of the activity of packet injection in the network is recorded within a single full-system simulation. Then, this trace can be played to generate the same traffic again. Modifications in the system architecture (not the interconnect architecture) require newly recorded traces. To cover the effects of slow interconnects, dependency-tracking can be used, as proposed in “Netrace” [114]. The traces are generated using full-system simulation of a 64-core system. To further increase the performance, these trace-driven simulations can be conducted on an FPGA [JM 19].

4.9.2 *Area*

The second important design metric is area costs. In general, it should be minimized, since auxiliary functions such as the interconnection network must occupy as little area as possible, in order to increase chip area for functional components. Usually the area costs are either evaluated via router architecture synthesis for FPGA or standard cells. In case of FPGAs, vendor specific compilers such as Xilinx Vivado are used to synthesize the very high speed integrated circuit hardware description language (VHDL) or Verilog description of routers. Since NoCs yield relatively large resource utilization on FPGAs, NoCs are normally not used to connect components. Therefore, FPGAs are used for NoC prototyping or verification rather than production purposes. A NoC performance evaluation framework for FPGAs was proposed as part of the work on this thesis [JM 19]. If NoCs are to be used in production, a synthesis for standard-cell technology will be conducted with synthesis tools; for instance from Synopsys [115] or from Cadence [116]. This also requires libraries of target technology nodes, which are available for commercial technologies from the vendors as closed-source. One finds open-source libraries for predictive technologies, as well. Synthesis for standard cells is the only reasonable evaluation for heterogeneous 3D integration because effects of disparate technologies can be revealed.

4.9.3 *Power*

Synthesis both for FPGA and standard cells provide power consumption estimations per transaction and for average usage patterns. NoC simulators sometimes also provide power estimations. Most often, power consumption of routers is estimated by simply counting events such as buffer writes, buffer reads, routing calculations or data transmissions [117]. This is highly unrealistic, since the power consumption depends on the actual transmitted data [118]. Therefore, novel methods and models for power estimation of NoCs are required, especially with heterogeneity being present.

4.9.4 *NoC evaluation via simulations*

A fair and comparable evaluation of NoCs design methods and implementations is rather complex, since an assessment in terms of performance and power consumption is complicated. The power consumption is vaguely estimated by common design tools and the estimation quality differs largely depending on the assumptions about the utilization of the on-chip network. A detailed knowledge about the network load is required for an exact power evaluation. These data can often only be received via simulations. Many simulators have been

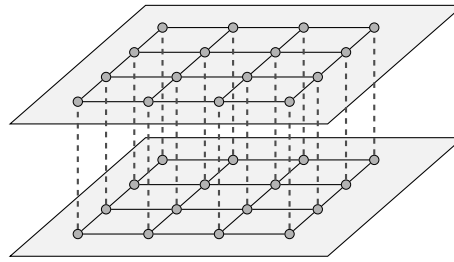


Figure 22: Exemplary 3D NoC.

proposed for this purpose. Both universal simulators, as well as specific tools dedicated to a specific use case exist (e.g. [119]). According to the *NoC Blog* [120] the most popular NoC simulators are BookSim 2.0 [121] and Noxim [122], which offer manifold features and are actively maintained.

Noxim is a cycle-accurate simulator. It is implemented in SystemC and many parameters, such as depth of buffers, size of packets and the routing algorithm used, can be set. For benchmarking, synthetic traffic patterns are implemented. Noxim measures throughput and packet delay and estimates power consumption. The power consumption is based on a simple, cycle-accurate model, which tracks different events. The timing behavior of routers in Noxim is rather static, yet can be modified via the source code. The router model is divided into a receive method, which handles incoming flits and stores them into buffers and a transmit process, which send flits and calculates routes.

The simulator *Booksim* is comparable in features to Noxim: Its router model is also cycle accurate and the implementation is written in C++. Booksim is more flexible, since many topologies can be read from configuration files. Also, multiple router architectures with diverse routing algorithms are already implemented. Again, only synthetic traffic patterns can be injected into the network. The evaluation features are similar to Noxim; a power model is not provided.

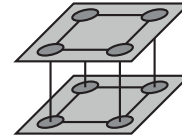
4.10 3D NOCS

In general, characteristics and properties of 3D NoCs are quite similar to those of 2D NoCs. An exemplary 3D NoC with two layers and 32 routers following 3D mesh topology is shown in Figure 22. Each router is connected vertically via an array of TSVs (dashed) and within the layer via links (solid). Most of the considerations and research results from 2D NoCs can be applied to 3D systems, as well. 3D NoCs offer similar advantages as general 3D technology, only applied to the interconnection networks. For instance, 3D NoCs profit from smaller network diameters. Some results from the research on 2D NoCs must be reconsidered. 3D technologies enable previously discarded topologies, since wire-length constraints from 2D NoCs are

not present [123]. As an example, torus topology is again promising for 3D designs, since its main drawback (long wires on 2D chips) is nullified [12]. Currently, there are three categories of 3D NoCs:

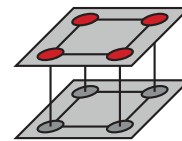
Homogeneous 3D NoCs

Homogeneous 3D NoCs extend 2D designs by a spatial dimension, yet not by new manufacturing technologies (e.g. see [124]): Existing works tacitly assume a multi-layer homogeneous 3D SoC, such that communication costs in each die are identical. [JM 3]



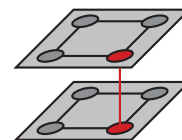
Heterogeneous 3D NoC

Heterogeneous 3D NoC [5, 125] denote NoC designs with non-uniform properties at the architectural level [JM 3].⁵ In one approach, a heterogeneous mixture of 2D and 3D routers is constructed. This inter-router heterogeneity has been investigated in [5] achieving power reductions of around 20% with a negligible performance degradation. Further on, [7] reports additional energy reductions by 40%. Interestingly, this later work claims that 3D systems require heterogeneous NoCs due to the technology asymmetry typically found in 3D systems. However, the technology asymmetry is not further investigated. In another approach, multiple router architectures are implemented side-by-side on homogeneous 3D SoCs targeting cost reductions. A standard router can be divided [59], providing a multi-layered 3D NoC router. Units, which can be separated, span multiple layers (e.g. crossbars and buffers). Routing and arbitration are inseparable and thus located in one layer. Signals within the router are transmitted using TSVs. This 3D router architecture with a 2D network topology has 42% area reduction and 51% performance increase for synthetic traffic patterns in comparison to a 2D router architecture. For real-world applications, energy savings are achieved by dynamically shutting down unused layers. In [6], link and crossbar sharing between routers decreases the network latency by up to 21% compared with a standard 3D router. [JM 3]



Hybrid 3D NoCs

Methods in which buses connect NoC routers are referred to as hybrid NoCs [8, 9, 125]. Buses can be vertical or horizontal. The key premise for vertical buses is that these transmissions do not need to be hop-to-hop: Routers communicating vertically via a local bus is sufficient. It not only saves area since the number of ports in the



⁵ The term can also be used for 2D NoCs. In this case, heterogeneous (non-uniform) designs combine different router architectures in a single silicon layer.

router are reduced but also improves performance since the packets can cross more than one die in one step. The logical bus can be implemented either with tri-state logic [125] or with a simple demultiplexing and multiplexing stage [9]. The number of buses connecting the silicon layers in the hybrid 3D mesh topology has been optimized in [125]; it produces an intra-router heterogeneity which saves up to 20% of area of 3D mesh NoCs. Finally, fault tolerant and thermal issues are discussed in [8].

4.10.1 *Performance, power and area*

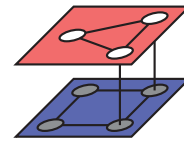
The same optimizations for throughput and latency from 2D NoCs are also valid for (homogeneous) 3D systems. Look-ahead routing, for instance, reduces the system latency and increases the throughput by approximately 45% in 3D systems [126], which is a similar result as in 2D networks [127]. While for 2D systems many benchmarks exist, there are few for 3D NoCs. Synthetic benchmarks can be applied to both 2D and 3D NoCs. Most of the real-world-based benchmarks are tailored for 2D ICs and therefore cannot be used. Common NoC simulators offer the capability to model 3D networks, as well, but are not focused on the special properties of 3D technology. [JM 3]

Area and power of routers will increase if 2D designs are extended to the third dimension: Additional buffer space is required for ports in vertical directions. The vertical links themselves require area for instance for KOZs (cf. Sec. 3.3.3). The crossbar is larger and, although marginal, additional logic is needed for arbitration and routing calculation. Reference [123] shows that the area increase per router is about 50% when moving from 2D mesh to 3D mesh topology without optimizing the router architecture. Therefore, area reductions in 3D NoCs are essential. The majority of works addressing this focus on the placement and number of vertical links: The TSV count is reduced by partially connecting layers [128, 129] or by sharing TSVs among neighbored routers [130]. The router architecture can be optimized, as well [131]. Optimization of power consumption is achieved by novel router architectures. For instance, reference [132] proposes to use partially activated crossbars, clock-frequency scaling and serial-link coding. This can be applied to 2D and 3D NoCs; different properties of vertical and horizontal links are only investigated recently, which has a vast influence on energy consumption [32]. [JM 3]

Part III
INNOVATION

5.1 DEFINITION

If NoCs are used for communication in heterogeneous 3D SoCs, routers in different layers are implemented in different manufacturing technologies such as (purely) digital and mixed-signal nodes. These NoCs are called technology asymmetric.



Definition 5.1 (Asymmetric 3D NoCs). Technology-asymmetric 3D NoCs (A-3D NoCs) are 3D NoCs that explicitly target heterogeneous 3D SoCs and exploit the technology-specific properties on each silicon die implemented in disparate technologies.

The novelty of this approach lies in the exploitation of technology heterogeneity for interconnection architectures. An implicit assumption of the current state of research on 3D NoCs is that the silicon layers in a 3D SoC are technologically homogeneous; the network designs are extended “only” by a dimension, but not by “new technologies”. A-3D NoCs overcome this limitation. A-3D NoCs will be useful in 3D SoCs if two conditions are met: First, the application itself demands disparate technologies since it incorporates analog circuits, mixed-signal circuits and digital logic. Otherwise, the electrical requirements of components are uniform and a homogeneous 3D SoC is advantageous in performance, or a 2D SoC is cheaper in production. Second, complex communication patterns must be prevalent between components and layers. Otherwise, a full-fledged network will yield disproportionate overhead if the prevalent data flow is unidirectional. [JM 1, JM 3]

5.2 LIMITATIONS OF TODAY’S APPROACHES

Existing approaches to communication networks in 3D SoCs only exploit incremental improvements of 3D technology that directly emerge from the potentials of 3D integration as discussed in Section 3.1. Thus, these works assume identical routers and manufacturing technologies on all silicon layers. The evaluated objective functions for NoC optimization incorporate parameters such as area, power, performance, Quality of Service (QoS), and reliability. However, these parameters differ significantly between dies manufactured in disparate technologies. Thus, neither limitations nor advantages of heterogeneity are considered so far. There are three main related approaches to A-3D

NoCs, which have already been introduced in Section 4.10. Each of these concepts is different to A-3D NoCs: *Homogeneous 3D NoCs* assume a homogeneous 3D SoC. If heterogeneity is present, costs and constraints of the communication infrastructure in each die will vary. While *heterogeneous 3D NoC* are designed with non-uniform architectural properties, they do not consider influence of disparate manufacturing technologies. Therefore, A-3D NoCs again add a new design dimension. *Hybrid 3D NoCs*, which combine NoCs and other interconnection architectures, are not directly implementable in a technology-asynchronous scenario. For instance, bus systems spanning multiple heterogeneous dies face severe limitations with the not purely synchronous clocks in heterogeneous 3D SoCs. To summarize, the research scope of A-3D NoCs is not considered in the previous works. [JM 3]

5.3 POTENTIAL FOR OPTIMIZATION THROUGH A-3D NOCS

The presence of different die manufacturing technologies has a vast influence on the overall architecture. It allows for NoC optimizations on multiple levels:

1. At *system-level*, technology characteristics can be exploited by combining different communication strategies as well as sets of components at different chip layers. System-level optimizations comprise: placement, mapping and topology.
2. At *architectural level* another optimization potential arises from the possibility to utilize technology heterogeneity to optimize parameters of sets of routers per layer. Architectural level optimizations comprise: routing and number of virtual channels.
3. At *micro-architectural level*, communication components themselves can be spatially split up and distributed among adjacent chip layers to exploit the heterogeneity in manufacturing technologies such that new features can emerge beyond the means of today's systems. Micro-architectural-level optimizations comprise: buffer depths, buffer distributions, router architectures and router synchronization.

The following examples, which will be researched within this thesis, illustrate the optimization potential:

First, considering routing: We will show that it is advantageous to route long-distance packages via the more efficient and higher performing adjacent layer, and to pay the extra delay required for the traversal of vertical links. Routing adaptation will also consider different clock frequencies in different layers. Thinking further, global routing decisions can be outsourced from a slower layer to a faster one. Run-time-adaptive routing mechanisms on faster chip layers can complement source routing on slower chip layers.

Second, considering router resources: These will be reduced in area-inefficient and power-inefficient layers. Providing more communication resources than strictly needed by its processing elements in the more efficient layers compensates for performance losses. Examples are buffers and links.

Third, considering locations of routers: The assignment of components to layers is critical, because the electrical characteristics of layers must suit the requirements of components. As the size of components potentially differs between layers, the positions of routers varies, as well. Thus, the network topology must be optimized not only with respect to the network but also the components.

As can be seen, these scenarios extend the design choices tremendously. The potential is large, but it demands that current NoC topologies and architectures are re-thought, re-analyzed and extended in a technology-asymmetrical scenario.

5.4 CHALLENGES SOLELY PRESENT IN A-3D NOCS

There are unique challenges for NoCs in heterogeneous 3D SoCs due to disparate technologies, as partially published in [JM 3]:

1. *Costs in mixed-signal layers*: If homogeneous routers are used, as state-of-the-art [128, 130], these will yield unbearable costs (in terms of area and power) in some layers.
2. *Not purely synchronous timing*: Clock speeds in A-3D NoCs are not purely synchronous; in particular, components in mixed-signal layers are clocked slower (as shown in Section 9.2). The clock differences between layers can be rather large, which must be reflected in the router architectures. Therefore, synchronous implementation is not realistic due to large performance losses: Heterogeneous 3D SoCs will implement globally asynchronous, locally synchronous (GALS) clocks.
3. *Design space complexity*: Asymmetry adds multiple degrees of freedom. In addition, design metrics are interleaved, i. e. design decisions influence multiple metrics at once with possibly contradictory tendencies. This is not necessarily unique to A-3D NoCs (buffer resources have a positive influence on performance and a negative on area), but it is more severe (e. g. memory is more expensive in mixed-signal layers). Therefore, an integrated power, area and performance assessment is mandatory.
4. *Irregular topologies*: Heterogeneity influences the network topology, as shown in Figure 23. The sectional drawing shows a three-layered SoC. The size of components and routers is larger in the mixed-signal layer, even for structural identical units. Therefore, routers must not be located in a regular and aligned topology as usually found in homogeneous 3D SoCs. Rather, any irregular connection scheme is possible.

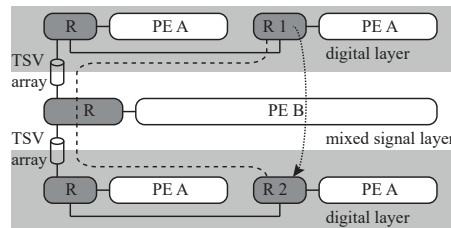


Figure 23: Sectional drawing of a heterogeneous 3D SoC in which dimension-ordered routing is impossible [JM 3].

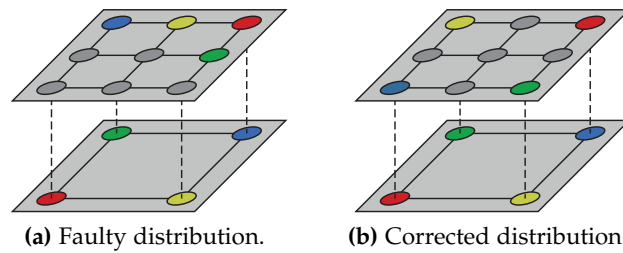


Figure 24: Bit-complement's spatial distribution. The first four sources and destinations are in the same color. Using the same mathematical relational between addresses as in 2D NoCs yields the faulty distribution (left-hand side) while the desired distribution is different (right-hand side).

5. *Routing algorithms:* Simple and robust routing algorithms such as dimension-order routing might not be connected as shown in Figure 23 for communication between routers R1 and R2. Routing algorithms must consider these technology constraints. This is a general principle in A-3D NoCs as shown by means of modeling in Section 9.5 (cf. Figure 75).
6. *Synthetic traffic patterns:* At first glance it appears to be trivial to extend synthetic patterns to heterogeneous 3D SoCs. However, due to irregular topologies, their definition cannot always be translated without losing the pattern's characteristics: For instance, in bit-complement traffic, packets are sent to the other side of the network. This is modeled using addresses as introduced in Section 4.9.1.1. For heterogeneous 3D SoCs, however, the bit-complement of the address is not necessarily located at the opposing side of the network in A-3D NoCs, cp. Figure 24. Despite same mathematical distribution, different spacial traffic patterns emerge. Therefore, synthetic traffic pattern must be handled with care.
7. *Real-world based benchmarks:* Standardized benchmarks and evaluation platforms are essential for evaluation. Tools such as Netrace [114] provide this for 2D NoCs; A-3D NoCs are not targeted. Using abstract task graphs [JM 12, 110] requires task-mapping algorithms that reflect technological properties. Application benchmarks must be defined specifically targeting heterogeneous 3D SoCs.

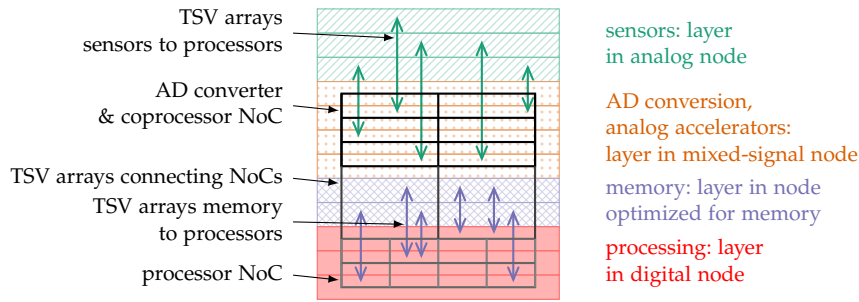


Figure 25: Sectional drawing of a heterogeneous 3D SoC with four types of layers: analog, mixed-signal, memory and digital. Two 3D NoCs span layers in digital and mixed-signal technology. Analog components and AD-converters, as well as memory and processors, connect via point-to-point connections [JM 1].

5.5 TYPICAL EXAMPLE FOR NOCS IN HETEROGENEOUS 3D SOCS

One possible example for a NoC in a heterogeneous 3D SoCs is explained here, as originally published in [JM 1]. It is derived from the typical properties of chips in the class of 3D VSoCs. It is shown in Figure 25. There are analog layers (□) typically implemented in a wide variety of technology nodes. The sensors on these layers connect to analog-digital (AD) conversion on mixed-signal layers (□). This communication is usually unidirectional and uses point-to-point connections. Next, there are layers in a purely digital technology node (□). For both mixed-signal and digital layers it is preferred to use one technology node each. The selection depends on the design targets; e.g. the fastest digital node and the most advanced mixed-signal node. 3D NoCs would typically span these two classes of layers. The networks are homogeneous within, yet both have disparate properties. Finally, there are memory layers (□). Those are placed between the layers in mixed-signal node and the layers for digital processing, since digital layers must be placed at the bottom for thermal reasons and the latency between logic and memory must be minimized, hence adjacent placement. Dedicated point-to-point connections are used due to their low latency, which is critical. It is important to highlight that it is especially relevant to the design of NoCs for heterogeneous 3D SoCs to consider two 3D NoCs, implemented in disparate nodes, and the interplay between those two networks. [JM 1]

5.6 DESIGN SPACE OF A-3D NOCS

In this section, we present a model to describe the design space of A-3D NoCs. We further propose a hierarchy which facilitates systematic exploration. Both results were originally published in [JM 6].

5.6.1 Terminology, structure and hierarchy

We highlight the structure of the design space and propose a hierarchy, which facilitates exploration. Due to the influence of varying technologies, the design space differs from conventional NoCs, of which the design is less dependent on technologies. Finally, we propose methods for systematic exploration. This is important, since the design space of A-3D NoCs has non-linear, non-convex and non-trivial properties, that impede exploration.

Definition 5.2. Design space. Let there be n design dimensions. The design space is defined as:

$$D = \{d_i \mid i = 1, \dots, n\}. \quad (1)$$

Each design dimension d_i represents a set of optimization variables. A simple graphical representation of the design space is given in Figure 26.



Figure 26: Representation of a design space with n design dimensions d_i .

Optimization variables determine the subject of the optimization in a particular design dimension. If for instance the design dimension indicates placement of components on a heterogeneous 3D SoC, the optimization variables will denote the actual position of each component or its bounding box. As another example, a design dimension denotes the properties of vertical connections. Then, the variables describe number, position and dimension of the TSV arrays. Considering a third example, design variables with the size of each buffer in the network denote the design dimension of router buffer depths. The organization of design variables into design dimensions thus follows the natural understanding of topics for design space exploration. Each optimization variable is given by its value s_k^i . It holds for every design dimension $d_i \in D$ that

$$d_i = \{s_1^i, \dots, s_{k_i}^i\}, \quad (2)$$

with $k_i \in \mathbb{N}^\infty$ for all indexes $i \in \{1, \dots, n\}$. In general, it is possible that dimensions consist of an infinite set of variables. For instance, the variable describing the position of components on the chip may be infinite as there exist an infinite number of locations in \mathbb{R}^3 . The design variables are depicted in Figure 27. The design dimension with index 1 consists of three variables, number 3 of four variables. The remaining dimensions depicted (2, $n - 1$ and n) consist of an infinite number of variables.

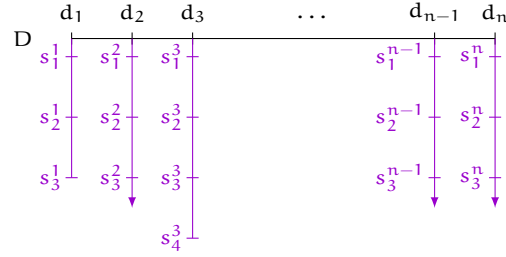


Figure 27: Each design dimension consist of several design variables $s_{k_i}^i$, which are optimized during design space exploration.

Optimization variables are set during design space exploration. The input constants define the scope of the exploration. There are two types of input constants:

Definition 5.3. Technology constants. The set T consists of technology constants, which describe technological characteristics, with an influence on the network architecture. These input constants, for example, comprise manufacturing technologies, component implementation costs or the size of KOZs.

Definition 5.4. Application constants. The set A gives application constants. These characterize applications, which are executed on heterogeneous 3D SoCs. Again, these constants are fixed during the design process. Examples for application constants are application or task graphs, or network traffic characteristics such as toggle or bit self-switching activity. We model application constants as invariant during application execution. However, the properties of the application might actually change, which cannot directly be taken into consideration during the design phase. For instance, a sensor yields different data depending on context and surroundings. This can influence the control flow of the application and thus its requirements for the communication infrastructure. The time-variant properties can only be modeled using statistical properties. For example, sensor data are uncorrelated and independent of the environment.

The optimization must be measured using cost functions:

Definition 5.5. Evaluation metrics. Variables are optimized for a set of evaluation metrics M . The most important evaluation metrics are PPA, i. e. power consumption, performance, and area costs. Less common, additional design metrics, such as reusability and scalability, can be assessed as well. If multiple metrics will be considered simultaneously during an optimization, this optimization is called “multi-objective”. Multi-objective optimization is mandatory for A-3D NoCs: Parameters of the interconnection network and the manufacturing technologies are interwoven. Both analytical and simulative design space exploration methods must reflect this.

Technology and application constants together with an instance d of the design space are used to calculate the evaluation metrics M . This is shown in Figure 28.

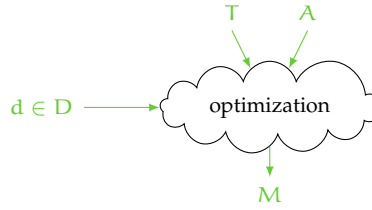


Figure 28: Technology and application constants influence the optimization of design variables under minimization of M .

The structure of the design space can be ordered hierarchically: There are different models and sub-models for application, network, routers, components, links, etc. For instance, the combination of buffer depths with other router-specific design variables, such as link widths or the number of virtual channels generate (sub-)models. This hierarchical classification is, however, not unique. Variables may be used in two models: The link width influences properties of routers, network interfaces, and links; even packet sizes may change for the application. To cope with this intersections, we propose a separation of the design space depending on mutual influence and not purely on functionality. This yields two subclasses, namely: *system-level decisions* and *architectural and microarchitectural features*. The distinguishing characteristic between the two classes is that system-level decisions influence optimizations of (micro-)architectural features but not vice versa – a hierarchical order.

Definition 5.6. System-level decisions. The system-level decisions are given by the set $G = \{d_1, \dots, d_j\}$, with $j \leq n$. Figuratively spoken, these are the first j design dimensions. These are depicted on the left-hand side of Figure 29.

System-level decisions are influencing optimization are influencing the optimization of every other design dimension. In other words, (micro-)architectural features are not independent of system-level decisions. System-level decisions also influence other variables on G . Therefore, a single, closed model is essential for system-level decisions. Examples for global decisions are physical dimensions of the chip, its component floor planning or placement, or the network topology.

Definition 5.7. Architectural and microarchitectural features. There are k architectural and microarchitectural features given by the set $L_l = \{d_{r_l}, \dots, d_{t_l}\}$, with $j < r_l \leq t_l \leq n$ and $l \leq k_i$. This is shown in Figure 29 in orange. Note that it holds $n = j + k$.

(Micro-)architectural features represent components, methods or techniques, whose design decisions are (at least partially) indepen-

dent from each other, i. e. have limited influence on each other. Examples for (micro-)architectural features are coding, properties of horizontal links and vertical links (TSV arrays) or router methods such as long range links [133] or router architectures with special features e.g. prioritization of semi-static data streams [JM 8]. In general, it cannot be ensured that all of these features do not influence one-another, i. e. are independent. Independence is assumed as approximation for efficient design optimization.

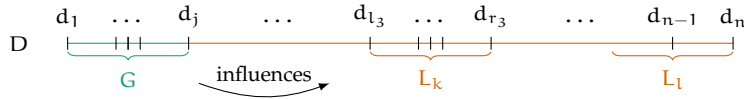


Figure 29: Exemplary design space with system-level decisions G and (micro-)architectural features L_k and L_l . The system-level decisions influence the individual features.

Considering a different hierarchical order, the design space can also be separated into different levels of abstraction. This allows to consider only some variables for faster optimization.

Definition 5.8. Abstraction levels. Abstraction levels reduce a specific subset of the design space by representation via a smaller set of variables. This allows for reduced model complexity.

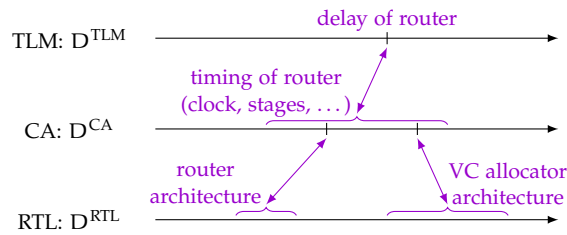


Figure 30: Design spaces for different abstraction levels. The timing behavior of the router is described using different variables per level.

Certain abstraction levels serve a specific purpose. For instance, both transaction-level modeling (TLM) and cycle accurate (CA) abstraction yield models of timing behavior. But TLM is less accurate, in general, since only single and important events and not the whole time is simulated. Both levels do not allow to evaluate the area requirements of a circuit. Register-transfer level (RTL) models allow to evaluate the area requirements yet are too complex for the simulation of larger systems and rapid prototyping. This is shown in Figure 30. The timing of a NoC router is described there. In a transaction-level model, a single variable with the router's delay is sufficient. This variable relates to multiple variables on cycle-accurate level. These describe the timing of the router and comprise, for example, the clock

rates or the stages of the router. Again, these variables relate to multiple variables on RT-level. The stages, for instance, are now given by the router architecture. The clock rate could be dominated by the architecture of the VC allocator for the sake of the example.

Different abstraction levels model different parts of the design space and allow to set different optimization variables. Let α be an abstraction level. Then, the subsets of variables $G^\alpha \subseteq G$ and $L_1^\alpha \subseteq L_1$ must be considered for system-level decisions and for features. Please note, that not all (technology and application) constants are relevant in every abstraction level α ; therefore only a subset $T^\alpha \subseteq T$ and $A^\alpha \subseteq A$, respectively, is considered. We denote the sets of all possible parameter sets $A^\alpha, T^\alpha, G^\alpha$, and L_1^α using the symbols $\mathcal{A}^\alpha, \mathcal{T}^\alpha, \mathcal{G}^\alpha, \mathcal{L}_1^\alpha$.

Definition 5.9. Objective function. The objective function guides the design space exploration towards an optimum using evaluation metrics. The objective function models the influence of the optimization variables on the evaluation metrics under a given set of constants. Objective functions for system-level decisions and (micro-)architectural features are:

$$\zeta^\alpha : \mathcal{A}^\alpha \times \mathcal{T}^\alpha \times \mathcal{G}^\alpha \longrightarrow M, \quad (3)$$

$$\xi_1^\alpha : \mathcal{A}^\alpha \times \mathcal{T}^\alpha \times \mathcal{L}_1^\alpha \longrightarrow M. \quad (4)$$

Definition 5.10. Accuracy. In general, different abstraction levels can yield different model accuracy because the design space is not always considered as a whole. A numerical value for this accuracy can be found per abstraction level and model. It is determined by comparison of models on different abstraction levels. For instance, the timing behavior of a circuit will be compared if modeled on transaction level, cycle-accurate level or on gate level. The downside of higher abstraction is descending accuracy. The upside is reduced model complexity and faster simulation, which allows for more efficient design space exploration.

5.6.2 Exploration

In general, design space exploration is a hard and complex topic, since many subproblems are NP hard (e. g. non-overlapping area minimization). Since NP hardness impedes optimization, approximate heuristic approaches are essential. Here, we explain how to exploit structural properties of the design space allowing for faster exploration. It is noteworthy that this is orthogonal to approximate solutions of individual subproblems within the hierarchy, which actually tackle NP hardness, such as proposed in Section 6.4.

The design space is ordered by hierarchy, such that system-level decisions influence the (micro-)architectural features but not vice versa. This is only possible, since independence between system level and

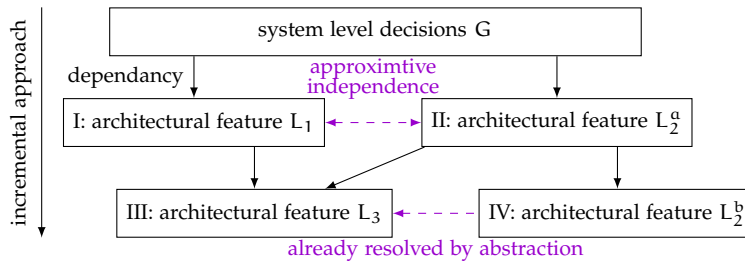


Figure 31: The incremental approach exploits structure and hierarchy of the design space. The exploration sets parameters following dependencies between features. Some are treated as independent despite actual dependence as approximation.

single (micro-)architectural features is assumed as an acceptable compromise. Furthermore, the design space can be ordered by abstraction levels. This allows to reduce the number of parameters by partial exploration, since models with fewer variables are less complex; this reduces design time. We call this an *incremental approach*: First, system-level decisions set the boundaries, in which further exploration is possible. Second, the (micro-)architectural level is explored. Therefore, abstraction levels allow to reduce the number of variables which must be set per optimization. Fixed variables on a higher abstraction level are constraints for lower abstraction levels. This is shown in Figure 31.

6.1 INTRODUCTION

The design space of 3D NoCs for heterogeneous 3D SoCs consists of system-level decisions and (micro-)architectural features, as introduced in Section 5.6. In this chapter, we optimize the system-level parameters, as defined in Definition 6.3. It is important to optimize at system-level because these decisions have a huge impact on (micro-)architectural features. Therefore, an extensive design-space exploration is essential. Before we define the problem itself, we start by defining two basic terms:

Definition 6.1 (Components). Components denote an abstract representation of PEs: Components are defined as parts of a SoC, which consume and generate data and have area requirements.

Definition 6.2 (Bounding boxes). Bounding boxes are rectangular shapes on the chip, in which there is enough space to place a component, a router and TSV arrays.

Using these terms, we can give a concrete definition of the problem:

Definition 6.3 (System-level optimization of NoCs in heterogeneous 3D SoCs). System-level optimization of NoCs in heterogeneous 3D SoCs consists of *NoC planning with simultaneous layer assignment and component positioning*. Using the basic terms, we specify:

- *NoC planning* is defined as the optimization of the interconnection network by assigning routers and TSVs to bounding boxes.
- *Layer assignment and component positioning* is defined as assignment of components to layers and then to bounding boxes.

The following properties of heterogeneous 3D SoCs are especially challenging and render system-level optimization difficult: Component positioning and NoC planning cannot be separated because the topology and locations of components and routers mutually influence each other. Also, the location and vertical link count is determined, which is interwoven with the components' locations. Furthermore, the optimization is multivariate; the objective function comprises models for area, power, performance, network load and network congestion.

In general, this optimization problem can be solved with two different approaches: The first approach is *analytical*, by defining a closed model for the system. The second approach is *empirical*, by defining sets of parameters with their properties and then analyzing their

effect after running several simulations. System-level optimization must be conducted analytically for two reasons: Firstly, system-level decisions must be accurate because of their impact on all features. Secondly, an extensive exploration is too time-consuming using simulations.

To tackle system-level optimization, we contribute a well-reasoned analytical formulation accounting for technology parameters in Section 6.2. Next, we provide a model for an optimal solution in Section 6.3. Due to infeasible computation time, we propose a heuristic algorithm which allows for a relaxed solution in Section 6.4. We discuss the performance of the solution in Section 6.5. Next, we apply the models and heuristics to different input sets. The results are presented in Section 6.6 and discussed in Section 6.7. Finally, we draw a conclusion in Section 6.8. The chapter is partly published in the articles [JM 4].

6.1.1 *Differentiation to today's approaches*

NoC planning with simultaneous layer assignment and component positioning has not been considered so far. The reason lies in the novelty of A-3D NoCs. A-3D NoCs have properties that are not accounted for in existing literature: Only in heterogeneous 3D SoCs the size of components varies with technology, which influences network topology. Also unique to heterogeneous 3D SoCs, is that performance and energy consumption of components differ with their location. Plus, properties of routers change between layers. Despite the novelty of A-3D NoCs and their unique features, there are similar approaches. Naturally, these are applied in different context and target other systems. Here, we introduce them and discuss distinguishing features and why they cannot be applied to A-3D NoCs.

There are two problem categories that can be found in system-level optimization for A-3D NoCs, which can be found in other optimizations, as well. These categories are *partitioning of resources* and *positioning of TSV (arrays)*.

Partitioning of resources is found during system-level optimization for A-3D NoCs when components are assigned to layers. A mathematically similar problem is partitioning for ASIC design [134]. This cannot directly be applied here: ASIC design targets gate-level inputs, in which circuits with a huge number of gates are partitioned and the gates have the same properties in each partition. Solely communication is optimized. In contrast, system-level optimizations target a network level, in which the number of components is significantly smaller, but their properties differ between layers.

Positioning of TSVs is a well-researched topic. There are three general approaches: Positioning of individual TSVs for ASIC design, po-

sitioning of TSV arrays modeled as macroblocks, and positioning of TSV arrays in 3D NoCs determining a topology for vertical links:

- *Positioning of individual TSVs for ASIC design:*
 In this scope, the objective is a reduction of wire lengths to increase the performance of the 3D chip by optimized placement of single TSVs. For instance, [135] proposes a method to shorten wire by an average of 20% using an analytical placer. The optimization can also effect the gates connected by the TSV. For instance, both [136] and [137] propose to combine layout placement and TSV placement by a 3D cell placement algorithm. All these publications target the placement of individual TSVs in homogeneous 3D chips at gate level. These are not directly applicable to A-3D NoCs, since we optimize at network level, with heterogeneous integration.
- *Positioning of TSV arrays modeled as macroblocks:*
 When moving the focus from individual TSVs to arrays of TSVs, their area becomes considerably larger. Therefore, a large area-consuming block must be placed in a chip. This is very similar to placement of macroblocks, which is a common problem found in ASIC design. To tackle macroblock placement, many works pursue a twofold approach: First, the macroblocks are placed but may overlap with other chip components. Second, the design is legalized by removing overlaps [137]. This approach is rather similar to our method but can also not be directly applied, as we explain using the following example: PolarBear [138] proposes a method for mixed-size placement, with less than 5% non-utilized space (white space). As usual, this approach uses graph-based legalization, which is very well applicable to unrestrained orders of placements. It cannot be applied to our problem, since components are positioned according to the network topology in A-3D NoCs and thus the order is in fact restrained. Other algorithms, such as Capo [139] are correct-by-construction, which avoid legalization; we do not pursue this because simultaneous TSV array placement and component positioning considerably increases the required computational effort. Please note that all methods for placement of TSVs can directly be applied in post-processing to the solutions for A-3D NoCs to determine the precise location of TSVs within their given bounding boxes.
- *Positioning of TSV arrays in 3D NoCs:*
 There are a few publications addressing placement of TSV arrays, i. e. vertical links, in homogeneous 3D NoCs. In general, all of these publications target NoCs with stacked layers of 2D meshes, in which routers are located at the same positions

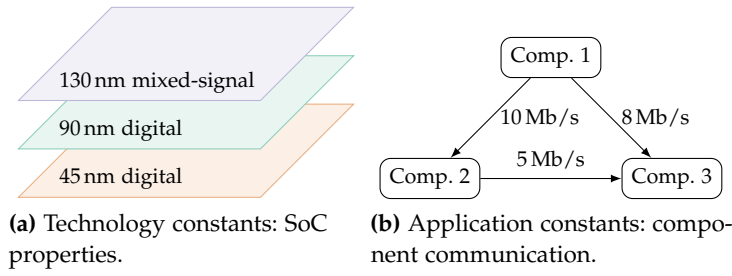
in each layer. The objective is optimization of network properties by determining a good topology for vertical connections. In [140] the target number of TSV arrays is given. For 25% of all possible locations filled with vertical links, the network performance is reduced by 18.7% while the energy of the NoC is reduced by 23.5% in comparison to a fully connected network. [141] presents another algorithm based on a similar principle. For 25% connectivity, 7.96% throughput decrease, 0.39% latency decrease and 7.69% packet energy reduction are measured in comparison to full connectivity. We differentiate from these approaches as follows: We do not assume homogeneous NoC with uniformly-sized grids in each layer, which is very wasteful for heterogeneous 3D SoC. Furthermore, our technology model is more accurate because we are using redistribution to increase the freedom of placement. Plus, our models for routers and TSV arrays are more modern, because we account for physical properties of state-of-the-art via middle process flow. Hence, our models are more area-precise.

6.2 PROBLEM FORMULATION AND TECHNOLOGY MODEL

Here, we define *input*, *objective function* and *solution space* of system-level optimization of NoCs for heterogeneous 3D SoCs:

The *input* of the system-level optimization consists of technology and application constants as defined in Section 5.6: As technology constants, the layers-count and a set of available technologies are given, along with ordering of layers and their technologies. As application constants, the expected communication between components is accounted for, including the amount of data each component consumes and produces over time, as well as the data flow direction (Figure 32b). This is analog to the definition of the core graph [142]. We extend this graph by a description of components that includes their properties, i. e. power and area requirements, as well as their performance, in different technologies (Figure 32c). For their estimation, standard approaches exist [143].

The *objective* is to minimize the area and power requirements of the resulting SoC while maximizing performance of components and network implemented on the SoC. The latter includes avoiding congestion and utilizing the available resources efficiently. The area is minimized by the physical dimensions of the chip; we target quadratic chips, which are rather common and require less computational effort during optimization (Figure 32a). We model this using a linear weighted objective function with five addends for area, power, component performance, network latency and network throughput. The resulting five weighted addends account for all relevant metrics which describe the quality of a design. Using PPA (power, performance,



	Comp. 1	Comp. 2	Comp. 3
130 nm mixed-signal	3	4	6
90 nm digital	5	12	3
45 nm digital	7	3	1

(c) Application constants: Component properties (abstract representation, thus unit-less).

Figure 32: Depiction of example inputs.

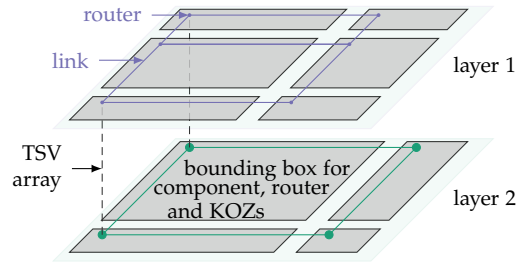


Figure 33: Depiction of a solution candidate.

area) is the de-facto standard and our first three addends models this for the components and routers on a SoC. Since this does not account for the network performance, we included the latter two addends that model latency (distance of communication times bandwidth) and throughput limitations. We chose a quite general approach, which includes a broad set of design targets. Their actual relevance in each individual design can be set through the weights. Of course, the linear model allows for an easy extension of the objective function, if necessary. Therefore, the objective function considers the targets for the optimization problem properly.

For a *solution*, bounding boxes are found, which provide space for components, routers and TSV arrays. Since components and routers have different sizes in each layer due to heterogeneity, the sizes of bounding boxes differ. A solution candidate is shown in Figure 33: Bounding boxes are shown in gray; Routers in different layers have varying properties depicted by color-coding. The floorplanning within each bounding box can be done using well-known methods. Methods for placement of individual TSVs of TSV arrays also exist. They require bounding boxes and have been discussed in Section 6.1.1. The KOZ's placement can be realized by modeling them as macroblocks. Solutions also define the network topology, indirectly, as

shown.¹ Routers are located near the upper left corner in the solution candidate shown in Figure 33. In general, their actual location in a bounding box will be a result of floorplanning.

6.2.1 Router model

Routers have up to four ports in each cardinal direction in their layer and one local port. To the latter, at maximum one component is connected. Furthermore, routers can implement one port to the layer above or one to the layer below. Routers consume area and power and are connected within their layer via links and vertically via TSV arrays. The router size depends on the port count, i. e. 2D routers are smaller than 3D routers. Please note that routers will have a different port count if implemented inlying or peripheral (e. g. a router at the edge of the chip cannot connect in every direction). Therefore, the size of routers varies depending on their position; this is only modeled in the heuristics, since it would further increase the complexity of the model. Furthermore, vertical links in 3D routers require additional area. This is shown in detail in Figure 34. The area of the 3D routers consists of the area for the router and the KOZ as shown in Figure 34a. If the router only connects downwards, the TSV area will still be required, see Figure 34b. A KOZ will not be required if routers connect only upwards, because we assume via-middle process flow, as shown in Figure 34c (cf. Section 3.3.2). Also, we allow connecting routers and TSV arrays via redistribution, as explained in the next paragraph on the technology model. Our model allows routers to implement any routing algorithm. As an example, we model elevator first dimension order routing algorithm. Packets are sent to the next TSV array following dimension order routing in each layer until the destination is reached, as implemented in LBDR3D [144]. Advantages are simplicity and reduced number of TSVs. An example of a path is shown in Figure 35.

6.2.2 3D technology model

As already explained, components and routers have varying sizes, implementation costs, performance and energy consumption per manufacturing technology in heterogeneous 3D SoCs. Some components cannot be implemented in certain manufacturing technologies due to physical limitations. For instance, analog sensors cannot be implemented in digital technologies. Furthermore, horizontal connections (wires) and vertical connections (TSVs) have different and even contradicting characteristics, such as KOZs of TSVs. Furthermore, for horizontal links, a redistribution (RD) can be used to route to TSV arrays. Using a RD yields additional freedom to place routers and

¹ Please note that the term network topology refers to the network graph.

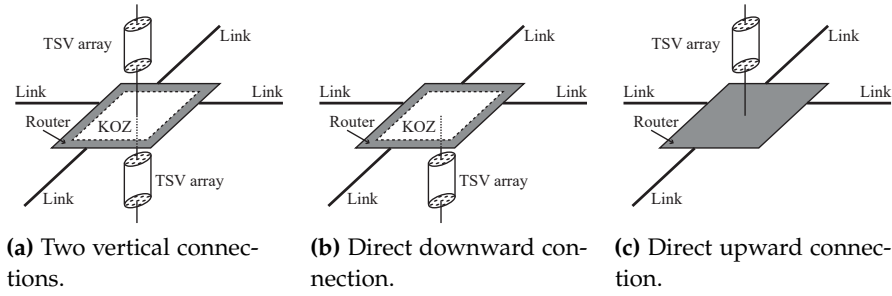


Figure 34: Router model with TSVs.

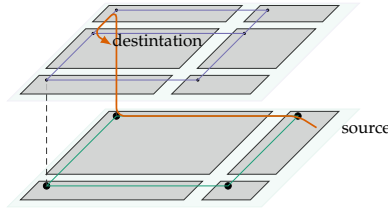


Figure 35: Elevator first DOR.

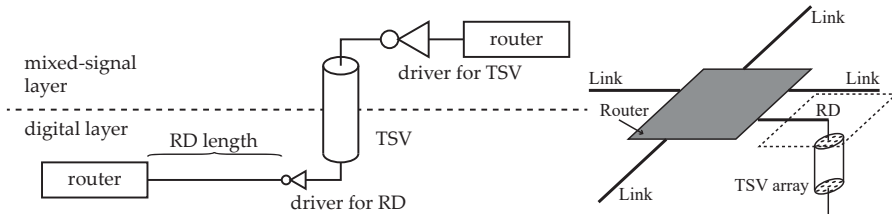


Figure 36: Example for redistribution connecting a router in a less advanced technology to a router in a more advanced technology.

Figure 37: Router model with vertical connection and redistribution.

components. An exemplary RD of a single bit connection is shown in Figure 36 for a scenario, in which the upper layer is implemented in a less advanced node than the lower layer. When connecting these two layers, the RD in the less advanced node is as short as possible to reduce the driver size, since it is relatively expensive. Therefore, the driver connects directly to the TSV and the driver of the RD in the more advanced layer is located directly after the TSV. In the opposite directions, the driver in the more advanced layer is located behind the router and drives both RD and TSV. The freedom in router's placement is given by the maximum length of the RD without violating the clock frequency of the routers; this RD length is called D . It can be calculated from data provided by the vendors in their library for the commercial technologies. Please note that D will vary for both directions, if technologies are heterogeneous. Since we assume bidirectional links between routers, we use the minimum of both directions as actual RD length. The router design using an RD is shown in Figure 37; KOZs are only required for downwards connections.

6.2.3 Model assumptions and simplifications

The proposed optimization is valid under these model assumptions and simplifications:

- Bounding boxes for components and routers are rectangular. This restricts optimization potential, yet is a usual approximation to reduce the problem complexity [134].
- The optimization only models system-level decisions and not (micro-)architectural features. For instance, routers are not separated on multiple layers.
- NoC synthesis does not account for direct links and buses. Hybrid architectures are not modeled.
- Components (and routers) are positioned in a structure implementing a grid, within layers, as shown in Figure 33.
- Routers are positioned at joints of the component grid whenever required. The routers are within bounding boxes.
- Routers are always connected by horizontal links in a layer. These wires form the network topology in a layer. For instance, neighboring routers are connected to form a mesh.

6.3 MIXED INTEGER LINEAR PROGRAM MODEL

In general, a MILP is defined as:

$$\begin{aligned} & \text{minimize } c^T x + d^T y, \\ & \text{subject to } Ax + By \leq b, \end{aligned}$$

$$\text{with } x \in \mathbb{Z}^n, y \in \mathbb{R}^k, A \in \mathbb{R}^{m \times n}, B \in \mathbb{R}^{m \times k}, b \in \mathbb{R}^m, c \in \mathbb{R}^n, d \in \mathbb{R}^k$$

We only highlight concepts of the model. The whole model is given in the Appendix, Section a. It begins with an overview of definitions, variables and cost functions, given in Sections a.1.1, a.1.2 and a.1.3; the definitions follow thereafter. The following notations are used. For $n \in \mathbb{N}$ we introduce the notation $[n] := \{1, \dots, n\}$. The symbols p_x, p_y and p_z denote the entries of a vector $p \in \mathbb{R}^3$: $p = (p_x, p_y, p_z)$. By χ we mean the *indicator function* as commonly defined: it will hold $\chi_A(x) = 1$, if x in A ; else it will hold $\chi_A(x) = 0$.

6.3.1 Constants and Definitions

Component and communication model

There are n components; the set of components is $[n]$.² There are m routers. The number of routers is set prior to optimization. It limits the number of components and bounding boxes (Proof: Lemma a.1,

² $[n]$ is treated as if it were the set of components in the way that each number represents a particular component. In fact, it is just a set of indices, which are used to reference components.

Section a.2). Within the MILP model, we refer to the bounding boxes as “tiles”. The reason is that tiles have a slightly different definition than bounding boxes: Similarly, tiles provide area for routers, components, and vertical links. But tiles can contain more than one router (cf. Section 6.3.2), because this allows to place routers without an own component at junctions required from the routing algorithm with higher packaging efficiency. The communication between components is modeled by a directed graph with weighted edges. The edge set of the *component digraph* (*directed graph*) contains all directed pairs of components which communicate. An edge follows the pattern (*sender, receiver*), which yields the edge set: $E_A = \{(i, j) \in [n] \times [n] \mid \text{component } i \text{ sends to } j\}$. Together this yields the directed and weighted *component graph*:

$$A = ([n], E_A) \quad (5)$$

The weights of the graph’s edges represent communication between components, called *bandwidth requirement*, given by the function:

$$u : E_A \rightarrow \mathbb{R}^+ \quad (6)$$

with $\mathbb{R}^+ := \{x \in \mathbb{R} \mid x > 0\}$. An example of this graph is depicted in Figure 32b. Three components communicate with a bandwidth requirement of 5-8 Mb/s.

Technologies and layers model

There are k available manufacturing technologies and ℓ layers. The sets $[k]$ and $[\ell]$ are regarded as the sets of manufacturing technologies and the set of layers. A *technology* is assigned to each layer using the function $\tau : [\ell] \rightarrow [k]$. This is depicted in Figure 32a, in which three different technologies are combined in one SoC.

Implementation costs model

Each component can be implemented in different technologies, yielding the set of available implementations $I = [n] \times [k]$. For each implementation of a component, *area costs* are given:

$$f_c : I \rightarrow \mathbb{R} \cup \{\infty\} \quad (7)$$

A component cannot be realized in a technology if costs are infinite. The implementation costs f_c are depicted in Figure 32c, as a matrix, with area costs per technology and component index. Furthermore, routers must be implemented, which are of same size within a layer but not between layers. *Router implementation costs* are given for 2D routers and 3D routers as:

$$f_{R_{2D}} : [k] \rightarrow \mathbb{R} \quad (8)$$

$$f_{R_{3D}} : [k] \rightarrow \mathbb{R} \quad (9)$$

TSV arrays have area costs, due to KOZs, as well. It does not depend on the layer's manufacturing technology, when via-middle process flow is used. Thus, a constant f_{KOZ} is given.

Energy model

Components will consume different amounts of energy, if implemented in different layers due to disparate technologies. The *energy consumption of components* is modeled per implementation by the function $f_E : I \rightarrow \mathbb{R} \cup \{\infty\}$. Furthermore, *routers consume energy* which is modeled as technology-dependent, i. e. the layer, in which the routers are implemented: $f_{E_R} : [k] \rightarrow \mathbb{R}$.

Performance model

Components have varying performance in different technologies. We use a *performance index*, which numerically estimates the performance given by: $f_P : I \rightarrow \{x \in \mathbb{R} \mid x \geq 0\}$.

Coordinates

The SoC is modeled to be spatially embedded in a three dimensional coordinate system, with upper bounds for the x - and y -dimension x_{\max} and y_{\max}

$$P := \{x \mid x \in \mathbb{R}, 0 \leq x \leq x_{\max}\} \times \{y \mid y \in \mathbb{R}, 0 \leq y \leq y_{\max}\} \times [\ell]$$

P is bounded to model logical OR relations: This allows for two constraints, of which only one must be satisfied (cf. Section a.3.1). The coordinates are shown in Figure 38. The upper left corner of the coordinate system is the source. The upper layer, in orange-colored shading, has one tile (area reserved for routers and components). The second layer, in green shading, has two tiles.

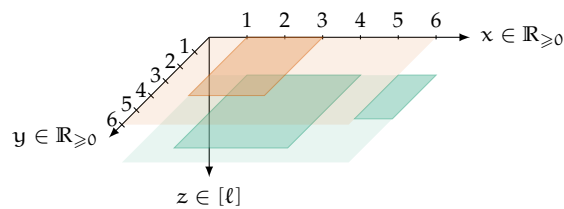


Figure 38: Coordinate system with three tiles in two layers.

6.3.2 Variables

During component positioning and NoC synthesis, positions of components, network topology, positions of routers and physical dimen-

sions are optimized. The *places of components* are given by their upper left corner, with positions $s_i \in P$ for all routers $i \in [n]$.³

This is also the place of the first n routers, which connect one component each. We introduce tiles, which are bounding boxes for routers and components belonging to the same place. A variable denotes tiles places, with positions $t_i \in P$ for all $i \in [m]$.

Tiles represent rectangular bounding boxes, with the summed size of their component, routers and KOZs, represented by length a_i and width b_i for all $i \in [m]$. A tile at position $p_i \in T$ is given by the interval

$$A_p = [p_{i,x}, p_{i,x} + a_i) \times [p_{i,y}, p_{i,y} + b_i) \times p_{i,z} \quad (10)$$

The *chip's physical dimensions* are given by the position of tiles and their size, i.e. the most exterior and largest tiles determine the production size of the chip.

We model the NoC as a spatially embedded graph. Router locations are given by their positions r_i for all $i \in [m]$.

A variable is defined that denotes the connection of two routers, p and q : $e_{\{p,q\}} \in \{0, 1\}$. The variable

$$e_{\{p,q\}} \in \{0, 1\}. \quad (11)$$

The variable is 1 for a physically implemented connection or 0 otherwise. Wires and TSVs are bidirectional. This yields the connections between all pairs of routers p, q : $E_N = \{(p, q) \mid e_{\{p,q\}} = 1\}$. The *network topology* N is defined as the spatially-embedded directed graph

$$N = ([m], E_N) \quad (12)$$

6.3.3 Objective function

Objective: SoC area minimization. The size of the actual production chip is determined by the size of the largest layer, since each layer is produced with the same dimensions. The external dimensions are equal to the size of the outer-most component in each dimension given by its position p and its size a_p and b_p , respectively. Thus the costs of the chip are defined as $\tilde{c}_{\text{area}} = \max_{i \in [m]}(t_{i,x} + a_i) \max_{i \in [m]}(t_{i,y} + b_i)$. This must be linearized. Therefore, we rather use a cost function that minimizes the area of the chip for square-shaped components:

$$c_{\text{area}} = \max\left\{\max_{i \in [m]}(t_{i,x} + a_i), \max_{i \in [m]}(t_{i,y} + b_i)\right\} \quad (13)$$

The inequalities are given in Section a.4.1.

³ Please note that the positions of routers cannot be directly at the upper limits of the coordinates P , since they have a size. This is not modeled here, but taken into account by constraints.

Objective: The total energy consumption reduction. It is given by the sum of router and component power consumption. This is given by the cost function:

$$c_{\text{power}} = \sum_{i \in [n]} f_E(i, \tau(t_{i,z})) + \sum_{j \in [m]} f_{E_r}(\tau(t_{i,z})) \quad (14)$$

Objective: System performance maximization: It is given by the sum of component performances. The sum is considered as negative for maximization.

$$c_{\text{perf}} = - \sum_{i \in [n]} f_P(i, \tau(t_{i,z})) \quad (15)$$

Objective: The data transmission in the network is optimized. To measure the utilization on each link, we use *flows*: The »flow« of packets is modeled by a source-sink-flow in the network digraph. The flow is smaller or equal to one, since we multiply with the bandwidth requirement u in the cost function. We define the function f which gives us this flow for each pair of components:

$$f : E_A \rightarrow \bigcup_{(i,j) \in E_A} \{f \mid f \text{ is a } s_i\text{-}s_j\text{-flow in } N, \text{value}(f) = 1\}. \quad (16)$$

The value of the flow is denoted by $\text{value}(f)$, following the convention used in [145]. If we write $f(s, t)$ (which is technically not defined), we mean $f((s, t))$. $f_{(k,l)}^{(i,j)}$ denotes the corresponding variable for all network links $k, l \in [m]$ and components $i, j \in [n]$. Flows allow modeling different routing algorithms. Deterministic routing algorithms have binary flows; for adaptive routing, flow values are in the interval $[0, 1]$ and represent the probability that packets use a link.

Objective: Optimize data flow. Congestion increases the average network delay, reduces throughput and increases the energy consumption. To *minimize risk of congestion*, the overall network load is reduced. The complete network load cannot be influenced, since it is an application constant. Still, the individual router's utilization can be reduced by ensuring that packets travel shorter distances and pass through fewer routers on their path. According to the following function, the costs to reduce the overall network utilization are given:

$$c_{\text{util}} = \sum_{e \in E_A} u(e) \left(\sum_{v \in E_N} (f(e))(v) \right) \quad (17)$$

To further reduce congestion risk, peak loads are avoided. Loads are defined by the function $\text{load} : E_N \rightarrow \mathbb{R}_{\geq 0}$, which returns the summed utilization on a link:

$$\text{load}(v) := \sum_{e \in E_A} u(e) (f(e))(v) \quad (18)$$

We propose a load-heterogeneity measure that penalizes loads larger than the average link load given by:

$$\mu_l = \frac{1}{|E_N|} \sum_{v \in E_N} \text{load}(v) \quad (19)$$

We estimate $|E_N| \approx n^2 - n$ as approximate linearization; this corresponds to a fully-connected network and underestimates μ_l . With the indicator function χ , the peak costs are defined as

$$c_{\text{peak}} = \sum_{v \in E_N} (\chi_{(\mu, \infty)}(\text{load}(v))(\text{load}(v) - \mu)) \quad (20)$$

$$= \sum_{(k,l) \in [m] \times [m]} (\max\{0, \text{load}((k,l)) - \mu\}) \quad (21)$$

The complete objective function comprises the summed costs, with weighting factors ω_1 to ω_5 :

$$c = \omega_1 c_{\text{util}} + \omega_2 c_{\text{power}} + \omega_3 c_{\text{perf}} + \omega_4 c_{\text{peak}} + \omega_5 c_{\text{area}}. \quad (22)$$

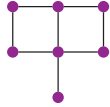
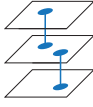



An overview of the costs functions is given in Section a.1.3.

6.3.4 Constraints

The model constraints define the boundaries, in which the optimization takes place. We explain and illustrate these; the inequalities are given in the Appendix, Section a.5. Prior to actual functional constraints for the chip's properties, we define four technical constraints for easy indexing: Components, tiles (the representation of bounding boxes in the MILP model, cf. Section 6.3.1) and routers with an index $i \in [n]$ are at the same location. Tiles and routers, which are not placed on the SoC, i. e. are not required, are at the end of the index range $[m]$. The rather technical constraints are given in Section a.5.1.


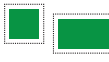





6.3.4.1 Modeling the network

The bounding boxes of routers and components, i. e. tiles, are modeled such that the router is located in the upper left corner of the bounding box. The reason for this lies in easier modeling in the MILP. The network implements a grid-like topology; therefore, neighboring routers with either same x- or y-coordinates are connected. Long-range links are prohibited. 3D routers connect adjacent layers. Which router can be connected depends on its projected distance within layers, which must be smaller than D ($|p_x - q_x| + |p_y - q_y| \leq D$). Manhattan distance linearizes the equation and is reasonable to model link manufacturing. Each component is connected to the local port of a router and routers are not connected to themselves. The resulting constraints can be illustrated as follows; inequalities are given in Section a.5.2:

<i>Topology as grid and TSVs connect to adjacent layers</i>	
	Within a layer, wires form grids.
	TSVs only connect neighboring layers.
<i>Components are connected to routers</i>	
	Each component has a router since it has a network interface with a unique address.
<i>Routers are not self-connected</i>	
	Routers connect among each other yet not to themselves.
<i>Forbid connections between non-neighboring routers</i>	
	Only neighboring routers can be connected. Long-range and diagonal links are prohibited.

6.3.4.2 Modeling tiles (bounding boxes)

The tiles provide SoC area to implement components, routers and TSV arrays. The assigned area of tiles is given by the product of its sides $a_p b_p$, which is constrained to be larger than its area requirement. The area requirement \hat{A}_p is the sum of the component size, the size of each router in the tile and their KOZs. This yields a constraint $a_p b_p \geq \hat{A}_p$ for all $p \in T$. A detailed formula is given within this section. Furthermore, tiles are without overlaps. Routers and components are assigned to tiles. In general, it is possible to have more routers than components to fulfill the connectivity of the routing algorithm: Routers may belong to a tile without a component, i.e. start their own tile, or belong to an existing tile. Links connect routers. These are only allowed to be along the borders of tiles and may not cross them. This router placement allows easy modeling. Routers must not be at the same location in the model. Since we are using tiles/bounding boxes, the actual router placement can be done in post-processing (routers can be at any location within the tile). Thus, routers can be freely placed in the tiles and links might cross them in an actual implementation after post processing. The resulting constraints can be illustrated as follows; inequalities are given in Section a.5.3:

<i>Starting positions of tiles</i>		<i>Components start tiles</i>	
	Each tile has a router.		Each component has its own tile.
<i>Routers and tiles</i>		<i>Sizes of tiles</i>	
	Routers are either part of a component's tile or their own.		Tiles provide enough space for their component, routers, TSVs.
<i>Tiles may not overlap</i>		<i>Links do not cross tiles</i>	
	Tiles are forbidden to overlap.		Links cannot be placed within a tile.
<i>Routers have different locations</i>			
	Routers are not at the same location.		

To model these constraints, we introduce a *minimum distance* and *area linearization*. The former allows modeling the inequality of positions and the latter enables the calculation of area, which intrinsically are a product, i. e. non-linear.

Modeling unequal positions on the SoC

It is not possible to state $a \neq b$ for two reals a and b through an MILP, since this would require the possibility to model open sets. To model this, a *minimum distance* δ around one of the variables is introduced. This allows modeling the unequal relations such as, $a \leq b - \delta$ or $b \leq a - \delta$. The physical representation of δ is the semiconductor scale (i. e. feature size). Modeling the unequal relation is shown in Figure 39: The variables $a, b \in \mathbb{R}$ are constrained to be unequal using the distance δ . The variable b_1 is within the interval $[a, a + \delta]$, which is not allowed. The variable b_2 is larger than $a + \delta$, which models the unequal relation.



Figure 39: In general, it holds $a \neq b_1$ (left-hand side). In the model however, $a \neq b_2$ is modeled using the distance δ and $a \equiv b_1$.

Linearization of area product

In the definition of constraints on sizes of tiles (Section a.5.3), the product of a tile’s edges $a_i b_i$ is calculated, which is not linear. Therefore, we use the following linearization. We follow the approach proposed by Montreuil [146]. The idea is illustrated in Figure 40. The size of the tile $i \in [m]$ must be larger than the summed size of its component i , α implemented routers (2D and 3D routers), of which β are 3D routers and γ KOZs with the technology costs of the component’s layer $\xi \in [\ell]$. For this given area $F_{\alpha, \beta, \gamma}^{\xi, i}$, the possible lengths and widths combinations of the tile are given by the half-space:

$$a_i b_i \geq F_{\alpha, \beta, \gamma}^{\xi, i} \tag{23}$$

In addition, each side of a tile is constrained by the maximum size of the layer:

$$a_i \leq x_{\max} \tag{24}$$

$$b_i \leq y_{\max} \tag{25}$$

To estimate the product from Equation 23, we introduce a limit on the aspect ratio η . In general, the aspect ratio is between 0 and 1, i. e.

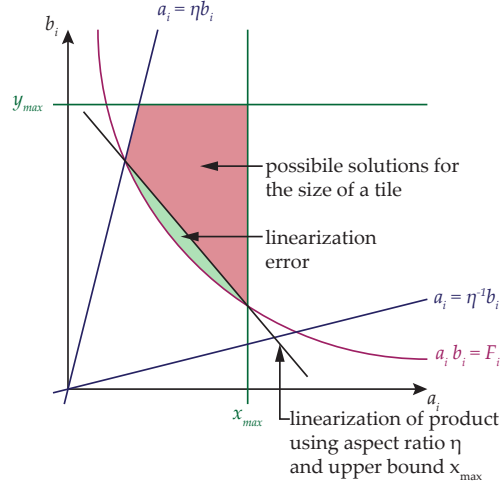


Figure 40: Linearization of a product $a_i b_i$, with aspect ratio η and bounds x_{\max} and y_{\max} .

$0 < \eta \leq 1$. Stricter bounds for η reduce the estimation error. The constraints for a given η , also shown in Figure 40, are:

$$a_i \geq \eta b_i \quad (26)$$

$$a_i \leq \eta^{-1} b_i \quad (27)$$

Connecting the intersection between $a_i b_i = F_{\alpha, \beta, \gamma}^{\xi, i}$ and $a_i = \eta b_i$ as well as the intersection between $a_i b_i = F_{\alpha, \beta, \gamma}^{\xi, i}$ and $a_i = \eta^{-1} b_i$ yields a line equation, which linearly approximates $a_i b_i$. This linearization is given by:

$$a_i + b_i \geq \sqrt{F_{\alpha, \beta, \gamma}^{\xi, i} \eta^{-1}} + \sqrt{F_{\alpha, \beta, \gamma}^{\xi, i} \eta} \quad (28)$$

This approximation will have an error of 9% to 12% if applied to the facility layout problem [147], which is similar to our model. The square root of $F_{\alpha, \beta, \gamma}^{\xi, i}$ cannot be calculated using the MILP model, since it contains variables for the area of component size, routers and KOZs. The 2D and 3D router count in a tile has an upper bound, i. e. $\alpha, \beta, \gamma \in [m - n + 1]$. Therefore, ℓm matrices are introduced which contain all precomputed square roots of tiles' sizes depending on the router and TSV count: $\tilde{F}^{\xi, i} = (\tilde{f}_{\alpha, \beta, \gamma}^{\xi, i}) \in \mathbb{R}^{[m-n+1] \times [m-n+1] \times [m-n+1]}$. The corresponding area requirement of a tile i implemented in layer ξ is given by the i, ξ -th matrix element: $\tilde{f}_{\alpha, \beta, \gamma}^{\xi, i}$. We introduce auxiliary binary variables $h_{\alpha, \beta, \gamma}^{\xi, i, 11}$ for each $i \in [m]$, $\xi \in [\ell]$, and $\alpha, \beta \in [m - n + 1]$, selecting the element of this matrix (Note that the exponent 11 is part of the variable name; variable names are generated from a counter and therefore only appear to be arbitrary at this point). These variables can be arranged in the form of a matrix of the same dimensions as $\tilde{F}^{\xi, i}$. Using an analogy of the Frobenius scalar product of matrices $A = (a_{i,j,k}) \in \mathbb{R}^{m \times n \times q}$ and $B = (b_{i,j,k}) \in \mathbb{R}^{m \times n$

$\langle A, B \rangle = \sum_{i=1}^m \sum_{j=1}^n \sum_{k=1}^q a_{i,j,k} b_{i,j,k}$, the constraint in Equation 23 can be written as m inequalities:

$$a_i + b_i \geq \sum_{\xi \in [\ell]} \langle H^{\xi,i,11} \tilde{F}^{\xi,i} \rangle \left(\sqrt{\eta^{-1}} + \sqrt{\eta} \right) \quad (29)$$

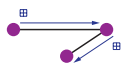
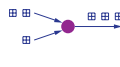
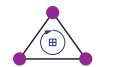
which is a linear equation using $h_{\alpha,\beta,\gamma}^{\xi,i,11}$ as elements in $H^{\xi,i,11}$. It must hold for all $i \in [m]$, $\xi \in [\ell]$ and $\alpha, \beta, \gamma \in [m - n + 1]$:

$$h_{\alpha,\beta,\gamma}^{\xi,i,11} = \begin{cases} 1 & \text{if tile } i \text{ is in layer } \xi, \text{ has } (\alpha - 1) \text{ routers,} \\ & (\beta - 1) \text{ 3D routers, and } (\gamma - 1) \text{ KOZs} \\ 0 & \text{else} \end{cases} \quad (30)$$

The linearization error can be reduced by using multiple linear equations that piecewise approximate a segment of the iso-area line $a_i b_i = F_{\alpha,\beta,\gamma}^{\xi,i}$. This method increases the number of inequalities.

6.3.4.3 Modeling common properties of routing algorithms

Common properties of all routing algorithms are modeled. The actual implemented routing algorithm must be defined separately. We model routing algorithms via the flow $f(s, t)$ [145]. The routing algorithm only traverses existing links between routers. Packets cannot be duplicated or lost, so flow is conserved. This approach allows to verify the routing algorithm: If the constraints are fulfilled, the routing will be connected. Also livelock freedom is proven, since the flow is acyclic. Deadlock freedom cannot be proved with this model, since it does not comprise channels. The resulting constraints are illustrated as follows; inequalities are given in the next paragraphs.

 <p>Flow in E_N</p>	<p>Only existing links can be traversed.</p>	 <p>Flow conservation</p>	<p>Packets are not lost or duplicated.</p>
 <p>Live-lock freedom</p>	<p>Routing algorithms are live-lock free.</p>		

Flow in E_N : The flow in E_N can traverse existing links only. It is defined on a fully connected network graph. If only the actual implemented links in each solution candidate has been used, the size of the constraints would depend on the solution and not on the input constants, which is illegal for MILPs. Therefore, all flow values of edges, which are not found in the network topology E_N , are set to zero by the following $m^2|E_A|$ inequalities for all $(i, j) \in E_A, k, l \in [m]$:

$$f_{(k,l)}^{(i,j)} \leq e_{\{k,l\}}. \quad (31)$$

Flow is conserved: Flow must be conserved, or else packets are duplicated or lost. We define flows in the network E_N with value 1, which are also non-circular. The latter is given by the subsequent constraint. According to [148] (Definition 8.1.), the flow for a vertex v will be conserved if the summed flow from all incoming edges $e \in$

$\delta^-(v)$ is equal to the sum of the flows on the outgoing edges $e \in \delta^+(v)$: $\sum_{e \in \delta^-(v)} f(e) = \sum_{e \in \delta^+(v)} f(e)$. In our model, the incoming edges e to a vertex with index l (i.e. a router) are given by $e \in [m] \times \{l\}$. Analog, the outgoing links e are given by: $e \in \{l\} \times [m]$. This yields the equations for the flow conservation rule for the fully connected network for all $q \in [m]$, $(i, j) \in E_A$ and $l \in [m] \setminus \{i, j\}$ (since the source and the sink do not have flow conservation):

$$\sum_{k \in [m]} f_{(k,l)}^{(i,j)} = \sum_{k \in [m]} f_{(l,k)}^{(i,j)} \quad (32)$$

This relates to $2(m-2)|E_A|$ inequalities. For the source i , the inequality is given by

$$1 + \sum_{k \in [m]} f_{(k,i)}^{(i,j)} = \sum_{k \in [m]} f_{(i,k)}^{(i,j)} \quad (33)$$

and for the sink j , it is given by:

$$\sum_{k \in [m]} f_{(k,j)}^{(i,j)} = \sum_{k \in [m]} f_{(j,k)}^{(i,j)} + 1 \quad (34)$$

Flow is acyclic: Flows in the network must be acyclic for live-lock freedom of routing algorithms. A digraph will have a topological order if and only if it is acyclic ([148], Definition 2.8. and Proposition 2.9.). To generate a topological order, variables $\Gamma_r^{e_A} \in \mathbb{Z}$ are defined such that they enumerate the vertexes in the network E_N for all $e_A \in E_A$ and $r \in [m]$: . The enumeration is modeled for all $k, l \in [m] \times [m]$:

$$f_{(k,l)}^{e_A} > 0 \rightarrow \Gamma_i^{e_A} < \Gamma_j^{e_A} \leftrightarrow f_{(k,l)}^{e_A} = 0 \text{ or } \Gamma_i^{e_A} < \Gamma_j^{e_A} \quad (35)$$

For a binary flow (i.e. $f_{(k,l)}^{(i,j)} \in \mathbb{B}$) this yields the following $m^2|E_A|$ inequalities for all $k, l \in [m] \times [m]$ and $(i, j) \in E_A$ using the constant $c^{17} = m + 1$ (Again, the “exponent” 17 is a variable name):

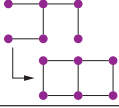
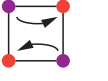
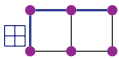
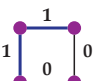
$$\Gamma_k^{(i,j)} + 1/2 \leq \Gamma_l^{(i,j)} + (1 - f_{(k,l)}^{(i,j)})c^{17}. \quad (36)$$

For non-binary flows, the auxiliary binary variable $h_{k,l \text{ OR}}^{i,j 17}$, which is 1 if the flow is non-zero ($h_{k,l \text{ OR}}^{i,j 17} \geq f_{(k,l)}^{(i,j)}$) is introduced. This changes the inequalities to $\Gamma_k^{(i,j)} + 1/2 \leq \Gamma_l^{(i,j)} + (1 - h_{k,l \text{ OR}}^{i,j 17})c^{17}$.

6.3.5 Case study: Modeling elevator-first dimension order routing

As an exemplary application, we model elevator-first dimension-order routing to demonstrate the expressiveness of our approach. For dimension-order routing, each pair of two routers within a layer must span a rectangle with two further routers. Thus, wires between all neighboring routers must be implemented. The flow is binary: the flow is 1 on the paths following the routing algorithm and zero on all other edges. The constraints can be illustrated as follows. The inequalities are provided in Section a.5.4. They require the implementation

of a search algorithm for the next router with a TSV by means of an MILP, which is rather inefficient.

	<p><i>Connect neighboring routers</i></p> <p>Forces wires between routers within a layer to enable routing.</p>		<p><i>Topology</i></p> <p>For dimension order routing routers must form rectangles.</p>
	<p><i>Flow for the routing algorithm</i></p> <p>“Extended” dimension order routing is performed.</p>		<p><i>Flow is binary</i></p> <p>Packets follow a single path in deterministic routing.</p>

6.4 HEURISTIC ALGORITHM

The proposed model in form of MILP provides an optimal solution. However, it is not efficient and therefore hardly provides any solution for large input sets. Defining a modified MILP model, with higher performance, is not a promising approach. Instead, we propose an efficient heuristic to construct approximate solutions. One major issue with the MILP is that a multitude of variables are optimized simultaneously. This is essential to achieve a precise solution but poses a severe performance limitation. To tackle this, we propose to dissect the problem and split it up into a set of steps, each of which can be efficiently optimized. This reduces the optimization potential, but a fast heuristic can be developed.

The proposed heuristic applies an incremental approach as proposed in Section 5.6.2. We identify the following five steps, for each of which a graphical representation is given in Figure 41.

① Components are partitioned into layers (Figure 41a). Since layers of identical size are stacked, the overall area consumption of the SoC is determined by its largest layer. Thus, the optimization adds up the area requirements of components per layer and targets a homogeneous distribution. This step appears to be similar to partitioning, which is known from VLSI (Very Large Scale Integration) design. In partitioning, units are clustered by their communication, yet here we optimize area, power and performance. At the end of the first step, power and performance of components is optimal; the area is only approximated, but yields similar-sized layers.

② A floor plan is given per layer for bounding boxes of components (Figure 41b). We use rectangular bounding boxes, which is in line with the tiles of the MILP model. This optimization targets two objectives: Minimizing the layer area by reducing the sizes of bounding boxes and tightly packaging them, and minimizing the communication within layers by locating components according to their bandwidth (pairs of components with large communication requirements are located adjacent in layers). This step determines the size of the layer rather precisely; only the area of 3D communication is missing.

Furthermore, by only optimizing communication within layers, the complexity of this problem is vastly reduced.

③ The communication between layers is optimized: The number of TSV arrays connecting layers is calculated (Figure 41c). There are two adversaries: Many TSV arrays reduce the bandwidth via TSV arrays, but for each array, the area of its KOZ and router ports are required. Therefore, an optimum with low link utilization and area is required. Determining only the TSV array count but not their precise location, allows for a very efficient solution.⁴ Up to now, interlayer and intralayer communication have been optimized separately.

④ The global communication is accounted for: TSV arrays and 3D routers are placed by adding them to bounding boxes of components (Figure 41d); this results in bounding boxes, as already shown in the MILP model (as tiles). This is done using the TSV connection model and the actual routing algorithm, so that the paths of data are known. Constrained by the given TSV count and the order of components in their layer, this is an extensive optimization which accounts for very detailed network information. At the end of this step the size of bounding boxes increases and they might overlap.

⑤ The solution is legalized to accommodate the area of the added connections.⁵

In order to make optimization fast, i. e. feasible, step ① splits the design in layers before floor planning each layer in step ②. Deciding step ③ as late as possible is beneficial for network design, since then component floor plans can be accounted for. This avoids overallocation of network resources. Steps ③ and ④ are separated to allow for comparison against standard approaches.

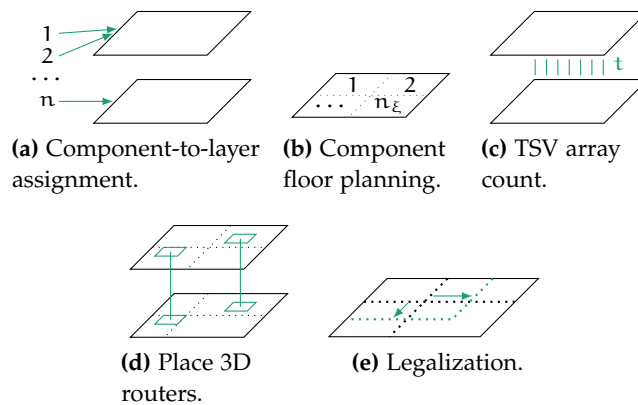


Figure 41: Visual representation of the steps.

⁴ The TSV array count is, in short, referred to by TSV count throughout this thesis.

⁵ Strictly speaking, this step might generate illegal solutions by violating the constraint that the distance between routers must be below D if connected vertically. However, this issue is easily fixed by post-processing or by using a slightly reduced D .

6.4.1 Heuristic algorithm

Using the introduced steps, we develop an efficient heuristic. The solutions proposed per step represent only one possible implementation. The structure of the resulting heuristic is illustrated in Figure 42 as a flow chart and explained in the following paragraphs.

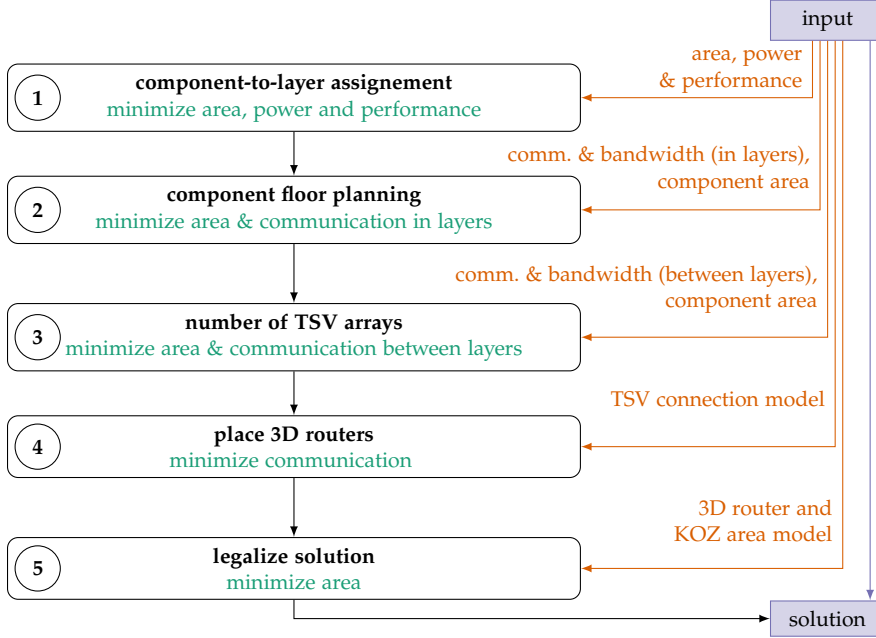


Figure 42: Flow chart of heuristic algorithm.

① The heuristic starts with a component-to-layer assignment. An optimized assignment function $\alpha : [n] \rightarrow [\ell]$ is found, which assigns components to layers w.r.t. an optimized area, power and performance. We use this objective function, with weights ω_C, ω_E and $\omega_P \in \mathbb{R}$:

$$c_{(1)} = \omega_C h + \omega_E \sum_{i \in [n]} f_E(i, \alpha(i)) - \omega_P \sum_{i \in [n]} f_P(i, \alpha(i)) \quad (37)$$

whereas

$$h = \max_{k \in [\ell]} \left\{ \sum_{i \in \{j | \alpha(j) = k\}} f_C(i, \alpha(i)) \right\}. \quad (38)$$

This is an integer linear program, of which the inequalities are given in Section 4.7.1. The first step is shown in Figure 42 on top; it takes technology parameters of components as input and returns an assignment.

② In a second step, relative positions and bounding boxes are determined for components in their layer w.r.t. minimized area and communication. This is called component floor planning. To remain with the structure of a grid, which is the well-known topological base of the MILP model, we define an order that describes the relative positioning of components by assigning them to rows and columns. Each

component gets its own bounding box in the grid, which we display by assigning a row number and a column number to the component. The advantages of using the order to describe the positioning of components are twofold: First, efficient (polynomial-time) area minimization via a linearized LP or an exact semidefinite programming (SDP) is possible under a given maximum aspect ratio η . Second, it allows to modify the size of bounding boxes (to add routers and TSVs) while maintaining the order. The solution can be legalized and the bounding boxes are non-overlapping after adding 3D infrastructure. The LP and SDP, originally published in [JM 4], are formulated as follows:

The area optimization problem is formulated as follows: Assume we have a given order of lk or fewer components in l rows and k columns. Each component has the size $a_{i,j}$, for certain $i \in [l]$ and $j \in [k]$. $a_{i,j} = 0$ if there is no component at row i , column j for all pairs $(i,j) \in [l] \times [k]$. The height of rows is denoted by $r_i \in \mathbb{R}$ for all $i \in [l]$. The width of columns is denoted by $c_j \in \mathbb{R}$ for all $j \in [k]$.

$$r_i c_j \geq a_{i,j} \quad \text{for all } i \in [l], j \in [k]. \quad (39)$$

The objective function is: Minimize the side length of a square that encloses all bounding boxes.

$$\max \left(\sum_{i \in [l]} r_i, \sum_{j \in [k]} c_j \right) \rightarrow \min \quad (40)$$

Using LP and linearization: The optimization is conducted subject to

$$r_i \geq \eta c_j \quad \forall i \in [l], \forall j \in [k] \quad (41)$$

$$c_j \geq \eta r_i \quad \forall i \in [l], \forall j \in [k] \quad (42)$$

$$r_i + c_j \geq \sqrt{a_{i,j}\eta} + \sqrt{a_{i,j}/\eta} \quad \forall i \in [l], \forall j \in [k] \quad (43)$$

with a given bounding box aspect ratio limit $\eta \in (0, 1)$. The linearization is conducted in Equation 43. The formulation as an LP allows for an efficient solution.

Using SDP: Variables in SDPs are positive semidefinite matrices. Here, these are lk variables $X_{k(i-1)+j}$. We define them such that the desired product $r_i c_j \geq a_{i,j}$ is directly modeled:

We set lk variables $X_{k(i-1)+j}$ such that

$$X_{k(i-1)+j} = \begin{bmatrix} r_i & \sqrt{a_{i,j}} \\ \sqrt{a_{i,j}} & c_j \end{bmatrix} \succeq 0, \quad \forall i \in [l], \forall j \in [k] \quad (44)$$

These are positive semidefinite matrices; thus each principal minor is greater or equal to 0:

$$\det(X_{k(i-1)+j}) \geq 0 \quad (45)$$

$$\Rightarrow r_i c_j - a_{i,j} \geq 0 \quad (46)$$

$$\Rightarrow r_i c_j \geq a_{i,j}, \quad \forall i \in [l], \forall j \in [k] \quad (47)$$

We formulate it as an SDP. The objective function minimizes the variable x subject to these constraints:

We assign the corresponding area values to each matrix:

$$2\sqrt{a_{ij}} \leq \left\langle \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, X_{k(i-1)+j} \right\rangle \leq 2\sqrt{a_{ij}}, \quad \forall i \in [l], \forall j \in [k] \quad (48)$$

For each $i \in [l]$, the upper left entry of the matrices $X_{k(i-1)+j}$ has the same value for all $j \in [k]$ (this models r_i):

$$0 \leq \left\langle \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}, X_{k(i-1)+1} \right\rangle + \left\langle \begin{bmatrix} -1 & 0 \\ 0 & 0 \end{bmatrix}, X_{k(i-1)+j} \right\rangle \leq 0 \quad (49)$$

For each $j \in [k]$, the lower right entry of the matrices $X_{k(i-1)+j}$ has the same value for all $i \in [l]$ (this models c_j):

$$0 \leq \left\langle \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix}, X_j \right\rangle + \left\langle \begin{bmatrix} 0 & 0 \\ 0 & -1 \end{bmatrix}, X_{k(i-1)+j} \right\rangle \leq 0 \quad (50)$$

We model the maximum variable x for the objective function:

$$0 \leq x + \sum_{i=1}^l \left\langle \begin{bmatrix} -1 & 0 \\ 0 & 0 \end{bmatrix}, X_{k(i-1)+1} \right\rangle \quad (51)$$

$$0 \leq x + \sum_{j=1}^k \left\langle \begin{bmatrix} 0 & 0 \\ 0 & -1 \end{bmatrix}, X_j \right\rangle \quad (52)$$

Areas of components are constrained by an aspect ratio η (to maintain a lower length of a critical path). Note that this aspect ratio is not violated by the relation between r_i and c_j . Rather, a component can find a rectangle inside the bounding box given by $r_i c_j$. This rectangle has the size of the component and the length of each one of its edges is within the aspect ratio η . We formulate for all $i \in [l]$ and for all $j \in [k]$:

$$\sqrt{\eta a_{i,j}} \leq \left\langle \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}, X_{k(i-1)+1} \right\rangle \quad (53)$$

$$\sqrt{\eta a_{i,j}} \leq \left\langle \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix}, X_j \right\rangle \quad (54)$$

We optimize the assignment of components to rows and columns by simulated annealing. In general, simulated annealing algorithms optimize as follows: An initial solution candidate is generated. Starting from there, neighboring candidates are constructed and their costs are evaluated. If costs decrease, the new solution will be accepted. If costs are higher, the solution will only be accepted if a random variable allows for it. The probability of accepting a solution with higher costs than the current (local) minimum depends on an initial "temperature" parameter, which is reduced by a "cooling" percentage parameter. The optimization terminates after a given number of iterations.

Regarding the relative placement of components, $n_\xi \times n_\xi$ rows and columns will be sufficient for every possible placement, if there are n_ξ components layer ξ . As an initial solution, we position components with an area-efficient approach. The optimization of row and column sizes will yield large white space if the size difference between adjacent components is large and the area requirements alternate. Therefore, we cluster components in rectangles by their sizes in descending order. Rows (columns) are filled with components, as long as the summed height (width) of the chip under assumption of squared component shapes is smaller than the width (height). Otherwise, a new column (row) is filled. In every iteration of the simulated annealing, we move a component, selected from a uniform random distribution, to any other row or column, also selected from a uniform random distribution. If another component is located there, the components will be swapped. After deletion of rows and columns without components, the aforementioned optimization minimizes the area of the layer. For this solution candidate, we evaluate the following objective function, with the weights ω_a to minimize area and ω_c for communication. The communication between two components i, j is calculated by summing their bandwidth $u(i, j)$ multiplied by their hop distance $\Delta(i, j)$, i. e. their distance in numbers of rows and columns. We minimize:

$$c_{(2)} = \omega_a \max \left(\sum_{i \in [n]} r_i, \sum_{j \in [n]} c_j \right) + \omega_c \left(\sum_{i \in [n]} \sum_{j \in [n]} u(i, j) \Delta(i, j) \right) \quad (55)$$

This second step is shown in Figure 42; it takes intralayer communication and component area as input and returns bounding boxes.

③ In a third step, the TSV array count connecting adjacent layers $t : [\ell - 1] \rightarrow [n]$ is determined by optimizing communication between layers and minimizing area requirements of KOZs and routers. Based on the router and TSV model as defined in Section 6.2.1, all possible implementations are modeled by a “TSV graph”, with all bounding boxes as vertices and all possible (physically implementable) TSV arrays as edges. The subgraphs of nodes located in adjacent layers are bipartite. An example is shown in Figure 43 for a chip with two layers. In this example, components 1 and 2 are located in the upper layer and components 3 to 6 in the lower. It is physically possible to connect component 1 to all other components in the lower layer; component 2 can only connect to 5 and 6. The vertical links, which can be implemented, form a matching in the TSV graph, as shown by the dashed links in Figure 43. The upper bound for TSV arrays connecting two layers is given by the largest maximum matching in these subgraphs and it is rather low, since it is bounded by the number of components per layer. Therefore, an optimal solution can be found efficiently by iterating all numbers of TSVs. The locations of bounding boxes for components are already known. The actual location of TSVs is unknown and only determined in the next step. Therefore, it

is approximated by dissecting the covered area of bounding boxes in equal-sized rectangles. The number of rectangles is given by the next

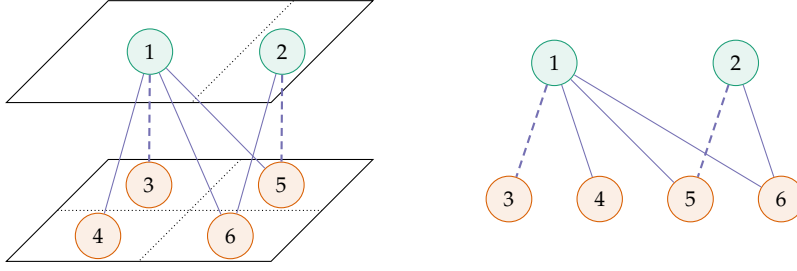


Figure 43: Exemplary chip with corresponding TSV graph, which is bipartite. Selected TSVs are dashed and demonstrate matching property.

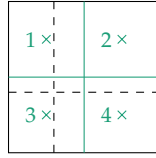


Figure 44: Approximated places of TSV arrays in green for four bounding boxes.

larger square number of the number of TSVs (i. e. , for 6 TSV arrays, 9 rectangles are defined, with 3 rows and 3 columns). This retains the well-known topological grid. TSVs are positioned in the center of these rectangles, as shown in Figure 44, filling the rectangles randomly. The distance of the center of each component bounding box to the next TSV array is calculated, in the form of a hop distance, and multiplied by the communication of the component to all other layers, both in upward and downward direction. The sum of these weighted distances is an estimate of the communication costs. We call the estimated communication costs c . The best solution candidate is selected using the objective function, with weights ω_a and ω_c :

$$c_{(3)} = \omega_a \sum_{\xi \in [\ell-1]} (f_{KOZ} + f_{R3D})t(\xi) + \omega_c c \longrightarrow \min \quad (56)$$

The third step is shown in Figure 42 in the middle. It takes interlayer communication and component area as input and returns the number of TSV arrays.

④ In the fourth step, the position of 3D routers are determined and TSV arrays are associated with bounding boxes. This increases the area requirement of bounding boxes. If necessary, routers are added to connect the routing algorithm, as already described in the MILP model. Simulated annealing is used to find an optimized network E_N . The possible connection candidates are already given in the “TSV graph” from the previous problem. Since routers are only allowed to connect to a single TSV array per direction, a legal solution must be a matching in the subgraph of the “TSV graph” with edges

from adjacent layers. The initial solution generates a random matching for every pair of adjacent layers, using a simple greedy algorithm, with a matching size smaller or equal to the number of TSV arrays as determined in step three. The neighbor function randomly selects a pair of layers and includes a new random TSV array from the “TSV graph”. If an added connection violates the matching, all TSVs, which connect the routers at the end of the novel connection, are deleted. If the maximum number of TSVs is violated, another random connection is deleted. Then, a greedy algorithm finds a novel matching, with a size smaller or equal to the number of TSV arrays. The objective function minimizes the communication for the given network E_N and the flow of the routing algorithm, with Equation 17. At the end of this step, a complete network graph is defined. The fourth step is shown in Figure 42 in the last node; it takes the TSV connection model as input and returns bounding boxes for TSVs and routers.

⑤ After adding 3D router area and KOZs, the area of bounding boxes may be too small or boxes may overlap. In the fifth step, the solution from step four is legalized while retaining the order of components in layers, rows, and columns. The LP or SDP, which is formulated in step two (floor planning of components), is used again. This step is shown in Figure 42 at the end. It returns an optimized solution based on the router and TSV area model. With this final step, a complete network graph and non-overlapping bounding boxes are defined and the network can be designed. The solution can be further optimized by determining the exact location of routers, components and TSV arrays within their bounding boxes. This post-processing is not discussed in this thesis, since standard approaches from layout synthesis can be used.

6.5 PERFORMANCE AND COMPUTATIONAL COMPLEXITY

Component positioning and NoC synthesis contain similar subproblems as layout synthesis, since problems similar to floor planning and partition must be solved. Layout synthesis is an NP-hard problem [134]. The solution space is significantly larger for 3D systems: As reported in [33], the possible arrangements in the solution space for 3D chip floorplanning is increased by factor $N^{n-1}/(n-1)!$ with N components and n layers in comparison to 2D. Thus, system-level optimization of A-3D NoCs is naturally NP hard, as well, and (possibly) has a larger solution space. Standard divide and conquer approaches, as conducted in layout synthesis, cannot be applied directly due to interdependencies: Positions of components in one layer influence the positions in other layers. Some constraints, such as non-overlapping tiles, are individually hard to satisfy.

6.5.1 MILP model

The complexity of the MILP model can be demonstrated by exemplary implementation. The model is implemented using MATLAB R2018a and optimized with IBM CPLEX 12.8.0 [149]. Inequalities and variables are generated automatically from input sets. With the use of sparse matrices, millions of inequalities can be stored with less than 16 GB of memory. MATLAB's internal `intlinprog` function is too slow to optimize even small input sets with only a few constraints activated. CPLEX is able to find a solution candidate in few minutes for less than 3 layers and 5 components on an Intel i7-6700 CPU running Windows 10. For larger, realistic sets (with more than 5 components), the performance of CPLEX is too low, as well. Therefore, efficient heuristics are necessary.

6.5.2 Heuristic algorithm

We analyze the computational time of the heuristic algorithm for each step depending on the input and parameters.

① The calculation time of component-to-layer assignment is difficult to estimate without experiments, since it is solved using MILP. Experiments show that the problem is solved in less than a second, even for larger input sets (cf. Table 4).

② The component floor planning uses simulated annealing; its run time is determined by the maximum number of iterations and the time required to generate solution candidates and assess the cost function. Calculating an initial solution and the neighbor function is costs $O(m)$, since a random vector is generated. For the cost function, the area of each configuration must be minimized. Therefore, an LP (with linearization) or a SDP is used. Both are polynomial time optimizations. We use an ellipsoid algorithm for an upper bound of the LP's computation time, i.e. a worst case approximation. Following Khachiyan's approach, solving an LP costs $O(n^4L)$, with n variables and L input bits [150]. A layer with k rows and l columns yields $O(k+l)$ variables. L is essentially the bit size of x_{\max} and η , so it is bounded. For the cost function, the pairwise distance between communicating components is calculated, i.e. takes $O(kl)$.

③ To calculate the number of TSVs, we inspect every possible solution and find the global minimum. The largest maximum matchings are calculated, which takes $O(|E_N|n) = O(n^3)$ ([145], Theorem 10.4). The number of inspected solutions is equal to the largest maximum matching, which is a constant smaller than n^{l-1} ; for each candidate, a cost function is evaluated by calculating pairwise distances between components, i.e. $O(n^2)$.

④ Placing the 3D routers uses simulated annealing. Initial and neighboring solutions are generated with a greedy algorithm, which

constructs a matching by iterating all possible edges in the network, i. e. $O(m^2)$. The cost function calculates the routing algorithm for all edges in E_A yielding $|E_A| \leq n^2$ iterations. Calculating the routing function is of similar complexity as breadth-first search in the network, i. e. takes $O(m + m^2)$ ([145], Theorem 2.19). Together this yields $O(n^2m^2)$.

⑤ The legalization uses the same LP or SDP as already discussed.

Summing up, the proposed heuristic allows for an efficient solution in polynomial time, since it is in $O(n^2m^2L)$.

6.6 RESULTS

Both the MILP model and the heuristic algorithm are implemented using MATLAB. The following results are generated from this implementation. We use a Core i7-6700 running at 3.4 GHz, Windows 10 operating system, MATLAB R2018a. For optimizing LPs, we use CPLEX 12.8.0 [149]. For optimizing SDPs, we use Mosek 8.1 [151]. Both tools utilize all 8 logical cores with 8 threads. The evaluation is conducted using homogeneous 3D SoC examples. Properties of “correct” solutions are known, since homogeneity allows for manual optimization. Therefore, we demonstrate that the results are correct and as expected (Section 6.7.2). Later on, we compare our heuristics with state-of-the-art NoC planning for a heterogeneous 3D SoC to demonstrate area, power and performance advantages of the proposed heuristic algorithms.

6.6.1 Case study for technology model

The RD length D can be calculated using the formula provided by technology vendors. We encapsulated this into python modules [JM 14], which are available as open source. As an exemplary case study, we connect two layers with a 9 bit vertical link. One layer is implemented in a 45 nm commercial digital technology and the other in a 180 nm commercial mixed-signal node. For a target frequency of 10 MHz, with 20% delay margin left for the remaining circuit, the length of the RD from 45 nm node to 180 nm node is 0.07 m; in opposite direction, it is 0.03 m. This shows the difference for both directions, for which we use the minimum to determine D . Please note, that we cannot assess the precision of the models as those are provided by the vendors.

6.6.2 Comparison to related work

The proposed heuristic algorithm, as it is, cannot be compared to the related work, since it targets designs not considered previously. Nonetheless, individual steps are similar and therefore can be com-

pared. It is not useful to compare steps ① and ③: The first step is merely a linear optimization as a preparation of the remainder of the heuristic algorithm. The third step determines the number of TSVs. The result is highly dependent on the parameters of the cost function. Therefore, realistic parameters must be found for each set of technologies. This is not possible here, since the properties of 3D technology are not completely available to academia. Therefore, the related works, such as [152], use a fixed ratio between implemented TSVs and available TSV positions. We compare steps ② and ④ in the next sections. The SDP or LP in step ⑤ is reused from the second step and therefore is not assessed individually.

6.6.2.1 Step 2 and 5: Placement of components

In the second step of the heuristic algorithm, components are placed. Since different types of components are integrated into SoCs, the size of each component may differ even within layers. Traditional approaches assume identical cores' sizes and thus, regular 2D mesh topologies [106]. However, [142] maps quadratic-shaped cores of varying size in a 2D mesh NoC. The work proposes a MILP and a heuristic approach. The aim is optimizing network performance by minimizing transmission energy. We compare the results of our heuristic algorithm, step ②, with the results of [142] using the three benchmarks provided by [142], namely H.256 decoder mp3 decoder, H.263 encoder mp3 decoder, and mp3 encoder mp3 decoder. Traffic patterns are taken from benchmarks in [106] and the cores' area from [142]. The results of area and network performance are given in Table 1. The area is measured by the area of the complete chip embracing the components. Network performance is measured by accumulated link load (measuring delay) and maximum link load (measuring throughput). Five data sets are given per benchmark: First, the baseline is given by [142] for their mesh-based solutions; we do not compare with non-mesh solutions, since they allow for additional degree of freedom and therefore provide an unfair baseline. Second, this configuration is optimized using the proposed SDP. The aspect ratio in the SDP is set to $\eta = 0.1$, so that non-quadratic, rectangular shapes of components with that maximum aspect ratio are possible. Thereby we assess the optimization potential from the additional degree of freedom in cores' shapes. Third, the initial solution for the subsequent simulated annealing is given, which is an area-efficient, communication-inefficient solution. It shows the optimization potential in terms of area. Fourth, the proposed simulated annealing is executed 30 times with 15,000 iterations, an initial temperature of 30 and a cooling of 0.98. The results of the 30 reruns are averaged and the standard deviation is calculated. We set the weight of the area in the cost function to zero to optimize communication only. Fifth, we balance the weights in the cost function and prioritizes neither area nor communication. A

single run of the simulated annealing terminates after approx. 17 minutes on a Windows 10 workstation using an Intel i7-7740X at 4.3 GHz processor. The results are discussed in Section 6.7.1.

6.6.2.2 Step 4: Placement of vertical links (TSV array placement)

There are multiple publications on methods to place vertical links in a 3D NoC under the assumption of a given TSV count (usually in the form of a ratio of implemented TSVs). At the time of writing this thesis, [152] is the most recent work on TSV placement with simultaneous application mapping on a 3D mesh NoC. The authors propose an ILP model and a particle swarm optimization (PSO) algorithm. Since the mapping is already conducted in step ②, we only compare with the TSV placement part of this work. Therefore, we take video object plane detection (VOPD) and double video object plane detection (DVOPD) [106] as benchmarks. The other benchmarks from [106] are smaller; comparison is not useful, because both the PSO and the proposed heuristic algorithm using a simulated annealing will find the global minimum in a very small search space. We chose an arbitrary but fixed mapping for both benchmarks. We use 20 reruns for both PSO and simulated annealing and both algorithms have approximately the same computation time budget. The parameters of the PSO are given by [152] ($k_1 = 1$, $k_2 = 0.04$, $k_3 = 0.02$). The parameters for the simulated annealing are: initial temperature 30, cooling 0.97, 1,000 iterations. Both [152] and the proposed heuristic algorithm use the same objective: minimizing bandwidth times communication hop distance. The results are shown in Table 2 for VOPD and in Table 3 for DVOPD. The proposed heuristic algorithm allows for up to 15% improved performance.

6.6.3 Small homogeneous 3D SoC

For very small input sets, both MILP model and heuristic algorithm can generate optimized results. This allows to compare them. We take the following input sets: We use (homogeneous) 3D SoCs with $\ell = 2$ layers and 5 components. Components require 10 area units (A), routers per port $3/5 A$ and KOZs are $2 A$ large. The performance and energy consumption of all components and routers is identical. The maximum length of the RD is set to $5\sqrt{A}$. We model a fully-adaptive routing algorithm, i. e. we only require connectivity of the network. We do not model elevator-first dimension order routing, since this has a significant negative impact on the optimization performance. All weights of the cost functions are set to 1. The component communication digraph has bidirectional link with bandwidth 1 between subsequent components, as shown in Figure 45 for five components.

The result of the optimization using the MILP is shown for 5 components in Figure 46. The model comprises 23,277 inequalities and

	AREA [A]			COMMUNICATION [HD MB]			BANDWIDTH [MB/s]		
	MEAN	STD	RATIO	MEAN	STD	RATIO	MEAN	STD	RATIO
H256 DEC mp3 DEC	BASLINE [142]	11301	—	—	19858	—	4060	—	—
	BASLINE WITH SDP	10178	—	-9.94%	19858	—	4060	—	0.0%
	INITIAL SOLUTION	7902	—	-30.1%	33707	—	7994	—	+96.9%
	SA COMMUNICATION	11699	1598	+3.52%	20449	404	4265	201	+5.05%
	SA BALANCED	8244	505	-27.1%	21280	624	4452	674	+9.66%
H263 ENC mp3 DEC	BASLINE [142]	12535	—	—	255324	—	84884	—	—
	BASLINE WITH SDP	10178	—	-18.8%	255324	—	84884	—	0.0%
	INITIAL SOLUTION	6993	—	-44.2%	525537	—	85244	—	+0.42%
	SA COMMUNICATION	15762	1723	-25.7%	241479	15333	73012	14302	-14.0%
	SA BALANCED	10474	2148	-16.4%	250187	14763	73161	17497	-13.8%
mp3 ENC mp3 DEC	BASLINE [142]	8568	—	—	17546	—	4085	—	—
	BASLINE WITH SDP	8091	—	-5.57%	17546	—	4085	—	0.0%
	INITIAL SOLUTION	7281	—	-15.0%	39171	—	6560	—	+60.1%
	SA COMMUNICATION	10779	1460	+25.8%	17341	342	5065	906	+24.0%
	SA BALANCED	8516	796	-0.61%	17572	487	4974	902	+21.8%

Table 1: Area and performance comparison with benchmarks [142]. The simulated annealing is executed with 30 reruns, initial temperature 30, cooling 0.98 and 15,000 iterations. The SDPs allow for an aspect ratio of up to 0.1.

TSV COUNT	NETWORK PERFORMANCE [Hd Mb]				DIFFERENCE
	PSO		PROPOSED		
	MEAN	STD	MEAN	STD	
1	12229	0	12229	0	0%
2	10591	581	9005	0	15%
3	8894	102	8659	0	3%
4	9013	364	8595	0	5%
5	8725	155	8595	0	1%
6	8723	148	8595	0	1%
7	8595	0	8595	0	0%
8	8595	0	8595	0	0%
AVERAGE IMPROVEMENT					3.125%

Table 2: VOPD benchmarks [152] network performance comparison (hop distance [HD] times bandwidth [Mb]) in a $4 \times 2 \times 2$ NoC. 20 reruns for PSO and simulated annealing with same computational time budget.

TSV COUNT	NETWORK PERFORMANCE [Hd Mb]				DIFFERENCE
	PSO		PROPOSED		
	MEAN	STD	MEAN	STD	
1	43330	0	43330	0	0%
2	38274	163	37954	395	1%
3	34636	0	33854	0	2%
4	34217	674	32382	0	5%
5	33249	555	31014	0	7%
6	32351	699	30168	0	7%
7	31920	575	29916	0	6%
8	30767	679	29744	0	3%
9	30767	679	29744	0	3%
10	30318	453	29712	0	2%
11	30235	409	29712	0	2%
12	29764	69	29712	0	0%
13	29996	340	29712	0	1%
14	29805	208	29712	0	0%
15	29712	0	29712	0	0%
16	29712	0	29712	0	0%
AVERAGE IMPROVEMENT					2.563%

Table 3: DVOPD benchmarks [152] network performance comparison (bandwidth times hop distance) in a $4 \times 4 \times 2$ NoC. 20 reruns for PSO and simulated annealing with same computational time budget.

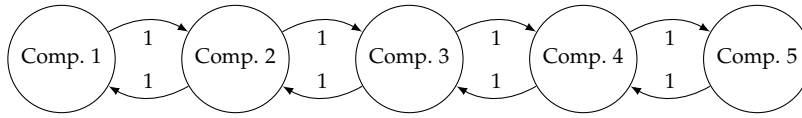


Figure 45: Component communication digraph for small input.

18,121 variables, and takes 7.95s to set up and requires 1.3GB of memory. By using sparse matrices, the memory consumption only changes for larger problems with more than 100,000 inequalities. The optimization is terminated after 10 minutes. It takes 94s to find an initial solution with a gap of 25.93%; after 599.34s a second solution is found and the gap is reduced to 25.00%. The area of the upper layer is 57.43 A and of the lower layer 29.35 A.

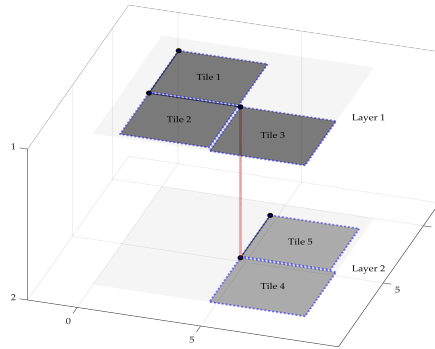


Figure 46: Result for 5 components using the MILP to optimize. Bounding boxes are shown in gray and TSV arrays in red. Skewed TSVs use RD.

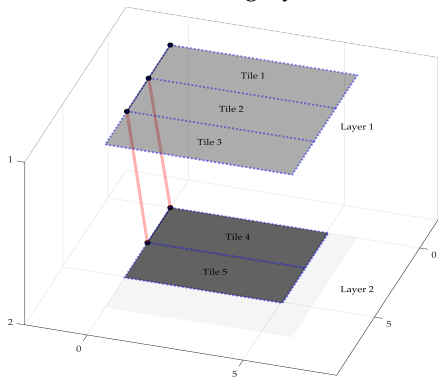


Figure 47: Result for 5 components using the heuristic algorithm with LP area optimization. Bounding boxes are shown in gray and TSV arrays in red. Skewed TSVs use RD.

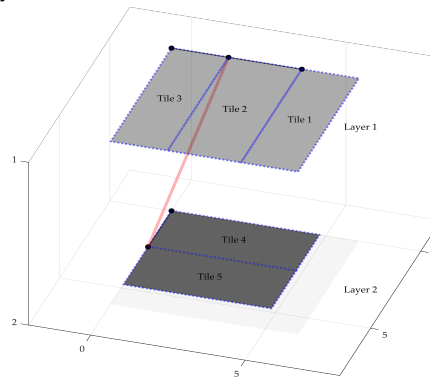


Figure 48: Result for 5 components using the heuristic algorithm with SDP area optimization. Bounding boxes are shown in gray and TSV arrays in red. Skewed TSVs use RD.

An exemplary result for the input using 5 components is shown in Figure 47 (optimization with linearized LP) and Figure 48 (optimization with SDP). To provide comparability with the results of the MILP model, we use the linearized LP. The SDP reduces the linearization error. The simulated annealing for component-floor planning is executed with an initial temperature of 20, 120 iterations and 0.97

PART OF HEURISTIC: EXECUTION TIME	HOMOGENEOUS 3D SoC					
	5 COMPONENTS		40 COMPONENTS		80 COMPONENTS	
	2 LAYERS		4 LAYERS		4 LAYERS	
	LP	SDP	LP	SDP	LP	SDP
① COMP. TO LAYER ASSIGNM.	0.4 s	0.4 s	0.4 s	0.4 s	0.4 s	0.4 s
② LAYER FLOOR PLANNING	20.2 s	146 s	194 s	375 s	609 s	802 s
③ NUMBER OF TSV ARRAYS	0.1 s	0.1 s	0.2 s	0.2 s	0.2 s	0.2 s
④ PLACEMENT OF TSV ARRAYS	3.9 s	3.8 s	308 s	298 s	1246 s	1300 s
⑤ LEGALIZATION	0.2 s	0.2 s	0.3 s	0.5 s	0.3 s	0.5 s
COMPLETE	24.7 s	152 s	503 s	675 s	1856 s	2104 s

Table 4: Execution time of heuristic algorithm for exemplary input sets.

cooling per iteration; the placement of routers uses an initial temperature of 100, and 50 iterations and 0.97 cooling per iteration. All weights of the cost functions are set to 1. As shown in Table 4, the heuristic algorithm using an LP requires approximately 24 s to find a solution; when using the SDP, 152.4 s elapse until termination. In both cases, 1.3 GB of memory are used. The area of the results are given in Table 5. For the LP, the size of the upper layer is 43.0 A and of the lower layer 25.7 A; for the SDP, the area requirements are smaller with 36.8 A and 23.0 A, respectively.

6.6.4 Large homogeneous 3D SoC

The proposed heuristic is executed for a large input instance with a homogeneous 3D SoC. The homogeneity allows to assess the solution from a practical perspective, since properties of optimal solutions are known. We use a SoC with $\ell = 4$ identical layers and a fully connected component graph E_A with 40 and 80 components. The bandwidth requirement is 1 between all components. Components require 10 A, routers 3/5 A per port and KOZs are 2 A large. The performance and energy consumption are identical for all components and routers. The RD can be $5\sqrt{A}$ long. The simulated annealing of component-floor planning is executed with an initial temperature of 20, 120 iterations and 0.97 cooling per iteration; the placement of routers uses an initial temperature of 100, 50 iterations and 0.97 cooling per iteration. All weights of all cost functions are set to 1. Again, both LP and SDP formulation are used.

One exemplary result for the optimization of a SoC with 80 components, using LP for optimization, is shown in Figure 49a. The corresponding exemplary result for the SDP is shown in Figure 49b. For an input set with 40 components, the heuristic algorithm terminates after 503 s using the LP, and after 675 s using the SDP. For an input set with 80 components, the heuristic algorithm terminates after 1856 s using the LP, and after 2103 s using the SDP. The execution times are

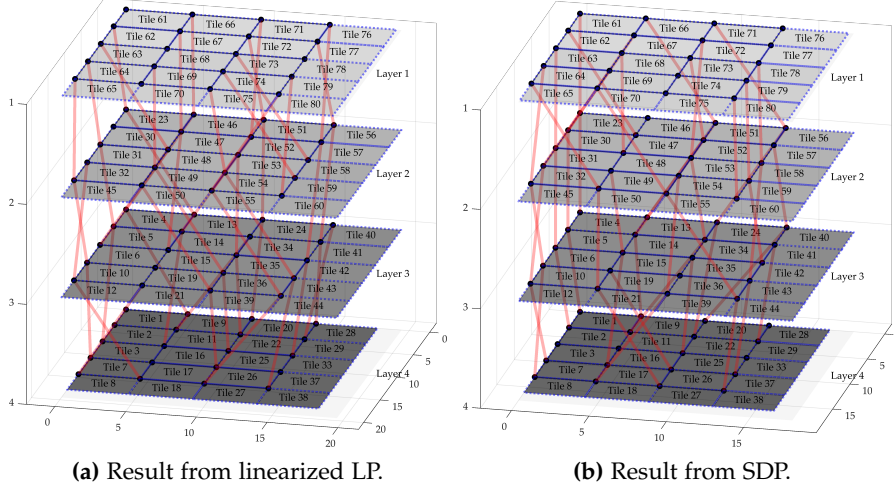


Figure 49: Result for homogeneous 3D SoC with 80 components and with fully-connected component communication graph using linearized LP and SDP. Bounding boxes are shown in gray and TSV arrays in red. Skewed TSVs use RD.

	AREA OF HOMOGENEOUS SoC [A]								
	5 COMPONENTS			40 COMPONENTS			80 COMPONENTS		
	LP	SDP	DIFF	LP	SDP	DIFF	LP	SDP	DIFF
LAYER 1	43.0	36.8	16.8%	211	178	18.5%	364	301	20.9%
LAYER 2	25.7	23.0	19.6%	222	180	23.3%	379	313	21.1%
LAYER 3	—	—	—	214	183	16.9%	378	313	20.8%
LAYER 4	—	—	—	185	154	27.6%	316	261	21.1%
AVERAGE	18.2%			21.6%			21.1%		

Table 5: Area comparison LP and SDP.

shown in detail in Table 4. The average hop distance in the network with 40 components is 3.19 (LP), and 3.29 (SDP). In the example with 80 components, it is 3.76 (LP) and 4.71 (SDP). The area of the 3D SoC with 40 components is 214 A for the LP, and 183 A for the SDP. The area of the 3D SoC with 80 components is 379 A, for the LP, and 313 A for the SDP. The detailed area results are shown in Table 5. The input set with 80 components requires approximately 1.2 GB of memory. Note, that the implementation is not optimized for memory.

6.6.5 3D VSoC case study

We use the proposed heuristic algorithm optimizing a heterogeneous 3D SoC, which follows a typical 3D VSoC application. The SoC consists of two layers: There is a mixed-signal layer on top for analog-digital conversion of image data from a sensor. (The analog sensor layer is not optimized in this example.) Furthermore, there is a digi-

tal layer, for digital data processing. The chip implements 18 components, 9 of which are analog-digital converters (ADCs), that can only be implemented in the mixed-signal layer, and 9 processors, that can be implemented in both layers, but are larger and perform less in the mixed-signal layer than in the digital layer (100 and 130 A). The ADCs are smaller than the processors in the digital layer with 25 A. Each ADC sends data to one processor. The processors send data among each other, i. e. a fully connected component subgraph. We do not consider energy and performance in this example and set it to the same value in each layer for each component. This is reasonable, since the LP in the first step must not be evaluated due to its reduced complexity.

The traditional design for this chip is shown in Figure 50a. The ADCs are located in a 3×3 grid in the mixed-signal layer; the processors are located in a 3×3 grid in the digital layer. Since traditional approaches do not use a RD and rely on same sized components, the grids in both layers are similar and routers are located at the same positions. Please note the white space in the mixed-signal layer and the dense packaging in the digital layer.

The design, as a result of the proposed heuristic algorithm, with 0.1 maximum aspect ratio, 500 iterations, a cooling of 0.97 and an initial temperature of 30 in the simulated annealing is shown in Figure 50b. Three processors are located in the mixed-signal layer to save overall implementation costs. Fewer TSV arrays between routers are implemented than in the traditional approach. The run time of the example is 93 s.

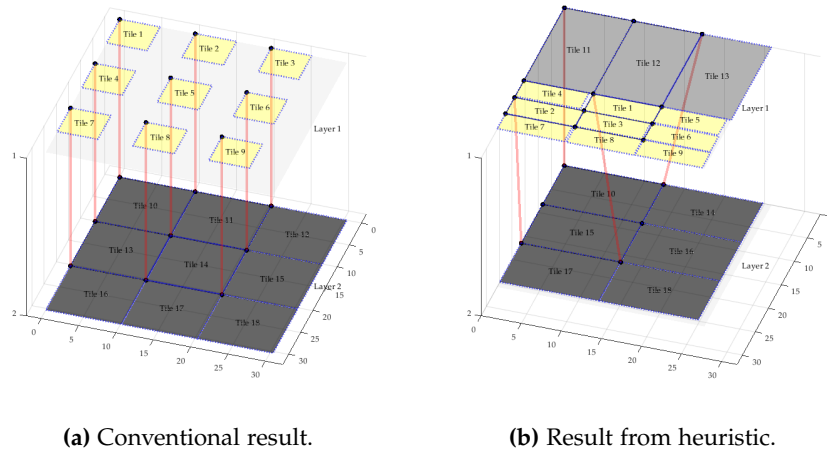


Figure 50: Design of a heterogeneous 3D SoC with 9 ADCs (yellow) in a mixed-signal layer (on top) and 9 processors (gray) with a digital layer (on bottom). Bounding boxes are shown in gray and TSV arrays in red. Skewed TSVs use RD.

6.7 DISCUSSION

6.7.1 *Comparison to related work*

6.7.1.1 *Step 2 and 5: Placement of components*

The placement of different sized components in each layer is compared to [142] in Table 1. Applying the SDP, which offers further optimization potential of area by non-quadratic component areas, shows a reduced area of between 5.57% and 18.8%. Communication is not changed, since the configuration of components is unchanged.

The initial solution for the simulated annealing only optimizes area without considering communication. This shows an area optimization potential between 15.0% and 44.2%. This is the magnitude by which area optimization is possible for each individual benchmark. Please note, that communication properties are worse, as expected, by an increase of up to 123.3%.

The proposed simulated annealing begins with this initial solution and is able to restore the good properties of the communication as given by the baseline: The communication is only 2.98% worse and the maximum link load 5.05% worse on average for the H.256 decoder mp3 decoder. Communication and link load is even better than the baseline on average for the H.263 encoder mp3 decoder, by 5.42% and 14.0% respectively, which is the result of different cost functions between our approach and [142]. In the case of the mp3 decoder mp3 encoder, communication is slightly improved by 1.17%, while the maximum link load is worse by 24%. These three benchmarks demonstrate that the proposed heuristic algorithm is able to optimize network performance even though it is given a very bad initial solution (in terms of communication/link load). Therefore, it is reasonable to use an area-optimized initial solution.

Both area and communication are optimized in a realistic scenario. The results for this are shown in the last rows of Table 1 by using a balanced cost function, in which the weights for area and network performance are set such that both of them participate approximately equally to the overall costs. The optimized H.256 decoder mp3 decoder has a area reduction of 27.1%, while the communication and link load are worse than the baseline on average between 7.16% and 9.66%. This shows that large area reductions will be possible for benchmarks with comparable communication properties. Optimization of the H.263 encoder mp3 decoder reduces area by 16.4%, the accumulated network load is better by 2% and the maximum link load is better by 13.8% on average. This is a very promising result: The heuristic algorithm finds a better solution than state-of-the-art. The result for the mp3 decoder mp3 encoder shows a reduced area of 0.61%, a communication reduction by 0.15% and a maximum link

load that is 21.8% worse, on average. In general, better results can be achieved by a longer optimization time.

In summary, the proposed heuristic algorithm allows for up to 16.4% reduced area with simultaneous improvements in network performance over state-of-the-art. In general, up to 44.2% area reduction is possible, while the communication properties hardly improve. The difference between the second and the third benchmark shows that precise adjustment of parameters of the heuristic algorithm is necessary. Furthermore, the quality of the results has increased when larger than 17 minutes optimization time was used.

6.7.1.2 *Step 4: Placement of vertical links*

The placement of vertical links is compared to [152]. The results are shown in Table 2 for the VOPD benchmark and in Table 3 for the DVOPD benchmark. In the case of the VOPD benchmark, the proposed simulated annealing finds better solutions than the PSO [152] with up to 15% better communication (measured in bandwidth times hop distance). Also, the SA is more efficient: Within the same time budget, it always finds the global optimum in every rerun. This is shown by a standard deviation of 0 and by means of ILP formulation. For a TSV count of 2 – 6, the PSO does not find the global minimum and therefore yields worse communication. We achieve an average improvement of 3.125% for the VOPD benchmark. The results for the DVOPD benchmark demonstrate the efficiency of the proposed simulated annealing, as well. It converges to the global optimum for every TSV count except 2. This allows for up to 7% better communication costs and 2.56% average improvement.

6.7.2 *Validity and quality of the results*

It is not trivial to assess the validity and the quality of the results of the heuristic algorithm, since the calculation of an optimal solution (using the MILP model) is impossible. Furthermore, the scope of this optimization is orthogonal to all existing approaches; thus, in comparison to existing benchmarks is not possible. Nonetheless, evaluation is possible using two scenarios: First, the MILP model can calculate solutions for very small input sets, which can be compared to the results of the heuristic for the same input set. Second, input scenarios can be used, for which properties of good solutions are already known. Therefore, we use a homogeneous 3D SoC with a fully-connected communication graph between all components.

6.7.2.1 *Small homogeneous 3D SoC*

We evaluate two small input sets to compare the results of the MILP model optimization and the proposed heuristic. In terms of perfor-

mance, the heuristic shows a clear advantage: To generate a first solution candidate for a homogeneous 3D SoC with two layers and 5 components, CPLEX requires a little over 3 minutes, while the heuristic terminates after 25 s (LP) or 152 s (SDP). Considering chip area, the heuristic algorithm outperforms the MILP for similar optimization times: The upper layer is 34% (LP) or 56% (SDP) smaller and the lower layer is 14% (LP) or 28% (SDP) smaller than the result from the MILP model. Both MILP model and the heuristic algorithm with the LP use the same linearization and can be directly compared. The better optimization capabilities of the SDP are demonstrated. Regarding power and (component) performance, the results are equivalent since homogeneous 3D SoCs are used and no further assessment is needed here. Considering network performance, the result of the MILP is clearly superior: The components are strung together like a necklace, which directly represents the properties of the application graph. Thus, the average hop distance of packets is 2. In the result from the heuristic, the average hop distance is 2.5, i. e. a 25% worse result. Actually, the proposed heuristic algorithm can provide results with a hop distance as low as the MILP model. However, these are purely the result of chance, since the heuristic algorithm considers interlayer and intralayer communication separately. Therefore, solutions with larger average hop distances are Pareto-optimal for the heuristic algorithm. In summary, results obtained with the heuristic algorithm are superior in terms of area, since non-linearized, non-approximated optimization is possible. The results are inferior in terms of network traffic, since communication spanning multiple layers is not considered.

6.7.2.2 *Large homogeneous 3D SoC*

Two exemplary results for a homogeneous 3D SoC with 4 layers and 80 components are shown in Figure 49a using the LP and in Figure 49b using the SDP. The component layer assignment assigns 20 components to each layer, so one of the Pareto-optimal solutions is selected (each solution with 20 components per layer is Pareto-optimal). As expected, the number of TSV arrays is identical between all layers; it is 16 in the current solutions. The assignment of components and routers to rows and columns also returns the expected result with 5×4 -NoCs in all layers. One would expect that the positions of TSV arrays are similar between all layers. In this homogeneous 3D SoC, we set the length of RD in all layers to $5\sqrt{A}$. Therefore, a hop distance between 2 and 3 routers can be spanned by the RD to TSV arrays. Since the component communication graph is fully connected and all edge weights are identical, every connection scheme within this distance is Pareto-optimal. This is illustrated in Figure 51 for four routers in two layers and XYZ routing. The distance for the RD is large enough that TSV arrays can potentially be crossed. Both shown configurations have the same link utilization. Therefore, the solutions

	CONVENTIONAL	PROPOSED	DIFFERENCE
AREA	3712 A	2676 A	-28%
WHITESPACE	663 A	110 A	-84%
COMMUNICATION	1890 Mb \sqrt{A}/s	2810 Mb \sqrt{A}/s	+49%
MAXIMUM LINK LOAD	100 Mb/s	160 Mb/s	+60%

Table 6: Comparison of traditional and proposed NoC planning for a heterogeneous 3D SoC, with one mixed-signal and one digital layer and with 18 components, 9 ADCs and 9 processors. Parameters: initial temperature 30, cooling 0.98 and 500 iterations. The aspect ratio of bounding boxes is set to a maximum of $\eta = 0.1$.

are Pareto-optimal and crossed TSVs can be found in solutions, as well. From a practical point of view, these crossings can be deleted in post-processing or used to merge two individual TSV arrays into a single one, placed in the middle, with less area overhead for KOZs. In an optimal solution, the area of all layers is identical. Since solely downward TSV connections have KOZs, the bottom layer is smaller. (The KOZs are not accounted for in the component-to-layer assignment.) The area of individual layers is given in Table 5. As expected, the lower layer is approximately 18% smaller than the average of the other layers for both LP and SDP of the current results. To summarize, the heuristic algorithm is validated for a homogeneous 3D SoC, in which optimal results are known.

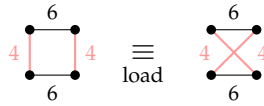


Figure 51: Two identical solutions in terms of link utilization, for fully-connected component communication graph and sufficient RD length for TSV arrays. Edge weight is the link load.

6.7.3 3D VSoC case study

The comparison for the heterogeneous yields the expected results as shown in Table 6: We are able to achieve an area reduction of 28% by more efficient packaging and 83% reductions in white space. The heuristic algorithm does not consider inter-layer communication. Thus, both communication and maximum link load are both worse by 49% and 60%, respectively. Please note, that this does not necessarily limit the system performance as long as the maximum link load is below the throughput capabilities of the corresponding link.

6.7.4 *Performance of the heuristic algorithm*

The individual execution times per part of the heuristic algorithm are shown in Table 4. The results show that the MILP for layer-to-component assignment is efficient and participates only at a fraction of the overall computation time. The layer floor planning requires $3.1\times$ more time for a doubled input size using the LP. This is not a result of a larger input size to the LP for area minimization, since the legalization in the last step does not suffer from a larger input size and the optimization is identical. This demonstrates that the worst case approximation of $O(n^4)$ vastly overestimates the execution time. The increase of computation time is rather a result of calculating the cost function. This matches the theoretical findings, since the complexity scales quadratically with the number of components. Using the SDP is more efficient for larger input sets in comparison to the LP; for small sets, the SDP has a high overhead. Iterating all possible solutions for the TSV count is a reasonable choice, since the increase of computation time was not measurable for the given precision. The placement of TSV arrays increases its computation time by a factor of 4 for a double-sized input. This also matches the theoretical findings, since the cost calculation scales quadratically. The legalization scales very well, as already discussed.

6.7.5 *Lost optimization potential*

In general, it is impossible to precisely determine the lost optimization potential using the proposed heuristic algorithm, since it is impossible to calculate an exact solution efficiently. Nonetheless, our results clearly highlight the potential, which is lost by the approximation of the heuristic. During component-to-layer assignment, only the component area is considered, since the count of TSV arrays is still unknown. This clearly influences the area of the layer: For the input set with 80 components, the bottom layer is 16% smaller than the average of the other layers. During the determination of the TSV count and the component floor planning, interlayer and intralayer communication is considered separately. As demonstrated in the example with 5 components, the result of the MILP model has up to 25% lower average hop distance for packets. The quality of results of the heuristic algorithm varies a lot and the results are rather often stuck in local minima. Further fine-tuning of algorithmic parameters is required.

One can either use SDP or LP to optimize the area of layers. Using the LP provides comparability to the MILP model but is expected to have worse results, since the LP is linearized with an approxima-

tion error while the SDP does not rely on an approximation.⁶ One exemplary result for a homogeneous 3D SoC with 80 components is shown in Figure 49a using the linearized LP and in Figure 49b using the SDP. The average hop distance of packets for both solutions is similar (4.71 and 4.78). The area of both chips should be similar, as well, with a linearization error of approximately 12% [147]. We can validate this for our input examples, as shown in the area comparison of results from LP and SD in Table 5; on average, the SDP performs 20.3% better for our benchmarks. In terms of computational speed, the LP offers up to $6\times$ higher performance than the SDP as shown in Table 4. Only during component floor planning there is an extensive use of the LP or SDP. For 80 components, the computation time for this part of the heuristic algorithm is approximately 40% higher, which also has a large effect on the complete execution time. In summary, the LP has higher speed but the SDP produces better results, as expected. The linearization error of the LP results in white spaces of approximately 20% more than in solutions from the SDP for our benchmarks. Competing approaches in the field of placement such as [138] report between 5% and 20% white space, depending on the benchmarks.

6.7.6 *Combining MILP and heuristic*

In general, it is possible to use the result of the heuristic as an initial solution for the MILP model. This potentially increases optimization time and further increases the quality of results. To demonstrate, we apply this principle to the small SoC with 5 components. We compare results using solely the MILP, running 10 minutes, and using the heuristic for an initial solution and then applying the MILP, in which the heuristic algorithm requires 24s and the MILP executes further 9.5 minutes. The MILP produces a gap of 25%, while the dual approach reduces the gap to 18.54% in the same computation time. The initial solution improves the optimization performance. This approach does not help with the larger input sets. The MILP is not able to improve the solution quality even after one week.

6.7.7 *Post-processing*

The provided solution requires further post-processing. For actual placement of gates in the provided bounding boxes, standard approaches exist. To place individual TSVs in the bounding boxes, methods as introduced in the related work can be used; thus, this issue is solved, as well. The placement of routers in the bounding boxes can

⁶ The SDP is optimal and does not yield an error, for instance from linearization. However, it will yield solutions with white space if the configuration of components is inefficient and does not allow for tight packaging.

also be optimized. For instance, it is useful to place routers at the edges of the network in the inner sides of the bounding boxes to reduce the lengths of links. The complexity of this optimization problem is rather low and can either be done manually or with standard methods by pin-placing the blocks [134]. Please note, that we are not discussing post-processing as part of this work, since solutions exist for these problems. Finally, the discussed Pareto-optimal solutions should be eliminated, which is possible using a trivial algorithm.

6.8 CONCLUSION

We describe the system-level optimization of NoC targeting heterogeneous 3D SoCs for the first time. The problem is also solved by means of modeling and a heuristic using a decomposition of the problem in individual, solvable parts: The MILP model provides a precise solution based on a well-reasoned technology model. Since solutions can not be found efficiently with this approach, we further provide an efficient heuristic. With this heuristic, it is possible to find an optimized solution for large input sets of 80 components in approximately 35 minutes. We compare different optimization methods within the heuristic and are able to remove 20% white spaces from the design. We have proven the validity of our results with input sets, for which properties of optimal solution are well-known and can highlight limitations of the heuristic algorithm, such as the estimation of inter-layer communication, as well as discuss Pareto-optimal solutions. The decomposition allows for using existing approaches to solve these. We find that our solution is better than state-of-the-art in terms of area, because we use an SDP with more degrees of freedom in placing and forming components. Please note, that better solutions in terms of communication exist, which can be used instead of the proposed approach. For TSV placement, we achieve up to 15% better communication. Furthermore, our approach converges faster. We also show that algorithms are required for further post-processing of the solution.

As highlighted multiple times throughout this thesis, the design space of 3D NoCs for heterogeneous 3D SoCs consists of system-level decisions and (micro-)architectural features (cf. Section 5.6). In the previous chapter, we introduced methods to optimize system-level decisions. We chose an analytical approach, for which a mathematical model is defined and optimized. We will focus on the optimization of individual features for the remainder of this thesis: We will propose optimizations of buffer size and buffer distribution in routers in Chapter 8 and we will contribute optimized co-designs of routing algorithms and router architectures in Chapter 9. The design space exploration for individual features is best done using simulations (and subsequent prototyping). This is due to the fact that fewer options are reasonable, and thus less effort is required to assess possible designs. Further, simulations are able to model all dynamic effects of the systems under test. In this chapter, we contribute the whole set of tools and methods for simulation-driven design space exploration of A-3D NoCs.

There are two popular NoC simulators, namely Noxim [122] and Booksim 2.0 [121] (cf. Section 4.9.4). Both software are targeting conventional NoCs, which are not implemented in a technology-heterogeneous setting. This, however, requires a novel approach, due to the following special challenges from heterogeneity:

- **Consideration of technology constants:** It is not sufficient only to account for architectural characteristics. Rather, the effects of disparate technologies require consideration of technology constants in simulations. Examples are discussed in Section 5.4 (1., 2., 4. and 5.). Regarding simulations, important ones are: Clock speeds of routers differ within a single chip, thus the simulation must support non-purely synchronous communication. The area footprint of routers and components varies, thus the simulation must cope with irregular network topologies.
- **Benchmarks and traffic generation:** Benchmark traffic patterns for heterogeneous 3D SoCs have more complex properties, since these chips typically combine sensing and processing, as introduced in Section 5.4 (6. and 7.). Therefore, the simulation must support application models with more sophisticated capabilities than related tools.
- **Analysis of interwoven design metrics:** There is mutual influence between different evaluation metrics, as introduced in Section 5.6 (Definition 5.5). Therefore, several metrics must be ac-

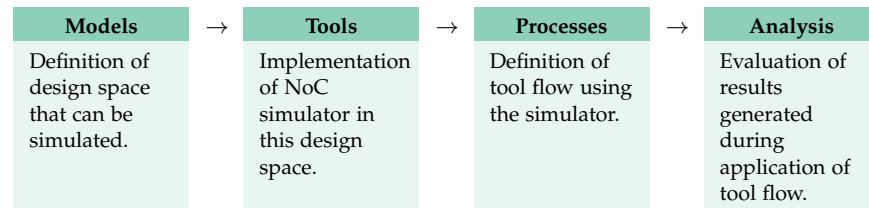


Figure 52: Models, tools, processes and analysis for design space exploration using simulations.

counted for simultaneously during evaluation of designs. Simple analysis of traditional simulation results is not sufficient. Considering buffer depths (Chapter 8) as an example: In general, reduced buffer depths increase the network latency especially for high loads. However, reduced buffer depths will vastly decrease the chip's costs, especially if the memory area is very expensive in certain chip layers. Therefore, area must be considered as an evaluation metric, as well, which is (naturally) not the result of a simulation. Therefore, we advocate methods, which embed simulations into a process considering results of prototyping, as well.

To tackle the aforementioned challenges, we do not only introduce a novel NoC simulator; Much more, we propose *models*, *tools*, *processes* and *analysis*, which cover all aspects of tools and processes for design space exploration as shown in Figure 52. We structure this chapter as follows:

- **Models:** First, we define simulation models in Section 7.1. The model serves as a basis for the implementation of tools, processes and analysis. Therefore, a precise model definition clearly shows properties and limitations of the remainder. Therefore, the influence of technological (i. e. in terms of routers: architectural) constants, as well as application constants, must be covered, and the application must be modeled.
- **Tools:** Second, we describe the implementation of the simulation models in a toolbox in Section 7.2. The tools also comprise the NoC simulator, which is configurable for design space exploration.
- **Processes:** Third, we explain the tool flow using the tools in Section 7.3. This facilitates design space exploration.
- **Analysis:** Fourth, the results of the simulations are evaluated using the tool flow, as explained in Section 7.4.

The four elements together yield integrated tools and methods for design space exploration. We conclude this chapter with the presentation of results (Section 7.5), which focus on properties of simulations using the proposed design space exploration. The whole chapter is based on peer-reviewed articles [JM 2, JM 5, JM 6, JM 10, JM 12].

7.1 MODELS

To fulfill the special characteristics of A-3D NoCs, novel simulation models are defined. As published in [JM 5], the models must be ...

- ... *accurate* to describe all system parameters.
- ... *precise* to measure each design metric.
- ... *adjustable* to cover every possible use case.
- ... *versatile* to adapt to many possible architectures.

As proposed in [JM 5], this is achieved by the following approach:

- Decomposition of the model into functional *components* that represent different parts of the system. This allows identifying parameters influencing and precisely determine their location of effect.
- Modeling components on the *most abstract level* as possible. This has a positive influence on the simulation speed. Furthermore, well-defined abstraction isolates important design parameters of the component.
- For different degrees of accuracy, multiple *levels of abstraction* must be covered by the model.
- Definition of *interfaces* to encapsulate the interaction between components. These interfaces include the relevant data, which are passed between components. They may include more data than required for an actual hardware implementation to enable efficient modeling.
- Definition of *parameter sets*, which encapsulate parameters influencing, as well as providing sufficient flexibility per component. This allows for rapid prototyping of different technologies, architectures, and design features.

The simulation model for A-3D NoCs comprises two parts, namely, application model and hardware model, as shown in Figure 53. The application model generates network traffic. It is modeled on transaction level. The hardware model consists of processing elements, network interfaces, routers, and links, i. e. all parts of a NoC, which are all modeled on cycle-accurate abstraction level. The models allows measuring network performance and energy consumption. Parameter sets allow setting properties of the hardware models. Their definition is delicate, since it sets boundaries to the expressiveness of the model: All relevant use cases and possible architectures must be accounted for. The interfaces, which connect components in the model, are shown in Figure 53, as well: Data are transmitted between tasks in the application model. These application data are converted to packets for injection into the network in the hardware model. Packets split up into flits, since routers implement flit based transmission. We provide a detailed definition of interfaces in Section 7.1.3.

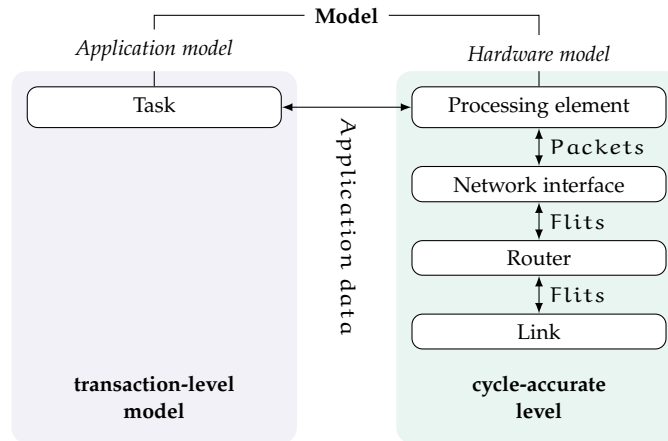


Figure 53: Simulation model for A-3D NoCs.

7.1.1 Application model

The application model must cover all possible use cases of heterogeneous 3D SoCs, which typically combine sensing and processing in a single chip. Further, it must be both abstract and accurate. There are three unique properties: First, the model includes timing of data processing. Processing times vary significantly depending on technology, data and component type. As an instance for the influence of component type, analog digital conversion has different properties than complex (digital) data processing. These can be found in heterogeneous 3D SoCs because these can efficiently combine sensing and processing. Second, there are dynamic effects of varying input data from sensors. For the sake of abstraction, the expected behavior for sets of inputs is modeled, rather than the actual control and data flows per input. Thus, statistical properties of control and data flow are relevant. Third, the dynamic energy consumption of links and routers, which depends on the properties of transmitted data, is calculated. Again, we use statistical modeling: Data are classified by similar statistical properties and annotated with these properties instead of modeling each data individually.

In general, Petri nets are one possible model of applications, with well-known properties. In these nets, data are represented as *tokens* and are passed between *places* via *transitions*. Conventional Petri nets are not sufficient here, since the three aforementioned properties are not covered. Therefore, we extend the model as follows:

1. **Timing:** Places have a delay, which models calculation times within the application. Places must be invalid for a time period after token consumption. We use the concept of *retention time on places* [153].
2. **Statistical communication selection:** Tokens are transmitted via enabled transitions, of which a random one is selected. This models applications with non-deterministic behavior originat-

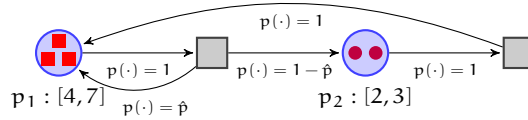


Figure 54: Colored stochastic Petri net with retention time for places [JM 5]. Please note that not only the color but also the form of the tokens is changed for the sake of accessibility.

ing from input data, e.g. sensing different environments influences the control flow. We use *statistical Petri nets* [154].

3. **Annotation with data:** Tokens must be annotated with data classes to compute the energy consumption of links and routers. The energy consumption depends on the statistical properties of data and the used coding method (since these influence the data properties, such as the bit self-switching activity [155]). Here, we colorize data transmitted in the network using *colored Petri nets*. Each color represents one data type; we denote colors by $\sigma \in C$.

For the sake of brevity, we kindly refer to Section b.1 for a detailed definition of the model, which was originally published in [JM 5]. A small example of such a Petri net is shown in Figure 54, which gives an illustrative overview of the model. There are two places in the model, which are depicted as blue circles. The retention time is given for both places; place p_1 has a retention time between 4 and 7 s and place p_2 between 2 and 3 s. Both places hold a different amount of tokens. The tokens are of varying type i. e. color, which is illustrated by red rectangles and purple circles. Please note that not only the color but also the form of the tokens is changed in the figure for the sake of accessibility. Transitions are shown as gray rectangles, which is a usual depiction. The probability to transmit tokens from place p_1 back to place p_1 is \hat{p} .¹ The probability to transmit tokens from place p_1 back to place p_2 is $1 - \hat{p}$. Tokens from place p_2 are always transmitted to place p_1 (i. e. $p(\cdot) = 1$).

The applicability of the model is demonstrated in Section 7.5.6, in which it is used to model a typical VSoC example.

7.1.2 Hardware model and parameter sets

The hardware model is given by all basic building blocks, the components, namely *PEs*, *NIs*, *routers* and *links*, as shown in Figure 53. Their role within the model, their functionality and their parameter sets are introduced here. The latter define the architectural properties and enable rapid prototyping.

¹ Transmission from a task to itself does not make sense for actual applications, but can be modeled.

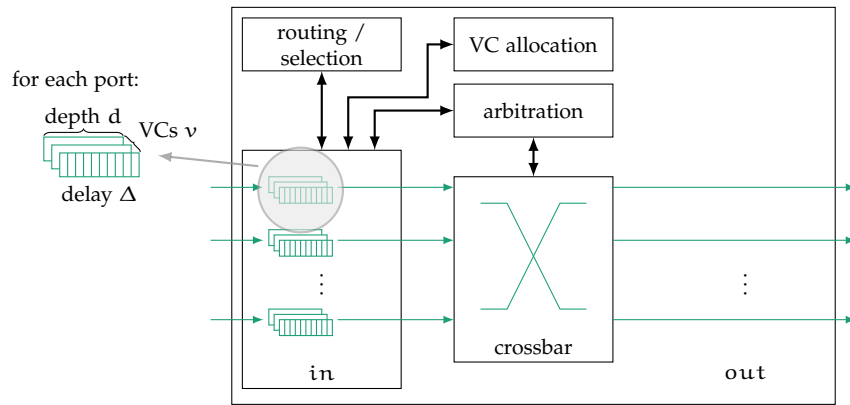


Figure 55: Router model with data path, control path and parameter set.

Processing Elements (PEs)

DEFINITION: The PE model is an abstract representation of any type of hardware with focus on traffic generation. PEs typically represent sensors, actors, processors, memory and hardware accelerators.

FUNCTION: PEs inject traffic into the network, as application tasks are mapped onto PEs. The model comprises a static mapping of tasks to PEs; dynamic remapping is not modeled, since dynamic properties can be defined using the stochastic property of the application model.

PARAMETER SETS: The number of PEs o is adjusted by defining their set $[o]$. The mapping function $\mathcal{M} : \mathcal{P} \rightarrow [o]$ maps places in the application to PEs. Places are in the set \mathcal{P} , as introduced in Section b.1.

Network Interface (NI)

DEFINITION: The NI serializes packet streams into packets, consisting of flits, and vice versa. There is one NI per PE.

FUNCTION: NIs are modeled as message queues (with infinite depth). Further, NIs realize the TLM interconnect between application model and hardware model in the simulation.

PARAMETER SETS: The NI divides the payload size of a packet into flits using the link bit width $b \in \mathbb{N}$. It adds a header flit of bit size $h \in \mathbb{N}$ and the last flit is marked as tail flit (it contains valid payload data). The NI is modeled on cycle-accurate abstraction level. As introduced in Section 5.4, A-3D NoCs are non-purely synchronous: Different layers of the 3D SoC are in disparate technologies. Thus, components have different clock speeds. Therefore, we model the NIs as GALS using the parameter $\text{clk}_{\text{NI}} : [\ell] \rightarrow \mathbb{R}_{>0}$ for ℓ layers.

Router

DEFINITION: Routers send packets through the network following a routing function. Here, we model input-buffered routers, with a architecture shown in Figure 55. Based on the flits in the input buffers,

routes are calculated, a route is selected and a path and VC is allocated.

FUNCTION: Routers are modeled cycle accurate with a state machine per input channel. The states are given by idle, routing calculation (RC), virtual channel calculation (VC), switch arbitration (SA) and switch traversal (ST): $\text{state} = \{\text{idle}, \text{RC}, \text{VC}, \text{SA}, \text{ST}\}$, which are assigned to each input port via $[m] \times \text{in} \rightarrow \text{state}$, in which m is the number of routers, in the set of input ports (see below). A translation relation is defined: $\delta \subset \text{state} \times \text{Flits} \times \text{state}$, which models the architectural behavior of the router. Function reports power-relevant events to the reporting, which can be counted and evaluated in post-processing.

PARAMETER SETS: The parameter set of routers is rather large, since it covers all possible architectures for heterogeneous 3D SoCs. The number of routers is adjusted via m , with $[m]$ their set. The network topology is defined as a network graph, in which vertices represent routers and edges represent links. The set of all possible directed edges is given by: $E_N = \{(k, l) \mid k, l \in [m]\}$. The set of network graphs is given by $N = ([m], E_N)$ (of course, not every router pair can be connected in a real system due to production constraints.). The graph is spatially embedded with positions of routers: $p : [m] \rightarrow \mathbb{R}_{>0}^2 \times [\ell]$. As shown in Figure 55, input and output ports of routers are given by $\text{in} : [m] \rightarrow \mathcal{P}(E_N)$ and $\text{out} : [m] \rightarrow \mathcal{P}(E_N)$, in which $\mathcal{P}(E_N)$ represents the power set of E_N .² All links connected to the input port of the router r are given by $\text{in}(r) = E_{N,r} = \{(k, l) \mid l = r, (k, l) \in E_N\} \subset E_N$. The number of input ports of this router is given by: $|\text{in}(r)|$. The definition of output port is analog.

Based on the same consideration as for the timing of NIs, routers are modeled cycle-accurate using GALS set by $\text{clk}_{\text{router}, \text{GALS}} : [\ell] \rightarrow \mathbb{R}_{>0}$.

The number of VCs can be set per router $v : [m] \rightarrow \mathbb{N}$ and the buffer depth can be set per port $d : [m] \times \text{in} \rightarrow \mathbb{N}$. Both are shown in Figure 55. This allows modeling asymmetric router buffer depths [JM 3].

Routing function, selection function and allocation can be set via $\text{routing} : [m] \times \text{Flits} \rightarrow \text{out}^3$ and $\text{selection} : \text{Flit} \times \text{state} \times \text{state} \rightarrow v$, depending on the state of the corresponding downstream router.

The timing behavior of each input port of a router can be set by its delay to evaluate the head flit in clock cycles: $\Delta : [m] \times \text{in} \rightarrow \mathbb{R}$. To exemplify: For a router with three stages until the head flit can be sent, Δ is set to three. Please note, that the head flit will leave the router without congestion after four clock cycles, since sending requires one additional cycle. Since this is required for all flits, it is not part of

² The power set of a set A denotes the set of all subsets of A .

³ This is valid for deterministic routing algorithms. Adaptive routing algorithms require the state of adjacent routers.

the parameter set. The parameter set enables versatile modeling of different router architectures.

Links

DEFINITION: Links send data between routers.

FUNCTION: Links calculate transmission matrices. These are used to evaluate their dynamic energy consumption, because this depends on different factors such as the transferred data [155] or the VCs [JM 2, JM 14].

PARAMETER SETS: The links do not require parameter sets. Their properties are already modeled by means of link widths (in the NIs) and their energy consumption is calculated using data flow matrices in post-simulation processing (cf. Section 7.2.3.1).

7.1.3 *Simulator interfaces*

Interfaces are the means of communication between components of the model. The interfaces, as published in [JM 5], are defined as follows:

Definition 7.1 (Application data). Application data are used for communication between application and hardware model, i. e. between places P in the application model and PEs. The amount of data is represented by the number of transmitted tokens. Statistical properties of the data are annotated using colors C . The amount of data per stream is given by u . This yields:

$$\text{ApplicationData} = (P \times P, C, u) \quad (57)$$

Definition 7.2 (Packets). Within the network, packets are transmitted between PEs. These contain source and destination addresses of the PEs ($[o] \times [o]$). This yields:

$$\text{Packets} = ([o] \times [o], P \times P, C) \quad (58)$$

Please note, that it is required to include the source and destination within the application ($P \times P$), since the inverse mapping function between task addresses and hardware addresses \mathcal{M}^{-1} is not always well-defined, as multiple tasks can be executed on a single PE. Packets always have the same size in this model; their number is calculated from u per stream in the NIs.

Definition 7.3 (Flits). Within the network, flits are transmitted among routers. The type of the flit is denoted by the set $\text{Type} = \{\text{head}, \text{body}, \text{tail}\}$. This yields:

$$\text{Flits} = ([o] \times [o], P \times P, \text{Type}, C) \quad (59)$$

Please note the following technical detail: additional information are transmitted between parts of the model in the implementation.

We transmit the application data generation time Γ between application and hardware model. The packet generation time, i. e. the time the head flit is generated, Γ_p is transmitted on packet level and the flit generation time Γ_f for each flit. The flit and packet generation time differ for body and tail flits, which allows measuring flit latency. Strictly speaking, these are not part of the interfaces, which model the actual hardware implementation. However, this information is required for analysis of the simulation, for instance to calculate network, packet and flit delay.

7.2 TOOLS

The tools implement the aforementioned model. Due to the special characteristics of A-3D NoC's design space, they further fulfill these requirements:

1. They cover technological and architectural parameters.
2. They comprise a simulator that is parameterized to modify parameters of the model, and configurable to modify parameters of the simulator. All these parameters are set during initialization to reduce unnecessary compile times and meta programming.
3. They are implemented at multiple levels of abstraction, i. e. the application is modeled with transactions and the hardware is modeled cycle-accurate.
4. They provide methods for an integrated tool flow, as proposed later in Section 7.3. Further, it is possible to generate evaluation results for different hardware models without rerunning simulations for certain evaluation metrics.

The proposed model for A-3D NoCs is implemented in Modern C++ using SystemC 2.3.1a class library [156]. It is open source and available on github at github.com/jmjjos/ratatoskr. The structure of the tools, implementing the models from the last section, is shown in Figure 56. There are three parts:

The *benchmark tool* sets up system benchmarks. It implements the application model, which is on transaction level. Further, it provides synthetic traffic patterns and multithreaded benchmark programs. Please note, that the latter two options have limited applicability for A-3D NoC, yet are implemented to provide comparability to other tools.

The *NoC simulator* implements the hardware model and is on cycle-accurate abstraction level. The PEs implement a technology-aware TLM interface to convert the different abstraction levels of hardware and application model.

The *reporting tool* is crucial for the process of evaluation during design space exploration. It is on the highest abstraction level. It generates (static) reports including network and application statistics,

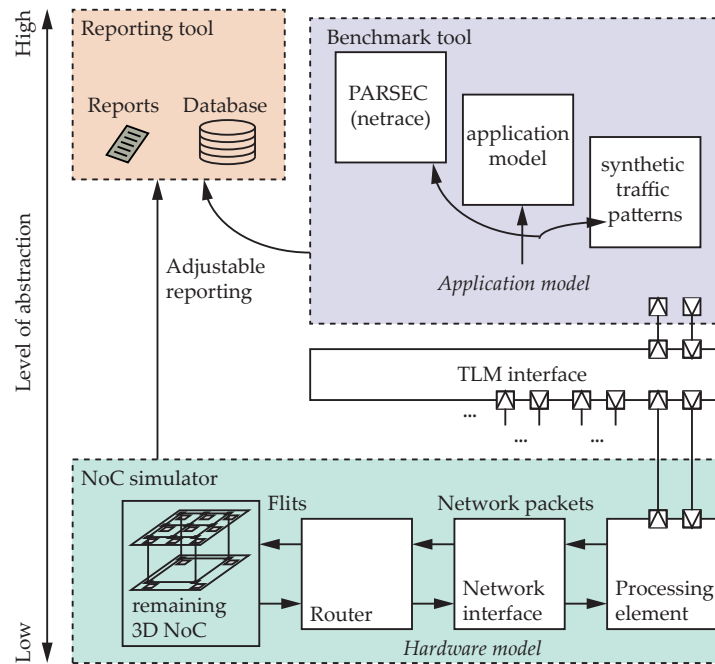


Figure 56: Architecture of simulation and evaluation tools. [JM 6]

which is similar to the two competitors. In addition, it is possible to evaluate different configurations of parameters without rerunning the simulation with the reporting tool. This is achieved by means of innovative data aggregation, see the data flow matrices in Section 7.2.3.1, and by means of a database, which stores events and allows for post-simulation data manipulation, see Section 7.2.3.2. Both features are not found in current simulators.

7.2.1 Implementation of application model – the benchmarking tool

The benchmarking tool provides network traffic and implements three options, as shown in Figure 56: An implementation of the application model, synthetic traffic patterns and PARSEC multithreaded benchmarks. Here, we explain the implementation and introduce the configuration of the application model using XML files.

The application model implements colored, stochastic Petri nets with retention time on places, as introduced in Section 7.1.1. Places are implemented in the class `Task`, while the class `TaskPool` stores the entirety of places. The application configuration is done via XML files. Each task is configured as shown in Listing 1. In and out going data connections are configured with the `requires` and `generates` fields. Application data are generated after all required data, from the `requires` field, are available. To model the stochastic property, data can be sent to different destinations, which are grouped into a possibility each that is selected following a given probability. The timing of tasks and the retention time on places is modeled using the

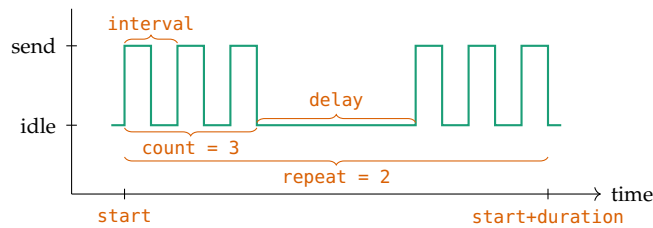


Figure 57: Timing of tasks with all parameters, which can be set using XML files.

fields `start`, `duration`, `repeat`, `interval` and `delay`. The behavior is exemplified in Figure 57: A task is only executed in between the start time and until its duration is finished. A task will also stop execution, if it was repeated for as many times as given by the field `repeat`. For each repetition, one data send possibility is taken. If tokens are available, a task sends data to this destination. It will send data `count` times. There is a delay in-between sending data, in which the task is idle; this implements the retention time. The colored Petri net is realized by defining data types: A set of data types can be defined as shown in Listing 2. Data in the network are identified by type allowing for precise energy estimation [JM 2].

The implementation of synthetic traffic patterns relies on the same implementation of tasks. The class `SyntheticPool` provides hotspot, uniform random, transpose and tornado traffic patterns and instantiates `Task`-objects automatically following the distribution. Multiple phases can be defined, for instance to generate periods with varying injection rates. The definition of the synthetic traffic patterns is done as exemplified in Listing 3 for uniform random traffic pattern. The `start` and `duration` fields set the length of the phase. Alternatively, a `count` can be defined to stop the execution after a given number of packets. The injection rate is set, as well. The spacial distribution of the patterns is as usual for 2D NoCs and it is adopted to reassemble the characteristic for A-3D NoCs (cf. Figure 24). Please note that all synthetic traffic patterns are realized by using the application model, which shows its expressiveness. Only uniform random traffic cannot be modeled, since the times between injecting traffic must be non-deterministic but within certain intervals to model correct injection rates, without biased results from bursts. Therefore, uniform random traffic extends the functionality of the task model and has a partly different implementation.

Finally, PARSEC benchmarks are implemented using the `netrace` library, in which traces of benchmark simulation for a 64 core processor with dependency tracking are sorted. Please note, that this option has limited applicability for heterogeneous 3D SoCs, since it targets a homogeneous multi-core processor. It will be useful, if traditional 2D NoCs are modeled using the proposed tool.

Listing 1: Definition of a task.

```

<task id = "1">
  <start min = "0" max = "0"/>
  <duration min = "100" max = "100"/>
  <repeat min = "2" max = "2"/>
  <requires>
    <requirement id = "0">
      <type value = "1"/>
      <source value = "0"/>
      <count min = "1" max = "1"/>
    </requirement>
  </requires>
  <generates>
    <possibility id = "0">
      <probability value = "1"/>
      <destinations>
        <destination id = "0">
          <delay min = "0" max = "50"/>
          <interval min = "10" max = "10"/>
          <count min = "3" max = "3"/>
          <type value = "1"/>
          <task value = "3"/>
        </destination>
      </destinations>
    </possibility>
  </generates>
</task>

```

Listing 2: Definition of data types.

```

<data>
  <dataTypes>
    <dataType id = "0">
      <name value = "image"/>
    </dataType>
  </dataTypes>
</data>

```

Listing 3: Exemplary synthetic traffic.

```

<synthetic>
  <phase name = "warmup">
    <distribution value = "uniform"/>
    <start min = "1000" max = "1000"/>
    <duration min = "500" max = "500"/>
    <injectionRate value = "0.04"/>
    <count min = "-1" max = "-1"/>
    <hotspot value = "0"/>
  </phase>
</synthetic>

```

7.2.2 Implementation of hardware model – the NoC simulator

As already introduced, the NoC simulator implements the hardware model. Here, we explain details of the implementation. Further, we describe the configuration of the simulator, which is parsed from XML files during initialization.

Top module

The class `LayerTop` is the top module. It contains all instances of routers, NIs, PEs and links. XML configuration files are read during initialization. Based hereupon, `LayerTop` instantiates classes of the hardware model, which results in a complete NoC. The available hardware is given by defining a set of nodes (in XML). Nodes either represent PEs or routers here. In general, the strength of the concept of nodes is that the model can easily be extended to include other interconnect architectures, such as direct links or buses, simply by defining and implementing new types of nodes; thus, hybrid designs can potentially be modeled in future work.

An exemplary definition of two nodes, namely a router and a PE, is shown in Listing 4. The other fields for further configuration are discussed later. The network topology is read from an XML file, as well. Next, a set of nodes is defined by assigning a node type and a position to each node. This is shown in Listing 5: The node with id 0 is at position (0,0,0). The network topology is defined by a set of links, which connect nodes. Connections are bidirectional. This is exemplified in Listing 6, in which a connection is defined in the `con`-field, which connects two nodes via ports. For the sake of brevity, only one of two ports is shown. These configuration options are discussed in detail later, as well.

Processing Elements (PEs)

The class `ProcessingElementTasks` implements the PE model, which is able to connect to the given benchmark tool. Please note, that the implementation provides an abstract class `ProcessingElement`, which would allow to extend the simulator by other PE models, if different application had been implemented.

There are two important functions: First, `void receive()` receives packets from the network-side, i.e. the corresponding NI, and executes the TL-modeled receive function in the application model. Second, `void execute(task*)` manages the application model's tasks, which are mapped to the PE. This function keeps track of the task's execution status by triggering the events for timing and updating relevant member variables of the tasks. Thereby, PEs provide the TLM interconnect between application model and hardware model. The function of the TLM interconnect is shown in Figure 58: Tasks *i* and *j*

Listing 4: Node types.

```

<nodeTypes>
  <nodeType id = "0">
    <nodeModel value = "RouterVC"/>
    <routing value = "DPR"/>
    <selection value = "EDXYZ"/>
    <clockDelay value = "1"/>
  </nodeType>
  <nodeType id = "1">
    <nodeModel value = "ProcessingElement"/>
    <clockDelay value = "2"/>
  </nodeType>
  <nodeType ... </nodeType>
</nodeTypes>

```

Listing 5: Position of nodes.

```

<nodes>
  <node id = "0">
    <xPos value = "0"/>
    <yPos value = "0"/>
    <zPos value = "0"/>
    <nodeType value = "0"/>
  </node>
  <node ... </node>
</nodes>

```

Listing 6: Connections.

```

<connections>
  <con id = "0">
    <width value = "32"/>
    <depth value = "1"/>
    <interface value = "0"/>
    <ports>
      <port id = "0">
        <node value = "0"/>
        <bufferDepth value = "16"/>
        <vcCount value = "4"/>
      </port>
      <port ... </port>
    </ports>
  </con>
  <con ... </con>
</connections>

```

Listing 7: Mapping.

```

<map>
  <bind>
    <task value = "0"/>
    <node value = "8"/>
  </bind>
</map>

```

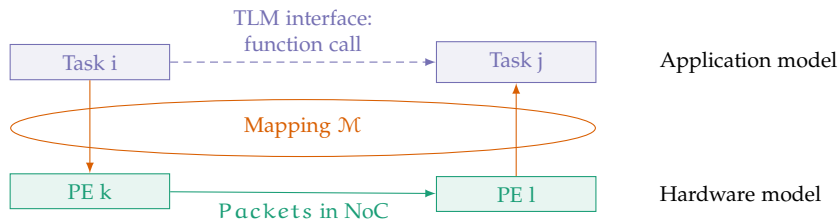


Figure 58: TLM interface between application and hardware model using mapping [JM 12].

communicate on transaction level in the form of a function call. This function call is redirected to the PE k , which sends the corresponding packets to PE l via the NoC. After reception, PE l calls the corresponding function in task j .

PEs have a mapping function \mathcal{M} , mapping tasks to PEs, as a parameter, as defined in the model. This can be set using a mapping file in XML. An example is shown in Listing 7, in which task number 0 is mapped to PE number 8. Please note that PEs are assigned a clock speed of the PE in Listing 4, which is passed to the corresponding NI (see below). Strictly speaking, this is not defined in the model, but are implemented as such for a less complicated description.

Network Interface (NI)

The class `NetworkInterfaceVC` implements the NI model with VCs based on the abstract class `NetworkInterface`. NIs convert application data into networks packets by knowledge of the bit width of links in the network, and vice versa. There is one function per direction of transmission, `void receivePacket()` and `void receiveFlit()`, which are executed for each clock cycle of the NI. The parameters of the NI are: The clock frequency, as defined in Listing 4, and the bit width of links in the network from Listing 6. Please note, that the bit width must be identical for each connection in the network.

Router

The class `RouterVC` implements the router model with VCs based on the abstract class `Router`. Routers are connected among each other and to PEs/NIs via a buffer model implementing a first-in-first-out (FiFo) memory. The buffer is asynchronous to model the GALS nature of heterogeneous 3D SoCs.

There are two important router functions: `void receive()` implements flits reception and `void thread()` flit sending. In the latter function, there are three sub-functions implementing the state machine of the router: A path is calculated for head flits in the function `void route()`, in which routing and selection function are in separate classes; further, a VC is allocated. The function `void arbitrate()` cal-

culates a matching between requests and resources; different arbitration schemes are implemented. Arbitrated flits are sent in the function `void send()`.

The parameters of the routers can be set using XML configuration. The basic parameters are set via the `nodes`-field as shown in Listing 4: Router model (with VCs), routing algorithm (e.g. DPR [100]), selection (e.g. EDXYZ [157] or round robin) and the clock speed (1 GHz) are set there. Defining multiple node types enables simulating different router architectures in a single model.

Please note, that the router is limited in capability by the model. Therefore, it is not possible to use other designs than input buffered routers, with architectures similar to [93], as is shown in this very work's Figure 55. For instance, routers, which accelerate semi-static data streams via pre-allocation of prioritized paths, require a different model as proposed with an implementation in [JM 8].

Links

The class `Link` implements the link model. It generates the data flow matrices (cf. Section 7.2.3.1) by connecting to the reporting tool and issuing data transitions clock-wise. The correct functionality of the link class is verified by output raw data. Links do not need to be further configured, since their energy model can be calculated in post-simulation processing (cf. Section 7.4.2).

7.2.3 Reporting Tool

7.2.3.1 Data flow matrices

Links are modeled in a more abstract way than calculating the energy consumption cycle-by-cycle: We propose the use of data flow matrices \mathcal{R} for each link defined as $\mathcal{R} : [m] \times \text{out} \rightarrow \mathbb{R}$, with matrices $\mathbb{R} : (\{\text{idle}\} \cup (\{\text{head}, \Sigma\} \times \{\text{idle}, \text{data}\}))^2 \rightarrow [0, 1]$. The data flow matrices yield the proportion, at which different data types subsequently pass through a link. The data types are determined by the colors Σ from the application model (cf. [JM 5]). Head flits are considered separately as their data are always uncorrelated. Tail flits are treated as body flits, since they only terminate the transmission and usually contain payload data, depending on the router hardware model. Both "idle" and active ("data") cycles of the link are saved with the last sent data type, read row-wise. Data flow matrices allow reconstructing many network statistics after simulation and calculating dynamic link energy and even assess the impact of different link-level codings from a single simulation; a short example is given in Section 7.4.2. Computational properties of the data flow matrices are discussed in Section 7.5.4 along with the accuracy of energy models that rely on these matrices [JM 2].

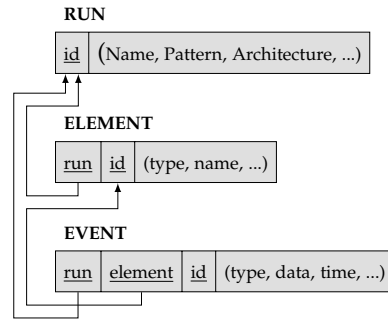


Figure 59: Database structure in the reporting tool [JM 10].

7.2.3.2 Event database

The report tool generates reports from simulation results. As usual, textual reports are generated for average network, packet and flit delay. Further, the data transmission matrices are stored in csv (comma separated value) file format for further data processing. As an innovative feature, the reporting tool connects a simulation database with all simulation events. The structure of the database is shown in Figure 59 with its three tables, *Run*, *Element* and *Event*. The *Run*-Table stores simulation run with an ID and attributes such as parameters, benchmark, and router architectures. The *Element*-table contains all components per simulation run, i. e. the hardware (routers, NIs, PEs). Each element has a unique identifier, also over multiple runs. A type and a name field associated to each element allow easy filtering. The actual simulation events are stored in the *Event*-table, with a time stamp, a type of event and a data field for debug text or references to other database entries. This structure of the database enables very flexible reporting. Even after the simulation is finished, detailed information about the simulation can be accessed. For instance, traces of individual packets can be generated from the *Event*-table by evaluating the packet's sending and receiving events. The class `Report` offers the API three important functions: A new simulation run is registered with the function `startRun()`. All components in this simulation register themselves via the function `registerElement()`. Events are reported via `reportEvent()` during simulations. The database is connected via network (TCP) and can be run on an external server to reduce the performance impact on simulations. Since the reporting tool uses a generic event-based approach, it can also be used for FPGA-based NoC prototyping such as [JM 19].

7.3 EXPLORATION PROCESS

The process of empirical design space exploration with simulations is depicted in Figure 60 as proposed in [JM 10]. Following the incremental approach from Section 5.6.2, the designer selects variable pa-

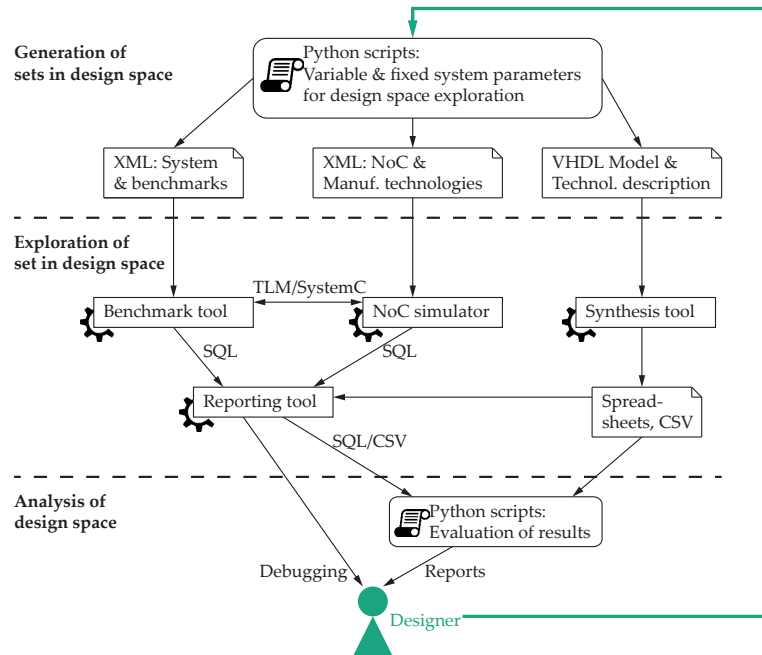


Figure 60: Design process using proposed tools [JM 10].

rameters and fixed system parameters; only the variable parameters are part of the exploration process. The parameter sets are passed to initialization scripts written in Python, as shown at the top of the figure. The scripts generate three sets of input data for the tools, as shown in first section of Figure 60. First, depicted on the left-hand side, there are the XML description files for the *benchmark tool*, with the application model. Second, in the middle part, the XML description for the *NoC simulator* is shown, with the hardware model. Third, as shown on the right-hand side, there are script files that select the correct VHDL models of the routers and assign parameters of the used manufacturing technologies.

After the initialization, simulation and syntheses are started, as shown in the second section of Figure 60. The NoC simulator is run with each parameter set to explore the design space. The traffic in the NoC is injected by the benchmark tool. The reporting tool collects the results of the simulations. Furthermore, the synthesis for standard cells is invoked. The output is stored in csv files. As a special feature of the reporting tool, it is possible to retrospectively modify results based on the synthesis, which for instance can be used to calculate the energy consumption.

After running all simulations and syntheses, python scripts generate complete reports for all variable parameters as shown in the lowermost section of Figure 60. If required, debugging information is contained either in the reporting tool or in traditional vcd-files (value change dump). The designer again interacts with the design process

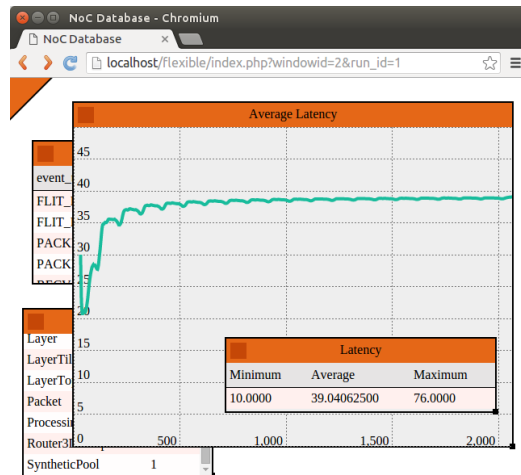


Figure 61: Screen shot of graphical reports. [JM 10].

by analyzing the results and setting the parameters for the next iteration in this particular incremental approach.

7.4 ANALYSIS

The analysis of the results utilizes the data in form of reports and events in the database. Furthermore, we highlight the usage of data flow matrices as those are unique and novel features.

7.4.1 Reports

During the design process, reports are generated for the designer to assess the properties of the units under test. In general, there are three types of reports:

First, the evaluation scripts generate *textual reports* about the design space exploration including the most relevant parameters of the simulation. Using the XML configuration files, the level of verbosity and thus the included results can be chosen by the designer.

Second, we provide a *flexible graphical user interface*, which presents the data from the database in the reporting tool. An HTML5 dashboard with interactive graphs can be accessed from any standard browser. It is shown in Figure 61: The average flit latency in a $2 \times 8 \times 8$ NoC for uniform random traffic is plotted, which converges already after 1,200 clock cycles. Also, average and maximum flit latency are printed. The advantage of the dashboard is its flexibility: All data are shown in frames, which can be moved, adjusted in size and whose contents can be freely set by SQL queries.

Third, *direct database access via SQL* is possible. This will be especially useful if the desired design metric is not already programmed or for debugging, e. g. by tracing objects in the network. The database

extends the functionality of vcd files, since these do not allow tracing of moving objects and must be programmed per component. Further, data can be modified in retrospection using the synthesis results. The disadvantage of the system is that even short simulations will generate many gigabytes of data if very detailed events are traced. Fortunately, this is only required for debugging and can be adjusted.

7.4.2 Data flow matrices

The data flow matrices \mathcal{R} allow to reconstruct the state of links during the simulation. In [JM 6] we demonstrated that the energy consumption of link depends on the coding of data and proposed models for both hop-to-hop coding and end-to-end coding. This model is extended in [JM 14] to calculate the dynamic energy consumption of links based on data flow matrices. Different hop-to-hop and end-to-end codings can be assessed without rerunning the simulation. We integrated these models into our simulation tool and assessed the influence of VCs on the energy consumption of NoCs in [JM 2].

We exemplify the form of the matrices with a small example as introduced in [JM 5]. We assume a network, in which two types of data σ_1 and σ_2 are transmitted. A basic router without VCs and a single pipeline stage is simulated. Both types of data pass the router and are transmitted via a single link. The average injection rate in the network is 0.75 flits/cycle. 25% of injected flits are of type σ_1 and 75% of injected flits are of type σ_2 . Packets have ten flits. The transmission starts after 100 clock cycles; after 10,000 clock cycles, the data flow matrix for the link is, as published in [JM 5]:

$$\mathbf{R} = \begin{matrix} & \begin{matrix} \text{idle} & (\text{head}, \text{data}) & (\text{head}, \text{idle}) & (\sigma_1, \text{data}) & (\sigma_1, \text{idle}) & (\sigma_2, \text{data}) & (\sigma_2, \text{idle}) \end{matrix} \\ \begin{matrix} \text{idle} \\ (\text{head}, \text{data}) \\ (\text{head}, \text{idle}) \\ (\sigma_1, \text{data}) \\ (\sigma_1, \text{idle}) \\ (\sigma_2, \text{data}) \\ (\sigma_2, \text{idle}) \end{matrix} & \begin{pmatrix} 0.010 & 0.000 & - & - & - & - & - \\ - & - & 0.019 & 0.042 & - & 0.014 & - \\ - & - & 0.007 & 0.013 & - & 0.005 & - \\ - & 0.041 & - & 0.326 & 0.126 & - & - \\ - & 0.013 & - & 0.113 & 0.039 & - & - \\ - & 0.014 & - & - & - & 0.114 & 0.045 \\ - & 0.006 & - & - & - & 0.040 & 0.014 \end{pmatrix} \end{matrix}$$

The matrix is read row-wise: Data σ_1 are succeeded by data of the same type in 32.6% of cases. Please note, that zero-element are marked by $-$, while 0.000 is rounded and non-zero. The matrix allows reconstructing many properties of the link: The idle state of the link for 100 clock cycles can be found in the first element. The active and idle times of the link can be calculated by summation of the rows. Also, the matrix shows the absence of VCs, since there the probability to switch from (σ_i, data) to (σ_j, data) is zero. These types of matrices can be used for energy estimation as explained in Section 7.5.4

7.5 RESULTS AND DISCUSSION

7.5.1 *Comparison with related work*

We compare the performance of the proposed implementation of the simulator with the two important competitors Booksim 2.0 and Noxim. All three simulators are compiled with gcc 7.3.0 with optimizations (-O3) and debug symbols (-g) using Linux Mint 19. The software is executed on a machine with an Intel i7-7740X clocked at a base frequency of 4.3 GHz and up to 4.5 GHz in turbo mode. We run 10 simulations per software; we model a 4×4 NoC, with 32 flit per packet, 4 VCs, 4 flit deep buffers, dimension-order routing and simulate for 100,000 clock cycles and 3% packet injection rate. The execution times are measured with the program time (Unix, using real), with suppressed terminal output. The results are shown in Table 7.

The comparison of the simulation speed shows, that Noxim has the fastest simulation speed with 28,000 clock cycles per second. Our simulator offers 79.3% worse performance with 5,800 clock cycles per second. However, it is 26% faster than Booksim 2.0, which offers 4,600 clock cycles per second simulation speed. Since neither Noxim nor Booksim implement a complex application model, we offer extended functionality with performance in between of the two most common simulators. Therefore, the proposed simulator offers good performance.

We compare the code complexity on basis of lines of code. Both Noxim and the proposed simulator use SystemC. Thus, these two are comparable. Although we offer additional functionality, the code size is smaller than Noxim. Please note, that BookSim 2 implements its own simulation engine and, therefore, is larger than the two competitors.

Finally, the proposed tools are the only ones providing a reporting tool with dataflow matrices support along side a report database. The advantages are, that energy estimation is possible for different codings and technologies without running simulations twice. In contrast, Booksim 2.0 does not generate power estimates and Noxim relies on a static power profile for each simulation run. Further, the proposed data flow matrices allow for energy models with exceptional performance (see Section 7.5.4).

7.5.2 *Performance of application model*

To evaluate the performance of the application model, a small exemplary application is executed with and without using a NoC for data transmission. A simulation without a NoC attached is a application simulation at transaction level. We use a VOPD for a MPEG-4 compatible decoder with data rates from [158]. We use a conventional 2D 4×4

SIMULATOR	EXECUTION TIME		LINES OF CODE
	AVERAGE	STD. DEV.	
BOOKSIM 2.0 [121] (27 JUN 2017, 28f4329)	21.9 s	0.11 s	25,000
NOXIM [122] (30 JUL 2018, df472c0)	3.61 s	0.04 s	8,600
PROPOSED HERE, VERSION 1.1.8, 30 AUG 2018, d7DBA08	17.32 s	0.25 s	7,700

Table 7: Comparison between NoC simulators; 10 simulations of 100,000 clock cycles, for a 4×4 NoC, with 32 flit per packet, 4 VCs, 4 flit deep buffers, dimension-order routing.

NoC. The mapping was done manually and minimizes traffic in the NoC with sequential tasks in adjacent PEs (cf. [JM 12, Figure 7]). In application-only mode, the simulation requires approximately 1/30 less CPU time than with a hardware model. This result is within the expected speed difference when using TL models [159]. This shows that the application model has a small and acceptable performance impact on the overall simulation. The simulation time is dominated by the context switches that cannot be avoided in cycle-accurate simulations.

7.5.3 Performance of the reporting tool

We assess the performance impact of the reporting tool’s database using a simulation of uniform random traffic in a $2 \times 8 \times 8$ NoC, as published in [JM 10]. After 2,200 simulated clock cycles, there are approximately 3 million events in the database including 500,000 flit send and receive events (*SEND_FLIT*, *RECV_FLIT*). The performance of the database is evaluated using a query to calculate the average flit latency. The query is published in [JM 10], Figure 6. On an Intel Xeon E3-1230, running Ubuntu Server 14.04.4 LTS and MySQL Server 5.6, the query takes 10 seconds. The time grows linearly with the number of assessed events. This is considerably longer than calculating a floating average during simulation and is the downside of the flexible reporting tool. Further, we compare the reporting tool against conventional vcd files. These are written to a storage device during simulation and can become rather large for long simulation runs. The advantage of the database is that an external server can be used to reduce the impact on performance. To summarize, only a subset of all events should be stored in the database. In that case, the reporting tool provides high flexibility and satisfying performance.

7.5.4 Accuracy, performance and complexity of data flow matrices \mathcal{R}

We assess the complexity of the data flow matrices. A single entry of the data flow matrix is updated per simulation time and per link. At the end of the simulation, all entries are divided by the number of

simulation times. Let n be the number of data stream types. Let l be the number of links in the network and t be the number of simulated clock cycles. Each data flow matrix has $2n^2 + 3$ entries. Then, the time complexity to estimate all data flow matrices is $O(n^2lt)$. The link matrices require $O(tn^2)$ memory. Storing the raw data of the links also requires $O(n^2lt)$ time, but $O(lt)$ memory is required. This is significantly more, since usual simulation times are larger by a few magnitudes than n and l . This impressively demonstrates the advantages of data flow matrices.

Energy modeling based on the data flow matrices is proposed in [JM 2]. Since the actual energy models are not part of this thesis, we kindly refer to the aforementioned publication for details. Nonetheless, we report the accuracy of the models. The matrices allow to cover the effects of VCs and coding for the dynamic energy consumption of links between routers, which is not possible with state-of-the-art [160]. We simulate a 3D VSoC, in which digitized image data from AD converters in a mixed signal layer store their data in a memory layer. We use daytime and nighttime 512×512 pixel image. We assign one color in the application model for each sensor and estimate the dynamic link energy consumption of links. We compare a novel high-level model [JM 2], a standard model (which does not model the effects of VCs) and bit-level simulations. Uncoded, gray-coded and correlated data are compared as presented in Tables 8 and 9. For the analyzed case study, the average flit and network latency is decreased by 52.2% and 45.5%, respectively, when using virtual channels. This performance gain results in a dramatic increase in link's power consumption of 74.9%. In terms of model accuracy, the proposed method, which relies on link matrices, predicts the energy consumption at a very low error rate below 1%. The previous model yields an error of almost 50%, for NoCs with VCs.

7.5.5 Productivity

The productivity of the proposed tools and processes is evaluated, as published in [JM 10]. The design of a 3D NoC with asymmetric buffer distributions and depths (cf. Chapter 9.6) is the exemplary use case. The proposed tools allow for an incremental approach, in which the level of detail is gradually increased. For a fast initial evaluation, small applications are modeled (video object plane decoder applications and MPEG-4, mp3, and h.256 converters from [106]). Although lacking generality, being application specific, these application models allow for a very fast evaluation of designs: simulations run only between 20 s and 60 s on an Intel Core i7-4770 CPU, with 16 GB RAM using CentOS 7.1. We thereby were able to exclude unnecessarily large buffer sizes at the beginning of the design space exploration and find a set of possible parameters. Next, we conducted detailed simulations

DATA	ENERGY PER TRANSMITTED PACKET [pJ]		
	BIT-LEVEL SIM.	PROPOSED MODEL [JM 2]	STANDARD MODEL [160]
UNCODED	4.18	4.15	2.40
GRAY	4.11 (-1.67 %)	4.09 (-1.44 %)	2.23 (-7.08 %)
CORR	2.69 (-35.64 %)	2.68 (-35.42 %)	2.71 (+12.92 %)
AVG. FLIT LATENCY: 19.2 ns		AVG. NETWORK LATENCY: 105.4 ns	

Table 8: Link energy quantities and network performance with 4 VCs for uncoded, gray-coded and correlated data [JM 2].

DATA	ENERGY PER TRANSMITTED PACKET [pJ]		
	BIT-LEVEL SIM.	PROPOSED MODEL [JM 2]	STANDARD MODEL [160]
UNCODED	2.39	2.40	2.40
GRAY	2.22 (-7.11 %)	2.23 (-7.08 %)	2.23 (-7.08 %)
CORR	2.70 (+12.94 %)	2.71 (+12.92 %)	2.71 (+12.92 %)
AVG. FLIT LATENCY: 40.2 ns		AVG. NETWORK LATENCY: 193.3 ns	

Table 9: Link energy quantities and network performance without VCs for uncoded, gray-coded and correlated data [JM 2].

using synthetic traffic patterns. Simulating a NoC, with 32 PEs, using uniform random traffic, with flit injection rates between 20% and 95% and 32 restarts per injection rate, took between 1.5 and 2 hours per set. SystemC only supports single thread simulations, but parallel execution on many cores is possible. In the next step, the best parameters can be further tested with real world applications. For instance, the simulation of 25 ms of CPU time in the PARSEC region of interest (ROI) from netrace files takes 41 hours. This demonstrates, why an incremental approach is essential for high productivity during design space exploration and how the proposed tools and processes allow for such an analysis.

7.5.6 Application example: adaptive face tracking



(a) Detection of a single face with bounding box (yellow) and feature points (white). (b) Additional faces enter scene; three faces detected with number of feature points. (c) Moving faces are tracked and right face (partially) leaves the scene. (d) Middle face is partially hidden by front face which reduces number of trackable features.

Figure 62: Face detection and tracking algorithm for a 3D VSoC [JM 5].

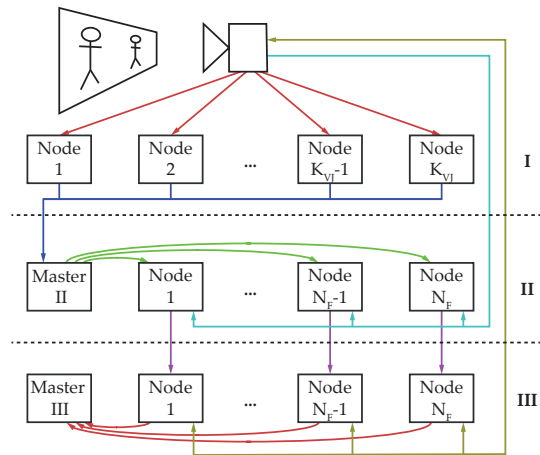


Figure 63: Stages of the application with nodes and data flows. Edges of same color belong to the same part of the algorithm and are similar in bandwidth. [JM 5].

We demonstrate the expressiveness of the proposed application model by a typical use case of heterogeneous 3D SoCs: An adaptive face tracking algorithm. It can be, for instance, used for video conferences or surveillance. Since a small, efficient and high-performance device is necessary, a 3D VSoC is a very good target. The function and model of the application is published in [JM 5]. Here, we only highlight the properties of the resulting Petri net. The four basic steps of the algorithm are shown in Figure 62: Viola-Jones algorithm [161] initially detects faces (Phase I). Shi and Tomasi algorithm [162] finds features, which are tracked (Phase II). Kanade-Lucas-Tomasi (KLT) algorithm [163] tracks these in the video (Phase III). KLT algorithm is fast and allows for real-time tracking but may lose some features. Thus, a periodic re-initialization is necessary. All stages of the algorithms run in parallel and are mapped to a 3D SoC as shown in Figure 63 including data flows. Edges of same color belong to the same part of the algorithm and are similar in bandwidth. For the sake of brevity, we kindly refer to [JM 5] for bandwidths and statistics. Python scripts are written to generate the XML description of the application depending on the number of available cores in the VSoC. From approximately 500 lines of python code, 35,305 lines of XML model are generated for 8 PEs. This example shows that it is possible to model even complex applications using the proposed tools. Since the XML description is rather bulky, generation via python is an easy and usable option.

7.6 CONCLUSION

In this chapter we introduced models, tools, processes and analysis for empirical design space exploration using simulations in A-3D NoCs. The clear definition of models highlights possibilities and limitations, i. e. the expressiveness, of the design space exploration. The

models are implemented as a box of tools for simulation and evaluation. These tools assist the designers during design space exploration process. A standard process is proposed based on the incremental approach as proposed in the previous chapter. The analysis of the results is also demonstrated. In comparison to existing solutions, the proposed simulator offers similar performance while providing more features: A novel application model is provided, which allows for simulation of traffic pattern found in heterogeneous 3D SoCs. The NoC simulator is configurable during execution using XML files enabling rapid prototyping via simulations of A-3D NoCs. Finally, innovative report features, such as a report database and data flow matrices, allow for efficient design evaluation, with high, unparalleled model accuracy. To summarize, the proposed software and processes enable design space exploration via simulations in A-3D NoCs and therefore will be used in the subsequent work to assess architectural level optimizations.

In the previous chapters, system-level optimizations for A-3D NoCs and tools and methods for empirical design space exploration have been presented. Using the incremental approach from Section 5.6.2, we will focus on (micro-) architectural features in this and the subsequent chapter. Here, we draw attention to router buffers, since these are one major source of area and power consumption: The area of the buffers is approx. 79% of the overall router area and buffers account for 85% of the energy consumption for a standard 3D router, with 4 VCs and 8 flit deep buffers synthesized for commercial 65 nm digital technology. Therefore, many works aim at reducing router area by tackling router buffer area as discussed in Chapter 4. Buffer area and power consumption is a topic of even higher importance in heterogeneous 3D SoCs: The same buffer space will require more area and energy if implemented in a mixed-signal node or a less advanced digital technology in comparison to modern digital nodes. Therefore, it is advantageous to reduce the implemented buffer space in expensive layers.

In general, area reductions via buffers is not a novel topic. Here, we peruse novel approaches for buffer distributions and asymmetric buffer depth, which both reduce area and power for heterogeneous 3D SoCs. The approaches are not useful for homogeneous systems. The novelty lies in exploitation of heterogeneity enabling new architectures. To the best of our knowledge, comparable ideas have not been investigated in literature so far. The proposed buffer redistribution scheme can be applied to many input-buffered router architectures, which are already optimized for other features.

8.1 BUFFER DISTRIBUTIONS AND BUFFER DEPTHS

Buffers differ in area and power requirements between layers in a heterogeneous 3D SoC, as already discussed. Therefore, the number of implemented buffers must be reduced in layers, in which memory is expensive; this has the largest positive effect on the system's costs. This approach is given by the principle that "*it is advantageous to place as many buffers as possible in a layer, which is optimized for memory*" [JM 3]. There are two options to realize the principle: On a router microarchitectural level, buffers can be redistributed among routers and layers. This results in novel *buffer distributions* with improvements as discussed in Section 8.4.1. On a router architectural level, buffer depths in layers can be different. This results in asymmetric *buffer*

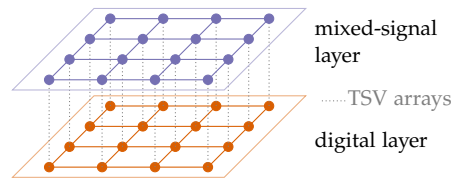


Figure 64: Two-layered 3D NoC.

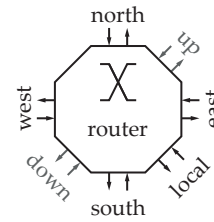


Figure 65: Router architecture.

depths, with advantages as discussed in Section 8.4.2. We also assess a combined approach for additional benefits. The router architectures and results of this chapter have originally been presented in [JM 3] and [JM 11]. As part of this thesis, the content of these publications is not substantially extended, since the work was completed. Rather, the presentation is clarified.

8.1.1 *Technology and network setting for evaluation*

We model a simplified technology and network scenario. This avoids unnecessary complications of a generalized heterogeneous 3D SoC model. We use a 3D SoC consisting of two, heterogeneous layers (Figure 64). The influence of buffer distributions and buffer depths can be evaluated using only two layers, as argued in Section 5.5 for a typical example for NoCs in heterogeneous 3D SoCs. The more advanced node, with smaller features size, is below the other layer. This layer, with a less advanced technology, is on top. Buffers are more expensive in the upper layer. The NoC topology is two stacked 4×4 grids. The results can be generalized to any combination of a more and a less expensive technology, such as digital and mixed-signal nodes.

We are only interested in the influence of buffers. This implies the following simplifications: We do not consider different topologies or clock frequencies per layer. We model a synchronous SoC, in which all components are clocked in the lowest speed. This is the most pessimistic scenario. In a separated approach, we assess the potential performance gains from a GALS implementation.¹

8.1.2 *Baseline: symmetric routers*

We use a symmetric input-buffered router as baseline, with deterministic dimension-order routing, 8 flit deep buffers and 4 VCs. Routers are connected to neighbors, which are directly adjacent, i. e. in each cardinal direction, to a local port and downwards or upwards, depending on the layer (Figure 65). The default router pipeline of conventional routers is used (comp. [76]), which was already introduced

¹ Implementing asynchronous communication is rather expensive and complicated. The actual microarchitectural modifications are assessed separately, see Section 9.6.

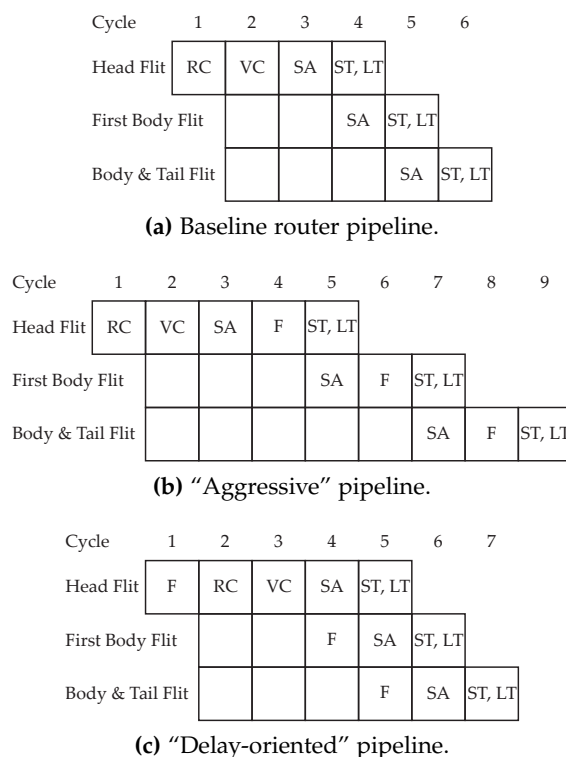


Figure 66: Time behavior of input buffers of vertical links (*RC* – routing calculation, *VC* – VC calculation, *SA* – switch allocation, *F* – fetch, *ST* – switch traversal, and *LT* – link traversal) [JM 3].

in Figure 17: There are four stages (Figure 66a): First, a routing is calculated (*RC*). Second, a virtual channel is allocated (*VC*). A route and VC arbitration are only calculated for head flits. Third, the switch is arbitrated among flits (*SA*). During arbitration, VCs with a lower number have a higher priority. Fourth, the arbitrated flits traverse switch (*ST*) and link (*LT*).

8.2 ROUTERS WITH OPTIMIZED BUFFER DISTRIBUTION

The microarchitecture of routers is optimized for area and power by distributing buffers among pairs of adjacent routers in different layers. Buffers are more expensive in the upper layer. To tackle this constraint of heterogeneity, we change the distribution of buffers of vertical, upward links: We place the input buffers of this very link from the upper router to the lower router as output buffers. We change the location of all VC buffers and do not consider individual VCs. The buffer status registers for state, route and output VC remain in the upper router. The buffer location of all other links in the network are not changed. This yields a novel router microarchitecture, as shown in Figure 67b: In the upper layer, input buffers are located on all links except the incoming vertical links. In the lower layer, input buffers are located in all links and output buffers are located on vertical, upward links.

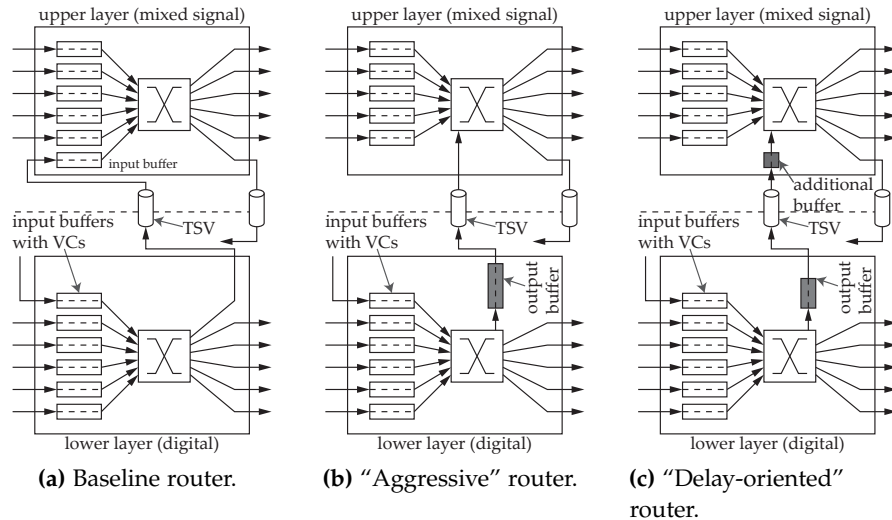


Figure 67: Architecture of router pairs for the proposed microarchitectural optimization [JM 3].

The area and power consumption of the routers in the upper layer is reduced by this optimization. However, the increased delay of buffer access impedes performance. Therefore, the architecture is called “*aggressive*”, in terms of area savings. To reduce this performance impact of heterogeneity, we additionally propose a “*delay-oriented*” microarchitectural optimization: We split the buffers of the vertical links into two parts. A flit can be stored in an intermediate buffer in the upper router, located between output buffers in the lower layer and the crossbar in the upper router. The remainder of the buffers is located as output buffers in the lower router. This is shown in Figure 67c.

8.2.1 Router pipelines

The router pipeline of routers in the faster (digital) layer is shown in Figure 66a. It is standard and not modified. The proposed modifications to the router buffer do not influence the router pipeline: Only flits transmitted from lower to upper layer could be influenced, because buffers from the upper layer are moved downwards as output buffer. Thus, the critical path from the input buffers via the crossbar to the output buffers in the lower layer is shorter. However, this does not yield a performance advantage. Only these links would be clocked faster, which is not realistic, since intrinsically asynchronous routers are very costly and the faster clocked part of the circuit could not be fed with incoming data.

The router pipeline of the “*aggressive*” architecture is shown in Figure 66b. Flits traveling the vertical, upward connections have a longer critical path due to the traversal of the TSVs. To fetch data, which is stored in the output buffers in the adjacent layer, an additional cycle

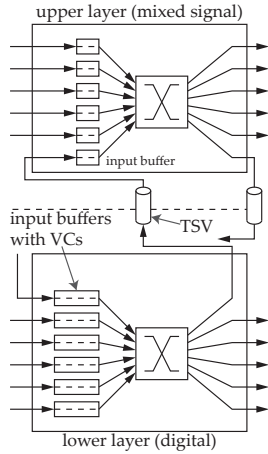


Figure 68: Conventional pair of routers with buffer depths optimization [JM 3].

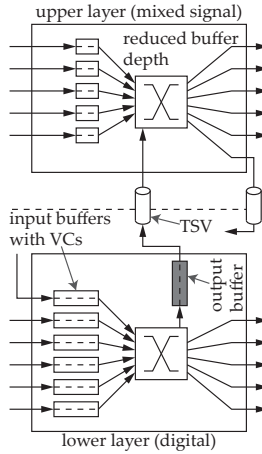


Figure 69: “Aggressive” pair of routers with buffer depths optimization [JM 3].

is introduced (“F”). Therefore, flits from this link leave the router only every other cycle (under zero load). All other input ports and their pipelines are standard, i. e. as shown in Figure 66a.

The router pipeline of the “delay-oriented” router is shown in Figure 66c. In this architecture, only head flits require the additional fetch cycle, since follow-up flits are sent in parallel to the intermediate buffer in the upper router while the previous flits traverse the router. Hence, flits from the vertical links leave the router in every clock cycle after an initial delay (under zero load).

8.3 ROUTERS WITH OPTIMIZED BUFFER DEPTHS

We also optimize routers on architectural level by reducing buffer depths in the more expensive layer. This is depicted for a pair of “aggressive” routers in Figure 69. Asymmetric buffer-depth reductions offer large optimization potential: While minimizing the area footprint of the whole NoC has a linear relation, the performance does not decline proportionally, since the relation between buffer depth and performance is non-linear: It declines slowly until network saturation is reached and then collapses. The theoretical relation between buffer depths, network costs, power consumption and performance (measured in package latency) is well-known for 2D systems (e.g. [77]). For NoCs targeting heterogeneous 3D SoCs, this relation is exemplified in Figure 70, with asymmetric buffer depths of 8 flits in the lower layer and 2 – 16 flits in the upper layer using uniform traffic at 3.2 GB/sec traffic injection. We synthesized a gate-level router model for 130 nm and 65 nm commercial digital technologies to generate the results. Area and power results are normalized by their maximum value. The plot shows a non-linear relation between buffer depth and costs; large buffer depths do not offer performance advantages. The

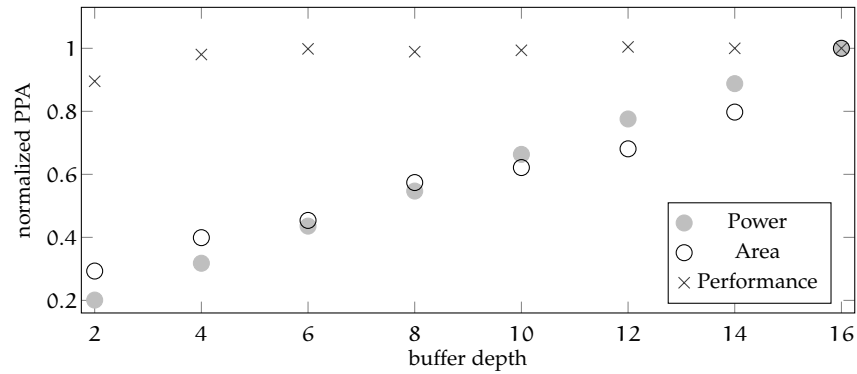


Figure 70: Relation between buffer depths and PPA, with area and power from synthesis and simulated performance (buffer depths: 8 flits in faster layer, 2 up to 16 flits in slower layer).

results imply that a small buffer depth will offer a good compromise with low costs and adequate performance if the buffer depth is still located on the left-hand side of the plateau. Based on these findings, we optimize buffer depths both for traditional router architectures with standard buffer organization, see Figure 68, and the proposed microarchitectural buffer reorganization, see Figure 69.

8.4 RESULTS

The results are originally presented in [JM 11] for buffer redistribution and in [JM 3] for buffer depths. As argued in Chapter 5, the design space exploration for buffer optimization is rather complicated, since both microarchitectural and architectural optimizations must be assessed and their effects are interwoven. Therefore, we chose the incremental approach from Section 5.6.2 using the introduced hierarchical order. First, we evaluate microarchitectural optimizations by benchmarking the proposed router architectures, with buffer reorganization and with 8 flit deep buffers in all layers in Section 8.4.1. We choose this buffer depth as baseline, since larger buffers only increase area and power overhead without performance advantages. Second, we evaluate architectural optimizations by different buffer depths from 4 – 8 flits to find the tripping point, at which buffer size increases add additional costs, without positive performance influence in Section 8.4.2. Third, the findings are combined by simultaneously applying microarchitectural and architectural optimizations. The “aggressive” router microarchitecture is evaluated with buffer depths of 8 flits in lower layer and 4 – 8 flits in the upper layer. Since the other architectures show similar results, we do not present the results in detail. Thereby, we find the optimal buffer depth per layer. This is applied to the baseline (symmetric) and the “delay-oriented” router architecture in Section 8.4.3.

ROUTER ARCHITECTURE	POWER CONSUMPTION			SAVINGS
	65 NM	130 NM	ROUTER PAIR	
Symmetric	20.7 mW	58.3 mW	79 mW	–
“Aggressive”	23.6 mW	49.7 mW	73.3 mW	7.2 %
“Delay-oriented”	22.9 mW	51.9 mW	74.8 mW	5.4 %

Table 10: Power of a router pair with symmetric buffer depths [JM 3].

We compare our results with a baseline, which is set by a symmetric standard router synthesized from RTL for commercial digital 130 nm and 65 nm technologies, with 8 flits per VC, a single VC and a flit size of 32 bit. The clock frequency of routers depends mainly on the technology: For the 65 nm-technology, the routers clock at approximately 1 GHz and for 130 nm at 820 MHz. We use this to compare power and area savings. Please note, that we refer to a *pair of routers* as those with the same x- and y-coordinates in adjacent layers.

8.4.1 Routers with optimized buffer distribution

Power savings:

The total power of the routers are given in Table 10 for a stacked pair of routers with 50% toggle activity. The savings are calculated from comparison with the baseline (conventional architecture, symmetric buffers). The power consumption in the 65 nm layer increases by moving buffers there. Since single buffers consume less power in this layer, the overall power consumption declines by 7.2% for the “aggressive”, asymmetric architecture and by 5.4% for the “delay-oriented” design, using a linear dependency, as presented in [JM 3].

Area savings:

The buffers occupy 84% of the router area for the 130 nm technology and 79% for the 65 nm technology in the conventional design. Different silicon nodes yield varying memory cell sizes; we measure an actual ratio of 3.7 for similar flip-flops in the commercial 65 nm and 130 nm technologies. By exploiting this area difference, the “aggressive” router and “delay-oriented” router are smaller than baseline by 9.6% and 8.3%, respectively (Table 11).

Performance loss:

For the performance evaluation we use the cycle-accurate simulation model, as proposed in Chapter 7. The timing of routers is modeled using the parameter Δ as introduced in Section 7.1.2 by setting it to 4 for downward input ports of routers in the upper layer and to 3 for all remaining ports. We benchmark with synthetic traffic patterns and

TRAFFIC/ APPLICATION	TASKS	PACKETS / INJ. RATE	SYM. ROUTER CLOCK CYCL.		"AGGRESSIVE" ROUTER CLOCK CYCL.		DIFF.	"DELAY-ORIENTED" ROUTER CLOCK CYCL.		DIFF.
UNIFORM (MEDIAN)	—	0.8 GB/s	1318	1310	1310	1322	.6%	1322	-3%	
COMPLEMENT	—	0.8 GB/s	170	210	190	178	-19%	178	-4.4%	
HOTSPOT	—	0.8 GB/s	1290	1930	55208	1656	-33%	1656	-4.9%	
VOPD & SHAPE DEC. [158]	17	3416	55188	55208	69766	55192	-.7%	55192	-.7%	
VOPD [106]	16	4044	63766	69766	77316	64368	-8.6%	64368	-.9%	
DVOPD[106]	32	8762	65202	77316	66458	66458	-16%	66458	-1.9%	
MPEG-4 [106]	12	3467	142010	142010	142010	142010	.0%	142010	.0%	
PIP [106]	8	576	9984	11364	10114	10114	-11%	10114	-1.3%	
MWVD [106]	12	1120	14114	16216	14338	14338	-13%	14338	-1.6%	
H.263 DEC., MP3 DEC. [106]	13	19636	172554	181068	173232	173232	-4.7%	173232	-.4%	
MP3 ENC., MP3 DEC. [106]	12	1652	18500	30752	19730	19730	-40%	19730	-6.2%	
H.263 ENC., MP3 DEC. [106]	14	24291	373778	430884	378704	378704	-13%	378704	-1.3%	
AVERAGE PERFORMANCE LOSS TO SYMMETRIC BASELINE										
AREA SAVING FOR ROUTER INPUT BUFFERS TO SYMMETRIC BASELINE										
							14%			2.1%
							9.6%			8.3%

Table 11: Benchmark results with symmetric buffer depth. [JM 3].

real-world based with application traffic from [106, 158]. A manual mapping of tasks to PEs prioritizes short communication distances to reduce the network load; further, the load in the upper layer in a less advanced technology is reduced. For some benchmarks, there are fewer tasks than PEs, in which some PEs are not used. The results of the simulations are shown in Table 11. We use a single instance of inputs per application and measure the total delay in clock cycles. Therefore, the performance measurement is independent of the target clock frequency.

It is expected, that the proposed router architectures have reduced performance due to the additional fetch cycles in the router pipelines. Further, the “aggressive” router has lower performance than the “delay-oriented” router. This is supported by the experimental results: The “aggressive” router has 14% worse average performance in comparison with baseline. The “delay-oriented” router is only 2.1% worse. The first approach to the worst case approximation is given by hot-spot traffic, in which all PEs send data to a single destination located in the more advanced layer. This yields high performance losses of 33% and 4.9%. If the hot-spot was located in the other layer, the performance would be even worse. This is demonstrated by the audio decoder and encoder benchmark, whose mapping yields a high utilization of vertical links: This benchmark yields the highest performance losses of 40% and 4.9% among the test cases. Please note, that the performance increase of 0.6% for uniform random traffic is an artifact with reasons in statistics of our measurement. The performance results demonstrates the full range of performance impact.

8.4.2 *Routers with optimized buffer depths*

To evaluate the effect of optimized buffer depths, we use the simulation model from Chapter 7. Modification of the parameter d as introduced in Section 7.1.2 allows for rapid design space exploration. The baseline is set by symmetric buffer depth. We assess both symmetric and asymmetric buffer depth reductions.

Symmetric buffer depth reductions are evaluated by comparison with the baseline using 8 flit deep buffers in both layers to a baseline router using 4 flit deep buffers. The results are shown in Table 12. Reducing the buffer depth in the conventional router pipeline yields a large performance loss of 49%. In combination with the proposed router architectures, the performance declines by additional 2.5% and 1.2%. The power and area savings of all three router architectures are between 36.1% and 41.6%. As expected, the proposed novel router architectures have lower performance, but also lower area and lower power than baseline. In general, the performance loss of symmetric buffer depth reductions is high, for which power and area savings are not justified.

TRAFFIC/ APPLICATION	BASELINE EXECUTION		SYMMETRIC ROUTER (4 FLIT)		"AGGRESSIVE" ROUTER (4 FLIT)		"DELAY-ORIENTED" ROUTER (4 FLIT)	
	CYC. CYCL.	LOSS	CYC. CYCL.	LOSS	CYC. CYCL.	LOSS	CYC. CYCL.	LOSS
UNIFORM	1318	45%	2411	45%	2480	47%	2448	46%
COMPLEMENT	170	42%	292	42%	310	45%	302	44%
HOT-SPOT	1290	46%	2380	46%	2542	49%	2438	47%
VOPD & SHAPE DECODER [158]	55188	56%	124574	56%	124578	56%	124578	56%
DVOPD [106]	63766	52%	131556	52%	134460	53%	132734	52%
VOPD [106]	65202	51%	132236	51%	133636	51%	132838	0.5%
MPEG-4 [106]	142010	47%	267264	47%	267264	47%	267264	47%
PIP [106]	9984	54%	21670	54%	21926	54%	21800	54%
MWID [106]	14114	48%	27120	48%	27868	49%	27414	49%
AV. PERF. LOSS		49%		49%		50%		50%
AREA SAVINGS		36.1%		36.1%		40.1%		39.8%
POWER SAVINGS		37.3%		37.3%		41.6%		40.6%

Table 12: Performance results for a symmetric buffer depth of four flits. [JM 3].

TRAFFIC / APPLICATION	"AGGRESSIVE" ROUTER, BUFFER DEPTH IN UPPER LAYER										
	BASELINE EXECUT. CLC. CYCL.	4		5		6		7		8	
		EXEC. TIME IN CLOCK CYCLES		LOSS - DEPENDING ON THE BUFFER SIZE (4 - 8)		IN UPPER LAYER					
UNIFORM (MEDIAN)	1318	1804	27%	1438	8.3%	1364	3.4%	1320	.2%	1310	-6%
COMPLEMENT	170	288	41%	218	22%	214	21%	210	19%	210	19%
HOTSPOT	1290	2466	48%	1978	35%	1930	33%	1930	33%	1930	33%
VOPD & SH. DEC. [158]	55188	118686	54%	90000	39%	70560	22%	55208	0%	55208	0%
VOPD [106]	63766	94754	33%	81490	22%	75038	15%	69766	8.6%	69766	8.6%
DVOPD [106]	65202	115532	44%	93300	30%	79830	18%	77316	16%	77316	16%
MPEG-4 [106]	142010	175972	19%	159158	11%	149680	5.1%	142010	.0%	142010	.0%
PIP [106]	9984	13382	25%	12148	18%	11642	14%	11264	11%	11264	11%
MWD [106]	14114	21214	33%	17864	21%	16808	16%	16220	13%	16216	13%
H.263 DEC., MF3 DEC. [106]	172554	370904	53%	283872	39%	230020	25%	181072	4.7%	181068	4.7%
MF3 ENC., MF3 DEC. [106]	18500	39622	53%	33808	45%	32086	42%	30756	40%	30752	40%
H.263 ENC., MF3 DEC. [106]	373778	554312	33%	484378	23%	453832	18%	430888	13%	430884	13%
AVERAGE PERF. LOSS	0%		40%		28%		21%		14%		14%
AREA SAVING	0%		44%		36%		27%		18%		9.6%
POWER SAVINGS	0%		29.3%		23.8%		18.2%		12.8%		7.2%

Table 13: Results for asymmetric buffer depths with asymmetric "aggressive" router architecture [JM 3].

Asymmetric buffer depth reductions are evaluated by comparison of the “aggressive” router architecture with 8 flit deep buffers in all layers to a NoC with 8 flit deep buffers in the lower, more advanced, layer and 4 – 8 flit deep buffers in the upper layer. The results are shown in Table 13. For small buffer depth in the upper layer, the performance losses are rather high, with up to 40%. Therefore, a large asymmetry in buffer depth is not advantageous. However, a light asymmetry is promising: Reducing the buffer depth from 8 to 7 flits in the upper layer yields a constant performance loss of 14% in comparison to the conventional baseline architecture, but doubles the area savings from 9.6% to 18% and increases the total power savings from 7.2% to 12.8%.

8.4.3 *Combination of both optimizations*

The results suggest evaluating a combination of both proposed optimization approaches, as those might reduce area and power while maintaining acceptable performance. In terms of microarchitectural optimizations, the “delay-oriented” router offers the best compromise. In terms of architectural optimizations, slight asymmetry of 7 and 8 flit deep buffers has the best properties. The combination of both approaches is compared in Table 14. Without microarchitectural optimizations, using the conventional router, with asymmetric buffer depth yields a negligible performance loss of 0.06%, with 13% reduced area and 9.3% power decline. With microarchitectural optimizations, i. e. using the “delay-oriented” router, the performance declines by 4.6%, with 28% area savings and 15% total power reduction.

8.4.4 *Influence of clock frequency deviation*

As worst case approximation, we assume globally synchronous clocks. Here, we assess the influence of this assumption. We use the maximum clock frequencies from RTL model synthesis for the routers and model a GALS system, in which the upper layer is clocked at 820 MHz and the lower at 1000 MHz. We do not consider the actual hardware implementation of router interfaces supporting different clock speed (cf. Chapter 9 for a detailed discussion). Uniform random traffic is injected at 3.2 GB/sec. The results vary by 2% on average.

8.5 DISCUSSION

Heterogeneity poses severe limitations to 3D SoCs, since mixed-signal layers or layers in a less advanced digital node are more restricted in terms of area, power and, thus ultimately, performance. Here we demonstrate that power and area consumption in the NoC will be

TRAFFIC/ APPLICATION	BASELINE EXECUTION CLC. CYCL.	SYM. ROUTER 7 FLIT BUFFERS		"DELAY-ORIENTED" 7 FL. BUF.	
		CLC. CYC.	LOSS	CLC. CYC.	LOSS
UNIFORM	1318	1316	-0.2%	1372	4.0%
COMPLEMENT	170	170	.0%	186	8.6%
HOT-SPOT	1290	1300	.8%	1412	8.6%
VOPD & SHAPE DECODER [158]	55188	55188	.0%	55192	.01%
VOPD [106]	63766	63766	.0%	65564	2.7%
DVOPD [106]	65202	65210	0.01%	67170	3.0%
MPEG-4 [106]	142010	142014	.003%	142014	.003%
PIP [106]	9984	9984	.0%	10366	3.7%
MWD [106]	14114	14118	0.03%	14476	2.5%
H.263 ENC, MP3 DEC. [106]	373778	373780	.0%	388554	3.8%
MP3 ENCODING, MP3 DEC. [106]	18500	18500	.0%	22182	17%
H.263 DEC, MP3 DEC. [106]	172554	172558	.0%	174976	1.4%
AVERAGE PERFORMANCE LOSS	0%		0.06%		4.6%
BUFFER AREA SAVINGS	0%		13%		28%
TOTAL POWER SAVINGS	0%		9.3%		15%

Table 14: Benchmarks for the symmetric and proposed asymmetric architectures with a buffer size of 7 flits. The execution time is measured in clock cycles [JM 3].

reduced if router buffers are optimized on an architectural and micro-architectural level using asymmetry.

Applying the proposed microarchitectural optimizations, only the "delay-oriented" router is promising: A relatively low performance loss of 2.1% offers area savings of 8.3%. For the "aggressive" router, the area savings are similar, with 9.6%, but the performance loss is rather high at 14%.

Applying architectural optimizations, large heterogeneity results in large performance losses of up to 41%. However, network performance can be maintained using light asymmetry; We achieved cost reductions of 13% and 9.6% at a negligible performance loss of 0.06%.

Applying both approaches simultaneously is promising for low-power designs, with lavish performance constraints. Using the proposed "delay-oriented" router with a light asymmetry in buffer depth of one flit offers large area reductions of 28% and power reductions of 15% at a minor performance loss of 4.6%.

8.6 CONCLUSION

Area and power consumption of NoCs in heterogeneous 3D SoCs can be reduced by applying architectural and microarchitectural optimization to buffer distribution and buffer depths. These types of optimizations require delicate fine-tuning of parameters, since the influence on performance is non-linear. Therefore, it is necessary to find buffer depths and distributions which offer the best compromise between performance penalty and reduced area and power consumption. The two proposed options offer either cost reductions of up to 13% and 9.6% for area and power, respectively, at negligible performance loss or larger area and power savings at a larger performance loss. Therefore, a universal design principle cannot be derived – it depends on the power and area budget as well as on the performance needs of an individual design. The proposed approach offers high re-usability: It can be applied to any router architecture, which has input buffers and does not require special buffer features. It is not applicable to router architectures with special buffer designs such as the SMART router [164] that implements buffers with repeaters to allow for single-cycle flit transmission.

OPTIMIZATION OF ROUTING AND ARCHITECTURES

In the previous chapter, we discussed (micro-) architectural router optimizations with regard to router buffers. In this chapter we continue to follow the incremental approach from Section 5.6.2 and shift focus on the design of routing algorithms and co-designed router architectures in A-3D NoCs. In general, there are many publications on routing algorithms in 3D NoCs, such as [126, 165]. These do not consider the effect of implementation technology, because homogeneous 3D SoCs offer the same properties in all parts of the chip. In contrast, novel routing algorithms are required for heterogeneous 3D SoCs, since the technology node, in which routers are implemented, influences their properties. For instance, the performance of routers in mixed-signal layers is smaller than the performance in advanced digital nodes. This can be exploited by routing: It is advantageous to send packets along path with high-performance routers. Models, which allow to calculate the transmission time along heterogeneous paths, i. e. path through layers in disparate technologies, do not exist so far. Since these allow finding paths for packets with optimized performance, we present novel models for this very purpose in this chapter. We further propose two routing algorithms and introduce an exemplary, bespoke router architecture. Both increase the network performance to beyond state-of-the-art in heterogeneous 3D SoCs as originally presented in [JM 1]. Since this publication concluded the topic, we only improve the clarity of presentation here.

9.1 INFLUENCE OF HETEROGENEITY ON ROUTING

If components in a heterogeneous 3D SoC are connected via a NoC, the network will span layers in different nodes. This causes a severe issue: Homogeneous routers, which are common for state-of-the-art 3D NoC designs [128, 130], yield unbearable costs (in terms of area and power) in layers in mixed-signal nodes. To solve this, routers with heterogeneous characteristics but with aligned properties are required. NoCs implementing this principle are called asymmetric, i.e. A-3D NoCs. Their architecture needs careful consideration of numerous trade-offs. Among the different challenges, the most relevant are:

Challenge 1: Routers in mixed-signal layers are disproportionately expensive.

Challenge 2: The different technology nodes influence the maximum number of routers per layer.

Challenge 3: Routers in mixed-signal layers are slower clocked. Routers in the more advanced or digital nodes can be clocked faster than mixed-signal layers.

Challenge 4: Interaction between routers in adjacent layers is not purely synchronous, intrinsically. Routers in different layers are clocked differently. This influences packet provision.

The challenges must be tackled, especially since communication in A-3D NoCs has a unique characteristic: Throughput and latency differ between layers because of varying router numbers and router clock speeds. We will contribute that low packet provision in some layers impedes the packet's performance in the whole network. We demonstrate this by means of modeling in Sections 9.2 and 9.3. Next, we quantify the impact of heterogeneity in Section 9.4 and tackle this obstacle by novel principles for routing algorithms and novel router microarchitectures. We apply the principles to concrete routing algorithms in Section 9.5 and design a fitting router architecture in Section 9.6. We quantify advantages in Section 9.7 and discuss the findings in Section 9.8.

9.2 MODELING TECHNOLOGY HETEROGENEITY

We model the influence of heterogeneity on area and timing. The models cover any type of commercial technology and any feature size under the following assumptions. We assume that the delay of TSVs is negligibly small compared to delay of links and logic. This is reasonable, since TSVs are only 50 μm long (cf. Section 3.3.4). Next, we do not model KOZs, because these are a constant overhead independent of technology node. Further, routers must not be located at the identical position in their layer, since a redistribution (RD) can connect routers and TSVs (cf. Section 6.2.1). The variability of the RD is modeled by converting router locations to router addresses. We assume a GALS design, in which components are clocked at different speeds. A synchronous model would waste performance, especially since mixed-signal components have very low clock speeds.

We do not contribute a power model for routing due to diverse influence parameters. For instance, the actual data transmitted vastly affect dynamic power consumption of links, which is hard to model a priori without simulations, as shown in Section 7.2.3.1.

We consider a chip with ℓ layers and their index set $[\ell] = \{1, \dots, \ell\}$. We assume n - m -mesh topologies of NoCs per layer. The structure size of the technology nodes of layers, measured in [nm], is given by $\tau : [\ell] \rightarrow \mathbb{N}$. A chip layer with index ι will be called »more advanced node« than a layer with index ξ , if $\tau(\iota)$ is smaller than $\tau(\xi)$ (for easy notation). Further, we define:

Definition 9.1 (Relative technology scaling factor [JM 1]). Let ξ and ι be the indexes of layers with technologies $\tau(\xi)$ and $\tau(\iota)$ and with $\tau(\xi) > \tau(\iota)$. The *relative technology scaling factor* Ξ is:

$$\Xi(\xi, \iota) := \frac{\tau(\xi)}{\tau(\iota)} \tag{60}$$

Please note, that the proposed model has been published in [JM 1]. Definitions are directly taken from this publication, since their definition is unique.

9.2.1 Area model

The area of the communication infrastructure in layers is determined by the size of individual routers and their number. We propose an abstract model covering the influence of technology nodes, synthesis constraints, synthesis tools and router architectures.

9.2.1.1 Area of routers

The technology node, in which a router is implemented, affects the size of each of its architectural components. Requirements for area of both combinatorial and sequential logic are influenced; thus, the size of routing computation, crossbars and buffers differs. The area of logic reduces its size (ideally) quadratically for more advanced nodes. The remainder of routers (e.g. power supply) does not scale. This yields a total area model of the form $\hat{\alpha} + \alpha\sigma^2$, in which $\hat{\alpha}$ is the constant part, α is a non-ideality factor, and σ is the feature size. This model is depicted in Figure 71. By this model we define the area scaling factor as the difference between baseline technology, i.e. the largest node, and any target technology:

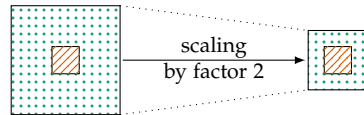


Figure 71: Area size model with constant area (orange, lined) and scalable area (green, dotted) [JM 1].

Definition 9.2 (Area scaling factor [JM 1]). Let ξ and ι be the indices of two chip layers with technologies $\tau(\xi)$ and $\tau(\iota)$ and with relative technology scaling factor $\Xi(\xi, \iota)$. The *area scaling factor* $s_f : (\mathbb{R}) \rightarrow \mathbb{R}$ is given by:

$$s_f(\Xi) := \frac{\alpha + \hat{\alpha}}{\Xi^2 + \hat{\alpha}} \tag{61}$$

The model assumes that the chip area is normalized to one area unit. The non-ideality factor α denotes, how well the technology scales quadratically. The base technology area offset $\hat{\alpha}$ is dominated by components which do not scale. Both parameters are evaluated for the

used set of technology nodes by synthesis of a small circuit with typical properties, such as a basic router model (see Sec. 9.7.1). Then, the parameters can be estimated using function fitting. In an ideal setting, $\alpha = 1$ and $\hat{\alpha} = 0$. For instance, if two layers are implemented in an ideal theoretical technology node with $\tau(1) = 180$ nm and $\tau(2) = 45$ nm, the technology scaling factor will be $s_f(\Xi(180, 45)) = 16$. For a setting with 90 nm and 180 nm nodes, it will be $s_f(\Xi(180, 90)) = 4$.

9.2.1.2 Number of routers

Not only the size of individual routers is influenced by varying nodes, but also the number of routers in layers is affected. This can also be modeled using the area scaling factor s_f for an approximate lower bound for the number of routers, which can be implemented in a more advanced node.

9.2.2 Timing model

We model the transmission time of packets in heterogeneous 3D SoCs. It is determined by the timing of individual routers and their interaction, i. e. the network topology. We account for the clock delay of individual routers, and deduct the propagation speed of packets, which traverse multiple routers.

9.2.2.1 Clock delays

The clock delays in heterogeneous 3D SoCs vary; in digital and more advanced layers, routers are potentially faster, while routers in mixed-signal layers are constrained by slower clock frequencies. The clock delay of routers is determined by two drivers: The interconnect delay does not scale and therefore limits router performance in small nodes (power constraints also limit the maximum achievable clock frequency); the logic delay does scale and is larger than the interconnect delay for large nodes. We model a clock scaling factor, which gives the ratio, at which the clock delays in different nodes scale. There are two drivers, one of which is scaling and one of which is non-scaling. Since the actual, physical influence is hard to estimate in models, we propose an empirical approach by fitting a sigmoid function. Unlike the area model, we do not propose a physical model. Since the proposed timing model shows high accuracy of the function fitting, as shown in Section 9.7.1, the results presented in this Chapter will not change, if another (physical or empirical) model with similar or higher accuracy is used.

Definition 9.3 (Clock scaling factor [JM 1]). Let ξ and ι be the indices of two chip layers with technologies $\tau(\xi)$ and $\tau(\iota)$, with $\tau(\xi) > \tau(\iota)$ and with technology difference $\Xi(\xi, \iota)$. Let c_b be the base clock delay of the layer with index ξ and c_c be the minimum achievable clock

delay, which is limited from physical effects such as power dissipation or interconnect delays. Let β be the maximum speedup achievable: $\beta := c_b/c_c$. The *clock scaling factor* $c_f : (\mathbb{R}) \rightarrow \mathbb{R}$ is given by:

$$c_f(\Xi) := \frac{\beta}{1 + \hat{\beta} \exp(-\tilde{\beta}(\Xi - \hat{\beta}))} \quad (62)$$

The function converges to the maximum achievable speedup β . The other parameters must be set by fitting the function to a set of synthesis results (see Section 9.7.1).

9.3 MODELING COMMUNICATION

We model latency, throughput and transmission speed in the NoC under zero load. Two separate models for communication are required. Horizontal communication within a layer is synchronous. Vertical communication between layers is not purely synchronous, depending on router architecture and technology nodes.

9.3.1 Horizontal communication

We call the speed at which packets are transmitted horizontally, under zero load, *propagation speed*. It varies with technology nodes, since the number and clock frequency of routers differ. Communication is synchronous. The *propagation speed is given by the distance traveled divided by the packet latency*.

To calculate the *distance*, we use the following notations: All possible positions of routers are given by the set $P = \mathbb{R} \times \mathbb{R} \times [\ell]$. The x - and y -coordinates of routers are measured in $[m]^1$. The symbols p_x , p_y and p_z denote the components of each position $p \in P$. Further, packets have a payload. This is modeled using the number of flits transmitted $l \in L = \mathbb{N}$. Together, the set of packets is given by $D = P \times P \times L$. Packets are transmitted from a current (source) position to a destination position. (Please note, that the current position describes the location of a packet during transmission, which does change. It does not refer to the position, at which the packet was injected.) This yields the definition of the transmission distance.

Definition 9.4 (Horizontal transmission distance [JM 1]). Let π be a packet with $\pi = (p_1, p_2, l)$, with source node p_1 , destination node p_2 and l flits. The horizontal transmission distance $s(\pi)$ is defined as the distance between source and destination positions in x - and y -dimension:

$$s(\pi) = \left\| (p_{1,x}, p_{1,y}) - (p_{2,x}, p_{2,y}) \right\| \quad (63)$$

¹ Please note that “measured in $[m]$ ” refers to SI-unit meter; “[m]” refers to the set $\{1, \dots, m\}$. Thereby we avoid ambiguity.

For example, in a mesh topology, the distance between source and destination position in x - and y -dimension of a packet $\pi = (p_1, p_2, l)$ is calculated by $s(\pi) := \|(p_{1,x}, p_{1,y}) - (p_{2,x}, p_{2,y})\|_1$. The norm $\|\cdot\|_1$ denotes the Manhattan norm.²

The *latency* of a packet is calculated by the accumulating the latency of routers on the packet path. This is shown in Figure 72. Each router requires $\delta(\xi)$ clock cycles to process the head flit in the layer $\xi \in [\ell]$. After this initial delay, one flit is sent per clock cycle until the packet is fully transmitted. If a single packet with l flits passes through a single router, the transmission will be finished after $\delta(\xi) + l$ cycles. Let the constant $\rho(\xi)$ be defined as the average distance between routers in the layer ξ . Hence, a packet traverses $s(\pi)/\rho(\xi) + 1$ routers including the destination router during its transmission. This yields the horizontal packet latency and throughput:

Definition 9.5 (Horizontal packet head latency under zero load [JM 1]). Let π be a packet with $\pi = (p_1, p_2, l)$ and $\xi \in [\ell]$ a layer. The average distance between routers in the layer ξ is $\rho(\xi)$ and the delay for processing head flits per router is $\delta(\xi)$. The clock delay of routers is $\text{clk}(\xi)$, measured in [s]. The *horizontal packet head latency under zero load*, measured in [s], in layer ξ is

$$\Delta_H(\pi, \xi) = \left(\frac{s(\pi)}{\rho(\xi)} + 1 \right) \delta(\xi) \text{clk}(\xi). \tag{64}$$

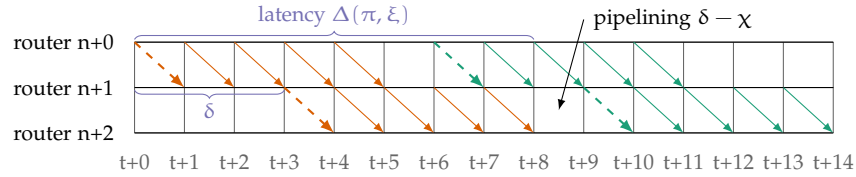


Figure 72: Horizontal communication of two consecutive packets (orange, green). Routers have delay $\delta = 3$ and pipelining $\chi = 2$. The latency $\Delta(\pi, \xi)$ is the difference between head flit send and receive time [JM 1].

Definition 9.6 (Horizontal router throughput [JM 1]). Let π be a packet with $\pi = (p_1, p_2, l)$ and $\xi \in [\ell]$ a layer. The delay for processing head flits per router is $\delta(\xi)$. The router is pipelined with $\chi(\xi) \in [0, \delta(\xi)]$ steps. The clock delay of routers is $\text{clk}(\xi)$, measured in [s]. The *horizontal router throughput*, measured in [flits/s], is given by the number of flits that a router can pass in a period:

$$\hat{\Delta}_H(\pi, \xi) = \frac{l}{(l + \delta(\xi) - \chi(\xi)) \text{clk}(\xi)} \tag{65}$$

² $\|p\|_1 = \sum_{i=1}^n p_n$ for $p \in \mathbb{R}^n$.

9.3.2 Vertical communication

While horizontal communication is essentially homogeneous, vertical communication may be effected by heterogeneity, especially by deviations in clock speed between routers. Therefore, the model must comprise non purely synchronous communication, but also different router and link architectures (such as the mesosynchronous router in Section 9.6).

Definition 9.7 (Vertical packet head latency under zero load [JM 1]). Let π be a packet with $\pi = (p_1, p_2, l)$ and ξ and $\lambda \in [\ell]$ layers with $p_{1_z} = \xi$ and $p_{2_z} = \lambda$. Without loss of generality, assume that $\xi \leq \lambda$ and that slower layers are above faster layers. The clock delay of routers is $\text{clk}(i)$ for all layers $i \in [\ell]$, measured in [s]. The *vertical packet head latency under zero load (downwards)*, measured in [s], is given by the delay each router adds during head flit processing

$$\Delta_V^\downarrow(\pi, \xi, \lambda) = \sum_{i=\xi}^{\lambda} \delta(i) \text{clk}(i). \quad (66)$$

The *vertical packet head latency under zero load (upwards)*, measured in [s], is given by the delay each router adds during head flit processing plus a clock cycle for synchronization as illustrated in Figure 73. This occurs only once during the path of the packet, since only two types of technology nodes are combined, for the typical scenario for heterogeneous 3D SoCs as considered here. Thus, the overhead for synchronization only occurs once. The slower clock frequency dominates.

$$\Delta_V^\uparrow(\pi, \xi, \lambda) = \sum_{i=\lambda}^{\xi} \delta(i) \text{clk}(i) + \text{clk}(\xi). \quad (67)$$

Definition 9.8 (Vertical router throughput [JM 1]). Let π be a packet with $\pi = (p_1, p_2, l)$ and ξ and $\lambda \in [\ell]$ layers with $p_{1_z} = \xi$ and $p_{2_z} = \lambda$. Without loss of generality, assume that $\xi \leq \lambda$. Routers are pipelined with $\chi(i) \in [0, \delta(\xi)]$ steps in each layer $i \in [\ell]$. The clock delay of routers is $\text{clk}(i)$, measured in [s]. The horizontal throughput of routers in any layer i is $\hat{\Delta}(\pi, i)$. The *vertical router throughput*, measured in [flits/s], is given by the slowest router:

$$\hat{\Delta}_V(\pi, \xi, \lambda) = \min_{i \in [\xi, \dots, \lambda]} \{ \hat{\Delta}(\pi, i) \} \quad (68)$$

Long delays for processing a head flit are not relevant in the case of pipelining. Figure 73 demonstrates that the slowest clock dominates the throughput of the transmission for asynchronous chips.

9.4 LIMITATIONS OF ROUTING DUE TO HETEROGENEITY

The main limiting factor for conventional routing of packets in heterogeneous 3D SoCs is the varying clock speed of routers in dis-

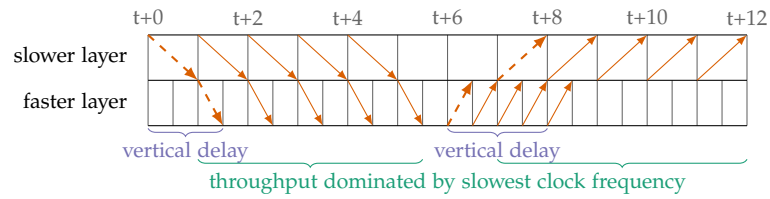


Figure 73: Vertical communication is dominated by the slowest clock frequency in a GALS model. Routers are clocked at a delay of 1 and 1/2. Routers have head delay $\delta = 0$ and pipelining $\chi = 0$ [JM 1].

parate technologies. This impedes latency; this issue can only be overcome by novel routing algorithms, which reduce the length of parts of the packet path with long latency. Heterogeneity also will impede throughput if interaction between routers is not purely synchronous. This issue can only be overcome by novel router architectures, which increase the throughput between layers in disparate technologies. Since efficient routing algorithms must provide low latency and high throughput, there is an essential need for a co-design of router architectures and routing algorithms in heterogeneous 3D SoCs.

9.4.1 Tackling latency limitations via novel routing algorithms

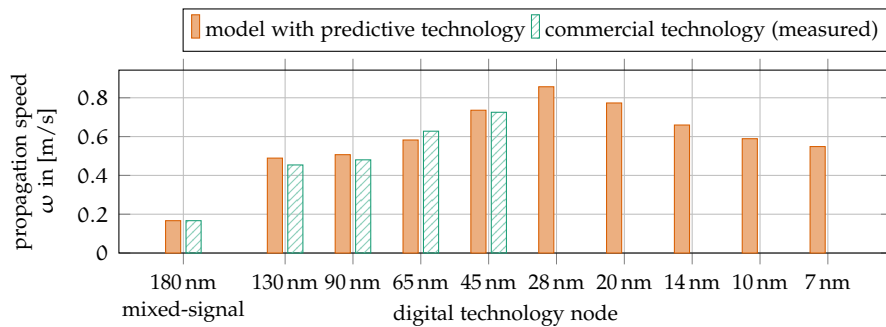


Figure 74: Propagation speed ω based on our experiments (green) for commercial 180 nm mixed-signal and 130 nm – 45 nm digital technology using the synthesis results for our NoC router with a head flit delay of $\delta = 3$ and a 2×2 NoC in the mixed-signal layer and based on the model (orange) for 180 nm – 7 nm predictive technology [JM 1].

Depending on the manufacturing technology of a layer, the communication speed differs. Routing algorithms can exploit this to reduce the limitations of heterogeneity. This requires identification of faster layers. In more advanced layers, routers are clocked faster, which increases the speed. But there are also more routers in these layers, since their individual size shrinks, which add delays to transmission. These two effects must be considered simultaneously to calculate the transmission speed. We use Equations 63, and 64, with derivation, which

yield the propagation speed of a packet under zero load in horizontal direction.

Definition 9.9 (Propagation speed [JM 1]). Let $\xi \in [\ell]$ be a layer. The propagation speed in layer ξ is

$$\omega(\xi) = \frac{\rho(\xi)}{\delta(\xi)\text{clk}(\xi)} \quad (69)$$

measured in [m/s]. It can be obtained by considering any packet π with $\pi = (p_1, p_2, l)$ with distance $s(\pi)$. The speed is distance per time, i.e. $\omega(\xi) = \frac{s(\pi)}{\Delta_H(\pi, \xi)}$.

The propagation speed ω is shown in Figure 74. We use a commercial 180 nm general purpose mixed-signal node and 130 nm – 45 nm general purpose digital nodes. The results are based on synthesis for a standard NoC router with a head flit delay of $\delta = 3$. The NoC in the mixed-signal layer has 2×2 routers; in the digital layer, there are larger, scaled NoCs. These transmission speeds are depicted in green. Comparing mixed-signal and digital technologies, we observe a propagation speed improvement of between $2.7 \times$ and $4.3 \times$. The effect of clock scaling is stronger than the effect of area scaling throughout all technologies, in our experiments.

For a further evaluation, we use the proposed models and fit synthesis results (cf. Section 9.7.1). This enables prediction of propagation speeds of layers manufactured in nodes with a feature size below 45 nm, which are not available to academia. We set the maximum clock frequency of the delay model to 5 GHz. We observe a maximum speed improvement of $5.1 \times$ for 28 nm. For smaller nodes, it is reduced to $3.3 \times$ as a result of the limit in clock speed. Summing up, the clock frequency scaling remains dominant over area scaling for all nodes, but its advantage declines. Hence, smaller nodes are faster than mixed-signal nodes, in general.

9.4.2 Tackling throughput limitations via novel router microarchitectures

We motivated in the last paragraph, that we can reduce communication latency by routing algorithms, which use paths through faster layers. Here we demonstrate that routing algorithms cannot increase throughput; the throughput limitation due to heterogeneity can only be tackled by means of router architectures. We consider packets with length l . According to Equation 65, the throughput of horizontal communication is $\hat{\Delta}_H = \frac{1}{\text{clk}(\xi)}$; it is determined by the layer's clock frequency. If communication spans layers in another technology (i.e. layer with another clock frequency), Equation 68 will yield the vertical throughput:

$$\hat{\Delta}(\pi, \lambda) = \min\{\hat{\Delta}_V(\pi, \xi, \lambda), \hat{\Delta}(\pi, \lambda)\} = \hat{\Delta}_V(\pi, \xi, \lambda) \leq \frac{1}{\text{clk}(\xi)} \quad (70)$$

This shows that the throughput on paths, which span multiple, heterogeneous layers, is limited by the slowest clock frequency. In other words, the chain is only as strong as its weakest link. This limitation is universal for routing in heterogeneous 3D SoCs; thus, “*communication may not span slower clocked layers if high throughput is required*” [JM 1]. Due to this principle, increases in transmission speed are still limited by reduced throughput. Even worse, packets from and to slower layers are inevitably throughout limited. Therefore, horizontal transmission in slower layers must be reduced to a minimum. Further, novel router architectures are required, which tackle the throughput limitations for packets, which originate from or are designated to a slower layer. In the next sections, we consider both novel routing algorithms (Section 9.5) and novel router architectures (Section 9.6) to tackle both throughput and latency limits.

9.5 TACKLING LATENCY: ROUTING ALGORITHMS

Conventional XYZ routing is not applicable to heterogeneous 3D SoCs because the routing algorithm is not connected. This was already introduced in Section 5.4. Using the models, this can also be proven as a general principle due to heterogeneity: Not all routers in digital layers are connected to a router in mixed-signal layers, since the number of routers differ. The routing algorithm is not connected: packets from a node in the mixed-signal layer to a destination in a digital layer cannot be routed if the destination router is not connected upwards. This is shown in Figure 75, orange path. The closest variation of conventional XYZ routing for heterogeneous 3D SoCs is also shown in Figure 75. We call this routing *heterogeneous XYZ*; packets are routed to the digital layers via the vertical connection, which is closest on the path to the destination. We do not prove livelock and deadlock freedom for this routing method, since the proof is merely rearranging the arguments from Section 9.5.5. Naturally, heterogeneous XYZ performs worse in every metric in comparison to the proposed novel routing algorithms and is only used as a baseline for comparison.

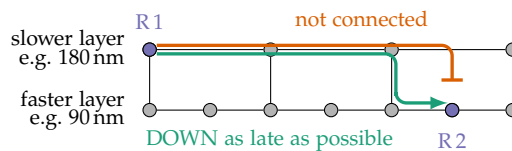


Figure 75: Sectional drawing of a 3D NoC with two layers in disparate technologies: conventional XYZ routing is not connected (orange, path from R1 to R2). The closest variation routes down as late as possible (green) [JM 1].

9.5.1 Principles for routing algorithms in heterogeneous 3D SoCs

As discussed, transmission through different layers can reduce the latency. This can be exploited by the following two paradigms for routing algorithms:

- "Stay in faster layers!": Packets should stay in layers with high propagation speed as long as possible. This is exemplified in Figure 76 using a sectional drawing of a two-layered chip, with two disparate technologies and $s_f = 4$. Usually, data transmitted from routers R1 to R2 stay in the upper layer until reaching the router above R2 (path in orange color). This yields higher latency than the way back via the lower layer in the more advanced technology node. Thus, it is favorable to route all packets via the preferred path, depicted in green.

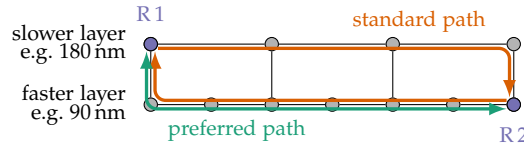


Figure 76: "Stay in faster layers!": A sectional drawing of the 3D NoC with routers plotted in gray and links in black. The green paths are faster than the orange paths [JM 1].

- "Go through faster layers!" If the performance gain is large enough, packets can be routed via adjacent, faster layers (even though this is a detour). Figure 77 exemplifies this with a sectional drawing of a two-layered chip, with two technologies and $s_f = 4$. The routers R1 and R2 are communicating. Usually, data is transmitted via the upper layer, which is slower than the lower layer. Therefore, it is favorable to route packets via the green path.

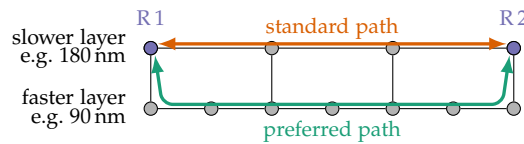


Figure 77: "Go through faster layers!": A sectional drawing of the 3D NoC with routers plotted in gray and links in black. The orange path between R1 and R2 is longer yet faster than the green path [JM 1].

We apply the two aforementioned paradigms and propose two exemplary routing algorithms. The proposed models provide relevant information to set parameters of the algorithms and assess, under which (technological) circumstances the routing algorithms are applicable, since the models are generally valid, i.e. can be applied to any

topology and set of technology parameters (beyond the proposed algorithms and the setting).

9.5.2 Preliminary considerations for the routing algorithms

9.5.2.1 Setting

A heterogeneous 3D SoC with $\ell \in \mathbb{N}$ layers is used. Its layers are ordered by technology node, as in the vast majority of works on 3D SoCs, e.g. [1]. The most coarse-grained technology is at the top whilst the most fine-grained technology is bottommost. Reordering the layers does not influence the models and principles and only requires minor changes to the proposed routing algorithms; hence, this does not lead to a loss of generality. But the order reduces the complexity of descriptions. Our approach is applicable to scenarios without ordered layers, with minor modifications.

A 3D NoC is implemented in the heterogeneous 3D SoC. As network topology, each layer has a grid with m_ξ rows and n_ξ columns, wherein $\xi \in [\ell]$ is the layer index; routers are arranged in rows and columns. Therein, neighboring routers connect horizontally which yields a m_ξ - n_ξ -mesh topology. We do not model long range links or express virtual channels; thus, routers only have one link in the same direction. All routers, except those on the bottommost layer, have a (bidirectional) vertical link to the adjacent router in the next lower layer. This is possible due to the ordering of layers (cf. Figure 78). The set of routers V is also the vertex set of the network digraph $T = (V, A)$.³ The set of arcs A contains the directed links between routers.

9.5.2.2 Addresses in the network

The coordinate system of the SoC is shown in Figure 79, which models location of routers. The origin of the coordinates is the SoC's top left corner. Calculation of routing algorithms must be efficient, thus using the (physical) locations in the 3D SoC is not realistic. We introduce row and column and layer number for routers as a solution. Rows and columns are based on the network digraph and not the physical locations: For example, pairs of neighbored routers in adjacent layers do not necessarily have the same physical x- and y-coordinate but the same column and row number. This is shown in Figure 78. We do not depict this in all figures for sake of simplicity. If routers are stacked, such as in Figure 76, this will model a physical placement comparable to Figure 78, in which routers are not necessarily stacked. Connecting these routers is done using redistribution. We notate row, column, and layer of each router as $w = (w_x, w_y, w_z)$ for

³ In *Duato* [99] the network digraph is called *interconnection network*.

$w \in W = \mathbb{N}^3$, which is also the network address space. An injective function

$$m : W \rightarrow P \tag{71}$$

converts addresses to locations of routers. Packets with source and destination address are given by $\tilde{D} = W \times W \times L$.

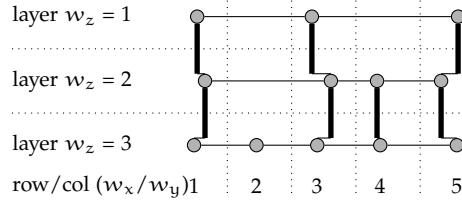


Figure 78: Layers are ordered by technology. Routers in same row and column must not be at the same physical location in their layer. Links connect routers via redistribution and TSV arrays (bold) [JM 1].

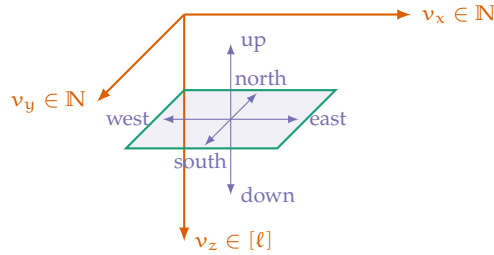


Figure 79: Cardinal directions in model coordinates W [JM 1].

9.5.2.3 Cardinal Directions

We use the six cardinal directions $C := \{\text{north, east, south, west, up, down}\}$ to sort the arcs, i.e. links in the network, as shown in Figure 79. We define functions which return the set of all links in one of these cardinal directions. These are given for all links $(v, w) \in A$, as published in [JM 1]:

$$\begin{aligned} (v, w) \in \text{north}(A) &\Leftrightarrow v_x = w_x, v_y > w_y, v_z = w_z \\ (v, w) \in \text{east}(A) &\Leftrightarrow v_x < w_x, v_y = w_y, v_z = w_z \\ (v, w) \in \text{south}(A) &\Leftrightarrow v_x = w_x, v_y < w_y, v_z = w_z \\ (v, w) \in \text{west}(A) &\Leftrightarrow v_x > w_x, v_y = w_y, v_z = w_z \\ (v, w) \in \text{up}(A) &\Leftrightarrow v_z > w_z \\ (v, w) \in \text{down}(A) &\Leftrightarrow v_z < w_z \end{aligned}$$

For example, $\text{north}(A)$ contains all links which point to north. We further introduce functions that will return neighbors of routers in

a certain cardinal direction, if a link exists.⁴ Routers at the edges of the network do not have links in that direction which is given by the value \emptyset . We define for all $f \in C$:

$$f : V \rightarrow V \cup \{\emptyset\}$$

$$v \mapsto \begin{cases} w & \text{if } (v, w) \in f(A) \\ \emptyset & \text{otherwise.} \end{cases}$$

9.5.3 Applying principle 1: $Z^+(XY)Z^-$ routing algorithm

We apply principle 1, "Stay in faster layers!" and develop a minimal and deterministic routing algorithm, as originally proposed in [JM 1]. Let $\tilde{\pi} = (v, w, l)$ be a packet. If source and destination are in different layers, i.e. $v_z \neq w_z$, the faster of the two layers is taken for transmission. Therefore, we apply Equation 69 to calculate the average propagation speed at design time. This yields the following rules for transmission of packet π (in router with address v):

- If $\omega(v_z) < \omega(w_z)$, XYZ routing will be applied.
- If $\omega(v_z) > \omega(w_z)$, ZXY routing will be applied.
- If $\omega(v_z) = \omega(w_z)$, either XYZ routing or ZXY routing will be applied, selected at design time, depending on other network properties such as energy consumption of routers.

We call this routing algorithm $Z^+(XY)Z^-$. Since layers are ordered by technology and hence by transmission speed, the implementation as shown in Listing 8 modifies deterministic XYZ simply by reordering if-statements. In a more general case, routers will require a flag storing information if faster layer is located below, above or is indeed this actual layer. The resulting routing is illustrated in Figure 80.

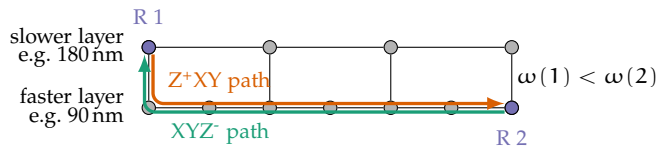


Figure 80: $Z^+(XY)Z^-$ routing: transmission through the lower layer is faster in this NoC with two layers in disparate technologies (sectional drawing) [JM 5].

Definition 9.10 (Routing function R_1 for $Z^+(XY)Z^-$ routing [JM 1]). Let $T = (V, A)$ be the topology digraph with the set of routers V and

⁴ Note, that the above functions will be only well-defined, if no router has more than one link to the same direction.

the set of links A . Further, $\mathcal{P}(A)$ is the power set of A . The routing function $R_1 : V \times V \rightarrow \mathcal{P}(A)$ is defined as:

$$(v, d) \mapsto \begin{cases} \emptyset & \text{for } v = d \\ \{\text{north}(v)\} & \text{for } v_x = d_x, v_y > d_y, v_z \geq d_z \\ \{\text{east}(v)\} & \text{for } v_x < d_x, v_z \geq d_z \\ \{\text{south}(v)\} & \text{for } v_x = d_x, v_y < d_y, v_z \geq d_z \\ \{\text{west}(v)\} & \text{for } v_x > d_x, v_z \geq d_z \\ \{\text{up}(v)\} & \text{for } v_x = d_x, v_y = d_y, v_z > d_z \\ \{\text{down}(v)\} & \text{for } v_z < d_z. \end{cases}$$

Please note that $\{\emptyset\}$ is impossible by construction, since all routers have a downward link in our setting (except those in the bottommost layer). This is formally proven in Lemma 9.3.

The following must be considered for minimality for this routing algorithm: Minimality refers to the shortest path in the interconnection network. Our routing algorithm will be minimal if links in the interconnection graph are weighted with their speed. In terms of hop distance (which are usually used to assess minimality) the proposed routing algorithm is not minimal.

9.5.4 Applying principle 2: ZXYZ routing algorithm

We apply principle 2, “Go through faster layers!”. Packets will be transmitted via detours if these paths are faster. Since there is an overhead when routing to the fastest layer for vertical transmission, it depends on the position of source and destination, if transmission to a faster layer is actually beneficial. Let $\tilde{\pi} = (v, w, l)$ be a packet with source address v , destination address w and l flits. Let π be the corresponding packet after applying m to convert addresses to locations. The transmission time under zero load in the layer v_z is $\Delta_H(\pi, v_z)$ (Equation 64). Let $\lambda \in [l]$ be another layer, through which the packet could be transmitted. The transmission time detouring via layer λ is the transmission time for traversing vertical links, plus time within layer λ . Applying the model will yield a condition for routing via layer λ , if this path has smaller latency:

$$\Delta_H(\pi, \xi) > \Delta_V^\downarrow(\pi, \xi, \lambda) + \Delta_H(\pi, \lambda) + \Delta_V^\uparrow(\pi, \xi, \lambda) - 2\delta(\lambda)\text{clk}(\lambda) \quad (72)$$

Please note, that delay of the routers at the edges of the vertical and horizontal paths are calculated double and therefore must be subtracted. The threshold distance $\phi(\xi, \lambda)$ determines the minimum distance in layer ξ , for which rerouting via layer λ is faster. In our setting it is not useful to use another than the top digital layer to save vertical transmission time. Non-adjacent layers in mixed-signal nodes have larger thresholds. Equation 72, with $\phi := s(\pi)$ yields

$\left(\frac{\phi\delta(\xi)}{\rho(\xi)} - \rho(\xi) - 1\right) \text{clk}(\xi) = \left(\frac{\phi}{\rho(\lambda)} + 1\right) \delta(\lambda)\text{clk}(\lambda)$. This term is transformed to:

$$\phi(\xi, \lambda) = \begin{cases} \frac{(\delta(\xi)\text{clk}(\xi) + \delta(\lambda)\text{clk}(\lambda) + \text{clk}(\xi))\rho(\xi)\rho(\lambda)}{\delta(\xi)\text{clk}(\xi)\rho(\lambda) - \delta(\lambda)\text{clk}(\lambda)\rho(\xi)} & \text{for } \xi < \lambda \\ \infty & \text{else} \end{cases} \quad (73)$$

Note, that ∞ can be replaced by any value larger the size of the chip. The two routing conditions are: (a) If a λ exists with $s(\pi) = s((m(v), m(w), l)) > \phi(v_z, \lambda)$, vertical routing will be applied in direction of $\arg \min_{\lambda \in [\ell]} \phi(v_z, \lambda)$ (i. e. ZXY routing). (b) If the condition $s(\pi) = s((m(v), m(w), l)) \leq \phi(v_z, \lambda)$ is true for all $\lambda \in [\ell]$, the horizontal paths will be taken and then vertical routing is applied (i. e. XYZ routing). There are two bottlenecks for run-time calculation: First, selection of the best layer by evaluation of $\arg \min$ is too expensive at runtime. Thus, a static layer Λ must be selected at design time. From a practical standpoint, the top digital layer is preferred because it offers high speed and low overhead for vertical transmission. (Without loss of generality, we set $\Lambda := \ell$ in proofs.) Second, locations must be converted into network address space. The location threshold distance ϕ is transformed into a hop distance threshold by division through the average router distance in the digital layers:

$$\Phi(\xi, \Lambda) := \lceil \phi(\xi, \Lambda) / \rho(\ell) \rceil \quad (74)$$

The threshold ϕ is required to be smaller than the outside measurements of the chip so that the routing can be applied. For a combination of a commercial 180 nm mixed-signal node with commercial 130 – 45 nm digital nodes and a 4×4 NoC in the layer in mixed-signal technology, ϕ is between 0.63 and 0.45 for a chip with edge length normalized to 1. Hence, packets traveling more than 2 or 3 hops in the layer in mixed-signal node are routed via the adjacent layer ($\Phi_{130,90 \text{ nm}} = 2$ and $\Phi_{65,45 \text{ nm}} = 3$).

To summarize, the routing algorithm has these conditions for a packet $\tilde{\pi} = (v, w, l)$ in router v :

- If $|v_x - w_x| + |v_y - w_y| \leq \Phi(\xi, \Lambda)$, XYZ routing will be applied.
- If $|v_x - w_x| + |v_y - w_y| > \Phi(\xi, \Lambda)$, the packet will be routed down.

We call this routing ZXYZ. It is illustrated in Figure 81. The algorithm is given in Listing 9. The routing function for ZXYZ is, as originally published in [JM 1]:

Definition 9.11 (Routing function R_2 for ZXYZ [JM 1]). Let $T = (V, A)$ be the topology digraph with the set of routers V and the set of links A . Let Λ be a layer which is selected for rerouting at design time. Let $\Phi(\xi, \Lambda)$ be a threshold for rerouting according to Equation 74. The routing function $R_2 : V \times V \rightarrow \mathcal{P}(A)$ is defined as:

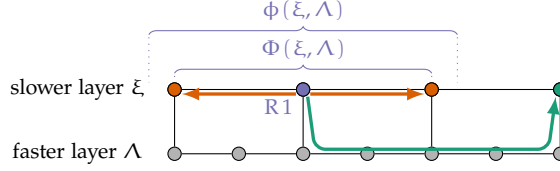


Figure 81: Different paths from router R1 in layer ξ based on a threshold $\Phi(\xi, \lambda)$ to the other routers in its layer; the distance to the destinations in green cases is short, thus XYZ routing. Sectional drawing of a 3D NoC with two layers in disparate technologies; layer ξ is slower than layer λ [JM 5].

$$(v, d) \mapsto \begin{cases} \emptyset & \text{for } v = d \\ \{\text{down}(v)\} & \text{for } |v_x - d_x| + |v_y + d_y| > \Phi(v_z, \Lambda), v_z \geq d_z \\ \{\text{down}(v)\} & \text{for } v_z < d_z. \\ \{\text{north}(v)\} & \text{for } v_x = d_x, v_y > d_y, v_z \geq d_z, |v_y - d_y| \leq \Phi(v_z, \Lambda) \\ \{\text{east}(v)\} & \text{for } v_x < d_x, v_z \geq d_z, |v_x - d_x| + |v_x - d_x| \leq \Phi(v_z, \Lambda) \\ \{\text{south}(v)\} & \text{for } v_x = d_x, v_y < d_y, v_z \geq d_z, |v_x - d_x| \leq \Phi(v_z, \Lambda) \\ \{\text{west}(v)\} & \text{for } v_x > d_x, v_z \geq d_z, |v_x - d_x| + |v_x - d_x| \leq \Phi(v_z, \Lambda) \\ \{\text{up}(v)\} & \text{for } v_x = d_x, v_y = d_y, v_z > d_z \end{cases}$$

Listing 8: $Z^+(XY)Z^-$.

```

1  if  $v_z - d_z < 0$  then
2    route to DOWN
3  else if  $v_x - d_x < 0$  then
4    route to EAST
5  else if  $v_x - d_x > 0$  then
6    route to WEST
7  else if  $v_y - d_y > 0$  then
8    route to NORTH
9  else if  $v_y - d_y < 0$  then
10   route to SOUTH
11 else if  $v_z - d_z > 0$  then
12   route to UP
13 else
14   route to LOCAL
15 end if

```

Listing 9: ZXYZ.

```

1  if  $|v_{x,y} - u_{x,y}| > \Phi(v_z, \Lambda)$  then
2    route to DOWN
3  else if  $v_z - d_z < 0$  then
4    route to DOWN
5  else if  $v_x - d_x < 0$  then
6    route to EAST
7  else if  $v_x - d_x > 0$  then
8    route to WEST
9  else if  $v_y - d_y > 0$  then
10   route to NORTH
11 else if  $v_y - d_y < 0$  then
12   route to SOUTH
13 else if  $v_z - d_z > 0$  then
14   route to UP
15 else
16   route to LOCAL
17 end if

```

9.5.5 Proof of validity: deadlock and livelock freedom

The proof of deadlock and livelock freedom have been originally published in [JM 1]. Due to their unique nature, these are taken directly from this work:

“We prove that the routing algorithms $Z^+(XY)Z^-$ and ZXYZ are free of deadlocks and livelocks. We make use of *Duato’s theorem* [99], ac-

g:		n.	e.	s.	w.	u.	d.
f:	n.	1	0	0	0	1	0
	e.	1	1	1	0	1	0
	s.	0	0	1	0	1	0
	w.	1	0	1	1	1	0
	u.	0	0	0	0	1	0
	d.	1	1	1	1	0	1

Table 15: Possible turns (f, g) in R_1 and R_2 .

ording to which a routing will be deadlock-free if the routing function is connected and the channel dependency graph is cycle free. We also use terms and definitions [99] without further explanation. Among them are *routing function*, *adaptive*, *connected*, *direct dependency*, and *channel dependency graph*. To clarify: If there is a direct dependency from a to b , we also say: » b is direct dependent on a .« Graph related terms like *path*, *closed walk*, or *cycle* are used as defined in [145].

We introduce the terms *possible turn* and *impossible turn* according to a routing function R . These terms will denote, if the routing functions permits consecutive flow of packets in these directions.

Definition 9.12. A pair of cardinal directions $(f, g) \in C \times C$ is called a possible turn according to R , if there exist two consecutive arcs, (u, v) and $(v, w) \in A$, with: $(u, v) \in f(A)$, $(v, w) \in g(A)$ and there is a direct dependency from (u, v) to (v, w) . A pair of cardinal directions that is not a possible turn is called an impossible turn according to R .

Lemma 9.1. *If there is a cycle in the channel dependency graph (CDG), then we can also find a closed walk $(v_1, a_1, v_2, \dots, v_k, a_k, v_1)$ (for $k \in \mathbb{N}$) in the topology digraph with*

- a_{i+1} is direct dependent on a_i for all $i \in \{1, \dots, k-1\}$,
- and a_1 is direct dependent on a_k .

Proof. Assume that there is a cycle $(\{a_1, \dots, a_k\}, \{(a_1, a_2), \dots, (a_{k-1}, a_k), (a_k, a_1)\})$ in the CDG. According to the definition of direct dependency, the destination node of a_i in the topology digraph is also the source node of a_{i+1} (for all $i \in \{1, \dots, k\}$, and $a_{k+1} := a_1$). Let us call this node v_{i+1} (for all $i \in \{1, \dots, k\}$). Then, $(v_{k+1}, a_1, v_2, \dots, v_k, a_k, v_{k+1})$ is a closed walk in the topology digraph. \square

9.5.6 $Z^+(XY)Z^-$: R_1 is deadlock-free

From the definition of R_1 , we determine the impossible turns and the possible turns, as shown in Table 15. Here, we assume that the numbers of rows, columns and layers m_ξ , n_ξ and ℓ are not too small. We assume $m_\xi, n_\xi, \ell \geq 2$ for all $\xi \in \{1, \dots, \ell\}$ as a precaution.

Lemma 9.2. *When R_1 gives a direction, then the necessary link always exists.*

Proof. Places without links in some directions are: (a) At the outside faces of the 3D NoC cuboid links at edges of layers, upward links from the topmost layers and downward links from the bottommost layer do not exist. (b) Some upward links do not exist between layers if one layer is in another technology than the other layer. a) By looking at the definition of R_1 , one can check that every routing step brings the packet nearer to d . Hence, the nonexistent links on the outer border of the 3D-NoC are never taken by R_1 . b) Not every router has a link in direction up. Every router, except those in the bottommost layer, has a down link by premise. Downward links in a router are upward links in the router below: When router v has the same x - and y -coordinates as the destination router d and v is below d , v has an up-link. These are also the conditions for traveling up in R_1 . \square

Lemma 9.3. *R_1 is connected.*

Proof. Let s and d be any two vertices in V . R_1 returns a direction for every vertex except d (it returns \emptyset). The links in the chosen direction always exist (Lemma 9.2). If we apply the routing function step by step and proceed through the network in the returned directions, we will find a route. As shown in the proof of livelock-freedom, the route is not infinite (Theorem 9.3). Hence, it terminates. Termination can only happen at d , by definition. Hence, with the routing function R_1 , we always find a path from s to d . \square

Theorem 9.1. *R_1 is deadlock-free.*

Proof. R_1 is connected, because of Lemma 9.3. Assume, that the CDG of T and R_1 has a cycle. Lemma 9.1 proves that T has a cycle where each two consecutive arcs are direct dependent.

Case 1: All vertexes of the cycle are in the same layer. We know by [104] that XY routing has a cycle free CDG due to impossible turns. Thus, Case 1 does not occur.

Case 2: The vertexes of the cycle are in at least two different layers. Since the vertexes are in different layers, there is at least one arc, which goes up. According to table 15, the only possible direction after »up« is »up« and the cycle could never be closed. Hence, Case 2 is also impossible. We have shown by contradiction that the CDG is cycle-free and apply Duato's Theorem on R_1 . \square

9.5.7 ZXYZ: R_2 is deadlock-free

Again, we can determine the set of possible turns. It can be seen in Table 15.

Lemma 9.4. *R_2 is connected.*

Proof. Let s and d be any two vertexes in V . We construct a path ($s = v_1, \dots, v_k = d$) with $v_i \in V$ for all $i \in [k]$, $k \in \mathbb{N}$ from s to d by using links $(c_1 = (v_1, v_2), \dots, c_{k-1} = (v_{k-1}, v_k))$ with $c_i \in A$ for all $i \in [k-1]$, which are consecutively delivered by the routing function R_2 .

Case 1 (The source is above the destination $s_z < d_z$): As in the proof of Lemma 9.3, the route starts with a sequence of downs until the destination layer is reached. Now the routing goes as explained in Case 2.

Case 2 (The source is below the destination or on the same layer $s_z \geq d_z$): The next links depend on the logical value of $\|s - d\| \geq \Phi(s_z)$.

Case 2.1 ($\|s - d\| \geq \Phi(s_z)$): If the condition is true, the next link will be down. The value of $\|s - d\|$ is the same as $\|v_2 - d\|$. The value of $\Phi(z)$ is the same for all $z < \Lambda$. Hence, layer Λ will be reached via a sequence of downs. The rest of the path is constructed as in Case 2.2.

Case 2.2 ($\|s - d\| < \Phi(s_z)$): Here, R_2 is identical to R_1 . Connectivity is proven in Lemma 9.3. \square

Theorem 9.2. R_2 is deadlock-free.

Proof. The proof is analog to the proof of Theorem 9.1. R_2 is connected because of Lemma 9.4. We assume that the CDG of T and R_2 has a cycle. Then T has a cycle, in which each two consecutive arcs are direct dependent, according to Lemma 9.1.

Case 1: All vertexes of the cycle are in the same layer. Case does not occur, cp. Theorem 9.1, Case 1.

Case 2: The vertexes of the cycle are in at least two different layers. There is at least one arc going up. According to table 15, the only possible direction after »up« is »up«. Thus, the cycle can not be closed. Hence, case 2 is impossible.

We have shown by contradiction that the CDG is cycle-free. We apply Duato's Theorem on R_2 . \square

9.5.8 Livelock freedom

Palesi et al. [166] define that "livelock is a condition where a packet keeps circulating within the network without ever reaching its destination". Hence the following definition.

Definition 9.13 (Livelock-free). A routing algorithm is livelock-free, if every packet has no other choice, but to reach its destination after a finite number of hops.

Remark. A routing algorithm consists of a routing function and a selection. R_1 and R_2 are examples for routing functions. If an adaptive routing function returns more than one link, the selection chooses one. The property *livelock-free* belongs to the routing algorithm. Nevertheless, we call a routing function *livelock-free* if every routing algo-

rithm with this routing function is livelock-free, independent of the selection.

Theorem 9.3. R_1 and R_2 are livelock-free.

Proof. Assume there were two vertexes s and d with the property that the routing R_1 makes infinite steps and never reaches d starting from s (the same arguments hold for R_2). Under this assumption, at least one cardinal direction must be traveled infinite times. We do a case-by-case analysis in which we assume that this applies to the different cardinal directions. We thereby show that it works for none of them. This contradicts the assumption that there could be a livelock.

Case 1: »up« is traveled infinite times. By the definition of R_1 (Definition 9.10), up is only used if $v_x = d_x$ and $v_y = d_y$ and $v_z > d_z$, with v being the current vertex. Traveling up one layer will remain $v_x = d_x$ and $v_y = d_y$ and results either in $v_z = d_z$ or $v_z > d_z$. The only possible direction after »up« is »up«. Since there are only $\ell < \infty$ layers, d will be reached after finite steps. Thus, Case 1 can not occur.

Case 2: »down« is traveled infinite times. Since up can not be traveled infinite times (Case 1), $down$ can not either. It is limited by the number of layers, ℓ , plus the number of times up could be traveled.

Case 3: »east« and »west« are traveled infinite times. Similar to Case 2, infinite steps to $west$ imply infinite steps to $east$ and vice versa. From the definition of R_1 , we know:

- $east$ and $west$ are the only directions, which affect the x -value of v .
- A step to $east$ is only done if $v_x < d_x$
- A step to $west$ is only done if $v_x > d_x$
- A step to $west$ or $east$ is only done if $v_z \geq d_z$.

We never step on a router with $v_x = d_x$. If we reached a router with $v_x = d_x$, $north$ -, $south$ -, up - or $down$ -routing would be done and the destination would be reached. Steps to $east$ or $west$ are only done in the destination layer or below. In these layers, each row has a router at position d_x . Routing from west to east and back without using one of these routers is impossible.

Case 4: »north« and »south« are traveled infinite times. This case is analog to Case 3.

None of the cases occur. Thus, the assumption is wrong. R_1 is livelock-free.

The same arguments hold for R_2 (defined in Definition 9.11). R_2 is livelock-free.

Remark: The proof relies on our special setting. It requires that for u and v with $down(u) = v$ it holds: $up(v) = u$, $u_x = v_x$, and $u_y = v_y$. It also requires the mesh topology in layers." [JM 1] \square

9.6 TACKLING THROUGHPUT: ROUTER ARCHITECTURES

In Section 9.4.2 a fundamental limitation of routing in heterogeneous 3D SoCs was revealed, described by the principle *the chain is as strong as its weakest link*: Throughput is limited by the slowest clock frequency of a router along a path of a packet. The limitation is only found in heterogeneous 3D SoCs, since clock frequencies of routers potentially vary by a large factor. The deviation of clock frequencies in homogeneous 3D or 2D systems is much smaller, so this principle is not relevant for these systems (even for GALS). This limitation can only be tackled by novel router architectures. We propose one possible solution here. We assume an integer relation between the clock frequencies c_f , with a constant phase shift. We propose a co-design of routing architectures and routing algorithms. Therefore, we exploit the finding of this work, that horizontal transmission in slower layers must be minimized. This is guaranteed by the routing algorithms propose in the previous section: If a packet traverses multiple heterogeneous layers, it will always be horizontally transmitted in the fastest layer. Thus, packets are directly forwarded from local ports or incoming upward port to the vertical port in direction of the fastest layer (i. e. down) in slower router. In opposite direction, packets from the downward direction are only transmitted to upward or local ports. We contribute a router architectures, which exploits this by enabling transmission along these paths in the router with multiple flits in parallel. Thus, the same throughput as in the faster layers can be reached between local and vertical ports. This architecture is called *high vertical-throughput router*.

9.6.1 *High vertical-throughput router architecture*

We propose a router architecture for the slower layers, which uses parallelism to achieve a higher throughput between vertical and local ports. The router architecture was developed in close collaboration with *Lennart Bamberg* from the university of Bremen. In the faster layers, the router architecture must not be modified; only the vertical link architecture is changed, as explained in the next section. To achieve the desired parallelism, we exploit that PEs, connected to the router's local port, can provide multiple flits in parallel, because the complete packet is available before transmission.

We modify a conventional input buffered 3D router, with link width of N , as shown in Figure 82. To achieve parallelism, the input buffers of vertical and local links can read up to c_f flits, each N bits wide, simultaneously. The modified buffers are shown in Figure 83. The crossbar also can read single flits or c_f flits, in parallel. Its modifications are shown in Figure 84, which comprise: First, some turns cannot occur in the proposed routing algorithms (e. g. down to north,

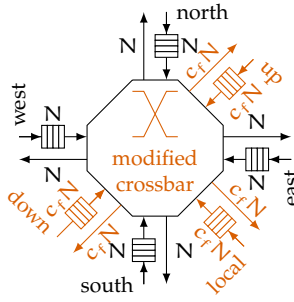


Figure 82: High vertical-throughput router architecture [JM 1].

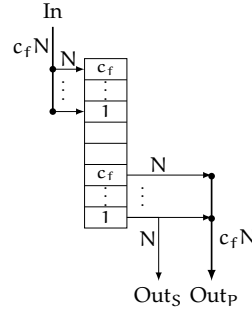


Figure 83: Novel, modified input buffer [JM 1].

east, west or south), which reduces the size of the crossbar. Second, it is extended to route up to c_f flits between local and vertical ports. For other connections, i. e. horizontal routes via the slower layer, the crossbar can switch single flits between horizontal ports, the upward port or the local port. In this case, the remaining $(c_f - 1)N$ lines of the output (to local or up) are zero and a single flit is transmitted, per cycle.

Despite these modifications, the complexity of this router architecture will be reduced in the most common scenario, in which a slower mixed-signal layer is located above and a faster digital layer at the bottom. The routers in the upper layer do not have a port in up direction; therefore, the high throughput path only connects the down and local ports. The modified crossbar with $c_f N$ bits (Figure 84, left-hand side) has two input and output ports. Thus, the local and down ports are directly connected without hardware costs. In comparison to the baseline router, this reduces the router's area costs. Further, only two input buffers are modified for parallelism, which reduces the complexity of the design. The complexity can be further reduced by considering possible paths in the routing algorithms: Single flits will not travel from horizontal to vertical ports. Therefore, the complexity of the crossbar is further decreased.

9.6.2 Pseudo-mesochronous high-throughput link

The architecture of the vertical links is modified for the increased throughput. We propose two designs: The first option targets future technologies with increased TSV yield. The second option is already implementable at the time of writing this thesis, yet its structure is more delicate. Both link architectures require that a circuitry in the mixed-signal layers is driven by clocks from a layer in a digital node.

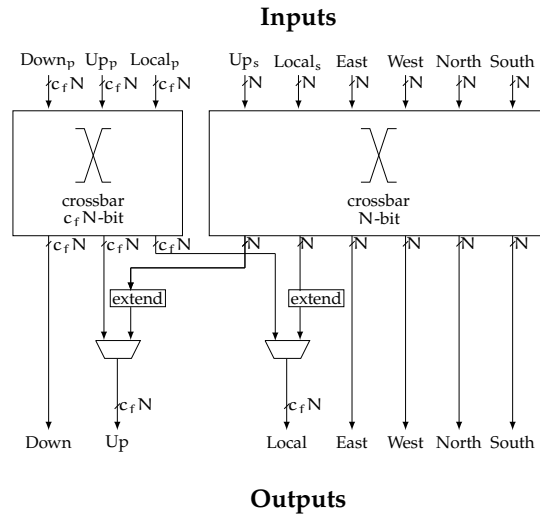
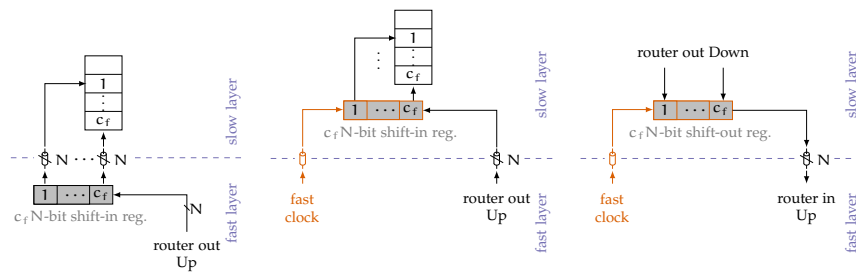


Figure 84: Modified crossbar which allows routing c_f flits between the local and the vertical ports [JM 1].

9.6.2.1 Implementation for future technologies

A large TSV array transmits $c_f N$ bits in parallel in this implementation. TSV yield is rather low at the time of writing this thesis. Thus, this architecture can not be manufactured, practically, and targets future technologies with higher yield. Before transmission, the data are paralleled in the faster layer by a shift register. From slower to faster layer, data is transmitted in parallel using the slower clock frequency via the large TSV array. The data are stored in the input buffers, which is modified to be able to fetch $c_f N$ bit in parallel at the slower clock frequency. The link architecture is shown in Figure 85a.



(a) Upward, with large TSV array. (b) Upward, with small TSV array and shift register. (c) Downward, with small TSV array and shift register.

Figure 85: High-throughput connection [JM 1].

9.6.2.2 Implementation for today's technologies

The previous solution requires higher TSV yield than available. To tackle this, we also propose a more delicate solution for today's state

of the art. We propose to clock parts of the logic in the slower layer at the faster clock speed. The clock is transmitted via a clock TSV from the faster layer. This architecture is shown in Figures 85b and 85c. The shift registers in the slower layer are triggered by the faster clock.

For the upward path (Figure 85b), a shift register is filled with data from the faster layer at its clock speed. Then, the flits are transmitted in parallel to the input buffers of the router in the upper layer at the slower clock frequency. Alternatively, the input buffers in the upper layer can be directly clocked at a higher rate by the faster clock, which removes the shift register's costs.

For the downward path (Figure 85c), in which data are sent from a slower to a faster layer, the shift register is loaded with data in parallel with up to c_f flits and at the slower clock rate. Then, the data is shifted out right, at the faster clock speed. Each flit is transmitted via an N-bit TSV array to the input buffers in the faster layer.

9.7 RESULTS

9.7.1 Model fitting

As published in [JM 1], we evaluate the accuracy of the model fit for the proposed area and timing models to synthesis results of a 3D NoC router. This router has two VCs, four flit deep input buffer per channel, credit based flow control, uses wormhole switching with decentralized arbiters. A deterministic XYZ routing algorithm is implemented. The synthesis is conducted with Synopsis design compiler for commercial 180 nm mixed-signal technology and commercial 45 – 130 nm digital technology. To exemplify potential differences, we use both general purpose (GP) and ULV mixed-signal technology for each structure size.

We show synthesis results and model fit for the *area model*, i. e. the area scaling factor, in Figure 86. Mathematica 10 [167] is used for curve fitting. As reported in [JM 1], a non-ideality factor $\alpha = 3462.7$ and an offset of $\hat{\alpha} = 29.8$ for 180 nm GP technology with a root mean square error (RMSE) of 0.1286 is calculated for our example. ULV technology yields $\alpha = 13.2$ and an offset of $\hat{\alpha} = 0.124$ with a RMSE of 0.1414.

We show synthesis results and model fit for the *timing model*, i. e. the clock scaling factor, in Figure 87. We use a predicted maximum achievable clock frequency of 5.0 GHz in this example.⁵ Mathematica 10 is used for curve fitting. The results for GP nodes are, as reported in [JM 1], $\beta = 32.85$, $\hat{\beta} = 7.88$, $\tilde{\beta} = 0.76$, and $\bar{\beta} = 1.26$ with a RMSE of 0.30. For ULV nodes, the model yields the parameters $\beta = 77.45$, $\hat{\beta} = 2.48$, $\tilde{\beta} = 0.76$ and $\bar{\beta} = 2.77$, with a RMSE of 0.71.

⁵ We must set β in the model, instead of fitting it, since commercial technology nodes below 45 nm are not available to academia.

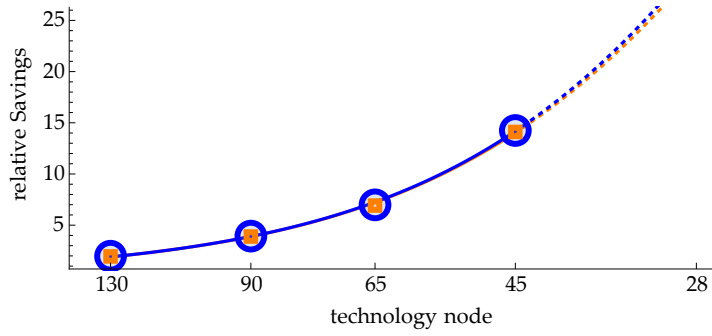


Figure 86: Area comparison of commercial 180 nm mixed-signal GP (blue) and ULV (orange) node with 45 – 130 nm digital GP nodes for synthesized router and model fit [JM 1].

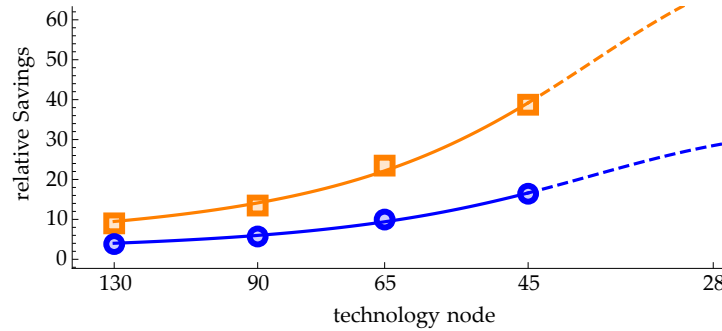


Figure 87: Timing comparison of commercial 180 nm mixed-signal GP node (blue) and ULV node (orange) with 45 – 130 nm ULV digital nodes for a synthesized NoC router and model fit with predictive 5 GHz maximum achievable clock speed [JM 1].

9.7.2 Latency of routing algorithms

We evaluate the performance of the routing algorithms by using the models to calculate the latency.

9.7.2.1 Latency of $Z^+(XY)Z^-$

The latency of packets from nodes in the mixed-signal layers to nodes in the digital layers is reduced by $Z^+(XY)Z^-$. We compare to the latency under zero load for conventional XYZ for a 3D SoC with two layers. In the upper layer, a 4×4 NoC is implemented in commercial 180 nm mixed-signal technology. In the lower layer, a NoC with more nodes according to the area model and the synthesis results (Equation 61) in one commercial 130 nm – 45 nm digital node is used. The latency speedup is calculated using Δ_H from Equation 64 and it is determined via simulations using the proposed NoC simulator (Section 7.2) with 16 flit deep buffer, wormhole routers and four VCs. The results, as originally presented in [JM 1] are shown in Figure 88 for all available hop distances in the layer in mixed-signal technology. Model and simulation yield identical results, as expected, since the

model is accurate under zero load. The results show a speedup between $1.5\times$ and $6.5\times$. The speedup will be larger if the mixed-signal is accompanied by a more advanced digital node. This is also consistent with the expectations (cf. Section 9.4). It is noteworthy, that $Z^+(XY)Z^-$, and thus the speedup, does not impose implementation costs (cf. Section 9.7.4).

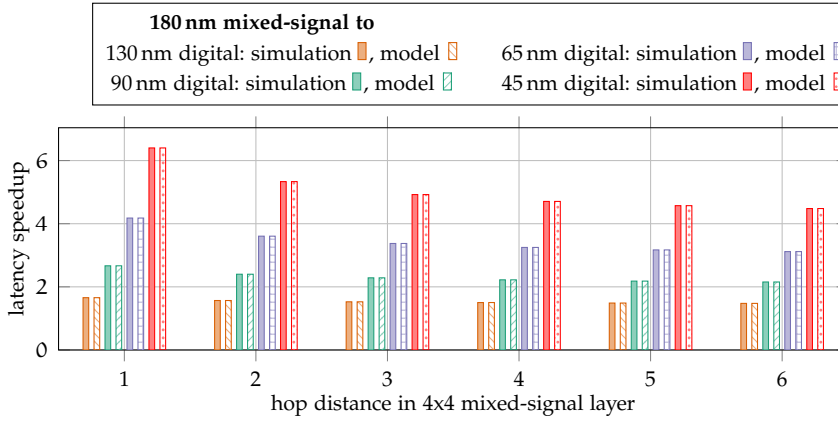


Figure 88: Latency speedup of $Z^+(XY)Z^-$ to conventional XYZ in simulations and model for packets from any node in a 4x4 NoC in 180 nm layer in commercial mixed-signal technology to any node in layers in different 130 nm – 45 nm commercial digital node [JM 1].

9.7.2.2 Latency of ZXYZ

The latency of packets from nodes in the mixed-signal layers to nodes in the mixed-signal layers is reduced by ZXYZ. Again, we compare to the latency under zero load for conventional XYZ for a 3D SoC with the same setting as in the previous evaluation of $Z^+(XY)Z^-$: In the upper layer, a 4×4 NoC is implemented in commercial 180 nm mixed-signal technology. In the lower layer, a NoC with more nodes according to the area model and the synthesis results (Equation 61) in one commercial 130 nm – 45 nm digital node is used. The latency speedup is calculated using Δ_V from Equations 64, 66 and 67 and it is determined via simulations using the proposed NoC simulator (Section 7.2) with 16 flit deep buffer, wormhole routers and four VCs. The results, as originally presented in [JM 1], are shown in Figure 89 for all available hop distances in the layer in mixed-signal technology. The latency speedup is between $0.54\times$ and $1.79\times$. Please note that the speedup is achieved at negligible hardware costs (cf. Section 9.7.4).

9.7.3 Throughput of high vertical-throughput router

The throughput for packets will be increased if a slower layer is part of the path of the packet. It will be increased up to the throughput

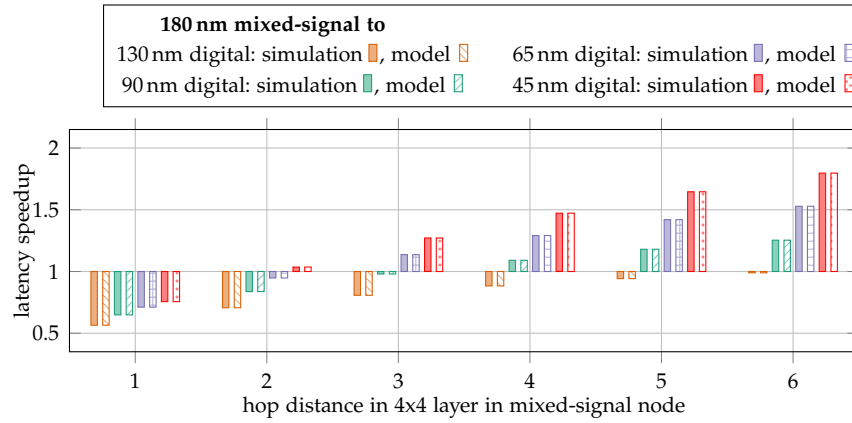


Figure 89: Latency speedup of ZXYZ to conventional XYZ for packets from any node to any node in a 4x4 NoC in 180 nm layer in commercial mixed-signal technology with layers in different 130 nm – 45 nm commercial digital node from simulations and model [JM 1].

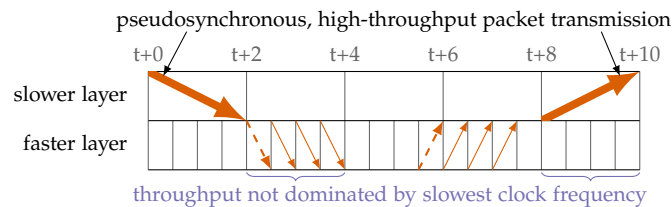


Figure 90: Throughput of high vertical-throughput router [JM 1].

in the faster layer if area for links and routers is expendable. For a transmission from a slower to a faster layer, the slower clock frequency does not dominate the throughput anymore, since the packet is transmitted at once as a whole. This is shown in Figure 90, on the left-hand side. For a transmission from a faster to a slower layer, the slower clock frequency does not dominate the throughput anymore, as well, since the packet is available at the faster router as a whole. This is shown in Figure 90, on the right-hand side.

9.7.4 Area costs of proposed router architecture and routing algorithms

As originally published in [JM 1], we evaluate the area costs of the proposed *high vertical-throughput router* using $Z^+(XY)Z^-/ZXYZ$ routing in a commercial 180 nm ULV mixed-signal technology. We assume a $4 \times 4 \times 2$ NoC (The mixed-signal layer is accompanied by one digital layer.). As baseline, we also synthesize a standard router using conventional XYZ routing, which has also an optimized crossbar including only possible turns. The flit width in all routers is 16 b, input buffers are four flit deep. The routers implement credit-based flow

control. VCs are only used in digital layers. Both the baseline, standard router and the high vertical-throughput router support a maximum frequency of 150 MHz, since the routing algorithm is not part of the critical path.

$Z^+(XY)Z^-$ routing has 0% overhead compared to conventional XYZ routing. ZXYZ routing increases the area by three gate equivalents from 18 to 21 gate equivalents, which affects the whole router area by less than 0.01%. Thus, the overhead for both routing algorithms is negligible.

The costs of the high vertical-throughput router depends on the clock frequency difference between routers in mixed-signal and digital technology. As published in [JM 1], for a clock frequency in the digital layer of 300 MHz ($c_f=2$), the silicon area required for the crossbar and the routers reduces by -7.29%. For routers clocked at 600 MHz ($c_f=4$), the area increases by 4.36%. A higher scaling factor is not possible for an input buffer depth of 4. The complete area increase of the router also depends on the implementation of the switch arbitration, i. e. centralized or decentralized; it is typically between 3% and 4%. Summing up, the router area will be reduced, if the throughput is doubled. The router area slightly increases for a speed up of 4.

9.7.5 Case Study

We analyze our approach for a 3D VSoC based on [11] with four layers as shown in Figure 91: The first layer is a sensing die, implementing a 180 nm CIS (CMOS Imaging Sensor). The second layer implements nine analog digital converters (ADCs) and three analog accelerators [74] in 180 nm mixed-signal node. The third layer implements 6 processors and 6 SIMD (single instruction multiple data) acceleration units in 90 nm digital node. In the fourth layer there are 12 processor cores in 90 nm digital node. The first and second layer are connected via point-to-point links. The second, third and fourth layer are connected via a 3D NoC with 32 b wide links, 8 flit deep buffers and 4 VCs. Packets are 32 flits long. Routers in the digital layer are clocked at 1 GHz and in the mixed-signal layer at 0.5 GHz.

The 3D VSoC implements an image processing pipeline for face recognition. The image sensor records at 720p. The ADCs send the digital raw image to the processors in the third layer, which apply Bayer filter. Then, the SIMD units reduce the resolution by a factor of 4 to increase feature extraction speed. The result is transmitted to the analog accelerators in the second layer, which extract features using Viola-Jones algorithm [161]. The resulting region of interest is transmitted to the fourth layer, in which the processors execute Shi and Tomasi algorithm [162] to find features to track and Kande-Lucas-Tomasi algorithm [163] tracks them. Work is split up equally among the available resources in each step.

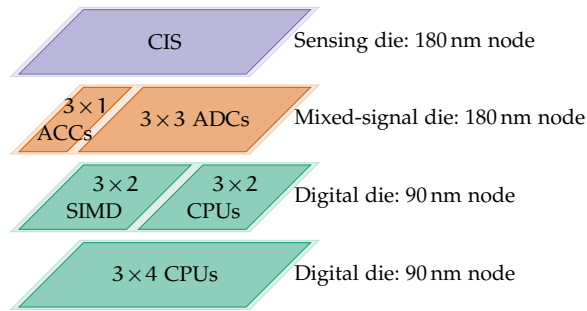


Figure 91: 3D VSoC case study based on [11]. The CIS directly connects to ADCs. The other components are connected by a NoC, with a $3 \times 3 \times 4$ mesh.

We simulate the VSoC's NoC using the described application traffic. Thereby, we compare $Z^+(XY)Z^-$ and $ZXYZ$ with conventional XYZ routing. We simulate 3M clock cycles in the digital layers and 1.5M in the mixed-signal layer. We measure the average flit latency as 145.91 ns for conventional routing and as 64.46 ns for the proposed routing. This equates to a speedup of $2.26\times$. Using the models, we calculate a theoretical speedup of $2.28\times$ under zero load. Average delay for whole packets is reduced from 229.23 ns to 123.07 ns, which is a speedup of $1.86\times$.

9.8 DISCUSSION

9.8.1 Model accuracy

The models assess the impact of heterogeneous integration on NoCs. Figures 86 and 87 show the area and timing model for an exemplary NoC router. Both models have a very good fit to technologies, which are available to academia. As a result of the physical foundation of the area model, it has small RMSEs. The model only covers a constant and a cubic term; it was not beneficial to add a linear term, due to increased RMSE. The timing model is empirical, and thus yields a less accurate fit than the area model, which is demonstrated by increased RMSEs. Also, the model converges to the target maximum clock frequency, as desired. If more modern technology nodes were available, either a better model with a physical foundation could be found or the fit of our model could be improved.

As demonstrated by the small RMSEs, the accuracy of the proposed area and timing model are sufficient to evaluate the influence of heterogeneity. We further evaluate their expressiveness in terms of routing. As shown in Figure 74, we use data from the model fits to calculate the propagation speed ω for predictive technologies. Comparing predictive technology to the synthesis results for 180 nm commercial mixed-signal and 130 nm – 45 nm commercial digital technologies

yields an accuracy of between 1.4% and 7.8%. This small error further supports the validity of the proposed models.

In terms of communication, the accuracy of the model for transmission latency is shown in Figures 88 and 89. Model and simulation yield identical results; the communication models are precise under zero load. Thus, there is no urgent need to model the behavior under load to develop routing for NoCs targeting heterogeneous 3D SoCs.

9.8.2 Implementation

We proposed exemplary routing algorithms and router architectures for heterogeneous 3D SoCs. Their implementation is evaluated. We are aiming at reductions of the negative effects of heterogeneity, i. e. limitations in throughput and latency, at as low area costs as possible. The limitations of heterogeneity are more severe for larger differences between mixed-signal and purely digital technology. The worst case, available here, is a combination of 180 nm commercial mixed-signal technology and 45 nm commercial digital technology, which we assess. Please note, that the results are valid for any other combination of technology nodes with similar relative technology scaling factor Ξ . The baseline for comparison is a conventional, homogeneous NoC, with dimension order routing.

The routing algorithms $Z^+(XY)Z^-$ and ZXYZ provide up to $6.5\times$ latency reductions for packets from routers in the mixed-signal nodes to routers in the digital layer and up to $1.79\times$ latency reductions for packets within the layer in the mixed-signal node. This is shown in Figures 88 and 89. For ZXYZ, there is a performance penalty for distances below Φ (Equation 74) of up to 45%, as expected (see Figure 89, left-hand side); therefore, for small distances, the packets are routed via the mixed-signal layer. The threshold distance shrinks for more advanced technology nodes and larger distances, which is also expected. It is noteworthy, that conventional XYZ outperforms ZXYZ for low technology differences, such as 180 nm to 130 nm, for all distances.

The vertical high-throughput router offers increased throughput of up to $2\times$, with router area savings comparing to a standard router for conventional XYZ routing. Further, a $4\times$ throughput increase is possible with a small router area increase of $\leq 1\%$. If a larger throughput increase will be desired, additional area costs must be expended for crossbar and buffer depth in the mixed-signal layer (which is unrealistic, due to high costs). The overhead for link area depends on the TSV technology; the area costs will be reduced by both improvements in yield and monolithic stacking.

For a real-world based benchmark, we simulate a face recognition image processing pipeline on a 3D VSoC based on [11] with 180 nm mixed-signal technology and 90 nm digital technology. We calculate

a flit latency speedup under zero load of $2.28\times$. In simulations, we achieve $2.26\times$, which shows the expressiveness of the models. The speedup demonstrates an impressive performance benefit of the proposed approach for typical applications of heterogeneous 3D SoCs.

To summarize, $Z^+(XY)Z^-$ and $ZXYZ$, in combination with the novel router architectures, have negligible area overhead and better performance than state-of-the-art both in theoretical and practical evaluations. Therefore, limitations of heterogeneity on routing in 3D NoCs are mitigated.

9.9 CONCLUSION

In this chapter, we contribute that conventional routing algorithms and router architectures pose severe limitations, with heterogeneity. This is an expected result, yet was not quantified by means of well-founded models that express relevant effects of heterogeneity at a low error of 1.4% – 7.8% for the evaluated scenarios. Particularly, we have shown that varying throughput and latency of NoCs in layers in disparate technologies drastically degrades network performance. We apply the models to develop universal principles for routing in heterogeneous 3D SoCs; further, we develop two exemplary routing algorithms and a co-designed router architecture implementing these principles. For an exemplary SoC, with layers in commercial 45 nm digital and commercial 180 nm mixed-signal technology, we achieve a latency reduction of up to $6.5\times$ at negligible hardware area overhead in comparison to conventional dimension ordered routing. The novel vertical high-throughput router architecture and a vertical link design overcome throughput limitations and increase it by up to $2\times$ at 6% reduced router hardware costs for the same exemplary set of technologies. Thus, a co-design of routing algorithms and router architectures based on the proposed principles exploit heterogeneity for performance advantages to mitigate the limitations of conventional routing algorithms without drawbacks in implementation costs. The area of vertical links connecting heterogeneous routers is an open issue, which will be tackled by advances in TSV production methods.

Part IV
FINALE

10.1 ASYMMETRY – A NOVEL DESIGN PARADIGM

Since the relevance of heterogeneous 3D integration is steadily increasing, research on communication systems specifically for these chips is highly important. As one integral approach, this thesis proposes asymmetry as a novel design paradigm for NoCs targeting heterogeneous 3D SoCs. Applying the paradigm of asymmetry to an on-chip interconnection network requires exploitation of the varying technology-specific properties on each silicon die. Thereby it is possible for the first time to find both efficient system parameters for the network and effective architectures for routers. Conventional approaches applied to heterogeneous 3D chips are either rather inefficient (e. g. homogeneous 3D NoCs with synchronous clocked routers (cf. Section 5.4)), or practically impossible (e. g. hybrid 3D NoCs with a bus spanning multiple layers (cf. Section 5.2)). Thus, the proposed design paradigm extends state-of-the-art and enables usage of NoCs in heterogeneous 3D chips. We call implemented networks *asymmetric 3D NoCs*, in short *A-3D NoCs*.

The proposed novel design paradigm is established by the following concrete contributions as shown in Figure 92: In Chapter 5 we specify A-3D NoCs and define the design space. As one very important contribution, we thereby define the incremental approach for design space exploration that is used throughout the whole thesis. Next, we focus on system-level optimization in Chapter 6 following the incremental approach. We contribute a model and a heuristic algorithm that enable NoC planning with simultaneous layer assignment and component positioning. It improves NoC planning over state-of-the-art and, for the first time, includes the effects of heterogeneity, redistribution and a realistic through-silicon via array area and router area model. Thereby, efficient networks can be found. Thereafter, we direct attention to tools and methods for simulation in Chapter 7. The introduced simulation models and the implemented tools facilitate an empirical design space exploration process, including network simulation. Unlike competitors, the NoC simulator accounts for the technology effects of heterogeneous 3D integration. Subsequently, we continue to follow the incremental approach and optimize router memory in Chapter 8. Novel buffer distributions among routers yield architectures that reduce the router area and power. To the best of our knowledge, these are the first optimizations of router costs specifically exploiting effects of heterogeneous integration. Fi-

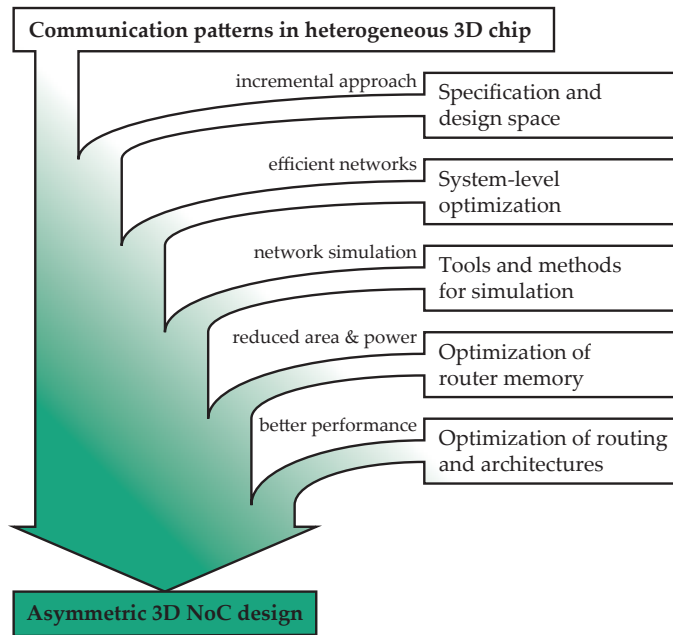


Figure 92: Thesis' contributions: From communication patterns in heterogeneous 3D SoCs to A-3D NoC designs.

nally, we further optimize routing and architectures in Chapter 9. We thereby contribute router architectures and routing algorithms with improved network performance, specifically reduced packet latency and improved throughput. In summary, this thesis covers the relevant aspects of design of A-3D NoCs by the incremental approach. Thereby, this thesis' contributions pave the way for novel, truly asymmetric interconnect architectures, which will play a significant role for communication in heterogeneous 3D SoCs.

10.2 IMPACT OF FUTURE TECHNOLOGIES

To conclude this thesis, we offer a discursive outlook of the impact of future technologies. Therefore, we will consider the most important trends, which can currently be found in heterogeneous 3D chips: The first trend is *increasing TSV yield* thanks to better production processes. This increases density of vertical interconnects. The second trend is *gradually reducing production costs* because of better yield and experience curve effects. The third trend is *permission for more applications* that will use heterogeneous integration. One example are chips combining processing and sensing. This principle can be found in 3D VSoC, as extensively applied as a use case in this thesis, and in many other application fields, such as mobile sensors. With cheaper production costs, heterogeneous 3D chips are increasingly usable. The impact of these trends on the findings of this thesis is as follows:

In Chapter 5 we specify the asymmetric design paradigm. It will still be reasonable for the aforementioned future technologies because

the limitations of today's approaches would still apply. The trends do not change our model of the design space. The proposed incremental approach will be even more relevant, because the increased TSV yield will allow for more architectural freedom. Therefore, more features must be explored. Hence, our approach to the design of NoCs targeting heterogeneous 3D chips will stay valid.

In Chapter 6 we introduced methods for system-level optimization. The technological models will stay valid, because the discussed trends do not change technological parameters, which the models account for. With increased TSV yield, the density of vertical interconnects will increase. Furthermore, reduced costs will drive the development for even more layers. Therefore, it will be very relevant that our proposed heuristic algorithm is also fast and efficient for high layer count and TSV array count.

In Chapter 7 we contributed models and tools for simulation. As already explained, higher TSV yield will increase the set of reasonable architectural and microarchitectural features. Therefore, our technological well-reasoned and well-structured definition of simulation models will still be applicable. Furthermore, the proposed parameter sets and introduced options to measure power and performance will be increasingly relevant with higher TSV array count in 3D NoCs.

In Chapter 8 we demonstrated a novel approach to router buffer area and power optimization exploiting heterogeneity. With more TSVs at lower costs, even more sophisticated architectures will be possible, extending this approach not only to memory but also to other parts of the router's microarchitecture.

In Chapter 9 we proposed routing and co-designed router architectures to increase network performance. The proposed models will stay valid. Therefore, different clock speeds in heterogeneous 3D chips will remain one important limitation to throughput. But increased TSV yield will mitigate this issue using approaches such as our novel link designs. Plus, reduced costs will increase the performance advantages of routing along paths spanning multiple heterogeneous layers.

To summarize, increased TSV yield and overall reduced manufacturing costs are very beneficial for NoCs as communication architecture in heterogeneous 3D chips. Therefore, it will be applicable to even more application areas and it will play a major role in future SoCs. The findings of this thesis will contribute to finally extend NoCs over multiple heterogeneous layers and optimally distribute parts of the interconnection architecture. This will allow for exploiting unprecedented optimization potential of heterogeneous integration, not only for on-chip communication networks.

Part V
APPENDIX

SYSTEM-LEVEL OPTIMIZATION

A.1 OVERVIEW OF SYMBOLS

A.1.1 Constants and definitions

The definitions slightly vary between Chapter 6 and the appendix due to conversion between places and indexes for locations of components, tiles and routers. We introduce the conversion in Section a.2 and Figure 94.

VALUES	MEANING
n	component count
k	available technology count
ℓ	layer count
m	maximum router count
x_{\max}, y_{\max}	upper bounds for size of chip, i.e. waver size
$[n], [k], [l], [m]$	Sets used as if they were sets of <i>components, technologies and layers and routers/tiles</i> .
E_A	set of ordered pairs of communicating components (<i>(sender, receiver)</i>), $E_A \subseteq [n] \times [n]$
$A = ([n], E_A)$	application digraph
u	bandwidth requirement, $u : E_A \rightarrow \mathbb{R}^+$
τ	$\tau : [l] \rightarrow [k]$, assignment of technologies to layers
f_c	implementation costs of components
f_{R2D}	implementation costs of a <i>2D router</i>
f_{KOZ}	area of KOZ
f_{R3D}	implementation costs of a <i>3D router</i>
d_{\max}	maximum length of links per layer, $d_{\max} : [k] \rightarrow \mathbb{R}_{>0}^{\infty}$. Used for redistribution and distance between routers in a layer.
P	Coordinates $P := \{x x \in \mathbb{R}, 0 \leq x \leq x_{\max}\} \times \{y y \in \mathbb{R}, 0 \leq y \leq y_{\max}\} \times [l]$
p_x, p_y, p_z	x -, y - and z -entries of a vector $p \in P$, $p = (p_x, p_y, p_z)$
<i>flow-Algorithm</i>	rule providing flows of the function f
$\varphi = \{(0, 0, 0)\}$	position of the non-placed position outside of the chip area
η	maximum aspect ratio of tiles

A.1.2 Variables

VALUES	MEANING
$r_1, \dots, r_m \in P \cup \varphi$	positions of routers in the network
$s_1, \dots, s_n \in P$	positions of components in the mapping
$t_1, \dots, t_m \in P \cup \varphi$	positions of tiles on the chip
R	$R := \{r_1, \dots, r_m\}$, set of router positions
S	$S := \{s_1, \dots, s_n\}$, set of positions of the components

T	$T := \{t_1, \dots, t_m\}$, set of positions where a <i>tile</i> starts
$\rho_1, \dots, \rho_m \in \mathcal{P}$	is one for placed routers, i.e. $\rho_i = 0 \Leftrightarrow r_i = \varphi$
$\tau_1, \dots, \tau_n \in \mathcal{P}$	is one for placed tiles, i.e. $\rho_i = 0 \Leftrightarrow r_i = \varphi$
$a_1, \dots, a_m \in \mathbb{R}$	length a_i of tile $t_i \in T$
$b_1, \dots, b_m \in \mathbb{R}$	width b_i of tile $t_i \in T$
$A_1, \dots, A_m \subset \mathcal{P}$	specific area of the tile $t_i \in T$
$e_{\{i,j\}} \in \{0, 1\}$	A_i is a closed two-dimensional interval embedded in a layer: $A_i := [(t_{i,x}, t_{i,y}), (t_{i,x} + a_i, t_{i,y} + b_i)] \times \{t_{i,z}\}$
E_N	for $i, j \in [m]$, information if the routers r_i and r_j are connected
$N = ([m], E_N)$	$E_N := \{(i, j) \in [m] \times [m] \mid e_{\{i,j\}} = 1\}$
f	<i>Topology</i> , digraph of routers links
$f^{(i,j)}$	$f : E_A \rightarrow \bigcup_{(i,j) \in E_A} \{f \mid f \text{ is an } i\text{-}j\text{-flow in } N, \text{value}(g) = 1\}$ ¹
$f^{(k,l)}$	variable for the flow between s_i and s_j , i.e. $f^{(i,j)} := f(\{(i, j)\})$
$f^{(k,l)}$	flow on the link $(k, l) \in [m] \times [m]$ of the flow $f^{(i,j)}$ (which is modeling flows in E_N)

A.1.3 Cost function

VALUES	MEANING
c_{area}	$c_{\text{area}} := \max\{\max_{i \in [m]}(t_{i,x} + a_i), \max_{i \in [m]}(t_{i,y} + b_i)\}$
c_{peak}	$c_{\text{peak}} := \sum_{v \in E_N} (\chi_{(\mu, \infty)}(\text{load}(v))(\text{load}(v) - \mu))$
c_{util}	$c_{\text{util}} := \sum_{e \in E_A} (u(e) \sum_{v \in E_N} (f(e))(v))$, costs for overall network utilization
load	$\text{load} : E_N \rightarrow \mathbb{R}_{\geq 0}$, $v \mapsto \sum_{e \in E_A} u(e)(f(e))(v)$, communication cost on a link or TSV
χ_A	<i>indicator function</i> of a set A
μ_l	average link load, $\mu_l := \frac{1}{ E_N } \sum_{v \in E_N} \text{load}(v)$
σ_l	sample variance of the loads in the network, $\sigma_l := \frac{1}{ E_N } \sum_{v \in E_N} (\text{load}(v) - \mu_l)^2$
σ_a	sample variance of bandwidth requirement u on E_A
μ	maximum value of u , $\mu := \max_{e \in E_A} u(e)$
ω_1, \dots	weights in the cost function
c_{total}	<i>cost function</i> $c_{\text{total}} = \omega_1 c_{\text{area}} \dots$ weighted for all costs

A.2 COMPONENT, ROUTER AND TILE COUNT

Lemma a.1. *The number of routers m limits the number of tiles and components. Hence, $n \leq m$.*

Proof. The auxiliary variable $\sigma_p \in \{0, 1\}$ is introduced. It holds that $\sigma_p = 1 \Leftrightarrow p \in S$, i.e. σ_p indicates each position in the chip, whether a component is placed there. Hence, the following inequalities are valid for every individual variable:

- For each position of a component, there is a router: $r_p \geq \sigma_p$. Since there is a router at each position of a component, the total number of routers is greater or equal to those of the components.

¹ The explanation of *flow* and its *value* can be found in Section 6.3.4.3.

- For each position of a component, there is a tile: $t_p \geq \sigma_p$. Since a tile starts at each position of a component or router, the total number of tiles is greater or equal to the number of the components.
- For each position of a tile, there is a router: $r_p \geq t_p$. Since there is at least one router at each position of a tile, the number of routers is larger or equal to the number of tiles.

Together, it holds that $r_p \geq t_p \geq \sigma_p$. Therefore, it is sufficient to limit the number of routers as this also limits the number of tiles and positions of components. \square

A.3 DEFINITIONS, NOTATIONS AND PREREQUISITES

A.3.1 Modeling a logical relations

Modeling the *logical OR relation* with the upper bounds is shown in Figure 93. On the left-hand side, the optimization curves of the OR-relation in z-dimension is shown. A variable should either be on the top or bottom orange line. Assuming, that the lines are not bounded, no convex optimization space can be defined as the red lines indicate. This is not the case with upper bounds as shown on the right-hand side of the Figure, in which x_{\max} and y_{\max} are used to limit the size of the optimization space. It now has a tetraedric shape and is convex. Therefore, bounds allow to model the logical OR. The *logical AND relation* can be modeled using multiple constraints, which must be satisfied together. The *logical implication* $a \rightarrow b$ can be transformed into an OR relation using the equivalence: $a \rightarrow b \leftrightarrow \bar{a} \vee b$. \bar{a} is equal to not a.

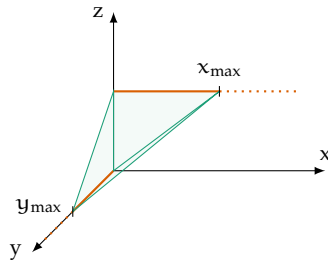


Figure 93: Modeling a logical or is possible via limits.

A.3.2 Line connections

Definition a.1 (Line between two points within a layer). The line between two points p and $q \in P$ within a layer is given by:

$$L(p, q) = \{p + \lambda(p - q) | \lambda \in [0, 1]\} = \text{conv}(p, q) \tag{75}$$

This definition is equivalent to the convex hull of p and q , since $(1 - \lambda)p + \lambda q = p + \lambda(p - q)$. In case the points p and q have the same x - or y -coordinate in our model, the line between the two points can also be defined as:

$$\begin{aligned} \hat{L}(p, q) &:= \{(\lambda_1, \lambda_2, p_z) \mid \\ &\lambda_1 \in [\min\{p_x, q_x\}, \max\{p_x, q_x\}], \\ &\lambda_2 \in [\min\{p_y, q_y\}, \max\{p_y, q_y\}]\} \end{aligned} \quad (76)$$

A.3.3 Neighbored routers

Definition a.2 (Neighbored routers (in a layer)). For indexes $i, j \in [m]$ we call two routers r_i and r_j *neighbored*, if:

- they are in the same layer ($r_{i,z} = r_{j,z}$)
- they share the same x - or y -coordinate ($r_{i,x} = r_{j,x}$ or $r_{i,y} = r_{j,y}$)
- there is no other router on the direct line between them ($\forall r_k \in [m] \setminus \{r_i, r_j\} : r_k \notin L(r_i, r_j) \setminus \{r_i, r_j\}$).
- and the indexes are not the same ($i \neq j$).

A.4 COST FUNCTION

A.4.1 SoC area

The SoC area c_{area} can be realized using a simple auxiliary variable $H_{\text{area}} \in \mathbb{R}_{\geq 0}$:

$$\forall i \in [m] : \quad t_{i,x} + a_i \leq H_{\text{area}}, \quad (77)$$

$$\forall i \in [m] : \quad t_{i,y} + b_i \leq H_{\text{area}} \quad (78)$$

and minimizing the cost function $c_{\text{area}} = H_{\text{area}}$.

A.4.2 Peak loads

We introduce the auxiliary variables h^μ and $h_{\text{peak}} \in \mathbb{R}$ as well as the auxiliary variables $h_{i,j}^{\text{load}}$ and $h_{i,j}^{\text{max}} \in \mathbb{R}$ for all $i, j \in [m]$. This yields $2m^2$ inequalities and $m^2 + 2$ equations:

$$h_{\text{peak}} = \sum_{i,j \in [m]} h_{i,j}^{\text{max}}, \quad (79)$$

$$h^\mu = n^2 - n, \quad (80)$$

$$h_{i,j}^{\text{load}} = \sum_{e \in E_\Lambda} u(e) f_e^{(i,j)} \quad \forall i \in [m], j \in [m] \quad (81)$$

$$h_{i,j}^{\text{max}} \geq 0 \quad \forall i \in [m], j \in [m] \quad (82)$$

$$h_{i,j}^{\text{max}} \geq h_{i,j}^{\text{load}} - h^\mu \quad \forall i \in [m], j \in [m] \quad (83)$$

A.5 CONSTRAINTS

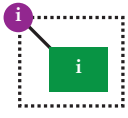
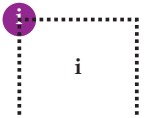
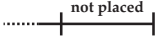

Constants

The following constants are given, which also allow modeling OR relations:

Definition a.3 (Constants). The following constants are given: $c_1 = x_{\max} + y_{\max}$, $c_2 = x_{\max} + y_{\max} + \ell$, $c_3 = y_{\max} + x_{\max} + \ell + 1$, and $c_4 = y_{\max} + x_{\max} + 1$.

A.5.1 Technical constraints

Technical constraints enable easy indexing within the other constraints. These constraints are not directly defining chip properties.

<p><i>Indexes of components and routers</i></p>  <p>Routers at the upper left corner of a tile have the same index as the tile's components.</p>	<p><i>Indexes of tiles and routers</i></p>  <p>Routers start tiles with the same index.</p>
<p><i>Indexes of unplaced Routers</i></p>  <p>Unplaced routers are at the end of the index range.</p>	<p><i>Force a_i, b_i to zero</i></p>  <p>Unplaced tiles have no size.</p>

Indexes of components and routers

As shown in Figure 94, components have the same indexes as routers:

$$\forall i \in [n] : s_i = r_i \tag{84}$$

This yields $6n$ inequalities:

$$s_{i,x} \leq r_{i,x}, \quad s_{i,y} \leq r_{i,y}, \quad s_{i,z} \leq r_{i,z}, \tag{85}$$

$$r_{i,x} \leq s_{i,x}, \quad r_{i,y} \leq s_{i,y}, \quad r_{i,z} \leq s_{i,z}. \tag{86}$$

Indexes of tiles and routers

Tiles will be started by the router with the same index if the tile is not started by a component and router together. Some routers will not start an own tile as shown in Figure 95. Thus, it holds for all $i \in \{n + 1, \dots, m\}$:

$$t_i \neq \varphi \Rightarrow t_i = r_i \tag{87}$$

$$\Leftrightarrow t_i = \varphi \text{ or } t_i = r_i. \tag{88}$$

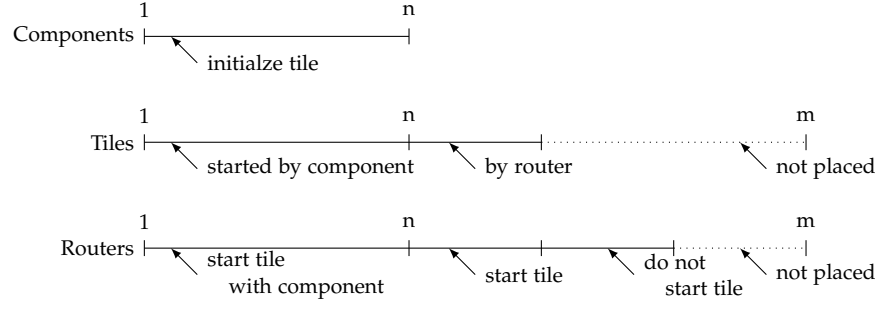


Figure 94: Indexes of routers, components, and tiles are the same if possible. Not placed routers and tiles have the highest indexes.

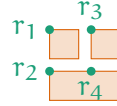


Figure 95: Area of router r_4 belongs to lower tile (orange). Router r_4 does not start individual tile for better packaging.

This yields the following $(m - n + 1)$ inequalities for all $i \in \{n + 1, \dots, m\}$ using the binary auxiliary variables h_i^z and the constant c_1 :

$$t_{i,z} \leq \varphi_z + h_i^z c_1 \quad (89)$$

$$\varphi_z \leq t_{i,z} + h_i^z c_1 \quad (90)$$

$$t_{i,x} \leq r_{i,x} + (1 - h_i^z) c_1 \quad (91)$$

$$r_{i,x} \leq t_{i,x} + (1 - h_i^z) c_1 \quad (92)$$

$$t_{i,y} \leq r_{i,y} + (1 - h_i^z) c_1 \quad (93)$$

$$r_{i,y} \leq t_{i,y} + (1 - h_i^z) c_1 \quad (94)$$

$$t_{i,z} \leq r_{i,z} + (1 - h_i^z) c_1 \quad (95)$$

$$r_{i,z} \leq t_{i,z} + (1 - h_i^z) c_1 \quad (96)$$

Indexes of unplaced tiles and routers

Non-placed tiles and routers are assigned to the highest indexes. Thus, it holds for all tiles with index $i \in [m - 1]$:

$$t_{i,z} = 0 \rightarrow t_{i+1,z} = 0 \quad (97)$$

$$\leftrightarrow t_{i,z} \neq 0 \text{ or } t_{i+1,z} = 0 \quad (98)$$

This can be written into the following $2(m - 1)$ inequalities using the auxiliary binary variable c_1 :

$$t_{i,z} c_1 \geq t_{i+1,z} \quad (99)$$

$$r_{i,z} c_1 \geq r_{i+1,z} \quad (100)$$

Force a_i and b_i to zero

The variables a_i and b_i for unplaced tiles with indexes $i \in [n + 1, \dots, m]$ must be zero:

$$t_i = \varphi \rightarrow a_i = 0 \text{ and } b_i = 0 \quad (101)$$

This leads to the following $2m - n + 1$ inequalities using the constant c_1 for all $i \in [n + 1, \dots, m]$:

$$a_i \leq c_1 t_{i_z} \quad (102)$$

$$b_i \leq c_1 t_{i_z} \quad (103)$$

Unplaced routers in φ

The auxiliary binary variables ρ_i for all $i \in [m]$ are introduced, which indicate whether a router is placed. Routers, which are unplaced, are assigned to the position $\varphi = (0, 0, 0)$ outside of the chip area, i.e. $\varphi \notin P$. Thus, the following inequalities for each $i \in [m]$ yield the total of $3m$ inequalities:

$$\rho_i \leq r_{i_z} \leq \varphi_i l \quad (104)$$

$$r_{i,y} \leq y_{\max} \varphi_i \quad (105)$$

$$r_{i,x} \leq x_{\max} \varphi_i \quad (106)$$

A.5.2 Network constraints

Connectivity

Utilizing the connectivity of routing algorithms via the definition of flow in the network, network graphs in valid solutions of the optimization problem are connected. Therefore, the connectivity is included via the constraints for flow and routing algorithms.

Grid-based topology and TSVs connect adjacent layers

The constraint is formulated as:

$$\begin{aligned} \forall (i, j) \in [m] \times [m] : \\ e_{\{i,j\}} = 1 \rightarrow \\ r_{i,z} = r_{j,z} \text{ and } (r_{i,x} = r_{j,x} \text{ or } r_{i,y} = r_{j,y}) \\ \text{or} \\ r_{i,z} \in \{r_{j,z+1}, r_{j,z-1}\} \text{ and } |r_{i,x} - r_{j,x}| + |r_{i,y} - r_{j,y}| \leq \\ d_{\max}(\max(r_{i,z}, r_{j,z})). \end{aligned} \quad (107)$$

This is translated into m^2 inequalities. Therefore, the binary auxiliary variables $h_{i,j}^4$, $h_{i,j}^4_{OR1}$, and $h_{i,j}^4_{OR2}$ are introduced using the constant c_3 . For all tuples $(i, j) \in [m] \times [m]$, $i \neq j$ these $14m(m - 1)$ inequalities must be satisfied:

$$r_{i,z} \leq r_{j,z} + h_{i,j}^4 c_3 + (1 - e_{\{i,j\}}) c_3 \quad (108)$$

$$r_{j,z} \leq r_{i,z} + h_{i,j}^4 c_3 + (1 - e_{\{i,j\}}) c_3 \quad (109)$$

$$r_{i,x} \leq r_{j,x} + h_{i,j}^4_{OR1} c_3 + h_{i,j}^4 c_3 + (1 - e_{\{i,j\}}) c_3 \quad (110)$$

$$r_{j,x} \leq r_{i,x} + h_{i,j}^4 \text{OR1} c_3 + h_{i,j}^4 c_3 + (1 - e_{\{i,j\}}) c_3 \quad (111)$$

$$r_{i,y} \leq r_{j,y} + (1 - h_{i,j}^4 \text{OR1}) c_3 + h_{i,j}^4 c_3 + (1 - e_{\{i,j\}}) c_3 \quad (112)$$

$$r_{j,y} \leq r_{i,y} + (1 - h_{i,j}^4 \text{OR1}) c_3 + h_{i,j}^4 c_3 + (1 - e_{\{i,j\}}) c_3 \quad (113)$$

$$(r_{j,z} + 1) \leq r_{i,z} + h_{i,j}^4 \text{OR2} c_3 + (1 - h_{i,j}^4 c_3) + (1 - e_{\{i,j\}}) c_3 \quad (114)$$

$$r_{i,z} \leq (r_{j,z} + 1) + h_{i,j}^4 \text{OR2} c_3 + (1 - h_{i,j}^4 c_3) + (1 - e_{\{i,j\}}) c_3 \quad (115)$$

$$(r_{j,z} - 1) \leq r_{i,z} + (1 - h_{i,j}^4 \text{OR2}) c_3 + (1 - h_{i,j}^4 c_3) + (1 - e_{\{i,j\}}) c_3 \quad (116)$$

$$r_{i,z} \leq (r_{j,z} - 1) + (1 - h_{i,j}^4 \text{OR2}) c_3 + (1 - h_{i,j}^4 c_3) + (1 - e_{\{i,j\}}) c_3 \quad (117)$$

$$|r_{i,x} - r_{j,x}| + |r_{i,y} - r_{j,y}| \leq d_{\max}(\max(r_{i,z}, r_{j,z})) + (1 - h_{i,j}^4 c_3) + (1 - e_{\{i,j\}}) c_3 \quad (118)$$

The last inequality (118) must be further transformed. We replace the absolute values and $d_{\max}(\max(r_{i,z}, r_{j,z}))$ by auxiliary variables. The Equation 118 changes to:

$$\text{dist}(r_{i,x}, r_{j,x}) + \text{dist}(r_{i,y}, r_{j,y}) \leq d_{ij} + \quad (119)$$

$$(1 - h_{i,j}^4 c_3) + (1 - e_{\{i,j\}}) c_3 \quad (120)$$

with the auxiliary variables

$$\text{dist}(r_{i,x}, r_{j,x}) \quad \forall (i, j) \in [m] \times [m], i \neq j \quad (121)$$

$$\text{dist}(r_{i,y}, r_{j,y}) \quad \forall (i, j) \in [m] \times [m], i \neq j \quad (122)$$

$$d_{ij} \quad \forall (i, j) \in [m] \times [m], i \neq j \quad (123)$$

The dist-"function" $\text{dist}(a, b) = |a - b|$ is given by these six inequalities which must be satisfied in the MILP model using the auxiliary binary variable $h_{a \leq b}$ and the constant c_3^2 :

$$\text{dist}(a, b) \geq a - b \quad (124)$$

$$\text{dist}(a, b) \geq b - a \quad (125)$$

$$a \leq b + (1 - h_{a \leq b}) c_3 \quad (126)$$

$$b \leq a + h_{a \leq b} c_3 \quad (127)$$

$$\text{dist}(a, b) \leq a - b + h_{a \leq b} c_3 \quad (128)$$

$$\text{dist}(a, b) \leq b - a + (1 - h_{a \leq b}) c_3 \quad (129)$$

For each of the dist-auxiliary-variables, these six inequalities with the respective variables instead of a and b are required to model the dist-function-value. Hence, $6 \cdot 2m(m - 1)$ extra inequalities are given for the absolute values.

Now, we need some inequalities to define the value of d_{ij} . Let us use the symmetry of these variables:

$$d_{ij} = d_{ji} \quad \text{for all } (i, j) \in [m] \times [m], i > j. \quad (130)$$

Since the value of d_{ij} depends on $r_{i,z}$ and $r_{j,z}$, we need further auxiliary variables.

The term $d_{ij} = d_{\max}(\max(r_{i,z}, r_{j,z}))$ is equivalent to

For all $(i, j) \in [m] \times [m], i < j$:

$$d_{ij} = \sum_{\xi=1}^{\ell} d_{\max}(\xi, h_{i,j,\xi}^4) \quad (131)$$

$$\text{and } h_{i,j,\xi}^4 \in \{0, 1\} \text{ for all } \xi \in \ell \quad (132)$$

2 Again, we exploit that a and b are bounded. Hence, dist is a function from $[0, x_{\max} + y_{\max}] \times [0, x_{\max} + y_{\max}]$ which maps to $[0, x_{\max} + y_{\max}]$

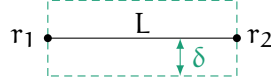


Figure 96: A small area around the line between r_1 and r_2 is defined since open sets cannot be modeled through an MILP.

$$\text{and } \sum_{\xi=1}^{\ell} h_{i,j,\xi}^4 = 1 \quad (133)$$

and for all $\xi \in \ell$:

$$r_{i,z} \leq \xi + (1 - h_{i,j,\xi}^4)\ell \quad \text{and} \quad (134)$$

$$r_{j,z} \leq \xi + (1 - h_{i,j,\xi}^4)\ell \quad \text{and} \quad (135)$$

$$(\xi \leq r_{i,z} + (1 - h_{i,j,\xi}^4)\ell) \quad \text{or} \quad (136)$$

$$\xi \leq r_{j,z} + (1 - h_{i,j,\xi}^4)\ell \quad (137)$$

which is, written in inequalities for all $(i,j) \in [m] \times [m]$, $i < j$, $\xi \in [\ell]$, with $h_{i,j,\xi}^4 \in \{0, 1\}$:

$$d_{ij} = \sum_{\xi=1}^{\ell} d_{\max}(\xi, h_{i,j,\xi}^4) \quad (138)$$

$$1 = \sum_{\xi=1}^{\ell} h_{i,j,\xi}^4 \quad (139)$$

For all $(i,j) \in [m] \times [m]$, $i < j$ and for all $\xi \in \ell$:

$$r_{i,z} \leq \xi + (1 - h_{i,j,\xi}^4)\ell \quad (140)$$

$$r_{j,z} \leq \xi + (1 - h_{i,j,\xi}^4)\ell \quad (141)$$

$$\xi \leq r_{i,z} + (1 - h_{i,j,\xi}^4)\ell + (1 - h_{i,j,\xi,\text{OR}}^4)\ell \quad (142)$$

$$\xi \leq r_{j,z} + (1 - h_{i,j,\xi}^4)\ell + h_{i,j,\xi,\text{OR}}^4\ell \quad (143)$$

Forbid connections between non-neighbored routers

We formulate:

$$\forall (i,j) \in E_{\mathcal{N}}, \forall k \in [m] \setminus \{i,j\}: r_k \notin L(r_i, r_j) \quad (144)$$

$$\Leftrightarrow \forall (i,j) \in [m] \times [m], i \neq j: e_{\{i,j\}} = 1 \quad (145)$$

$$\rightarrow \forall k \in [m] \setminus \{i,j\}: r_k \notin L(r_i, r_j)$$

Since $r_k \in \mathbb{R} \setminus L(r_i, r_j)$ is an open set, we introduce a small area around the connecting line as shown in Figure 96. To fulfill the latter part $\forall k \in [m] \setminus \{i,j\}: r_k \notin L(r_i, r_j)$, at least one of the four inequalities in Figure 97 must be fulfilled. This yields these four inequalities, which are connected by a logical OR relation for any $k \in [m]$:

$$\text{I: } a_1^T r_k \leq b_1 \quad \text{or} \quad (146)$$

$$\text{II: } -a_1^T r_k \leq b_2 \quad \text{or} \quad (147)$$

$$\text{III: } a_2^T r_k \leq b_3 \quad \text{or} \quad (148)$$

$$\text{IV: } -a_2^T r_k \leq b_4 \quad (149)$$

These line equations are solved to calculate the values of b_1 to b_4 (the normal is calculated). Therefore, points which are located on each line are required. A known point on the line (I) is r_i , on line (II) it is r_j , on line (III) it is $r_i - \delta a_2$, and on line (IV) it is $r_i + \delta a_2$. We further know

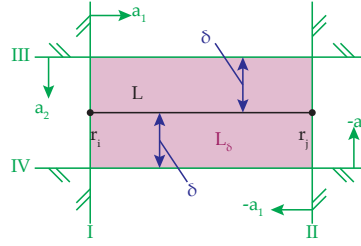


Figure 97: The routers r_i and r_j are connected by a 2D-link L . Around the link, the δ -area L_δ is defined and shown in red. Part of the constraint is $\forall k \in [m] \setminus \{i, j\} : r_k \notin L(r_i, r_j)$. Therefore, at least one of the four inequalities given by the line equations (I) to (IV) must be fulfilled, which are shown in green.

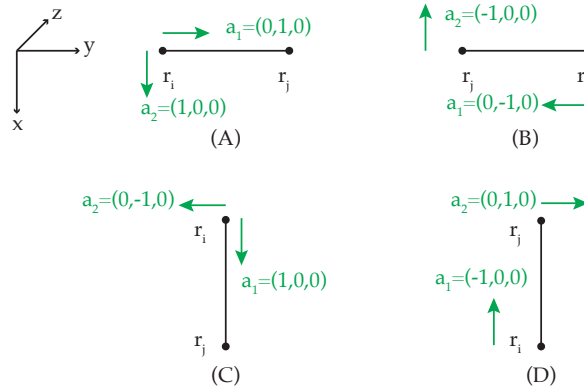


Figure 98: Only allowed configurations for grid based topology for the location of links and the resulting line equations.

that in our model it holds $r_{i,z} = r_{j,z}$ and $r_{i,x} = r_{j,x}$ or $r_{i,y} = r_{j,y}$. This yields

$$\text{I: } \mathbf{a}_1^\top \mathbf{r}_i = b_1 \quad (150)$$

$$\text{II: } -\mathbf{a}_1^\top \mathbf{r}_j = b_2 \quad (151)$$

$$\text{III: } \mathbf{a}_2^\top (\mathbf{r}_i - \delta \mathbf{a}_2) = b_3 \quad (152)$$

$$\text{IV: } -\mathbf{a}_2^\top (\mathbf{r}_j + \delta \mathbf{a}_2) = b_4 \quad (153)$$

as well as \mathbf{a}_1 :

$$\mathbf{a}_1 = \frac{\mathbf{r}_j - \mathbf{r}_i}{|\mathbf{r}_j - \mathbf{r}_i|} = \frac{1}{|\mathbf{r}_j - \mathbf{r}_i|} \begin{pmatrix} r_{j,x} - r_{i,x} \\ r_{j,y} - r_{i,x} \\ 0 \end{pmatrix} \quad (154)$$

and the analog for \mathbf{a}_2 . As shown in Figure 98 only the four configurations (A), (B), (C), and (D) are possible. Solving the linear equations in Equation 150 to Equation 153 yields:

	a_1	a_2	b_1	b_2	b_3	b_4
(A)	$(0, 1, 0)^T$	$(1, 0, 0)^T$	$r_{i,y}$	$-r_{j,y}$	$r_{i,x} - \delta$	$-r_{i,x} - \delta$
(B)	$(0, -1, 0)^T$	$(-1, 0, 0)^T$	$-r_{i,y}$	$r_{j,y}$	$-r_{i,x} - \delta$	$r_{i,x} - \delta$
(C)	$(1, 0, 0)^T$	$(0, -1, 0)^T$	$r_{i,x}$	$-r_{j,x}$	$-r_{i,y} - \delta$	$r_{i,y} - \delta$
(D)	$(-1, 0, 0)^T$	$(0, 1, 0)^T$	$-r_{i,x}$	$r_{j,x}$	$r_{i,y} - \delta$	$-r_{i,y} - \delta$

The line equations are used to formulate the constraint:

$$\begin{aligned} \forall (i, j) \in [m] \times [m], i \neq j: e_{\{i,j\}} = 1 \text{ and } r_{i,z} = r_{j,z} \\ \rightarrow \forall k \in [m] \setminus \{i, j\}: \text{(A) or (B) or (C) or (D)}, \end{aligned} \quad (155)$$

which, including the case differentiation, yields:

$$\begin{aligned} \forall (i, j) \in [m] \times [m], i \neq j: e_{\{i,j\}} = 1 \text{ and } r_{i,z} = r_{j,z} \rightarrow \forall k \in [m] \setminus \{i, j\}: \\ r_{i,y} \leq r_{j,y} \text{ and } r_{i,x} = r_{j,x} \text{ and } (1_{(A)} \text{ or } 2_{(A)} \text{ or } 3_{(A)} \text{ or } 4_{(A)}) \\ \text{or} \\ r_{j,x} \leq r_{i,x} \text{ and } r_{i,x} = r_{j,x} \text{ and } (1_{(B)} \text{ or } 2_{(B)} \text{ or } 3_{(B)} \text{ or } 4_{(B)}) \\ \text{or} \\ r_{i,x} \leq r_{j,x} \text{ and } r_{i,y} = r_{j,y} \text{ and } (1_{(C)} \text{ or } 2_{(C)} \text{ or } 3_{(C)} \text{ or } 4_{(C)}) \\ \text{or} \\ r_{j,x} \leq r_{i,x} \text{ and } r_{i,y} = r_{j,y} \text{ and } (1_{(D)} \text{ or } 2_{(D)} \text{ or } 3_{(D)} \text{ or } 4_{(D)}) \end{aligned} \quad (156)$$

For easy notation, we write

$$A_0 = e_{\{i,j\}} = 1 \text{ and } r_{i,z} = r_{j,z} \quad (157)$$

$$(A)_1 = r_{i,y} \leq r_{j,y} \text{ and } r_{i,x} = r_{j,x} \quad (158)$$

$$(A)_2 = I_{(A)} \text{ or } II_{(A)} \text{ or } III_{(A)} \text{ or } IV_{(A)} \quad (159)$$

$$(B)_1 = r_{j,x} \leq r_{i,x} \text{ and } r_{i,x} = r_{j,x} \quad (160)$$

$$(B)_2 = I_{(B)} \text{ or } II_{(B)} \text{ or } III_{(B)} \text{ or } IV_{(B)} \quad (161)$$

$$(C)_1 = r_{i,x} \leq r_{j,x} \text{ and } r_{i,y} = r_{j,y} \quad (162)$$

$$(C)_2 = I_{(C)} \text{ or } II_{(C)} \text{ or } III_{(C)} \text{ or } IV_{(C)} \quad (163)$$

$$(D)_1 = r_{j,x} \leq r_{i,x} \text{ and } r_{i,y} = r_{j,y} \quad (164)$$

$$(D)_2 = I_{(D)} \text{ or } II_{(D)} \text{ or } III_{(D)} \text{ or } IV_{(D)} \quad (165)$$

Using this, Equation 156 is transformed³:

$$\begin{aligned} \forall (i, j) \in [m] \times [m]: \bar{A}_0 \text{ or } \forall k \in [m] \setminus \{i, j\}: \\ ((A)_1 \text{ and } (A)_2) \text{ or } ((B)_1 \text{ and } (B)_2) \text{ or} \\ ((C)_1 \text{ and } (C)_2) \text{ or } ((D)_1 \text{ and } (D)_2) \end{aligned} \quad (166)$$

Due to the complexity of the constraint, it is split up into modules to write the inequalities. In addition, the inequalities are color coded: **green** denotes inequalities, **the or relation between cases A, B, C, and D** is coded in **red**, **the or relation between \bar{A}_0 and the rest** is given in **purple**.

Module 1 is $(A)_1$ and $(A)_2$. The auxiliary binary variables $h_{i,j,k}^5$, $h_{i,j,k}^{5A1}$, $h_{i,j,k}^{5A2}$, $h_{i,j,k}^{5A3}$, $h_{i,j,k}^{5A4}$, and $h_{i,j,OR}^5$, as well as the constant c_4 are used. The following inequalities must be satisfied for all $(i, j) \in [m] \times [m], i \neq j$ and per tuple (i, j) are given for all $k \in [m] \setminus \{i, j\}$, hence,

³ \bar{A}_0 denotes not A_0 .

$m - 2$ times. All in all, the inequalities are given $m(m - 1)(m - 2)$ times.

$$r_{i,y} \leq r_{j,y} + h_{i,j,k1}^5 c_4 + h_{i,j,OR}^5 c_4 \quad (167)$$

$$r_{i,x} \leq r_{j,x} + h_{i,j,k1}^5 c_4 + h_{i,j,OR}^5 c_4 \quad (168)$$

$$r_{j,x} \leq r_{i,x} + h_{i,j,k1}^5 c_4 + h_{i,j,OR}^5 c_4 \quad (169)$$

$$r_{k,y} \leq r_{i,y} + h_{i,j,kA1}^5 c_4 + h_{i,j,k1}^5 c_4 + h_{i,j,OR}^5 c_4 \quad (170)$$

$$-r_{k,y} \leq -r_{j,y} + h_{i,j,kA2}^5 c_4 + h_{i,j,k1}^5 c_4 + h_{i,j,OR}^5 c_4 \quad (171)$$

$$r_{k,x} \leq r_{i,x} - \delta + h_{i,j,kA3}^5 c_4 + h_{i,j,k1}^5 c_4 + h_{i,j,OR}^5 c_4 \quad (172)$$

$$-r_{k,x} \leq -r_{i,x} - \delta + h_{i,j,kA4}^5 c_4 + h_{i,j,k1}^5 c_4 + h_{i,j,OR}^5 c_4 \quad (173)$$

The logical OR is given by:

$$h_{i,j,kA1}^5 + h_{i,j,kA2}^5 + h_{i,j,kA3}^5 + h_{i,j,kA4}^5 \leq 3 \quad (174)$$

Module 2 is (B)₁ and (B)₂. The auxiliary binary variables $h_{i,j,k2}^5$, $h_{i,j,kB1}^5$, $h_{i,j,kB2}^5$, $h_{i,j,kB3}^5$, $h_{i,j,kB4}^5$, and $h_{i,j,OR}^5$. Again, the inequalities must be satisfied for all $(i, j) \in [m] \times [m]$, $i \neq j$ and for all $k \in [m] \setminus \{i, j\}$. Thus $m(m - 1)(m - 2)$ inequalities are given.

$$r_{j,y} \leq r_{i,y} + h_{i,j,k2}^5 c_4 + h_{i,j,OR}^5 c_4 \quad (175)$$

$$r_{i,x} \leq r_{j,x} + h_{i,j,k2}^5 c_4 + h_{i,j,OR}^5 c_4 \quad (176)$$

$$r_{j,x} \leq r_{i,x} + h_{i,j,k2}^5 c_4 + h_{i,j,OR}^5 c_4 \quad (177)$$

$$-r_{k,y} \leq -r_{i,y} + h_{i,j,kB1}^5 c_4 + h_{i,j,k2}^5 c_4 + h_{i,j,OR}^5 c_4 \quad (178)$$

$$r_{k,y} \leq r_{j,y} + h_{i,j,kB2}^5 c_4 + h_{i,j,k2}^5 c_4 + h_{i,j,OR}^5 c_4 \quad (179)$$

$$-r_{k,x} \leq -r_{i,x} - \delta + h_{i,j,kB3}^5 c_4 + h_{i,j,k2}^5 c_4 + h_{i,j,OR}^5 c_4 \quad (180)$$

$$r_{k,x} \leq r_{i,x} - \delta + h_{i,j,kB4}^5 c_4 + h_{i,j,k2}^5 c_4 + h_{i,j,OR}^5 c_4 \quad (181)$$

$$h_{i,j,kB1}^5 + h_{i,j,kB2}^5 + h_{i,j,kB3}^5 + h_{i,j,kB4}^5 \leq 3 \quad (182)$$

Module 3 is (C)₁ and (C)₂. The auxiliary binary variables $h_{i,j,k3}^5$, $h_{i,j,kC1}^5$, $h_{i,j,kC2}^5$, $h_{i,j,kC3}^5$, $h_{i,j,kC4}^5$, and $h_{i,j,OR}^5$ are used. Again, the inequalities must be satisfied for all $(i, j) \in [m] \times [m]$, $i \neq j$ and for all $k \in [m] \setminus \{i, j\}$. This yields $m(m - 1)(m - 2)$ inequalities.

$$r_{i,x} \leq r_{j,x} + h_{i,j,k3}^5 c_4 + h_{i,j,OR}^5 c_4 \quad (183)$$

$$r_{j,y} \leq r_{i,y} + h_{i,j,k3}^5 c_4 + h_{i,j,OR}^5 c_4 \quad (184)$$

$$r_{i,y} \leq r_{j,y} + h_{i,j,k3}^5 c_4 + h_{i,j,OR}^5 c_4 \quad (185)$$

$$r_{k,x} \leq r_{i,x} + h_{i,j,kC1}^5 c_4 + h_{i,j,k3}^5 c_4 + h_{i,j,OR}^5 c_4 \quad (186)$$

$$-r_{k,x} \leq -r_{j,x} + h_{i,j,kC2}^5 c_4 + h_{i,j,k3}^5 c_4 + h_{i,j,OR}^5 c_4 \quad (187)$$

$$-r_{k,y} \leq -r_{i,y} - \delta + h_{i,j,kC3}^5 c_4 + h_{i,j,k3}^5 c_4 + h_{i,j,OR}^5 c_4 \quad (188)$$

$$r_{k,y} \leq r_{i,y} - \delta + h_{i,j,kC4}^5 c_4 + h_{i,j,k3}^5 c_4 + h_{i,j,OR}^5 c_4 \quad (189)$$

$$h_{i,j,kC1}^5 + h_{i,j,kC2}^5 + h_{i,j,kC3}^5 + h_{i,j,kC4}^5 \leq 3 \quad (190)$$

Module 4 is (D)₁ and (D)₂. The auxiliary binary variables $h_{i,j,k4}^5$, $h_{i,j,kD1}^5$, $h_{i,j,kD2}^5$, $h_{i,j,kD3}^5$, $h_{i,j,kD4}^5$, and $h_{i,j,OR}^5$ are used. Again, the inequalities must be satisfied for all $(i, j) \in [m] \times [m]$, $i \neq j$ and for all

$k \in [m] \setminus \{i, j\}$. Thus, all in all, the equations are given $m(m-1)(m-2)$ times.

$$r_{j,x} \leq r_{i,x} + h_{i,j,k}^5 c_4 + h_{i,j,OR}^5 c_4 \quad (191)$$

$$r_{j,y} \leq r_{i,y} + h_{i,j,k}^5 c_4 + h_{i,j,OR}^5 c_4 \quad (192)$$

$$r_{i,y} \leq r_{j,y} + h_{i,j,k}^5 c_4 + h_{i,j,OR}^5 c_4 \quad (193)$$

$$-r_{k,x} \leq -r_{i,x} + h_{i,j,k}^5 c_{D1} + h_{i,j,k}^5 c_4 + h_{i,j,OR}^5 c_4 \quad (194)$$

$$r_{k,x} \leq r_{j,x} + h_{i,j,k}^5 c_{D2} + h_{i,j,k}^5 c_4 + h_{i,j,OR}^5 c_4 \quad (195)$$

$$r_{k,y} \leq r_{i,y} - \delta + h_{i,j,k}^5 c_{D3} + h_{i,j,k}^5 c_4 + h_{i,j,OR}^5 c_4 \quad (196)$$

$$-r_{k,y} \leq -r_{i,y} - \delta + h_{i,j,k}^5 c_{D4} + h_{i,j,k}^5 c_4 + h_{i,j,OR}^5 c_4 \quad (197)$$

$$h_{i,j,k}^5 c_{D1} + h_{i,j,k}^5 c_{D2} + h_{i,j,k}^5 c_{D3} + h_{i,j,k}^5 c_{D4} \leq 3 \quad (198)$$

There are $m(m-1)(m-2)$ inequalities for the OR relations between the modules for all $i, j, [m] \times [m], i \neq j$ and for all $k \in [m] \setminus \{i, j\}$:

$$h_{i,j,k1}^5 + h_{i,j,k2}^5 + h_{i,j,k3}^5 + h_{i,j,k4}^5 \leq 3 \quad (199)$$

Module 5 denotes \bar{A}_0 . We define the binary auxiliary variables $h_{i,j,1}^5, h_{i,j,2}^5$, and $h_{i,j,3}^5$. There are $m(m-1)$ inequalities.

$$e_{\{i,j\}} \leq 0 + h_{i,j,1}^5 c_4 + (1 - h_{i,j,OR}^5) c_4 \quad (200)$$

$$r_{i,z} \leq r_{j,z} - 1/2 + h_{i,j,2}^5 c_4 + (1 - h_{i,j,OR}^5) c_4 \quad (201)$$

$$r_{j,z} \leq r_{i,z} - 1/2 + h_{i,j,3}^5 c_4 + (1 - h_{i,j,OR}^5) c_4 \quad (202)$$

$$h_{i,j,1}^5 + h_{i,j,2}^5 + h_{i,j,3}^5 \leq 2 \quad (203)$$

Summing up, this constrain consists of $4(m(m-1)) + 4 * 8 * m(m-1)(m-2)$ inequalities.

Components are connected to routers

Due to the indexing as defined in the technical constraints, the first n routers have the same index as the n components. Thus, the condition of this constraints is already given since each component is associated with a router at the same index and position.

Routers are not self-connected

The constraint is modeled for all $i \in [m]$ by $e_{\{i,i\}} = 0$. This yields m inequalities for all $i \in [m]$:

$$e_{\{i,i\}} \leq 0. \quad (204)$$

A.5.3 Bounding boxes

Starting position of tiles

This is formulated as $\forall t_i \in T: t_i \neq \varphi \rightarrow \exists r_i : t_i = r_i$. Utilizing the indexing, this condition is separated in two cases. First, for all $i \in [n] : t_i = r_i$. This yields $6n$ inequalities representing the equality between t_i and r_i for all $i \in [n]$ in each dimension. Second, for all $i \in \{n+1, \dots, m\} : t_i \neq \varphi \rightarrow t_i = r_i$, which is already given by the constraint in Figure a.5.1.

Components start tiles

This is formulated as $\forall i \in [n] \exists j \in [m] : s_i = t_j$. Due to the indexing, this can be further simplified for all $i \in [n] : s_i = t_i$, which is already given by Section a.5.3 and Figure a.5.1 ($r_i = s_i = t_i$). In addition, all components must be placed. Therefore, for all $i \in [n]$ it holds that $t_i \neq \varphi$. Since $\varphi_z = 0$, this can be modeled with a lower bound for all $i \in [n] : t_{i,z} \geq 1$.

Routers and tiles

This is formulated for all routers with indexes $i \in [m]$ and tiles with index $k \in [m]$:

$$\forall i \in [m] : r_i \neq \varphi \rightarrow \exists k \in [m] : r_i \in A_k, \quad (205)$$

$$\leftrightarrow \forall i \in [m] : r_i = \varphi \text{ or } (\exists t_k : r_i \in A_{t_k}). \quad (206)$$

In our model, the first n routers are located at the same position as the first n components. Thus, this constraint must only consider routers with indexes $i \in [n + 1, \dots, m]$. Furthermore, additional routers cannot be placed within a tile, because links between borders of tiles cannot pass through tiles: Routers are always placed on borders of tiles. Hence, it is sufficient to consider the borders of each tile and ensure that each router i is located on one of the four edgings of a tile k :

$$\begin{aligned} \forall i \in [n + 1, \dots, m] : r_i = \varphi \text{ or } \exists k \in [m] : \\ r_i \in [(t_{k,x}, t_{k,y}), (t_{k,x} + a_k, t_{k,y})] \times t_{k,z} \text{ or} \\ r_i \in [(t_{k,x}, t_{k,y}), (t_{k,x}, t_{k,y} + b_k)] \times t_{k,z} \text{ or} \\ r_i \in [(t_{k,x}, t_{k,y} + b_k), (t_{k,x} + a_k, t_{k,y} + b_k)] \times t_{k,z} \text{ or} \\ r_i \in [(t_{k,x} + a_k, t_{k,y}), (t_{k,x} + a_k, t_{k,y} + b_k)] \times t_{k,z}. \end{aligned} \quad (207)$$

This can be further simplified since the first part of the equation, $r_i = \varphi$, can be omitted. If a router is unplaced, $r_i = \varphi$, there will also be at least one tile that is unplaced, i. e. $\exists j \in [m] : t_j = \varphi$. Thus, the inequalities are satisfied since $r_i = t_j = \varphi$.

For each of the four intervals there are 6 inequalities. The four blocks of inequalities are connected by a logical OR since it is sufficient that a router is in one of these intervals. The inequalities for the interval of the upper edging of the tile are given by:

$$t_{k,x} \leq r_{i,x} \leq t_{k,x} + a_t \quad (208)$$

$$t_{k,y} \leq r_{i,y} \leq t_{k,y} \quad (209)$$

$$t_{k,z} \leq r_{i,z} \leq t_{k,z} \quad (210)$$

The inequalities are given by for the interval of the tile's left edging:

$$t_{k,x} \leq r_{i,x} \leq t_{k,x} \quad (211)$$

$$t_{k,y} \leq r_{i,y} \leq t_{k,y} + b_t \quad (212)$$

$$t_{k,z} \leq r_{i,z} \leq t_{k,z} \quad (213)$$

The inequalities for the intervals of the right and lower edging of the tile are given by:

$$t_{k,x} + a_t \leq r_{i,x} \leq t_{k,x} + a_t \quad (214)$$

$$t_{k,y} \leq r_{i,y} \leq t_{k,y} + b_t \quad (215)$$

$$t_{k,z} \leq r_{i,z} \leq t_{k,z} \quad \text{and} \quad (216)$$

$$t_{k,x} \leq r_{i,x} \leq t_{k,x} + a_t \quad (217)$$

$$t_{k,y} + b_t \leq r_{i,y} \leq t_{k,y} + b_t \quad (218)$$

$$t_{k,z} \leq r_{i,z} \leq t_{k,z}. \quad (219)$$

We introduce the binary auxiliary variables $h_i^{1_0}$, $h_i^{2_0}$, $h_i^{3_0}$, and $h_i^{4_0}$ for each $i \in [n+1, \dots, m]$ and $h_{i,k}^{1_0}$ for each $i \in [n+1, \dots, m]$ and $k \in [m]$. Thus, the $25m(m-n+1)$ inequalities for each $i \in [n]$ and $k \in [m]$ are given:

$$t_{k,x} \leq r_{i,x} + c_3(h_i^{1_0} + h_{i,k}^{1_0}) \quad (220)$$

$$r_{i,x} \leq t_{k,x} + a_t + c_3(h_i^{1_0} + h_{i,k}^{1_0}) \quad (221)$$

$$t_{k,y} \leq r_{i,y} + c_3(h_i^{1_0} + h_{i,k}^{1_0}) \quad (222)$$

$$r_{i,y} \leq t_{k,y} + c_3(h_i^{1_0} + h_{i,k}^{1_0}) \quad (223)$$

$$t_{k,z} \leq r_{i,z} + c_3(h_i^{1_0} + h_{i,k}^{1_0}) \quad (224)$$

$$r_{i,z} \leq t_{k,z} + c_3(h_i^{1_0} + h_{i,k}^{1_0}) \quad (225)$$

$$t_{k,x} \leq r_{i,x} + c_3(h_i^{2_0} + h_{i,k}^{2_0}) \quad (226)$$

$$r_{i,x} \leq t_{k,x} + c_3(h_i^{2_0} + h_{i,k}^{2_0}) \quad (227)$$

$$t_{k,y} \leq r_{i,y} + c_3(h_i^{2_0} + h_{i,k}^{2_0}) \quad (228)$$

$$r_{i,y} \leq t_{k,y} + b_t + c_3(h_i^{2_0} + h_{i,k}^{2_0}) \quad (229)$$

$$t_{k,z} \leq r_{i,z} + c_3(h_i^{2_0} + h_{i,k}^{2_0}) \quad (230)$$

$$r_{i,z} \leq t_{k,z} + c_3(h_i^{2_0} + h_{i,k}^{2_0}) \quad (231)$$

$$t_{k,x} + a_t \leq r_{i,x} + c_3(h_i^{3_0} + h_{i,k}^{3_0}) \quad (232)$$

$$r_{i,x} \leq t_{k,x} + a_t + c_3(h_i^{3_0} + h_{i,k}^{3_0}) \quad (233)$$

$$t_{k,y} \leq r_{i,y} + c_3(h_i^{3_0} + h_{i,k}^{3_0}) \quad (234)$$

$$r_{i,y} \leq t_{k,y} + b_t + c_3(h_i^{3_0} + h_{i,k}^{3_0}) \quad (235)$$

$$t_{k,z} \leq r_{i,z} + c_3(h_i^{3_0} + h_{i,k}^{3_0}) \quad (236)$$

$$r_{i,z} \leq t_{k,z} + c_3(h_i^{3_0} + h_{i,k}^{3_0}) \quad (237)$$

$$t_{k,x} \leq r_{i,x} + c_3(h_i^{4_0} + h_{i,k}^{4_0}) \quad (238)$$

$$r_{i,x} \leq t_{k,x} + a_t + c_3(h_i^{4_0} + h_{i,k}^{4_0}) \quad (239)$$

$$t_{k,y} + b_t \leq r_{i,y} + c_3(h_i^{4_0} + h_{i,k}^{4_0}) \quad (240)$$

$$r_{i,y} \leq t_{k,y} + b_t + c_3(h_i^{4_0} + h_{i,k}^{4_0}) \quad (241)$$

$$t_{k,z} \leq r_{i,z} + c_3(h_i^{4_0} + h_{i,k}^{4_0}) \quad (242)$$

$$r_{i,z} \leq t_{k,z} + c_3(h_i^{4_0} + h_{i,k}^{4_0}). \quad (243)$$

The OR relation is modeled by the following equation for each $i \in [n+1, \dots, m]$

$$h_i^{1_0} + h_i^{2_0} + h_i^{3_0} + h_i^{4_0} \leq 3 \quad (244)$$

Additionally, satisfying the inequality

$$\sum_{k \in [m]} h_{i,k}^{1_0} \leq m - 1 \quad (245)$$

for all routers $i \in [n+1, \dots, m]$ ensures that each router is located in a tile. This can be further simplified using the next constraint about the sizes of tiles (cf. Section a.5.3). There the auxiliary binary variable $h_{i,j}$ is defined in Equation 247 that is 1 iff the router j is located

in tile i . Hence, the constraint can be simply modeled by $m - n + 1$ inequalities fore each router $j \in [n + 1, \dots, m]$:

$$\sum_{i \in [m]} h_{i,j} \geq 1. \quad (246)$$

Size of tiles

The size of each tile must be larger than the summed size of its component, routers and TSVs. Using the same indexes of routers, tiles, and components and the modified function f_c , this yields m inequalities for all $i \in [m]$:

$$\begin{aligned} a_i b_i \geq & f_c(i, \tau(r_{i,z})) + h_{i \text{ Router}} f_{R2D}(\tau(r_{i,z})) + \\ & h_{i \text{ TSV}} (f_{R3D}(\tau(r_{i,z})) - f_{R2D}(\tau(r_{i,z}))) + h_{i \text{ KOZ}} f_{\text{KOZ}} \end{aligned} \quad (247)$$

This requires the three auxiliary integer variables $h_{i \text{ Router}}$, $h_{i \text{ TSV}}$, and $h_{i \text{ KOZ}}$ which contain the number of routers and TSVs (KOZs) in a tile. Their definition is given in Section a.6.10, Section a.6.9, and Section a.6.8. Using the variables defined in the appendix, Equation 247 is given by m inequalities for all $i \in [m]$:

$$\begin{aligned} a_i b_i \geq & f_c(i, \tau(r_{i,z})) + h_{i \text{ Router}} f_{R2D}(\tau(r_{i,z})) + \\ & h_{i \text{ TSV}} (f_{R3D}(\tau(r_{i,z})) - f_{R2D}(\tau(r_{i,z}))) + h_{i \text{ KOZ}} f_{\text{KOZ}} \quad \text{or } t_i = \varphi. \end{aligned} \quad (248)$$

This equation is linearized according to Section 6.3.4.2. For $h_{\alpha,\beta,\gamma}^{\xi,i,11}$, the variables $h_{i \text{ Router}}$, $h_{i \text{ TSV}}$, and $h_{i \text{ KOZ}}$ can be re-used. The following two constraints must be satisfied. First, only one element of each matrix $H^{\xi,i,11}$ is allowed to be one using these m inequalities:

$$\sum_{\xi \in [\ell], \alpha, \beta, \gamma \in [m-n+1]} h_{\alpha,\beta,\gamma}^{\xi,i,11} \leq 1 \quad (249)$$

In addition, the implication "tile i is in layer ξ and has $(\alpha - 1)$ routers, $(\beta - 1)$ 3D-routers, and $(\gamma - 1)$ KOZs $\rightarrow h_{\alpha,\beta,\gamma}^{\xi,i,11} = 1$ " $\leftrightarrow t_{i,z} \neq \xi$ or $h_{i \text{ Router}} \neq (\alpha - 1)$ or $h_{i \text{ TSV}} \neq (\beta - 1)$ or $h_{i \text{ KOZ}} \neq (\gamma - 1)$ or $h_{\alpha,\beta,\gamma}^{\xi,i,11} = 1$ must be modeled. This yields $9m\ell(m - n + 1)(m - n + 1)$ inequalities for all $i \in [m]$, $\xi \in [\ell]$, $\alpha \in [m - n + 1]$, $\beta \in [m - n + 1]$, and $\gamma \in [m - n + 1]$:

$$1/2 + t_{i,z} \leq \xi + h_{\alpha,\beta,1}^{\xi,i,11} c_3, \quad (250)$$

$$1/2 + \xi \leq t_{i,z} + h_{\alpha,\beta,2}^{\xi,i,11} c_3, \quad (251)$$

$$1/2 + h_{i \text{ Router}} \leq (\alpha - 1) + h_{\alpha,\beta,3}^{\xi,i,11} c_3, \quad (252)$$

$$1/2 + (\alpha - 1) \leq h_{i \text{ Router}} + h_{\alpha,\beta,4}^{\xi,i,11} c_3, \quad (253)$$

$$1/2 + h_{i \text{ TSV}} \leq (\beta - 1) + h_{\alpha,\beta,5}^{\xi,i,11} c_3, \quad (254)$$

$$1/2 + (\beta - 1) \leq h_{i \text{ TSV}} + h_{\alpha,\beta,6}^{\xi,i,11} c_3, \quad (255)$$

$$1/2 + h_{i \text{ KOZ}} \leq (\gamma - 1) + h_{\alpha,\beta,7}^{\xi,i,11} c_3, \quad (256)$$

$$1/2 + (\gamma - 1) \leq h_{i \text{ KOZ}} + h_{\alpha,\beta,8}^{\xi,i,11} c_3, \quad (257)$$

$$\begin{aligned} h_{\alpha,\beta,1}^{\xi,i,11} + h_{\alpha,\beta,2}^{\xi,i,11} + h_{\alpha,\beta,3}^{\xi,i,11} + h_{\alpha,\beta,4}^{\xi,i,11} + h_{\alpha,\beta,5}^{\xi,i,11} + \\ h_{\alpha,\beta,6}^{\xi,i,11} + h_{\alpha,\beta,7}^{\xi,i,11} + h_{\alpha,\beta,8}^{\xi,i,11} + (1 - h_{\alpha,\beta,\gamma}^{\xi,i,11}) \leq 8 \end{aligned} \quad (258)$$

In addition, not placed tiles i in layer 0 must force all values of their matrices $H^{\xi,i,11}$ to zero by the following $\ell(m-n+1)^3$ inequalities:

$$h_{\alpha,\beta,\gamma}^{\xi,i,11} \leq t_{i,z} \quad (259)$$

The actual inequalities for this model here are skipped due to brevity.

Tiles do not overlap

There may not exist pairs of placed tiles with index i and j of which the areas intersect:

$$\begin{aligned} \forall i \in [m] : t_i = \varphi \text{ or } (\forall j \in [m], i \neq j : t_j = \varphi \\ \text{or } A_{t_i} \cap A_{t_j} = \emptyset) \end{aligned} \quad (260)$$

$$\begin{aligned} \leftrightarrow \forall (i, j) \in [m] \times [m], i \neq j : t_i = \varphi \text{ or } t_j = \varphi \\ \text{or } (A_{t_i} \cap A_{t_j} = \emptyset) \end{aligned} \quad (261)$$

The intersection of the areas $A_{t_i} \cap A_{t_j}$ separates into two cases: First, it can only occur in the same layer, i. e. it will always be false if $t_{i,z} \neq t_{j,z}$. Second, if the areas are located in the same layer, they will only overlap if and only if their x - and y -dimension both overlap as shown in Figure 99. This yields:

$$\begin{aligned} \forall (i, j) \in [m] \times [m], i \neq j : t_i = \varphi \text{ or } t_j = \varphi \text{ or } t_{i,z} \neq t_{j,z} \text{ or} \\ ([t_{i,x}, t_{i,x} + a_i] \cap [t_{j,x}, t_{j,x} + a_j] = \emptyset \text{ and} \\ [t_{i,y}, t_{i,y} + b_i] \cap [t_{j,y}, t_{j,y} + b_j] = \emptyset) \end{aligned} \quad (262)$$

The equation $[t_{i,x}, t_{i,x} + a_i] \cap [t_{j,x}, t_{j,x} + a_j] = \emptyset$ can be transformed into $\forall p \in [t_{i,x}, t_{i,x} + a_i] : p \notin [t_{j,x}, t_{j,x} + a_j]$. It is sufficient to test the borders of the intervals:

Lemma a.2. *The closed $I_1 = [a, b] \subset P$ and $I_2 = [c, d] \subset P$ overlap iff $c \leq b$ and $a \leq d$.*

Proof. The intervals $I_1 = (a, b) \subset P$ and $I_2 = (c, d) \subset P$ overlap, i. e. $I_1 \cap I_2 \neq \emptyset$, if and only if $\exists p \in P$ with $p \in I_1$ and $p \in I_2$. For p it holds that: $a < p < b$ and $c < p < d$. Therefore, this two conditions are met iff the intervals overlap: $c < b$ and $a < d$. Consideration of the boarder cases leads to the extension for closed intervals. \square

Using this lemma, the areas of the tiles overlap iff

$$\begin{aligned} (t_{j,x} \leq t_{i,x} + a_i \text{ and } t_{i,x} \leq t_{j,x} + a_j \text{ and} \\ t_{j,y} \leq t_{i,y} + b_i \text{ and } t_{i,y} \leq t_{j,y} + b_j) \end{aligned} \quad (263)$$

Using negation and DeMorgan's Law, Equation 262 yields:

$$\begin{aligned} \forall (i, j) \in [m] \times [m], i \neq j : t_i = \varphi \text{ or } t_j = \varphi \text{ or } t_{i,z} \neq t_{j,z} \text{ or} \\ (t_{j,x} \geq t_{i,x} + a_i + \delta \text{ or } t_{i,x} \geq t_{j,x} + a_j + \delta \text{ or} \\ t_{j,y} \geq t_{i,y} + b_i + \delta \text{ or } t_{i,y} \geq t_{j,y} + b_j + \delta) \end{aligned} \quad (264)$$

We introduce for all $i \in [n]$ and for all $j \in [n]$ with $i \neq j$ binary auxiliary variables $h_{i,j,1}^{1,2}$, $h_{i,j,2}^{1,2}$, $h_{i,j,3}^{1,2}$, $h_{i,j,4}^{1,2}$, $h_{i,j,5}^{1,2}$, $h_{i,j,6}^{1,2}$, $h_{i,j,7}^{1,2}$, and $h_{i,j,8}^{1,2} \in$

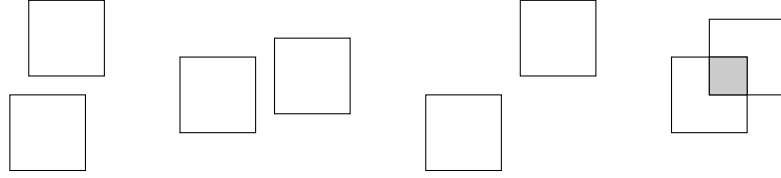


Figure 99: Only of the x - and y -dimension of the areas overlap, there is a non-empty intersection.

$\{0, 1\}$. This yields $11(m^2 - m)$ inequalities for all $(i, j) \in [m] \times [m]$ with $i \neq j$:

$$t_{i,z} \leq \varphi_z + h_{i,j}^{1z} c_1 \quad (265)$$

$$t_{j,z} \leq \varphi_z + h_{i,j}^{2z} c_1 \quad (266)$$

$$t_{i,z} \leq t_{j,z} - 1/2 + h_{i,j}^{3z} c_1 \quad (267)$$

$$t_{j,z} \leq t_{i,z} - 1/2 + h_{i,j}^{4z} c_1 \quad (268)$$

$$t_{j,x} \geq t_{i,x} + a_i + \delta - h_{i,j}^{5x} c_1 \quad (269)$$

$$t_{i,x} \geq t_{j,x} + a_j + \delta - h_{i,j}^{6x} c_1 \quad (270)$$

$$t_{j,y} \geq t_{i,y} + b_i + \delta - h_{i,j}^{7y} c_1 \quad (271)$$

$$t_{i,y} \geq t_{j,y} + b_j + \delta - h_{i,j}^{8y} c_1 \quad (272)$$

$$h_{i,j}^{1z} + h_{i,j}^{2z} + h_{i,j}^{3z} + h_{i,j}^{4z} + h_{i,j}^{5x} + h_{i,j}^{6x} + h_{i,j}^{7y} + h_{i,j}^{8y} \leq 7 \quad (273)$$

Components have different locations

All components with index $i \in [n]$ and $j \in [n], j \neq i$ must have different locations. Due to the indexing and the non-overlapping property of tiles, this constraint is already modeled. Components have a minimum distance of δ .

Links are not allowed to intersect tiles

Links are not allowed to intersect tiles, i. e. the line between two connected routers with index $i \in [m]$ and $j \in [m], j \neq i$ must not intersect a tile area:

$$\begin{aligned} \forall i, j \in [m] \times [m], i \neq j : e_{\{i,j\}} = 1 &\rightarrow (r_{i,z} = r_{j,z} \\ &\rightarrow \forall t \in T : \tilde{A}_t \cap L(r_i, r_j) = \emptyset) \end{aligned} \quad (274)$$

$$\begin{aligned} \leftrightarrow \forall i, j \in [m] \times [m], i \neq j : e_{\{i,j\}} = 0 &\text{ or } r_{i,z} \neq r_{j,z} \\ &\text{ or } \forall t \in T : \tilde{A}_t \cap L(r_i, r_j) = \emptyset \end{aligned} \quad (275)$$

The set equality $\tilde{A}_t \cap L(r_i, r_j) = \emptyset$ yields inequalities since a link and a tile do not overlap iff the link is left, right, below, or above the tile. Modeling this requires the auxiliary binary variables $h_{i,j,k}^{14}$, $h_{i,j,k}^{24}$, $h_{i,j,k}^{34}$, $h_{i,j,k}^{44}$, $h_{i,j,k}^{54}$, $h_{i,j,k}^{64}$, and $h_{i,j,k}^{74}$. There are $13m(m-1)(m-$

2) inequalities for all $k \in [m] \setminus \{i, j\}$. The link between routers r_i and r_j intersects the tile with index k if one of these conditions is met:

$$r_{i,x} \leq t_{k,x} + h_{i,j,k}^{14} c_1 + h_{i,j}^{14} \text{OR} c_1 \quad (\text{above}) \quad (276)$$

$$r_{j,x} \leq t_{k,x} + h_{i,j,k}^{14} c_1 + h_{i,j}^{14} \text{OR} c_1 \quad (\text{above}) \quad (277)$$

$$r_{i,y} \leq t_{k,y} + h_{i,j,k}^{14} c_2 + h_{i,j}^{14} \text{OR} c_1 \quad (\text{left}) \quad (278)$$

$$r_{j,y} \leq t_{k,y} + h_{i,j,k}^{14} c_2 + h_{i,j}^{14} \text{OR} c_1 \quad (\text{left}) \quad (279)$$

$$r_{i,x} \geq t_{k,x} + a_k - h_{i,j,k}^{14} c_3 - h_{i,j}^{14} \text{OR} c_1 \quad (\text{below}) \quad (280)$$

$$r_{j,x} \geq t_{k,x} + a_k - h_{i,j,k}^{14} c_3 - h_{i,j}^{14} \text{OR} c_1 \quad (\text{below}) \quad (281)$$

$$r_{i,y} \geq t_{k,y} + b_k - h_{i,j,k}^{14} c_4 - h_{i,j}^{14} \text{OR} c_1 \quad (\text{right}) \quad (282)$$

$$r_{j,y} \geq t_{k,y} + b_k - h_{i,j,k}^{14} c_4 - h_{i,j}^{14} \text{OR} c_1 \quad (\text{right}) \quad (283)$$

It remains to model whether the tile and the routers are located on the same layer, if $r_{i,z} \neq r_{j,z}$, and the OR relation between the link locations:

$$1/2 + t_{k,z} \leq r_{j,z} + h_{i,j,k}^{14} c_5 + h_{i,j}^{14} \text{OR} c_1 \quad (284)$$

$$1/2 + r_{j,z} \leq t_{k,z} + h_{i,j,k}^{14} c_6 + h_{i,j}^{14} \text{OR} c_1 \quad (285)$$

$$1/2 + r_{i,z} \leq r_{j,z} + h_{i,j,k}^{14} c_7 + h_{i,j}^{14} \text{OR} c_1 \quad (286)$$

$$1/2 + r_{j,z} \leq r_{i,z} + h_{i,j,k}^{14} c_8 + h_{i,j}^{14} \text{OR} c_1 \quad (287)$$

$$h_{i,j,k}^{14} + h_{i,j,k}^{14} + \dots + h_{i,j,k}^{14} \leq 7. \quad (288)$$

Finally, there must be a link between the two routers for all $i, j \in [m], i \neq j$:

$$e_{\{i,j\}} \leq (1 - h_{i,j}^{14} \text{OR}) c_1^{14}. \quad (289)$$

All in all, this constraint adds $13m(m-1)(m-2) + m(m-1)$ inequalities.

Routers have different locations

Placed routers are not allowed to be at the same position. Routers with index $i \in [n]$ are located at different positions due to forbidden intersections between tiles. Only routers with indexes larger n are relevant here. Therefore, it holds for all $(i, j) \in \{n+1, \dots, m\} \times [m]$:

$$r_i \neq r_j \text{ OR } r_i = \varphi. \quad (290)$$

This translates into the following $4(m-1)m$ inequalities using the auxiliary binary variable $h_{i,j}^{14} \text{OR}$ and $h_{i,j,k}^{14}$ and the constant $c = x_{\max} + y_{\max} + l$ for all $i, j \in [m] \times [m], i \neq j$:

$$\begin{aligned} r_{i,z} = \varphi_z \text{ OR } r_{i,x} > r_{j,x} \text{ OR } r_{i,x} < r_{j,x} \text{ OR } r_{i,y} > r_{j,y} \\ \text{OR } r_{i,y} < r_{j,y} \text{ OR } r_{i,z} > r_{j,z} \text{ OR } r_{i,z} < r_{j,z} \end{aligned} \quad (291)$$

Since the positions of the routers are vectors, this in total yields these $9(m-1)m$ inequalities using the binary auxiliary variables $h_{\neq i,j,1}^{15}$, $h_{\neq i,j,2}^{15}$, $h_{\neq i,j,3}^{15}$, $h_{\neq i,j,4}^{15}$, $h_{\neq i,j,5}^{15}$, $h_{\neq i,j,6}^{15}$, and $h_{i,j}^{14} \text{OR}$:

$$r_{i,z} \leq \varphi_z + h_{i,j}^{14} \text{OR} c_3 \quad (292)$$

$$\varphi_z \leq r_{i,z} + h_{i,j}^{14} \text{OR} c_3 \quad (293)$$

$$\delta + r_{i,x} \leq r_{j,x} + h_{\neq i,j,1}^{15} c_3 \quad (294)$$

$$\delta + r_{j,x} \leq r_{i,x} + h_{\neq i,j,2}^{15} c_3 \quad (295)$$

$$\delta + r_{i,y} \leq r_{j,y} + h_{\neq i,j,3}^{15} c_3 \quad (296)$$

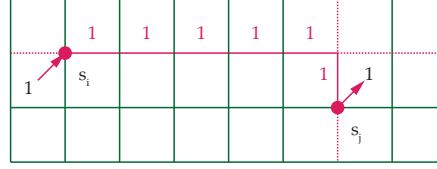


Figure 100: Constraints for 2D DOR. At the source s_i a flow of one is given into the network and it is removed from the network at destination s_j . Along the red path, the flow is 1. The green links have flow 0. The dotted red links could potentially have flow 1 following the constraints from Equation 301. However, the flow from these edges cannot be removed due to flow conservation, binary flows and acyclic flows. Therefore, these links indirectly also have flow 0.

$$\delta + r_{j,y} \leq r_{i,y} + h_{\neq i,j,4}^{15} c_3 \quad (297)$$

$$1/2 + r_{i,z} \leq r_{j,z} + h_{\neq i,j,5}^{15} c_3 \quad (298)$$

$$1/2 + r_{j,z} \leq r_{i,z} + h_{\neq i,j,6}^{15} c_3 \quad (299)$$

$$h_{\neq i,j,1}^{15} + h_{\neq i,j,2}^{15} + h_{\neq i,j,3}^{15} + h_{\neq i,j,4}^{15} + h_{\neq i,j,5}^{15} + h_{\neq i,j,6}^{15} + h_{i,j}^{14} \text{ OR} \leq 6 \quad (300)$$

A.5.4 Constraints for elevator first dimension order routing

A.5.4.1 Flow as binary variable

For “extended” DOR the flow must be binary since only a single path through the network is valid per source and destination pair. Thus, it must hold, that for all $i, j, k, l \in [m]$: $f_{(k,l)}^{(i,j)} \in \{0, 1\}$.

Flow $f(i, j)$ for DOR

The variables of the flow $f_{(k,l)}^{(i,j)}$ must be set within the network to follow elevator-first DOR. As a prerequisite, we consider 2D DOR. This is depicted in Figure 100. 2D DOR starts by minimizing the difference of the packet’s destination and location first in y -dimension and then in x -dimension. Thus, during transmission, the x -coordinate of the packet remains the same until the packet’s transmission direction changes. Then, the y -coordinate is static. Thus, for an edge (k, l) on the packet’s path from component s_i to s_j it must hold:

$$s_{i,x} = r_{k,x} = r_{l,x} \text{ OR } s_{j,y} = r_{k,y} = r_{l,y} \quad (301)$$

The flow-constraints, namely the definition of a flow (Section 6.3.4.3), flow conservation (Equation 6.3.4.3), acyclic flows (Equation 6.3.4.3), and the binary flow variable for DOR (Section a.5.4.1) ensure that only the desired flow following 2D DOR can be selected from source to destination with the aforementioned constraints. In case of elevator-first DOR, the source component s_i and the destination component s_j can be located in any two arbitrary layers in the chip. Therefore two auxiliary variables \tilde{s}_{ξ}^{ij} and $\tilde{d}_{\xi}^{ij} \in P$ are introduced for each layer

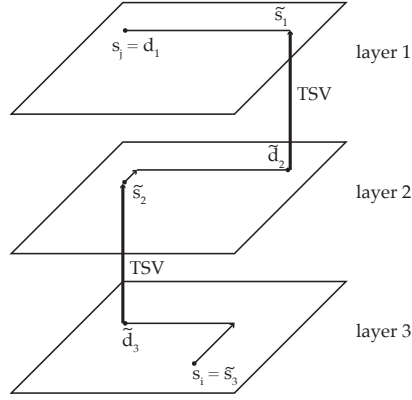


Figure 101: A packet travels from the lowermost to the uppermost layer via two TSVs. The variables \tilde{s}_ξ^{ij} and \tilde{d}_ξ^{ij} are set to the positions of routers, which revive or send the packet through the TSVs.

$\xi \in [\ell]$ and per source-destination-pair $i, j \in [m]$ that represent the start- and endpoint of the 2D DOR, which is performed in this layer⁴. This is shown in Figure 101. A packet travels from the lowermost to the uppermost layer via two TSVs. The variables \tilde{s}_ξ^{ij} and \tilde{d}_ξ^{ij} are set to the positions of routers, which receive or send the packet through the TSVs. For these variables the following conditions must be fulfilled for the upward transmission direction ($s_{i,z} > s_{j,z}$):

- In the start layer $s_{i,z}$ the packets start at the start component: $\tilde{s}_\xi^{ij} = s_i$.
- In the destination layer $s_{j,z}$ the packets must travel to the destination: $\tilde{d}_\xi^{ij} = s_j$.
- In layers, which are not traversed by packets, \tilde{s}_ξ^{ij} and \tilde{d}_ξ^{ij} are not set. Thus in layers ξ with $\xi > s_{i,z}$ and $\xi > s_{j,z}$ or $\xi < s_{i,z}$ and $\xi < s_{j,z}$ it holds that $\tilde{s}_\xi^{ij} = \tilde{d}_\xi^{ij} = \varphi$.
- For all (temporary) destinations \tilde{d}_ξ^{ij} in each traversed layer, a link from \tilde{d}_ξ^{ij} is directed upward for all $\xi \in \{s_{j,z} + 1, s_{j,z} + 2, \dots, s_{i,z}\}$ and $i, j \in [m]$. Thus, the link can be used to transmit packets.
- In layers, which are traversed by packets, the destination of the previous layer must be connected to the source of the next layer.
- For each (temporary) destination in each layer, which is used as a gateway, the packet travels the shortest possible distance. Thus, the distance between the source in this layer and each other router with a TSV in the correct direction must be larger: For all routers r_l with r_l has a downward TSV and is in a traversed layer ($r_{l,z} = \xi$) the following is valid:

$$\|r_l - \tilde{s}_\xi^{ij}\| \leq \|\tilde{d}_\xi^{ij} - \tilde{s}_\xi^{ij}\| \quad \forall \xi \in \{s_{j,z} + 1, \dots, s_{i,z}\}, i, j \in [m] \quad (302)$$

⁴ The z -component of \tilde{s}_ξ^{ij} and \tilde{d}_ξ^{ij} is ξ .

For the downward direction ($s_{i,z} < s_{j,z}$), the directions of the TSVs are altered.

Summing up, there are three cases: the packet travels upwards, downwards, or stays within the same layer. The third case can be seen as a special case of either case I or case II. We have chosen to assign the case, in which both routers are in the same layer, to case I (traveling upwards). In both of these cases the following must be valid:

- In source layer $s_{i,z} : \tilde{s}_\xi^{ij} = s_i$.
- In destination layer $s_{j,z} : \tilde{d}_\xi^{ij} = s_j$.
- The condition for 2D DOR holds for each edge $e_N = (k, l) \in E_N$:

$$\begin{aligned} f_{(k,l)}^{(i,j)} = 1 \rightarrow e_N > 0 \text{ and } r_{k,z} = r_{l,z} \\ \text{and } \tilde{s}_\xi^{ij} = r_{k,x} = r_{l,x} \text{ or } \tilde{d}_\xi^{ij} = r_{k,y} = r_{l,y} \end{aligned} \quad (303)$$

In the first case (I,) the packets travel upwards or stay within the same layer ($s_{i,z} \geq s_{j,z}$), these conditions must be fulfilled for all $i, j \in [m]$:

- The traversed layers are combined in the set $D_1 = \{s_{j,z} + 1, s_{j,z} + 2, \dots, s_{i,z}\}$. Note, that we are excluding the destination layer, i. e. the layer, in which the destination component s_j is located.
- For all $\xi \in D_1$ a link is heading upwards from \tilde{d}_ξ^{ij} .
- For all $\xi \in D_1$ the destination in the previous layer are identical with the source in the next layer. Thus, for all $k, l \in [m]$: $r_k = \tilde{d}_\xi^{ij}$ and $r_l = \tilde{s}_{\xi-1}^{ij}$ implies $e_{(k,l)} = 1$.
- For all routers r_l with r_l has a upward TSV and $r_{l,z} = \xi$: $\|r_l - \tilde{s}_\xi^{ij}\| \geq \|\tilde{d}_\xi^{ij} - \tilde{s}_\xi^{ij}\|$.
- For each connection $\{k, l\}$ with the link (k, l) heading upwards (i. e. $r_{k,z} = r_{l,z} + 1$):

1. $f_{(l,k)}^{(i,j)} = 0$
2. If $r_k = \tilde{d}_{r_{k,z}}^{i,j}$ and $r_l = \tilde{s}_{r_{l,z}}^{i,j}$ and $r_{k,z} \leq s_{i,z}$ and $r_{l,z} \geq s_{j,z}$ then $f_{(k,l)}^{(i,j)} = 1$
3. The otherwise case (" $f_{(k,l)}^{(i,j)} = 0$ else") is fulfilled: Because of the flow conservation and the first condition ($f_{(l,k)}^{(i,j)} = 0$), the sum of flows into and out of layers is always zero except for the start and destination layer, which have flow 1 and -1 respectively. In addition, the flow between two traversed layers is 1. Using the second condition, a single TSV between two traversed layers has a flow of 1. Therefore, all other TSV are forced to flow 0.

In the second case (II), the packets travel downwards ($s_{i,z} < s_{j,z}$), these conditions must be fulfilled for all $i, j \in [m]$:

- The traversed layers are combined in the set $D_0 = \{s_{i,z}, s_{i,z} + 1, \dots, s_{j,z} - 1\}$. Again the destination layer is excluded.
- A link heads downwards from \tilde{d}_ξ^{ij} for all $\xi \in D_0$.

- For all $\xi \in D_1$ the destination in the previous layer are identical with the source in the next layer. Thus, for all $k, l \in [m]$: $r_k = \tilde{d}_\xi^{ij}$ and $r_l = \tilde{s}_{\xi+1}^{ij}$ implies $e_{(k,l)} = 1$.
- For all routers r_l with r_l has a downward link and $r_{l,z} = \xi$: $\|r_l - \tilde{s}_\xi^{ij}\| \geq \|\tilde{d}_\xi^{ij} - \tilde{s}_\xi^{ij}\|$.
- For each TSV $\{k, l\}$ with the link (k, l) heading upwards (i. e. $r_{k,z} = r_{l,z} + 1$):
 1. $f_{(k,l)}^{(i,j)} = 0$
 2. If $r_l = \tilde{d}_{r_{l,z}}^{ij}$ and $r_k = \tilde{s}_{r_{k,z}}^{ij}$ and $r_{l,z} \geq s_{i,z}$ and $r_{k,z} \leq s_{j,z}$: $f_{(l,k)}^{(i,j)} = 1$
 3. The otherwise case (" $f_{(l,k)}^{(i,j)} = 0$ else") is automatically fulfilled (with the same arguments).

We use the auxiliary variable $h_{ij}^{18}_{UP}$ from Section a.6.13 to model the packet travel direction.

To fulfill the five conditions, we start by modeling the first one, namely the *sets of traversed layers* D_0 and D_1 . Therefore, auxiliary variables $h_{ij}^{18}_{D\xi}$ are required indicating layers, in which a packet from any source component with index i to destination component with index j is transmitted. The variable $h_{ij}^{18}_{D\xi}$ is given in Section a.6.12.

Next, for case I we model the third condition for all $\xi \in D_1^5$, $k, l \in [m]$: $r_k = \tilde{d}_\xi^{ij}$ and $r_l = \tilde{s}_{\xi-1}^{ij}$ implies $e_{(k,l)} = 1$. These inequalities exist for all $\xi \in [\ell] \setminus \{1\}$:

$$r_{k,x} + \delta \leq \tilde{d}_{\xi,x}^{ij} + h_{ijk1\xi}^{18}c + (1 - h_{ijD\xi}^{18})c + (1 - h_{ijUP}^{18})c \quad (304)$$

$$\tilde{d}_{\xi,x}^{ij} + \delta \leq r_{k,x} + h_{ijk1\xi}^{18}c + (1 - h_{ijD\xi}^{18})c + (1 - h_{ijUP}^{18})c \quad (305)$$

$$r_{k,y} + \delta \leq \tilde{d}_{\xi,y}^{ij} + h_{ijk1\xi}^{18}c + (1 - h_{ijD\xi}^{18})c + (1 - h_{ijUP}^{18})c \quad (306)$$

$$\tilde{d}_{\xi,y}^{ij} + \delta \leq r_{k,y} + h_{ijk1\xi}^{18}c + (1 - h_{ijD\xi}^{18})c + (1 - h_{ijUP}^{18})c \quad (307)$$

$$r_{k,z} + 1/2 \leq \tilde{d}_{\xi,z}^{ij} + h_{ijk1\xi}^{18}c + (1 - h_{ijD\xi}^{18})c + (1 - h_{ijUP}^{18})c \quad (308)$$

$$\tilde{d}_{\xi,z}^{ij} + 1/2 \leq r_{k,z} + h_{ijk1\xi}^{18}c + (1 - h_{ijD\xi}^{18})c + (1 - h_{ijUP}^{18})c \quad (309)$$

$$r_{l,x} + \delta \leq \tilde{s}_{\xi-1,x}^{ij} + h_{ijk1\xi}^{18}c + (1 - h_{ijD\xi}^{18})c + (1 - h_{ijUP}^{18})c \quad (310)$$

$$\tilde{s}_{\xi-1,x}^{ij} + \delta \leq r_{l,x} + h_{ijk1\xi}^{18}c + (1 - h_{ijD\xi}^{18})c + (1 - h_{ijUP}^{18})c \quad (311)$$

$$r_{l,y} + \delta \leq \tilde{s}_{\xi-1,y}^{ij} + h_{ijk1\xi}^{18}c + (1 - h_{ijD\xi}^{18})c + (1 - h_{ijUP}^{18})c \quad (312)$$

$$\tilde{s}_{\xi-1,y}^{ij} + \delta \leq r_{l,y} + h_{ijk1\xi}^{18}c + (1 - h_{ijD\xi}^{18})c + (1 - h_{ijUP}^{18})c \quad (313)$$

$$r_{l,z} + 1/2 \leq \tilde{s}_{\xi-1,z}^{ij} + h_{ijk1\xi}^{18}c + (1 - h_{ijD\xi}^{18})c + (1 - h_{ijUP}^{18})c \quad (314)$$

$$\tilde{s}_{\xi-1,z}^{ij} + 1/2 \leq r_{l,z} + h_{ijk1\xi}^{18}c + (1 - h_{ijD\xi}^{18})c + (1 - h_{ijUP}^{18})c \quad (315)$$

$$\sum_{i=1}^{12} h_{ijk1\xi}^{18} + (1 - e_{(k,l)}) \leq 12 \quad (316)$$

The second case II requires modeling $r_k = \tilde{d}_\xi^{ij}$ and $r_l = \tilde{s}_{\xi+1}^{ij}$ implies $e_{(k,l)} = 1$. Thus, we have the following inequalities for all $\xi \in [\ell - 1]$ and $k, l \in [m]$:

$$r_{k,x} + \delta \leq \tilde{d}_{\xi,x}^{ij} + h_{ijk1\xi}^{18}c + (1 - h_{ijD\xi}^{18})c + (1 - h_{ijUP}^{18})c \quad (317)$$

$$\tilde{d}_{\xi,x}^{ij} + \delta \leq r_{k,x} + h_{ijk1\xi}^{18}c + (1 - h_{ijD\xi}^{18})c + (1 - h_{ijUP}^{18})c \quad (318)$$

5 For $\xi \notin D_1$, the inequalities are always true.

$$r_{k,y} + \delta \leq \tilde{d}_{\xi,y}^{ij} + h_{ijk\xi 15}^{18}c + (1 - h_{ijD\xi}^{18})c + (1 - h_{ijUP}^{18})c \quad (319)$$

$$\tilde{d}_{\xi,y}^{ij} + \delta \leq r_{k,y} + h_{ijk\xi 16}^{18}c + (1 - h_{ijD\xi}^{18})c + (1 - h_{ijUP}^{18})c \quad (320)$$

$$r_{k,z} + 1/2 \leq \tilde{d}_{\xi,z}^{ij} + h_{ijk\xi 17}^{18}c + (1 - h_{ijD\xi}^{18})c + (1 - h_{ijUP}^{18})c \quad (321)$$

$$\tilde{d}_{\xi,z}^{ij} + 1/2 \leq r_{k,z} + h_{ijk\xi 18}^{18}c + (1 - h_{ijD\xi}^{18})c + (1 - h_{ijUP}^{18})c \quad (322)$$

$$r_{l,x} + \delta \leq \tilde{s}_{\xi+1,x}^{ij} + h_{ijk\xi 19}^{18}c + (1 - h_{ijD\xi}^{18})c + (1 - h_{ijUP}^{18})c \quad (323)$$

$$\tilde{s}_{\xi+1,x}^{ij} + \delta \leq r_{l,x} + h_{ijk\xi 20}^{18}c + (1 - h_{ijD\xi}^{18})c + (1 - h_{ijUP}^{18})c \quad (324)$$

$$r_{l,y} + \delta \leq \tilde{s}_{\xi+1,y}^{ij} + h_{ijk\xi 21}^{18}c + (1 - h_{ijD\xi}^{18})c + (1 - h_{ijUP}^{18})c \quad (325)$$

$$\tilde{s}_{\xi+1,y}^{ij} + \delta \leq r_{l,y} + h_{ijk\xi 22}^{18}c + (1 - h_{ijD\xi}^{18})c + (1 - h_{ijUP}^{18})c \quad (326)$$

$$r_{l,z} + 1/2 \leq \tilde{s}_{\xi+1,z}^{ij} + h_{ijk\xi 23}^{18}c + (1 - h_{ijD\xi}^{18})c + (1 - h_{ijUP}^{18})c \quad (327)$$

$$\tilde{s}_{\xi+1,z}^{ij} + 1/2 \leq r_{l,z} + h_{ijk\xi 24}^{18}c + (1 - h_{ijD\xi}^{18})c + (1 - h_{ijUP}^{18})c \quad (328)$$

$$\sum_{i=13}^{24} h_{ijk\xi i}^{18} + (1 - e_{\{k,l\}}) \leq 12 \quad (329)$$

Next, we model: "A TSV is heading upwards (case I) (or downwards for case II) from location \tilde{d}_{ξ}^{ij} for all $\xi \in D_1$ (or D_0)". We use the properties of \hat{h}_k^+ and \hat{h}_k^- (cf. Section a.6.7). They indicate for router k if it has a vertical link downwards (or upwards). The destination routers \tilde{d}_{ξ}^{ij} need to be such routers.

For all routers $r_k \in [m]$ it holds for case I that if $r_k = \tilde{d}_{\xi}^{ij}$ for a $\xi \in D_1$, then $\hat{h}_k^- \geq 1$. Analog, for case II it holds if $r_k = \tilde{d}_{\xi}^{ij}$ for a $\xi \in D_0$, then $\hat{h}_k^+ \geq 1$. For case I this yields:

$$r_k = \tilde{d}_{\xi}^{ij} \quad \text{and case I and } \xi \in D_1 \Rightarrow \hat{h}_k^- \geq 1 \quad (330)$$

$$\Leftrightarrow r_k \neq \tilde{d}_{\xi}^{ij} \quad \text{or not case I or } \xi \notin D_1 \text{ or } \hat{h}_k^- \geq 1 \quad (331)$$

$$\Leftrightarrow r_k \neq \tilde{d}_{\xi}^{ij} \quad \text{or } h_{ijUP}^{18} = 0 \text{ or } h_{ijD\xi}^{18} = 0 \text{ or } \hat{h}_k^- \geq 1 \quad (332)$$

$$\Leftrightarrow r_k < \tilde{d}_{\xi}^{ij} \text{ or } r_k > \tilde{d}_{\xi}^{ij} \text{ or } h_{ijUP}^{18} = 0 \text{ or } h_{ijD\xi}^{18} = 0 \text{ or } \hat{h}_k^- \geq 1 \quad (333)$$

$$\begin{aligned} \Leftrightarrow r_{k,x} < \tilde{d}_{\xi,x}^{ij} \text{ or } r_{k,x} > \tilde{d}_{\xi,x}^{ij} \text{ or } r_{k,y} < \tilde{d}_{\xi,y}^{ij} \text{ or} \\ r_{k,y} > \tilde{d}_{\xi,y}^{ij} \text{ or } r_{k,z} < \xi \text{ or } r_{k,z} > \xi \text{ or} \\ h_{ijUP}^{18} = 0 \text{ or } h_{ijD\xi}^{18} = 0 \text{ or } \hat{h}_k^- \geq 1 \end{aligned} \quad (334)$$

This yields the following $7(m-1)|E_A|$ inequalities for all $(i,j) \in E_A$, $\xi \in [\ell]$, and $k \in [m] \setminus \{j\}$ ⁶ using the auxiliary binary variables $h_{ijk\xi 1}^{18}$, $h_{ijk\xi 2}^{18}$, $h_{ijk\xi 3}^{18}$, $h_{ijk\xi 4}^{18}$, $h_{ijk\xi 5}^{18}$, and $h_{ijk\xi 6}^{18}$:

$$1/2 + r_{k,x} \leq \tilde{d}_{\xi,x}^{ij} + h_{ijk\xi 1}^{18}c, \quad (335)$$

$$1/2 + r_{k,y} \leq \tilde{d}_{\xi,y}^{ij} + h_{ijk\xi 2}^{18}c, \quad (336)$$

$$1/2 + r_{k,z} \leq \xi + h_{ijk\xi 3}^{18}c, \quad (337)$$

$$r_{k,x} \geq \tilde{d}_{\xi,x}^{ij} + 1/2 - h_{ijk\xi 4}^{18}c, \quad (338)$$

$$r_{k,y} \geq \tilde{d}_{\xi,y}^{ij} + 1/2 - h_{ijk\xi 5}^{18}c, \quad (339)$$

$$r_{k,z} \geq \xi + 1/2 - h_{ijk\xi 6}^{18}c, \quad (340)$$

$$h_{ijk\xi 1}^{18} + \dots + h_{ijk\xi 6}^{18} + h_{ijUP}^{18} + h_{ijD\xi}^{18} + (1 - \hat{h}_k^-) \leq 8 \quad (341)$$

⁶ Router with index j can be excluded since its layer is not in D_0 (or D_1 depending on the case).

For case II the analog is given:

$$\begin{aligned} &\Leftrightarrow r_{k,x} < \tilde{d}_{\xi,x}^{ij} \text{ or } r_{k,x} > \tilde{d}_{\xi,x}^{ij} \text{ or } r_{k,y} < \tilde{d}_{\xi,y}^{ij} \text{ or} \\ &\quad r_{k,y} > \tilde{d}_{\xi,y}^{ij} \text{ or } r_{k,z} < \xi \text{ or } r_{k,z} > \xi \text{ or} \\ &\quad h_{ijUP}^{18} = 0 \text{ or } h_{ijD\xi}^{18} = 1 \text{ or } \hat{h}_k^+ \geq 1 \end{aligned} \quad (342)$$

This adds one additional inequality for all $(i, j) \in E_A$, $\xi \in [\ell]$, and $k \in [m] \setminus \{j\}$

$$\begin{aligned} &h_{ijk\xi 1}^{18} + h_{ijk\xi 2}^{18} + h_{ijk\xi 3}^{18} + h_{ijk\xi 4}^{18} + h_{ijk\xi 5}^{18} + h_{ijk\xi 6}^{18} \\ &\quad + (1 - h_{ijUP}^{18}) + h_{ijD\xi}^{18} + (1 - \hat{h}_k^+) \leq 8 \end{aligned} \quad (343)$$

Summing up, $8(m-1)|E_A|$ inequalities are given here.

Now, the fourth condition about the *distance of the gateway to other routers* is modeled. The condition is given for all $(i, j) \in E_A$, for all routers $\iota \in [m] \setminus \{j\}$, and for all $\xi \in [\ell]$ by $\hat{h}_\iota^- = 1$ and $r_{\iota,z} = \xi$, and $\xi \in D_1$ (i. e. $h_{ijD\xi}^{18} = 1$) and $h_{ijUP}^{18} = 1$ (i. e. indicating case I) $\Rightarrow \|r_\iota - \tilde{s}_\xi^{ij}\| \geq \|\tilde{d}_\xi^{ij} - \tilde{s}_\xi^{ij}\|$ with $\|\cdot\|$ denoting the Manhattan norm. This can be rewritten:

$$\begin{aligned} &\hat{h}_\iota^- = 0 \text{ or } r_{\iota,z} \neq \xi \text{ or } h_{ijD\xi}^{18} = 0 \text{ or } h_{ijUP}^{18} = 0 \\ &\text{or } \|r_\iota - \tilde{s}_\xi^{ij}\| \geq \|\tilde{d}_\xi^{ij} - \tilde{s}_\xi^{ij}\| \end{aligned} \quad (344)$$

Next, the norm distance $\|r_\iota - \tilde{s}_\xi^{ij}\| \geq \|\tilde{d}_\xi^{ij} - \tilde{s}_\xi^{ij}\|$ must be brought into the MILP model. Due to the grid-based network topology within each layer the Manhattan norm is used. $\|r_i - r_j\|$ denotes the hop distance between routers i and j of the same layer.

$$\|r_\iota - \tilde{s}_\xi^{ij}\|_1 \geq \|\tilde{d}_\xi^{ij} - \tilde{s}_\xi^{ij}\|_1 \quad (345)$$

$$\begin{aligned} &\Leftrightarrow |r_{\iota,x} - \tilde{s}_{\xi,x}^{ij}| + |r_{\iota,y} - \tilde{s}_{\xi,y}^{ij}| + |r_{\iota,z} - \tilde{s}_{\xi,z}^{ij}| \geq \\ &\quad |\tilde{d}_{\xi,x}^{ij} - \tilde{s}_{\xi,x}^{ij}| + |\tilde{d}_{\xi,y}^{ij} - \tilde{s}_{\xi,y}^{ij}| + |\tilde{d}_{\xi,z}^{ij} - \tilde{s}_{\xi,z}^{ij}| \end{aligned} \quad (346)$$

The difference in the z -dimension is always zero, thus $|r_{\iota,z} - \tilde{s}_{\xi,z}^{ij}| = 0$ and $|\tilde{d}_{\xi,z}^{ij} - \tilde{s}_{\xi,z}^{ij}| = |\xi - \xi| = 0$. For the other terms of the sum an auxiliary absolute value "function" $\text{dist}(\cdot, \cdot)$ is introduced, which is given by $\text{dist}(a, b) = |a - b|$. The dist -function is given by these six inequalities which must be satisfied in the MILP model using the auxiliary binary variable $h_{a \leq b}$ and the constant c ⁷:

$$\text{dist}(a, b) \geq a - b \quad (347)$$

$$\text{dist}(a, b) \geq b - a \quad (348)$$

$$a \leq b + (1 - h_{a \leq b})c \quad (349)$$

$$b \leq a + h_{a \leq b}c \quad (350)$$

$$\text{dist}(a, b) \leq a - b + h_{a \leq b}c \quad (351)$$

$$\text{dist}(a, b) \leq b - a + (1 - h_{a \leq b})c \quad (352)$$

⁷ Again, we exploit that a and b are bounded. Hence, dist is a function from $[0, x_{\max} + y_{\max}] \times [0, x_{\max} + y_{\max}]$ which maps to $[0, x_{\max} + y_{\max}]$

Here, the dist-function values are auxiliary variables. These exist for those very cases in which they are required in:

$$\text{dist}(r_{\iota,x}, \tilde{s}_{\xi,x}^{ij}) \quad \forall (i,j) \in E_A, \iota \in [m] \setminus \{j\}, \xi \in [\ell] \quad (353)$$

$$\text{dist}(r_{\iota,y}, \tilde{s}_{\xi,y}^{ij}) \quad \forall (i,j) \in E_A, \iota \in [m] \setminus \{j\}, \xi \in [\ell] \quad (354)$$

$$\text{dist}(\tilde{d}_{\xi,x}^{ij}, \tilde{s}_{\xi,x}^{ij}) \quad \forall (i,j) \in E_A, \xi \in [\ell] \quad (355)$$

$$\text{dist}(\tilde{d}_{\xi,y}^{ij}, \tilde{s}_{\xi,y}^{ij}) \quad \forall (i,j) \in E_A, \xi \in [\ell] \quad (356)$$

For each of these auxiliary variables, 6 further inequalities are required to model the dist-function value. Therefore, a total of $12|E_A|\ell(m-1) + 12|E_A|\ell$ inequalities are given. Using these auxiliary variables, Equation 346 can be written as:

$$\text{dist}(r_{\iota,x}, \tilde{s}_{\xi,x}^{ij}) + \text{dist}(r_{\iota,y}, \tilde{s}_{\xi,y}^{ij}) \geq \text{dist}(\tilde{d}_{\xi,x}, \tilde{s}_{\xi,x}^{ij}) + \text{dist}(\tilde{d}_{\xi,y}, \tilde{s}_{\xi,y}^{ij}) \quad (357)$$

Continuing, the term 344 is given for case (I) by the inequalities using the binary auxiliary variables $h_{\iota\xi}^{18}_{\text{OR}1}$, $h_{\iota\xi 1}^{18}$, and $h_{ij\iota\xi 2}^{18}$ for all $\iota \in [m]$, and $\xi \in [\ell]$:

$$r_{\iota,z} \geq \xi + 1/2 - h_{\iota\xi}^{18}_{\text{OR}1}c - h_{\iota\xi 1}^{18}c, \quad (358)$$

$$r_{\iota,z} \leq \xi - 1/2 + (1 - h_{\iota\xi}^{18}_{\text{OR}1})c + h_{\iota\xi 1}^{18}c, \quad (359)$$

Furthermore, for all $(i,j) \in E_A$, $\iota \in [m]$, and $\xi \in [\ell]$ it is given:

$$\text{dist}(r_{\iota,x}, \tilde{s}_{\xi,x}^{ij}) + \text{dist}(r_{\iota,y} - \tilde{s}_{\xi,y}^{ij}) \geq \quad (360)$$

$$\text{dist}(\tilde{d}_{\xi,x}^{ij} - \tilde{s}_{\xi,x}^{ij}) + \text{dist}(\tilde{d}_{\xi,x}^{ij} - \tilde{s}_{\xi,x}^{ij}) - h_{ij\iota\xi 2}^{18}c$$

$$\hat{h}_{\iota}^- + h_{\iota\xi 1}^{18} + h_{ijD\xi}^{18} + h_{ijUP}^{18} + h_{ij\iota\xi 2}^{18} \leq 4 \quad (361)$$

Case II is modeled by condition for all routers $\iota \in [m] \setminus \{j\}$ and for all $\xi \in [\ell]$ by $\hat{h}_{\iota}^+ = 1$ and $r_{\iota,z} = \xi$ and $\xi \in D_0$ (i.e. $h_{ijD\xi}^{18} = 1$) and $h_{ijUP}^{18} = 0$ (indicating case I) implies $\|r_{\iota} - \tilde{s}_{\xi}^{ij}\| \geq \|\tilde{d}_{\xi}^{ij} - \tilde{s}_{\xi}^{ij}\|$. This can be rewritten:

$$\hat{h}_{\iota}^+ = 0 \text{ or } r_{\iota,z} \neq \xi \text{ or } h_{ijD\xi}^{18} = 0 \text{ or } h_{ijUP}^{18} = 1$$

$$\text{or } \|r_{\iota} - \tilde{s}_{\xi}^{ij}\| \geq \|\tilde{d}_{\xi}^{ij} - \tilde{s}_{\xi}^{ij}\| \quad (362)$$

Thus, the equations 358 to 360 are identical in this case, only the following must be added for all $(i,j) \in E_A$, $\iota \in [m] \setminus \{j\}$, and $\xi \in [\ell]$:

$$\hat{h}_{\iota}^+ + h_{\iota\xi 1}^{18} + h_{ijD\xi}^{18} + (1 - h_{ijUP}^{18}) + h_{ij\iota\xi 2}^{18} \leq 4 \quad (363)$$

Summing up, $3|E_A|(m-1)\ell + 2(m-1)\ell$ inequalities are given.

Next, the fifth condition is modeled concerning *flow in the network*. The flow from layers to layers in the opposite direction of the source to destination must be set to zero. Therefore, in case I for each TSV $\{k, l\}$ which is an upwards connection from router k to router l , the flow $f_{(l,k)}^{(i,j)} = 0$. Hence, the conditions can be formulated as follows:

$$h_{ijUP}^{18} = 1 \text{ and } e_{\{k,l\}} = 1 \text{ and } r_{k,z} = r_{l,z+1} \Rightarrow f_{(l,k)}^{(i,j)} = 0 \quad (364)$$

$$\Leftrightarrow h_{ijUP}^{18} = 0 \text{ or } e_{\{k,l\}} = 0 \text{ or } r_{k,z} \neq r_{l,z+1} \text{ or } f_{(l,k)}^{(i,j)} = 0 \quad (365)$$

These condition yields the inequalities using the auxiliary binary variables $h_{ijk\ell}^{18}$ and h_{ijkl}^{18} for all $(i, j) \in E_A$, and for all $k, l \in [m]$:

$$r_{k,z} \leq r_{l,z} + 1/2 + h_{ijk\ell}^{18}c + h_{ijkl}^{18}c \quad (366)$$

$$r_{l,z} + 3/2 \leq r_{k,z} + (1 - h_{ijk\ell}^{18})c + h_{ijkl}^{18}c \quad (367)$$

$$f_{(l,k)}^{(i,j)} + h_{ij}^{18}UP + e_{\{k,l\}} + h_{ijkl}^{18} \leq 3 \quad (368)$$

For case II, one additional inequality is added since the first two are the same again:

$$f_{(k,l)}^{(i,j)} + (1 - h_{ij}^{18}UP) + e_{\{k,l\}} + h_{ijkl}^{18} \leq 3 \quad (369)$$

Next, we model $f_{(k,l)}^{(i,j)} = 1$ if $r_k = \tilde{d}_{r_{k,z}}^{i,j}$ and $r_l = \tilde{s}_{r_{l,z}}^{i,j}$ and $r_{k,z} \leq s_{i,z}$ and $r_{l,z} \geq s_{j,z}$ and $h_{ij}^{18}UP = 1$. The formulations of form $r_k = \tilde{d}_{r_{k,z}}^{i,j}$ are elegant yet permitted. Therefore, we remodel the conditions for all $(i, j) \in E_A$, $k, l \in [m]$, and $\xi \in [\ell]$:

$$r_k = \tilde{d}_{\xi}^{i,j} \text{ and } r_l = \tilde{s}_{\xi-1}^{i,j} \text{ and } \xi \in D_1 \text{ and } h_{ij}^{18}UP = 1 \Rightarrow f_{(k,l)}^{(i,j)} = 1 \quad (370)$$

$$\Leftrightarrow r_k \neq \tilde{d}_{\xi}^{i,j} \text{ or } r_l \neq \tilde{s}_{\xi-1}^{i,j} \text{ or } \xi \notin D_1 \text{ or } h_{ij}^{18}UP = 0 \text{ or } f_{(k,l)}^{(i,j)} = 1 \quad (371)$$

Some parts of this conditions are already evaluated in terms 335 to 340. Therefore, the auxiliary variables defined there are re-used here. This yields $6|E_A|m^2(l-1)$ inequalities using the binary auxiliary variables $h_{ijk\ell\xi 1}^{18}$ to $h_{ijk\ell\xi 12}^{18}$ for all $(i, j) \in E_A$, $k, l \in [m]$, and $\xi \in [\ell - 1]$:

$$1/2 + r_{l,x} \leq \tilde{s}_{\xi-1,x}^{i,j} + h_{ijk\ell\xi 1}^{18}c \quad (372)$$

$$1/2 + r_{l,y} \leq \tilde{s}_{\xi-1,y}^{i,j} + h_{ijk\ell\xi 2}^{18}c \quad (373)$$

$$1/2 + r_{l,z} \leq (\xi - 1) + h_{ijk\ell\xi 3}^{18}c \quad (374)$$

$$1/2 + \tilde{s}_{\xi-1,x}^{i,j} \leq r_{l,x} + h_{ijk\ell\xi 4}^{18}c \quad (375)$$

$$1/2 + \tilde{s}_{\xi-1,y}^{i,j} \leq r_{l,y} + h_{ijk\ell\xi 5}^{18}c \quad (376)$$

$$1/2 + (\xi - 1) \leq r_{l,z} + h_{ijk\ell\xi 6}^{18}c \quad (377)$$

The next inequality is defined for all $(i, j) \in E_A$, $k, l \in [m]$ with $k \neq j$, and $\xi \in [\ell - 1]$. Technically, this inequality stands for $|E_A|(m-1)m(l-1)$ inequalities:

$$\begin{aligned} & h_{ij}^{18}UP + h_{ijD\xi}^{18} + (1 - f_{(k,l)}^{(i,j)}) + h_{ijk\ell\xi 1}^{18} + h_{ijk\ell\xi 2}^{18} + h_{ijk\ell\xi 3}^{18} + h_{ijk\ell\xi 4}^{18} + h_{ijk\ell\xi 5}^{18} + \\ & h_{ijk\ell\xi 6}^{18} + h_{ijk\ell\xi 1}^{18} + h_{ijk\ell\xi 2}^{18} + h_{ijk\ell\xi 3}^{18} + h_{ijk\ell\xi 4}^{18} + h_{ijk\ell\xi 5}^{18} + h_{ijk\ell\xi 6}^{18} \leq 14 \end{aligned} \quad (378)$$

The condition for case II is rewritten to

$$r_l = \tilde{d}_{\xi}^{i,j} \text{ and } r_k = \tilde{s}_{\xi+1}^{i,j} \text{ and } \xi \in D_0 \text{ and } h_{ij}^{18}UP = 0 \Rightarrow f_{(l,k)}^{(i,j)} = 1 \quad (379)$$

$$r_l \neq \tilde{d}_{\xi}^{i,j} \text{ or } r_k \neq \tilde{s}_{\xi+1}^{i,j} \text{ or } \xi \notin D_0 \text{ or } h_{ij}^{18}UP = 1 \text{ or } f_{(l,k)}^{(i,j)} = 1 \quad (380)$$

This yields additional $6|E_A|m^2(l-1)$ inequalities using the binary auxiliary variables $h_{ijk\ell\xi 13}^{18}$ to $h_{ijk\ell\xi 18}^{18}$ for all $(i, j) \in E_A$, $k, l \in [m]$,

and $\xi \in [\ell] \setminus \{1\}$. The auxiliary variables $h_{ijkl\xi 1}^{18}$ to $h_{ijkl\xi 6}^{18}$ can be reused along with their inequalities.

$$1/2 + r_{k,x} \leq \tilde{s}_{\xi+1,x}^{ij} + h_{ijkl\xi 7}^{18} \quad (381)$$

$$1/2 + r_{k,y} \leq \tilde{s}_{\xi+1,y}^{ij} + h_{ijkl\xi 8}^{18} \quad (382)$$

$$1/2 + r_{k,z} \leq (\xi + 1) + h_{ijkl\xi 9}^{18} \quad (383)$$

$$1/2 + \tilde{s}_{\xi+1,x}^{ij} \leq r_{k,x} + h_{ijkl\xi 10}^{18} \quad (384)$$

$$1/2 + \tilde{s}_{\xi+1,y}^{ij} \leq r_{k,y} + h_{ijkl\xi 11}^{18} \quad (385)$$

$$1/2 + (\xi + 1) \leq r_{k,z} + h_{ijkl\xi 12}^{18} \quad (386)$$

The next inequality is like 378 not defined for $l = j$, but for all $(i, j) \in E_A$, $k, l \in [m]$ with $l \neq j$, and $\xi \in [\ell] \setminus \{1\}$. Technically, this inequality stands for $|E_A|m(m-1)(l-1)$ inequalities:

$$\begin{aligned} & (1 - h_{ijUP}^{18}) + (1 - h_{ijD\xi}^{18}) + (1 - f_{(l,k)}^{(ij)}) + \\ & h_{ijkl\xi 7}^{18} + h_{ijkl\xi 8}^{18} + h_{ijkl\xi 9}^{18} + h_{ijkl\xi 10}^{18} + h_{ijkl\xi 11}^{18} + h_{ijkl\xi 12}^{18} + \\ & h_{ijl\xi 1}^{18} + h_{ijl\xi 2}^{18} + h_{ijl\xi 3}^{18} + h_{ijl\xi 4}^{18} + h_{ijl\xi 5}^{18} + h_{ijl\xi 6}^{18} \leq 14 \end{aligned} \quad (387)$$

Finally, we model the conditions that are valid for both case I and II. First, in the *source is given* in the source layer $s_{i,z} = \xi$ with $\tilde{s}_{\xi}^{ij} = s_i$. This is given by the implication

$$s_{i,z} = \xi \Rightarrow \tilde{s}_{\xi}^{ij} = s_i, \quad (388)$$

$$\Leftrightarrow s_{i,z} \neq \xi \text{ or } \tilde{s}_{\xi}^{ij} = s_i \quad (389)$$

This yields $9m$ inequalities using the auxiliary binary variables $h_{ij\xi 1}^{18}$, $h_{ij\xi 2}^{18}$, and $h_{ij\xi 3}^{18}$ for all $(i, j) \in E_A$ and $\xi \in [\ell]$:

$$s_{i,z} + 1/2 \leq \xi + h_{ij\xi 1}^{18} \quad (390)$$

$$\xi + 1/2 \leq s_{i,z} + h_{ij\xi 2}^{18} \quad (391)$$

$$\tilde{s}_{\xi,x}^{ij} \leq s_{i,x} + h_{ij\xi 3}^{18} \quad (392)$$

$$\tilde{s}_{\xi,y}^{ij} \leq s_{i,y} + h_{ij\xi 3}^{18} \quad (393)$$

$$\xi \leq s_{i,z} + h_{ij\xi 3}^{18} \quad (394)$$

$$s_{i,x} \leq \tilde{s}_{\xi,x}^{ij} + h_{ij\xi 3}^{18} \quad (395)$$

$$s_{i,y} \leq \tilde{s}_{\xi,y}^{ij} + h_{ij\xi 3}^{18} \quad (396)$$

$$s_{i,z} \leq \xi + h_{ij\xi 3}^{18} \quad (397)$$

$$h_{ij\xi 1}^{18} + h_{ij\xi 2}^{18} + h_{ij\xi 3}^{18} \leq 2 \quad (398)$$

Similar considerations lead to $s_{j,z} \neq \xi$ or $\tilde{d}_{\xi}^{ij} = s_j$ for the condition in the destination layer $s_{j,z} = \xi$, in which $\tilde{d}_{\xi}^{ij} = s_j$. This yields $9m$ ad-

ditional inequalities using the auxiliary binary variables $h_{ij\xi 4}^{18}$, $h_{ij\xi 5}^{18}$, and $h_{ij\xi 6}^{18}$ for all $(i, j) \in E_A$ and $\xi \in [\ell]$:

$$s_{j,z} + 1/2 \leq \xi + h_{ij\xi 4}^{18}c \quad (399)$$

$$\xi + 1/2 \leq s_{j,z} + h_{ij\xi 5}^{18}c \quad (400)$$

$$\tilde{d}_{\xi,x}^{ij} \leq s_{j,x} + h_{ij\xi 6}^{18}c \quad (401)$$

$$\tilde{d}_{\xi,y}^{ij} \leq s_{j,y} + h_{ij\xi 6}^{18}c \quad (402)$$

$$\xi \leq s_{j,z} + h_{ij\xi 6}^{18}c \quad (403)$$

$$s_{j,x} \leq \tilde{d}_{\xi,x}^{ij} + h_{ij\xi 6}^{18}c \quad (404)$$

$$s_{j,y} \leq \tilde{d}_{\xi,y}^{ij} + h_{ij\xi 6}^{18}c \quad (405)$$

$$s_{j,z} \leq \xi + h_{ij\xi 6}^{18}c \quad (406)$$

$$h_{ij\xi 4}^{18} + h_{ij\xi 5}^{18} + h_{ij\xi 6}^{18} \leq 2 \quad (407)$$

Next, the conditions for 2D DOR are modeled:

$$e_{\{k,l\}} = 1 \text{ and } r_{k,z} = r_{l,z} \Rightarrow \left[f_{(k,l)}^{(i,j)} = 1 \Rightarrow (\tilde{s}_{r_{k,z},x}^{ij} = r_{k,x} = r_{l,x}) \text{ or } (\tilde{d}_{r_{k,z},y}^{ij} = r_{k,y} = r_{l,y}) \right] \quad (408)$$

$$e_{\{k,l\}} = 1 \text{ and } r_{k,z} = r_{l,z} \Rightarrow \left[f_{(k,l)}^{(i,j)} = 0 \text{ or } (\tilde{s}_{r_{k,z},x}^{ij} = r_{k,x} = r_{l,x}) \text{ or } (\tilde{d}_{r_{k,z},y}^{ij} = r_{k,y} = r_{l,y}) \right] \quad (409)$$

$$e_{\{k,l\}} = 0 \text{ or } r_{k,z} \neq r_{l,z} \text{ or } \left[f_{(k,l)}^{(i,j)} = 0 \text{ or } (\tilde{s}_{r_{k,z},x}^{ij} = r_{k,x} = r_{l,x}) \text{ or } (\tilde{d}_{r_{k,z},y}^{ij} = r_{k,y} = r_{l,y}) \right] \quad (410)$$

$$e_{\{k,l\}} = 0 \text{ or } r_{k,z} \neq r_{l,z} \text{ or } \left[f_{(k,l)}^{(i,j)} = 0 \text{ or } (\tilde{s}_{\xi,x}^{ij} = r_{k,x} = r_{l,x} \text{ or } r_{k,z} \neq \xi) \text{ or } (\tilde{d}_{\xi,y}^{ij} = r_{k,y} = r_{l,y} \text{ or } r_{k,z} \neq \xi) \right] \quad (411)$$

This yields the following $11m^2$ inequalities for all $(i, j) \in E_A$, $k, l \in [m] \times [m]$, and $\xi \in [\ell]$ using the auxiliary binary variables :

$$1/2 + \xi \leq r_{k,z} + h_{ijk1\xi 19}^{18}c \quad (412)$$

$$1/2 + r_{k,z} \leq \xi + h_{ijk1\xi 20}^{18}c \quad (413)$$

$$1/2 + r_{k,z} \leq r_{l,z} + h_{ijk1\xi 23}^{18}c \quad (414)$$

$$1/2 + r_{l,z} \leq r_{k,z} + h_{ijk1\xi 24}^{18}c \quad (415)$$

$$\tilde{s}_{\xi,x}^{ij} \leq r_{k,x} + h_{ijk1\xi 25}^{18}c \quad (416)$$

$$r_{k,x} \leq \tilde{s}_{\xi,x}^{ij} + h_{ijk1\xi 25}^{18}c \quad (417)$$

$$r_{k,x} \leq r_{l,x} + h_{ijk1\xi 25}^{18}c \quad (418)$$

$$r_{l,x} \leq r_{k,x} + h_{ijk1\xi 25}^{18}c \quad (419)$$

$$\tilde{d}_{\xi,y}^{ij} \leq r_{k,y} + h_{ijk1\xi 26}^{18}c \quad (420)$$

$$r_{k,y} \leq \tilde{d}_{\xi,y}^{ij} + h_{ijk1\xi 26}^{18}c \quad (421)$$

$$r_{k,y} \leq r_{l,y} + h_{ijk1\xi 26}^{18}c \quad (422)$$

$$r_{l,y} \leq r_{k,y} + h_{ijk1\xi 26}^{18}c \quad (423)$$

$$h_{ijk1\xi 19}^{18} + h_{ijk1\xi 20}^{18} + h_{ijk1\xi 23}^{18} + h_{ijk1\xi 24}^{18} + h_{ijk1\xi 25}^{18} + h_{ijk1\xi 26}^{18} + f_{(k,l)}^{(i,j)} + e_{\{k,l\}} \leq 7 \quad (424)$$

Connect neighbored routers

Using the *neighbored*-property from definition a.2, the constraint is given by:

$$\forall i, j \in [m] \times [m] : \quad \text{neighbored}(r_i, r_j) \rightarrow e_{\{i,j\}} = 1 \quad (425)$$

$$\Leftrightarrow \forall i, j \in [m] \times [m] : \quad \text{not neighbored}(r_i, r_j) \text{ or } e_{\{i,j\}} = 1 \quad (426)$$

$$\begin{aligned} & \forall i, j \in [m] \times [m] : e_{\{i,j\}} = 1 \text{ or } i = j \text{ or } r_{i,z} \neq r_{j,z} \\ \Leftrightarrow & \quad \text{or } (r_{i,x} \neq r_{j,x} \text{ and } r_{i,y} \neq r_{j,y}) \\ & \quad \text{or } \exists k \in [m] : r_k \in L(r_i, r_j) \setminus \{r_i, r_j\} \end{aligned} \quad (427)$$

The part $\exists k \in [m] : r_k \in L(r_i, r_j) \setminus \{r_i, r_j\}$ is given by these inequalities for all $k \in [m] \setminus \{i, j\}$:

$$a_1^T r_k \geq \tilde{b}_1 \quad (428)$$

$$-a_1^T r_k \geq \tilde{b}_2 \quad (429)$$

$$a_2^T r_k \geq \tilde{b}_3 \quad (430)$$

$$-a_2^T r_k \geq \tilde{b}_4 \quad (431)$$

with $\tilde{b}_1 = b_1$, $\tilde{b}_2 = b_2$, $\tilde{b}_3 = b_3 + \delta$, and $\tilde{b}_4 = b_4 + \delta$. We introduce the binary auxiliary variables $h_{i,j}^{2^0} 1$, $h_{i,j}^{2^0} 2$, $h_{i,j}^{2^0} 3$, $h_{i,j}^{2^0} 4$, $h_{i,j,k}^{2^0} 5$, $h_{i,j}^{2^0} 6$, $h_{i,j}^{2^0} \text{OR1}$, $h_{i,j}^{2^0} \text{OR2}$ and the constants $c_1^{2^0} = x_{\max} + y_{\max} + l$ and $c_2^{2^0} = x_{\max} + y_{\max}$. For all $i, j \in [m]$ the following $7m + 1$ inequalities are satisfied for the first part of Equation 427:

$$h_{i,j}^{2^0} 1 + e_{\{i,j\}} \geq 1 \quad (432)$$

$$r_{i,z} \leq r_{j,z} - 1/2 + h_{i,j}^{2^0} 2 c_1^{2^0} \quad (433)$$

$$r_{i,z} \geq r_{j,z} + 1/2 - h_{i,j}^{2^0} 3 c_1^{2^0} \quad (434)$$

$$r_{i,y} - r_{j,y} \leq -\delta + h_{i,j}^{2^0} \text{OR1} c_1^{2^0} + h_{i,j}^{2^0} 4 c_1^{2^0} \quad (435)$$

$$r_{i,y} - r_{j,y} \geq \delta - (1 - h_{i,j}^{2^0} \text{OR1}) c_1^{2^0} - h_{i,j}^{2^0} 4 c_1^{2^0} \quad (436)$$

$$r_{i,x} - r_{j,x} \leq -\delta + h_{i,j}^{2^0} \text{OR2} c_1^{2^0} + h_{i,j}^{2^0} 4 c_1^{2^0} \quad (437)$$

$$r_{i,x} - r_{j,x} \geq \delta - (1 - h_{i,j}^{2^0} \text{OR2}) c_1^{2^0} - h_{i,j}^{2^0} 4 c_1^{2^0} \quad (438)$$

and

$$h_{i,j}^{2^0} 6 = \begin{cases} 1, & i \neq j \\ 0, & \text{otherwise.} \end{cases} \quad (439)$$

Furthermore, for all $i, j \in [m]$ and $k \in [m] \setminus \{i, j\}$ these $4m^2(m-2)$ inequalities from Equation 428 with the logical operations are given as:

$$c_2^{2^0} h_{i,j,k}^{2^0} 5 + a_1^T r_k \geq \tilde{b}_1 \quad (440)$$

$$c_2^{2^0} h_{i,j,k}^{2^0} 5 - a_1^T r_k \geq \tilde{b}_2 \quad (441)$$

$$c_2^{2^0} h_{i,j,k}^{2^0} 5 + a_2^T r_k \geq \tilde{b}_3 \quad (442)$$

$$c_2^{2^0} h_{i,j,k}^{2^0} 5 - a_2^T r_k \geq \tilde{b}_4 \quad (443)$$

Finally, the logical OR operations must be formulated as MILP constraint as well:

$$h_{i,j}^{2^0} 1 + h_{i,j}^{2^0} 2 + h_{i,j}^{2^0} 3 + h_{i,j}^{2^0} 4 + h_{i,j}^{2^0} 6 + \sum_{k \in [m] \setminus \{i,j\}} h_{i,j,k}^{2^0} 5 \leq m + 2 \quad (444)$$

Topology for (gateway) DOR

To enable 2D dimension order routing within each layer, for each pair of placed routers there must be another pair of routers so that these

four routers are located at the edges of a rectangle. This constraint is formulated as follows:

$$\begin{aligned}
 r_{i,z} = r_{j,z} &\rightarrow \\
 \exists k \in [m] : r_{k,z} = r_{i,z} \text{ and } r_{k,x} = r_{i,x} \text{ and } r_{k,y} = r_{j,y} \text{ and} & \quad (445) \\
 \exists l \in [m] : r_{l,z} = r_{i,z} \text{ and } r_{l,x} = r_{j,x} \text{ and } r_{l,y} = r_{i,y} &
 \end{aligned}$$

This can be transformed into:

$$\begin{aligned}
 r_{i,z} \neq r_{j,z} \text{ or} \\
 \exists k \in [m] : r_{k,z} = r_{i,z} \text{ and } r_{k,x} = r_{i,x} \text{ and } r_{k,y} = r_{j,y} \text{ and} & \quad (446) \\
 \exists l \in [m] : r_{l,z} = r_{i,z} \text{ and } r_{l,x} = r_{j,x} \text{ and } r_{l,y} = r_{i,y} &
 \end{aligned}$$

The binary auxiliary variable h_{ij}^{z1} indicates whether the z -component of the routers with index i and j are identical. The binary auxiliary variable $h_{ij}^{z1}_{\text{OR1}}$ realized the OR relation for the inequality. This yields for all $i, j \in [m]$ with $i < j$ the inequalities:

$$1/2 + r_{i,z} \leq r_{j,z} + h_{ij}^{z1}_{\text{OR1}}c_1 + h_{ij}^{z1}c_1, \quad (447)$$

$$1/2 + r_{j,z} \leq r_{i,z} + (1 - h_{ij}^{z1}_{\text{OR1}})c_1 + h_{ij}^{z1}c_1 \quad (448)$$

Additionally, m^4 binary auxiliary variables h_{ijkl}^{z1} are required to realize the OR relation between the \exists -quantifiers for all $i, j, k, l \in [m]$:

$$r_{k,z} \leq r_{i,z} + h_{ijkl}^{z1}c_1, \quad (449)$$

$$r_{i,z} \leq r_{k,z} + h_{ijkl}^{z1}c_1, \quad (450)$$

$$r_{l,z} \leq r_{i,z} + h_{ijkl}^{z1}c_1, \quad (451)$$

$$r_{i,z} \leq r_{l,z} + h_{ijkl}^{z1}c_1, \quad (452)$$

$$r_{k,x} \leq r_{i,x} + h_{ijkl}^{z1}c_1, \quad (453)$$

$$r_{i,x} \leq r_{k,x} + h_{ijkl}^{z1}c_1, \quad (454)$$

$$r_{k,y} \leq r_{j,y} + h_{ijkl}^{z1}c_1, \quad (455)$$

$$r_{j,y} \leq r_{k,y} + h_{ijkl}^{z1}c_1, \quad (456)$$

$$r_{l,x} \leq r_{j,x} + h_{ijkl}^{z1}c_1, \quad (457)$$

$$r_{j,x} \leq r_{l,x} + h_{ijkl}^{z1}c_1, \quad (458)$$

$$r_{l,y} \leq r_{i,y} + h_{ijkl}^{z1}c_1, \quad (459)$$

$$r_{i,y} \leq r_{l,y} + h_{ijkl}^{z1}c_1 \quad (460)$$

The OR relation is given for all $i, j \in [m]$ with $i < j$:

$$\sum_{k \in [m]} \sum_{l \in [m]} h_{ijkl}^{z1} \leq m^2 - h_{ij}^{z1} \quad (461)$$

A.6 AUXILIARY VARIABLES

A.6.1 Not all routers placed are in a subset of all routers

Name

For all $Q \subseteq [m]$:

h_Q

Definition

$$h_Q = \begin{cases} 1 & \text{if not all placed routers are in } Q \\ 0 & \text{otherwise.} \end{cases} \quad (462)$$

Logical Constraint

This is given by the following 2^m constraints:

$$h_Q \leq \sum_{i \in [m]} \rho_i - \sum_{i \in Q} \rho_i \leq m h_Q \quad (463)$$

A.6.2 *At least one router placed is in a subset of all routers*

Name

For all $Q \subseteq [m]$:

$$\tilde{h}_Q$$

Definition

$$\tilde{h}_Q = \begin{cases} 1 & \text{if at least one placed router is in } Q \\ 0 & \text{otherwise.} \end{cases} \quad (464)$$

Logical Constraint

This is given by the following 2^m constraints:

$$\tilde{h}_Q \leq \sum_{i \in [m]} \rho_i - \sum_{i \in [m] \setminus Q} \rho_i \leq m \tilde{h}_Q \quad (465)$$

A.6.3 *Router j is in tile i*

Name

For all $i, j \in [m]$:

$$h_{i,j}$$

Definition

$$h_{i,j} = \begin{cases} 1 & \text{if router } j \text{ is in tile } i \\ 0 & \text{otherwise.} \end{cases} \quad (466)$$

Logical Constraint

The conditions for router r_j , which does not start a tile itself, is located in the tile i for all $j \in \{n+1, \dots, m\}$ is given by:

$$t_{i,x} \leq r_{j,x} \leq t_{i,x} + a_i, \quad (467)$$

$$t_{i,y} \leq r_{j,y} \leq t_{i,y} + b_i, \quad (468)$$

$$t_{i,z} \leq r_{j,z} \leq t_{i,z} \quad (469)$$

For this, first, we model the implication "Router j is located in tile $i \rightarrow (h_{i,j} = 1)" \leftrightarrow "h_{i,j} = 1$ or router j is not located in tile i ". Using the auxiliary binary variables $h_{i,j}^{11}_1, h_{i,j}^{11}_2, h_{i,j}^{11}_3, h_{i,j}^{11}_4, h_{i,j}^{11}_5$, and $h_{i,j}^{11}_6$ this yields:

$$r_{j,x} + 1/2\delta \leq t_{i,x} + h_{i,j}c_3 + h_{i,j}^{11}_1c_3, \quad (470)$$

$$t_{i,x} + a_i + 1/2\delta \leq r_{j,x} + h_{i,j}c_3 + h_{i,j}^{11}_2c_3, \quad (471)$$

$$r_{j,y} + 1/2\delta \leq t_{i,y} + h_{i,j}c_3 + h_{i,j}^{11}_3c_3, \quad (472)$$

$$t_{i,y} + b_i + 1/2\delta \leq r_{j,y} + h_{i,j}c_3 + h_{i,j}^{11}_4c_3, \quad (473)$$

$$r_{j,z} + 1/2 \leq t_{i,z} + h_{i,j}c_3 + h_{i,j}^{11}_5c_3, \quad (474)$$

$$t_{i,z} + 1/2 \leq r_{j,z} + h_{i,j}c_3 + h_{i,j}^{11}_6c_3, \quad (475)$$

$$h_{i,j}^{11}_1 + \dots + h_{i,j}^{11}_6 \leq 5. \quad (476)$$

Here, addition with $1/2\delta$ is used in Equation 470 to 473 to model the $<$ -relation from the negation of Equation 467 and 468. Hence, there is an area around each tile with width $1/2\delta$. The variable $h_{i,j}$ is set to 0 and 1 simultaneously for a router located in this area. In consequence, within this region no router can be placed. This is coherent with the other constraints requiring that routers are located within tiles and tiles do not overlap with a distance of at least δ .

Second, we model the implication "Router j is not located in tile $A_i \implies (h_{i,j} = 0)" \leftrightarrow "h_{i,j} = 0$ or router j is located in tile A_i ". This yields:

$$t_{i,x} \leq r_{j,x} + (1 - h_{i,j})c_3 \quad (477)$$

$$r_{j,x} \leq t_{i,x} + a_i + (1 - h_{i,j})c_3 \quad (478)$$

$$t_{i,y} \leq r_{j,y} + (1 - h_{i,j})c_3 \quad (479)$$

$$r_{j,y} \leq t_{i,y} + b_i + (1 - h_{i,j})c_3 \quad (480)$$

$$t_{i,z} \leq r_{j,z} + (1 - h_{i,j})c_3 \quad (481)$$

$$r_{j,z} \leq t_{i,z} + (1 - h_{i,j})c_3 \quad (482)$$

A.6.4 Router j is in tile i and is a 3D router*Name*

For all $i, j \in [m]$:

$$\hat{h}_{i,j}$$

Definition

It must hold for all $i, j \in [m]$:

$$\hat{h}_{i,j} = \begin{cases} 1 & \text{if router } j \text{ has a TSV and is located in tile } i \\ 0 & \text{otherwise.} \end{cases} \quad (483)$$

Logical Constraint

The variable $\hat{h}_{i,j}$ can be modeled using the variables $h_{i,j}$ and \hat{h}_j :

$$\hat{h}_{i,j} = \begin{cases} 1, & \text{if } h_{i,j} = 1 \text{ and } \hat{h}_j = 1 \\ 0, & \text{otherwise.} \end{cases} \quad (484)$$

For $\hat{h}_{i,j}$ there are four cases:

1. For all tiles $i \in [n]$ and routers $j \in [n]$ the routers' and tiles' indexes are identical and the routers are associated with this very tile. Therefore, $\hat{h}_{i,j} = 1$ iff $i = j$ and $\hat{h}_j = 1$ and $\hat{h}_{i,j} = 0$ otherwise. This case is not of special interest since for $i = j$ the variable $\hat{h}_{i,j}$ is already given by $\hat{h}_{i,j} = \hat{h}_j$.
2. For all tiles $i \in [n]$ and routers $j \in [n+1, \dots, m]$ it holds that $\hat{h}_{i,j} = 1$ if and only if $h_{i,j} = 1$ and $\hat{h}_j = 1$. This case must be modeled as it is not already given by any other variable.
3. For all tiles $i \in [n+1, \dots, m]$ and routers $j \in [n]$ the variable $\hat{h}_{i,j}$ is always zero since those routers are located in the tiles with indexed in $[n]$.
4. For all tiles $i \in [n+1, \dots, m]$ and routers $j \in [n+1, \dots, m]$ it holds that $\hat{h}_{i,j} = 1$ if and only of $h_{i,j} = 1$ and $\hat{h}_j = 1$. This case, again, must be modeled as it is not already given by any other variable.

The cases 2 and 3 are modeled for all tiles with index $i \in [m]$ and routers with index $j \in [n+1, \dots, m]$. Thus, the variable $\hat{h}_{i,j}$ is given $m(m-n)$ times. First, we model the implication " $(h_{i,j} = 1 \text{ and } \hat{h}_j = 1) \rightarrow \hat{h}_{i,j} = 1$ " which yields $\hat{h}_{i,j} = 1$ or $h_{i,j} = 0$ or $\hat{h}_j = 0$ using DeMorgan's laws. This is given by $m(m-n)$ inequalities for all $i \in [m]$ and $j \in [n+1, \dots, m]$:

$$(1 - \hat{h}_{i,j}) + h_{i,j} + \hat{h}_j \leq 2. \quad (485)$$

Second, we model the implication " $(h_{i,j} = 0 \text{ or } \hat{h}_j = 0) \rightarrow \hat{h}_{i,j} = 0$ ", which is equivalent to $\hat{h}_{i,j} = 0$ or $(h_{i,j} = 1 \text{ and } \hat{h}_j = 1)$. This yields $2m(m-n)$ inequalities for all $i \in [m]$ and $j \in [n+1, \dots, m]$:

$$\hat{h}_{i,j} \leq \hat{h}_j, \quad (486)$$

$$\hat{h}_{i,j} \leq h_{i,j}. \quad (487)$$

A.6.5 Router j is in tile i and has a vertical and downward link

Name

For all $i, j \in [m]$:

$$\hat{h}_{i,j}^+$$

Definition

Similar to $\hat{h}_{i,j}$, we now define $\hat{h}_{i,j}^+$:

$$\hat{h}_{i,j}^+ = \begin{cases} 1, & \text{if router } j \text{ has a TSV leading downwards} \\ & \text{and is located in tile } i \\ 0, & \text{otherwise.} \end{cases} \quad (488)$$

Logical Constraint

Equation 488 is equivalent to:

$$\hat{h}_{i,j}^+ = \begin{cases} 1, & \text{if } h_{i,j} = 1 \text{ and } \hat{h}_j^+ = 1 \\ 0, & \text{otherwise,} \end{cases} \quad (489)$$

which is easily obtained by the following four inequalities, using the auxiliary variable $h_{i,j,OR2}^{\dagger\dagger}$:

$$\hat{h}_{i,j}^+ \leq h_{i,j} \quad (490)$$

$$\hat{h}_{i,j}^+ \leq \hat{h}_j^+ \quad (491)$$

$$\hat{h}_{i,j}^+ \leq h_{i,j} + h_{i,j,OR2}^{\dagger\dagger} c_3 \quad (492)$$

$$\hat{h}_{i,j}^+ \leq \hat{h}_j^+ + (1 - h_{i,j,OR2}^{\dagger\dagger}) c_3 \quad (493)$$

$$(494)$$

While inequality 490 and inequality 491 model the implication

$$(\text{not } (h_{i,j} = 1 \text{ and } \hat{h}_j^+ = 1)) \rightarrow \hat{h}_{i,j}^+ = 0, \quad (495)$$

Inequation 492 and inequation 493 guarantee that

$$(h_{i,j} = 1 \text{ and } \hat{h}_j^+ = 1) \rightarrow \hat{h}_{i,j}^+ = 1. \quad (496)$$

A.6.6 Router j is a 3D router (

Name

For all $j \in [m]$:

$$\hat{h}_j$$

Definition

Router j has at least a vertical link:

$$\hat{h}_j = \begin{cases} 1, & \text{if router } j \text{ has a TSV} \\ 0, & \text{otherwise.} \end{cases} \quad (497)$$

Logical Constraint

For all $k, j \in [m]$ it holds that routers r_k and r_j have a TSV iff $e_{\{k,j\}} = 1$ and $r_{k,z} \in \{r_{j,z} + 1, r_{j,z} - 1\}$ and $r_{k,x} = r_{j,x}$ and $r_{k,y} = r_{j,y}$. For Equation 497 we first model the relation "router j has a TSV $\rightarrow (\hat{h}_j = 1)$ " \leftrightarrow " $\hat{h}_j = 1$ OR router j has no TSV". The latter is given if for all $k \in [m]$ with $k \neq j$ if it holds $e_{\{j,k\}} = 1 \rightarrow r_{j,z} = r_{k,z}$. It is sufficient to evaluate whether two routers are connected and have the same z -coordinates. This can be rewritten: $\forall k \in [m], k \neq j : r_{j,z} = r_{k,z}$ or $e_{\{j,k\}} = 0$. This yields $3m(m-1)$ inequalities using the auxiliary binary variables $h_{j,k}^{11}$ OR for all $j \in [m]$ and $k \in [m], k \neq j$:

$$r_{j,z} \leq r_{k,z} + h_{j,k}^{11} c_1^{11} + \hat{h}_j c_3, \quad (498)$$

$$r_{k,z} \leq r_{j,z} + h_{j,k}^{11} c_1^{11} + \hat{h}_j c_3, \quad (499)$$

$$e_{\{j,k\}} \leq 0 + (1 - h_{j,k}^{11}) c_1^{11} + \hat{h}_j c_3. \quad (500)$$

Second, we model the implication "router j has no TSV $\implies (\hat{h}_j = 0)$ " \leftrightarrow " $\hat{h}_j = 0$ OR router j has a TSV". We model the latter by: $\exists k \in [m] : e_{\{j,k\}} = 1$ and $r_{j,z} \neq r_{k,z}$. This can be expressed in form of $4m(m-1)$ inequalities using the auxiliary binary variables $\tilde{h}_{j,k}^{11}$ and $\tilde{h}_{j,k}^{11}$ OR for all $j \in [m]$ and $k \in [m], k \neq j$:

$$1 \leq e_{\{j,k\}} + \tilde{h}_{j,k}^{11} c_3, \quad (501)$$

$$r_{j,z} \leq r_{k,z} - 1/2 + \tilde{h}_{j,k}^{11} c_3 + \tilde{h}_{j,k}^{11} c_3, \quad (502)$$

$$r_{k,z} \leq r_{j,z} - 1/2 + (1 - \tilde{h}_{j,k}^{11}) c_3 + \tilde{h}_{j,k}^{11} c_3 \quad (503)$$

Using this, the variable \hat{h}_j is given by satisfying the inequalities for all $j \in [m]$:

$$\hat{h}_j + \sum_{k \in [m], k \neq j} \tilde{h}_{j,k}^{11} \leq m - 1. \quad (504)$$

In this sum, $\tilde{h}_{j,k}^{11}$ is 1 if there is no TSV between routers j and k . There are $m-1$ possible routers to connect from router j . Thus, using the sum $\sum_{k \in [m], k \neq j} \tilde{h}_{j,k}^{11} \leq m-2$ it would be required that at least one TSV connects to router j , i. e. $\tilde{h}_{j,k}^{11} = 0$ for this router. This is impossible if router j does not have a TSV, and therefore, in the sum in Equation 504 the variable \hat{h}_j adds the missing addend.

A.6.7 Router k has a vertical upward/downward link

Name

For all $k \in [m]$:

$$\hat{h}_k^-, \hat{h}_k^+$$

Definition

The auxiliary binary variables \hat{h}_k^- and \hat{h}_k^+ are introduced, which indicate whether a router has a upward (or downward) vertical link:

$$\hat{h}_k^+ = \begin{cases} 1, & \text{iff router } k \text{ has a link heading downwards} \\ 0, & \text{otherwise.} \end{cases} \quad (505)$$

$$\hat{h}_k^- = \begin{cases} 1, & \text{iff router } k \text{ has a link heading upwards} \\ 0, & \text{otherwise.} \end{cases} \quad (506)$$

Logical Constraint

The implication "router k has a link leading downwards $\Rightarrow \hat{h}_k^+ = 1$ " is modeled. This is equivalent to $\hat{h}_k^+ = 1$ or router k has no link leading downwards. This is given for all $k \in [m]$ by:

$$\forall j \in [m] \setminus \{k\} \text{ with } r_{j,z} = r_{k,z} + 1 : \quad e_{\{k,j\}} = 0 \text{ or } \hat{h}_k^+ = 1, \quad (507)$$

$$\Leftrightarrow \forall j \in [m] \setminus \{k\} : \quad r_{j,z} \neq r_{k,z} + 1 \text{ or } e_{\{k,j\}} = 0 \text{ or } \hat{h}_k^+ = 1 \quad (508)$$

This yields $3m(m-1)$ inequalities for all $k \in [m]$, $j \in [m] \setminus \{k\}$ using the binary auxiliary variables $h_{kj}^+_{\text{OR}}$, h_{kj1}^+ and the constant c_1^{11} :

$$r_{j,z} \leq r_{k,z} + 1/2 + h_{kj}^+_{\text{OR}} c_1^{11} + h_{kj1}^+ c_1^{11}, \quad (509)$$

$$r_{j,z} \geq r_{k,z} + 3/2 - (1 - h_{kj}^+_{\text{OR}}) c_1^{11} - h_{kj1}^+ c_1^{11}, \quad (510)$$

$$e_{\{k,j\}} \leq 0 + (1 - h_{kj1}^+) + \hat{h}_k^+ \quad (511)$$

The analog is given for \hat{h}_k^- for all $k \in [m]$:

$$\forall j \in [m] \setminus \{k\} : \quad r_{j,z} \neq r_{k,z} - 1 \text{ or } e_{\{k,j\}} = 0 \text{ or } \hat{h}_k^- = 1 \quad (512)$$

This yields $3m(m-1)$ inequalities for all $k \in [m]$, $j \in [m] \setminus \{k\}$ using the binary auxiliary variables $h_{kj}^-_{\text{OR}}$, h_{kj1}^- and the constant c_1^{11} :

$$r_{j,z} \leq r_{k,z} - 3/2 + h_{kj}^-_{\text{OR}} c_1^{11} + h_{kj1}^- c_1^{11}, \quad (513)$$

$$r_{j,z} \geq r_{k,z} - 1/2 - (1 - h_{kj}^-_{\text{OR}}) c_1^{11} - h_{kj1}^- c_1^{11}, \quad (514)$$

$$e_{\{k,j\}} \leq 0 + (1 - h_{kj1}^-) + \hat{h}_k^- \quad (515)$$

The implication "router k has a no link leading downwards $\Rightarrow \hat{h}_k^+ = 0$ " is modeled, as well. This is equivalent to $\hat{h}_k^+ = 0$ or router k has a link heading downwards. This is given for all $k \in [m]$ by:

$$\exists j \in [m] \setminus \{k\} \text{ with } r_{j,z} = r_{k,z} + 1 \text{ and } e_{\{k,j\}} = 1 \text{ or } \hat{h}_k^+ = 0 \quad (516)$$

This yields the $3m(m-1) + m$ inequalities for all $k \in [m]$ and $j \in [m] \setminus \{k\}$ using the auxiliary binary variables h_{kj2}^+ :

$$r_{j,z} \leq r_{k,z} + 1 + h_{kj2}^+ c_1^{11}, \quad (517)$$

$$r_{j,z} \geq r_{k,z} + 1 - h_{kj2}^+ c_1^{11}, \quad (518)$$

$$e_{\{k,j\}} \geq 1 - h_{kj2}^+ \quad (519)$$

In addition, this inequality must be satisfied for all $k \in [m]$ ⁸:

$$\sum_{j \in [m] \setminus \{k\}} h_{kj2}^+ \leq m - 2 + (1 - \hat{h}_k^+) \quad (520)$$

The analog is given for \hat{h}_k^- with $r_{j,z} = r_{k,z} - 1$ for all $k \in [m]$ and for all $j \in [m] \setminus \{k\}$:

$$r_{j,z} \leq r_{k,z} - 1 + h_{kj2}^- c_1^{11}, \quad (521)$$

$$r_{j,z} \geq r_{k,z} - 1 - h_{kj2}^- c_1^{11}, \quad (522)$$

$$e_{\{k,j\}} \geq 1 - h_{kj2}^- \quad (523)$$

In addition, this inequality must be satisfied for all $k \in [m]$:

$$\sum_{j \in [m] \setminus \{k\}} h_{kj2}^- \leq m - 2 + (1 - \hat{h}_k^-) \quad (524)$$

In total, we have now required $12m(m-1) + 2m$ inequalities to set \hat{h}_k^+ and \hat{h}_k^- .

A.6.8 Number of KOZs in tile i

Name

For all $i \in [m]$:

$$h_{i \text{ KOZ}}$$

Definition

The auxiliary variable $h_{i \text{ KOZ}}$ is the number of TSVs which are leading downwards from tile i .

⁸ This model excludes the case $m = 0$ for a chip without routers (a trivial case).

Logical Constraint

The m inequalities are very similar to the inequalities of $h_{i \text{ TSV}}$.

$$\forall i \in [n] : \quad h_{i \text{ KOZ}} = \hat{h}_i^+ + \sum_{j \in [n+1, \dots, m]} \hat{h}_{i,j}^+, \quad (525)$$

$$\forall i \in [n+1, \dots, m] : \quad h_{i \text{ KOZ}} = \sum_{j \in [n+1, \dots, m]} \hat{h}_{i,j}^+, \quad (526)$$

A.6.9 Number of routers in tile i *Name*

For all $i \in [m]$:

$$h_{i \text{ Router}}$$

Definition

The auxiliary variable $h_{i \text{ Router}} \in \mathbb{N}$ is equal to the number of routers in tile i .

Logical Constraint

The variable $h_{i \text{ Router}}$ is defined as:

$$\forall i \in [n] : \quad h_{i \text{ Router}} = \sum_{j \in [n+1, \dots, m]} h_{i,j} + 1, \quad (527)$$

$$\forall i \in [n+1, \dots, m] : \quad h_{i \text{ Router}} = \sum_{j \in [n+1, \dots, m]} h_{i,j}. \quad (528)$$

A.6.10 Number of 3D routers in tile i *Name*

For all $i \in [m]$:

$$h_{i \text{ TSV}}$$

Definition

The auxiliary variable $h_{i \text{ TSV}} \in \mathbb{N}$ is equal to the number of routers in tile i .

Logical Constraints

This auxiliary variable has the value of the number of routers r_j in the tile i with a vertical link. This can be modeled using the auxiliary variables $h_{i,j}$ and \hat{h}_j by:

$$\begin{aligned} \forall j \in [m] : \quad & h_{i \text{ TSV}} \text{ is incremented} \\ & \leftrightarrow h_{i,j} = 1 \text{ and } \hat{h}_j = 1 \end{aligned} \quad (529)$$

The variable $h_{i \text{ TSV}}$ is given by the m inequalities:

$$\forall i \in [n]: \quad h_{i \text{ TSV}} = \hat{h}_i + \sum_{j \in [n+1, \dots, m]} \hat{h}_{i,j}, \quad (530)$$

$$\forall i \in [n+1, \dots, m]: \quad h_{i \text{ TSV}} = \sum_{j \in [n+1, \dots, m]} \hat{h}_{i,j}, \quad (531)$$

A.6.11 Find correct element in matrix with tile areas

Name

For all $i \in [m]$, $\xi \in [\ell]$ and $\alpha, \beta, \gamma \in [m - n + 1]$:

$$h_{\alpha, \beta, \gamma}^{\xi, i, 11}$$

Definition

It must hold for all $i \in [m]$, $\xi \in [\ell]$ and $\alpha, \beta, \gamma \in [m - n + 1]$:

$$h_{\alpha, \beta, \gamma}^{\xi, i, 11} = \begin{cases} 1 & \text{if tile } i \text{ is in layer } \xi, \text{ has } (\alpha - 1) \text{ routers,} \\ & (\beta - 1) \text{ 3D-routers, and } (\gamma - 1) \text{ KOZs} \\ 0 & \text{otherwise.} \end{cases} \quad (532)$$

Logical Constraint

The variables $h_{i \text{ Router}}$, $h_{i \text{ TSV}}$, and $h_{i \text{ KOZ}}$ can be re-used. The following two constraints must be satisfied. First, only one element of each matrix $H^{\xi, i, 11}$ is allowed to be one using these m inequalities:

$$\sum_{\xi \in [\ell], \alpha, \beta, \gamma \in [m - n + 1]} h_{\alpha, \beta, \gamma}^{\xi, i, 11} \leq 1 \quad (533)$$

The implication "tile i is in layer ξ and has $(\alpha - 1)$ routers, $(\beta - 1)$ 3D-routers, and $(\gamma - 1)$ KOZs $\rightarrow h_{\alpha, \beta, \gamma}^{\xi, i, 11} = 1$ " $\leftrightarrow t_{i,z} \neq \xi$ or $h_{i \text{ Router}} \neq (\alpha - 1)$ or $h_{i \text{ TSV}} \neq (\beta - 1)$ or $h_{i \text{ KOZ}} \neq (\gamma - 1)$ or $h_{\alpha, \beta, \gamma}^{\xi, i, 11} = 1$ must be modeled. This yields $9m\ell(m - n + 1)(m - n + 1)$ inequalities for all $i \in [m]$, $\xi \in [\ell]$, $\alpha \in [m - n + 1]$, $\beta \in [m - n + 1]$ and $\gamma \in [m - n + 1]$:

$$1/2 + t_{i,z} \leq \xi + h_{\alpha, \beta 1}^{\xi, i, 11} c_3, \quad (534)$$

$$1/2 + \xi \leq t_{i,z} + h_{\alpha, \beta 2}^{\xi, i, 11} c_3, \quad (535)$$

$$1/2 + h_{i \text{ Router}} \leq (\alpha - 1) + h_{\alpha, \beta 3}^{\xi, i, 11} c_3, \quad (536)$$

$$1/2 + (\alpha - 1) \leq h_{i \text{ Router}} + h_{\alpha, \beta 4}^{\xi, i, 11} c_3, \quad (537)$$

$$1/2 + h_{i \text{ TSV}} \leq (\beta - 1) + h_{\alpha, \beta 5}^{\xi, i, 11} c_3, \quad (538)$$

$$1/2 + (\beta - 1) \leq h_{i \text{ TSV}} + h_{\alpha, \beta 6}^{\xi, i, 11} c_3, \quad (539)$$

$$1/2 + h_{i \text{ KOZ}} \leq (\gamma - 1) + h_{\alpha, \beta 7}^{\xi, i, 11} c_3, \quad (540)$$

$$1/2 + (\gamma - 1) \leq h_{i \text{ KOZ}} + h_{\alpha, \beta 8}^{\xi, i, 11} c_3, \quad (541)$$

$$\begin{aligned} h_{\alpha, \beta 1}^{\xi, i, 11} + h_{\alpha, \beta 2}^{\xi, i, 11} + h_{\alpha, \beta 3}^{\xi, i, 11} + h_{\alpha, \beta 4}^{\xi, i, 11} + h_{\alpha, \beta 5}^{\xi, i, 11} + \\ h_{\alpha, \beta 6}^{\xi, i, 11} + h_{\alpha, \beta 7}^{\xi, i, 11} + h_{\alpha, \beta 8}^{\xi, i, 11} + (1 - h_{\alpha, \beta, \gamma}^{\xi, i, 11}) \leq 8 \end{aligned} \quad (542)$$

Tiles i not placed in layer 0 must force all values of their matrices $H^{\xi,i,11}$ to zero by the following $\ell(m-n+1)^3$ inequalities:

$$h_{\alpha,\beta,\gamma}^{\xi,i,11} \leq t_{i,z} \quad (543)$$

A.6.12 Layer with index ξ is in set D_0/D_1

Name

For all $\xi \in [\ell], i, j \in [m]$:

$$h_{ijD\xi}^{18}$$

Definition

This variable is the equivalent of the sets D_0 and D_1 in terms of inequalities. It is given for all $\xi \in [\ell]$:

$$h_{ijD\xi}^{18} = \begin{cases} 1, & \text{iff } \xi \in D_{h_{ijUP}^{18}} \\ 0, & \text{otherwise.} \end{cases} \quad (544)$$

Logical Constraint

The first implication " $\xi \notin D_{h_{ijUP}^{18}} \rightarrow h_{ijD\xi}^{18} = 0$ " is expressed. This is equivalent to

$$h_{ijD\xi}^{18} = 0 \text{ or } \xi \in D_{h_{ijUP}^{18}} \quad (545)$$

For case I this is equivalent to:

$$h_{ijD\xi}^{18} = 0 \text{ or } (\xi \leq s_{i,z} \text{ and } s_{j,z} + 1 \leq \xi), \text{ (or } h_{ijUP}^{18} = 0 \text{),} \quad (546)$$

for case II, we have:

$$h_{ijD\xi}^{18} = 0 \text{ or } (\xi \leq s_{j,z} - 1 \text{ and } s_{i,z} \leq \xi), \text{ (or } h_{ijUP}^{18} = 1 \text{).} \quad (547)$$

For the first case, this yields these two inequalities for all $i, j \in [m], \xi \in [\ell]$:

$$\xi \leq s_{i,z} + (1 - h_{ijUP}^{18})c_1^{11} + (1 - h_{ijD\xi}^{18})c_1^{11} \quad (548)$$

$$s_{j,z} + 1 \leq \xi + (1 - h_{ijUP}^{18})c_1^{11} + (1 - h_{ijD\xi}^{18})c_1^{11} \quad (549)$$

and these two for the second case:

$$\xi \leq s_{j,z} - 1 + h_{ijUP}^{18}c_1^{11} + (1 - h_{ijD\xi}^{18})c_1^{11} \quad (550)$$

$$s_{i,z} \leq \xi + h_{ijUP}^{18}c_1^{11} + (1 - h_{ijD\xi}^{18})c_1^{11} \quad (551)$$

Second, the implication " $\xi \in D_{h_{ijUP}^{18}} \rightarrow h_{ijD\xi}^{18} = 1$ " is expressed. This is equivalent to

$$h_{ijD\xi}^{18} = 1 \text{ or } \xi \notin D_{h_{ijUP}^{18}} \quad (552)$$

For the first case this is given by two inequalities $\xi \leq s_{j,z}$ and $\xi \geq s_{i,z} + 1$, which yields the inequalities using the binary auxiliary variable

$$\xi \leq s_{j,z} + h_{ij\xi\text{OR1}}^{18} c_1^{11} + (1 - h_{ij\text{UP}}^{18}) c_1^{11} + h_{ij\text{D}\xi}^{18} c_1^{11}, \quad (553)$$

$$s_{i,z} \leq \xi + (1 - h_{ij\xi\text{OR1}}^{18}) c_1^{11} + (1 - h_{ij\text{UP}}^{18}) c_1^{11} + h_{ij\text{D}\xi}^{18} c_1^{11} \quad (554)$$

and the two inequalities $\xi \leq s_{i,z} - 1$ and $\xi \geq s_{j,z}$, for the second case, which yields:

$$\xi \leq s_{i,z} - 1 + h_{ij\xi\text{OR2}}^{18} c_1^{11} + h_{ij\text{UP}}^{18} c_1^{11} + h_{ij\text{D}\xi}^{18} c_1^{11}, \quad (555)$$

$$s_{j,z} \leq \xi + (1 - h_{ij\xi\text{OR2}}^{18}) c_1^{11} + h_{ij\text{UP}}^{18} c_1^{11} + h_{ij\text{D}\xi}^{18} c_1^{11}. \quad (556)$$

A.6.13 Packet travel direction

Name

For all $i, j \in [m]$:

$$h_{ij\text{UP}}^{18}$$

Definition

A packet may be transmitted upwards or downwards. The auxiliary variable indicates, whether for a pair of components i, j the packets are traveling upwards or downwards. We introduce an auxiliary binary variable $h_{ij\text{UP}}^{18}$ to separate the two cases for upward and downward direction (I) and (II).

Logical Constraint

We use the constant $c_1^{11} = x_{\max} + y_{\max} + \ell$. Note that the case $s_{i,z} = s_{j,z}$ is within case I.

$$h_{ij\text{UP}}^{18} = \begin{cases} 1, & \text{iff packet travels upwards: } s_{i,z} \geq s_{j,z} \\ 0, & \text{otherwise.} \end{cases} \quad (557)$$

This yields these two inequalities:

$$s_{i,z} \geq s_{j,z} - (1 - h_{ij\text{UP}}^{18}) c_1^{11}, \quad (558)$$

$$s_{i,z} \leq s_{j,z} - 1/2 + h_{ij\text{UP}}^{18} c_1^{11} \quad (559)$$

A.7 HEURISTIC ALGORITHM

A.7.1 Layer component assignment:

The optimization problem

$$\sum_{i \in [n]} \omega_C h + \omega_E \sum_{j \in [\ell], i \in [n]} f_E^{i,j} x_{i,j} + \omega_P \sum_{j \in [\ell], i \in [n]} f_P^{i,j} x_{i,j} \longrightarrow \min \quad (560)$$

is solved, with $\alpha : [n] \rightarrow [\ell]$, the layer component assignment, also being the solution of the optimization problem. This is modeled using the binary variables $x_{i,j} \in \{0, 1\}$ for all components $i \in [n]$ and layers $j \in [\ell]$, which will be 1 if and only if component i is in layer j . Further, the auxiliary variable h is used to model the maximum layer area. The optimization is subject to constraints that all components are in one layer and that h is the size of the largest layer:

$$\sum_{j \in [\ell]} x_{i,j} \geq 1 \quad \forall i \in [n] \quad (561)$$

$$h \geq \sum_{i \in [n]} f_C^{i,j} x_{i,j} \quad \forall j \in [\ell] \quad (562)$$

SIMULATION MODELS

B.1 APPLICATION MODEL

The application model was originally published in [JM 5]. Since the definition is unique, the following definition is directly taken from this reference:

"We propose to use *stochastic, colored Petri nets with retention time on places* to model the timing in the application model. We define the model successively. A conventional Petri net is a starting point:

Definition b.1 (Petri net). A Petri net $\mathcal{N} = (P, T, F, V, m_0)$ is a tuple such that:

- The finite sets P and T are called *places* and *transitions* with $P \cap T = \emptyset$ and $P \cup T \neq \emptyset$.
- The relation $F \subseteq (P \times T) \cup (T \times P)$ is called *flow relation* and consists of *arcs*. Arcs connect places and transitions.
- The arcs are weighted by the *weight-function* $V : F \rightarrow \mathbb{N} \setminus \{0\}$.
- The *initial marking* assigns a number of initial marks to each place: $m_0 : P \rightarrow \mathbb{N}$.

A Petri net is shown in Figure 102. There are two places p_1 and p_2 with 3 and 2 tokens. Data are sent between the two places and from the first place to itself.

The distribution of tokens in the net is given by a *marking*:

Definition b.2 (Marking). Let \mathcal{N} be a Petri net with places P . The function $m : P \rightarrow \mathbb{N}$ is a *marking*. We call G the set of markings. An element $m \in G$ is given by $m = (m(1), \dots, m(|T|))$ with a number of tokens $m(p)$ in all places $p \in P$, and $|T|$ the number of transitions.

Tokens are transmitted via enabled transitions:

Definition b.3 (Enabled). Let \mathcal{N} be a Petri net and let m be a marking in \mathcal{N} . A transition $t \in T$ is *enabled*, if the minimum number of necessary tokens is on the place is given in a marking m , i. e. $V(t, p) \geq m$. The set of pre-places of a transition t is denoted by $\bullet t = \{p \in P : (p, t) \in F\}$. The set of enabled transitions for a given marking $m \in G$

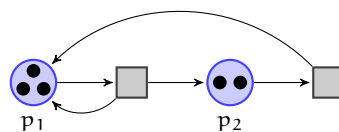


Figure 102: Petri net.

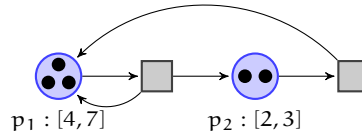


Figure 103: Petri net with retention time for places [JM 5].

is thus $T(m) = \{t \in T : m(t) \geq V(\bullet t, p)\}$. The set of markings, in which a transition t is enabled is $G(t) = \{m \in G : t \in T(m)\}$. In a dual manner, $t^\bullet = \{p \in P : (t, p) \in F\}$ denotes the set of post-places the transition t .

There are many definitions of *Petri nets with timing*. We aim at modeling the calculation times of tasks. So, places must be invalid for a time period although they already consumed enough tokens. This is modeled by *duration of invalidity* for places. An exemplary net is shown in Figure 103 in which both places have retention times of $[4, 7]$ and $[2, 3]$. We extend our model by Petri nets with retention times on places [153].

Definition b.4 (Petri net with retention time on places). The tuple $J = (\mathcal{N}, I)$ is called *Petri net with retention time on places*. The set $\mathcal{N} = (P, T, F, V, m_0)$ is a Petri net, cp. Def. b.1. The function I assigns a retention time interval to each place:

$$I : P \rightarrow \mathbb{Q}_0^+ \times (\mathbb{Q}_0^+ \cup \{\infty\}) \tag{563}$$

It must further hold that the bounds of the retention time interval are well-ordered ascending, i.e. it holds for all places $p \in P$ with $I(p) = (l_p, u_p) : l_p \leq u_p$ ¹.

The retention time influences the enabling of transitions. Places are only allowed to fire if their retention time is over. We kindly refer to [153], Def. 5.6 – Def. 5.11, for a complete definition which is skipped for the sake of brevity and to avoid redundancy.

Petri nets with non-deterministic transitions are called *stochastic Petri nets* [154]. A firing probability is assigned to each transition:

Definition b.5 (Firing probability function). Consider a $T^* \subseteq T$. The probability of a new marking m' from an old marking m for the transition T^* is $p(m', m, T^*)$. Further properties of p are:

- For each marking $m \in G$ and each transition set $T^* \in T(m)$, the function $p(\cdot, m, T^*)$ is a probability mass function on G : $\sum_{m' \in G} p(m', m, T^*) = 1$

¹ Note, that it is also possible use $\mathbb{N}_0^+ \times (\mathbb{N}_0^+ \cup \{\infty\})$ as target set. It is sufficiently precise to use \mathbb{N} [168].

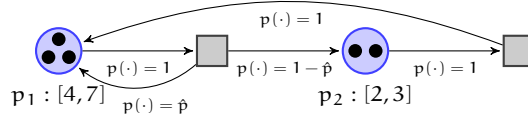


Figure 104: Stochastic Petri net with retention time for places [JM 5].

- The function $p(m', m, T^*)$ must be positive. We permit this if $m = (m(1), \dots, m(|T|))$ and $m' = (m(1)', \dots, m(|T|)')$ and T^* satisfies for all $j \in [1, \dots, |T|]$:

$$\begin{aligned} m_j - \sum_{t^* \in T^*} V(t^*, \bullet t^*) \mathbb{1}_{\bullet t^*}(d_j) &\leq m'_j \\ &\leq m_j + \sum_{t^* \in T^*} V(t^*, t^* \bullet) \mathbb{1}_{t^* \bullet}(d_j) \end{aligned} \quad (564)$$

Here, $\mathbb{1}_S$ is the indicator function of a set S . The sum $\sum_{t^* \in T^*} \mathbb{1}_{\bullet t^*}(d_j)$ is the number of transitions t^* with the input place d_j . Similar, $\sum_{t^* \in T^*} \mathbb{1}_{t^* \bullet}(d_j)$ is the number of transitions t^* with the output place d_j . Multiplication with the edge weights in V ensures that the tokens increase and decrease as wished.

We extend our Petri net with timing annotation from Definition b.4 using the firing probability function:

Definition b.6 (Stochastic Petri net with retention time on places). The tuple $\mathcal{P} = (\mathcal{S}, I)$ is called *stochastic Petri net with retention time on places*. The set $\mathcal{S} = (P, T, F, V, m_0, p(\cdot, m, T^*))$ is a stochastic Petri net. The function I assigns a retention time interval to each place (cp. Def. b.4).

An example for such a Petri net is shown in Figure 104. Two transitions are non-deterministic², which are taken with probability \hat{p} and $1 - \hat{p}$.

We use *colors* to annotate tokens with data type classes.

Definition b.7 (Colored stochastic Petri net with retention time on places). The tuple $\mathcal{A} = (\mathcal{P}, \Sigma, C)$ is called *colored stochastic Petri net with retention time on places*. The set $\mathcal{P} = (P, T, F, V, m_0, p(\cdot, m, T^*), I)$ is a stochastic Petri net with retention time on places. The set Σ is the set of colors. The function C assigns a color Σ to each place P : $C : P \rightarrow \Sigma$.

An example for such a Petri net is shown in Figure 54. Colors model statistical properties of traffics in SoCs. For instance, the red tokens are uncorrelated sensor data. The purple tokens represent correlated data of a PE. This allows to calculate the dynamic energy consumption of links and routers and evaluate coding for the first time in an NoC simulation model." [JM 5]

² We refer to [154], Def. 1.5. for a definition of deterministic transitions.

LIST OF FIGURES

Figure 1	3D CPU-FPGA-DRAM [1].	3
Figure 2	A-3D NoCs trends	4
Figure 3	Wire length estimation for chips	12
Figure 4	3D chip floorplanning	15
Figure 5	TSV	17
Figure 6	3D chip with TSVs	17
Figure 7	BOSCH process	18
Figure 8	Issues TSV production	19
Figure 9	Via middle process flow	20
Figure 10	“e-Cube” [17].	22
Figure 11	3D VSoC	22
Figure 12	Network-on-Chip	25
Figure 13	Packet switching	26
Figure 14	Virtual cut through switching	27
Figure 15	Wormhole switching	27
Figure 16	Input-buffered router design	28
Figure 17	Exemplary router pipeline	29
Figure 18	NoC topologies	32
Figure 19	Deadlock	34
Figure 20	Synthetic traffic patterns.	37
Figure 21	Modeling random injection rates	38
Figure 22	Exemplary 3D NoC	40
Figure 23	Heterogeneous 3D SoC	48
Figure 24	Bit-complement traffic pattern	48
Figure 25	Heterogeneous 3D SoC	49
Figure 26	Design space	50
Figure 27	Design space, with variables	51
Figure 28	Instance of design space	52
Figure 29	Exemplary design space	53
Figure 30	Abstraction levels	53
Figure 31	Incremental approach	55
Figure 32	Input of system-level optimization	61
Figure 33	System-level optimization solution	61
Figure 34	Router model	63
Figure 35	Elevator first DOR	63
Figure 36	Redistribution	63
Figure 37	Vertical connection with RD	63
Figure 38	Coordinate system with three tiles in two layers.	66
Figure 39	Unequal relation interval	71
Figure 40	Product linearization	72
Figure 41	Subproblems visually	76

Figure 42	Flow chart	77
Figure 43	TSV graph	81
Figure 44	TSV placement	81
Figure 45	Small input component communication	89
Figure 46	Result MILP 5 components	89
Figure 47	Result heuristic LP 5 components	89
Figure 48	Result heuristic SDP 5 components	89
Figure 49	Placed homogeneous 3D SoC	91
Figure 50	Placed heterogeneous 3D SoC	92
Figure 51	Link utilization	96
Figure 52	Overview chapter 7	102
Figure 53	Model	104
Figure 54	Colored stochastic Petri net with retention times	105
Figure 55	Router model	106
Figure 56	Architecture of simulation tools	110
Figure 57	Timing of tasks	111
Figure 58	TLM adaptor	115
Figure 59	Database structure in the reporting tool [JM 10].	117
Figure 60	Process	118
Figure 61	Exemplary user interface	119
Figure 62	3D VSoC application example	124
Figure 63	Model of face tracking algorithm	125
Figure 64	Two-layered 3D NoC.	128
Figure 65	Router architecture.	128
Figure 66	Router time behavior	129
Figure 67	Microarchitectural optimization	130
Figure 68	Architectural optimization	131
Figure 69	(Micro-)architectural optimization	131
Figure 70	PPA model of buffer depths	132
Figure 71	Area scaling model	143
Figure 72	Horizontal communication model	146
Figure 73	Vertical communication model	148
Figure 74	Propagation speed ω	148
Figure 75	Connectivity of XYZ routing	150
Figure 76	Routing principle 1	151
Figure 77	Routing principle 2	151
Figure 78	Router ordering	153
Figure 79	Cardinal directions	153
Figure 80	$Z^+(XY)Z$ routing	154
Figure 81	Routing threshold	157
Figure 82	High vertical-throughput router	163
Figure 83	Modified input buffer	163
Figure 84	Modified crossbar	164
Figure 85	High-throughput connection	164
Figure 86	Area comparison	166
Figure 87	Timing comparison	166

Figure 88	Latency speedup of $Z^+(XY)Z^-$	167
Figure 89	Latency speedup of $ZXYZ$	168
Figure 90	Throughput of modified router architecture . .	168
Figure 91	3D VSoC case study	170
Figure 92	Overview on contributions	176
Figure 93	logical or relation	183
Figure 94	Indexes of routers, components, and tiles . . .	186
Figure 95	Router not starting a tile.	186
Figure 96	Link Area	189
Figure 97	Link inequalities	190
Figure 98	Line Equations Configurations	190
Figure 99	Overlapping Areas	198
Figure 100	Optimization tetraeder OR	200
Figure 101	Auxiliary variables \tilde{d}_ξ^{ij} and \tilde{s}_ξ^{ij}	201
Figure 102	Petri net	225
Figure 103	Petri net with time windows	226
Figure 104	Stochastic, times Petri net	227

LIST OF TABLES

Table 1	Layer planning	87
Table 2	VOPD TSV planning	88
Table 3	DVOPD TSV planning	88
Table 4	Execution time of optimization	90
Table 5	Area LP vs. SPD	91
Table 6	NoC planning for heterogenous 3D SoC . .	96
Table 7	Simulator comparison	122
Table 8	Dynamic link energy with VCs	124
Table 9	Dynamic link energy without VCs	124
Table 10	Power results	133
Table 11	Benchmark results	134
Table 12	Performance results	136
Table 13	Results microarchitectural optimization	137
Table 14	Benchmark results	139
Table 15	Possible turns (f, g) in R_1 and R_2	158

LISTINGS

Listing 1	Definition of a task.	112
-----------	-------------------------------	-----

Listing 2	Definition of data types.	112
Listing 3	Exemplary synthetic traffic.	112
Listing 4	Node types.	114
Listing 5	Position of nodes.	114
Listing 6	Connections.	114
Listing 7	Mapping.	114
Listing 8	$Z^+(XY)Z^-$	157
Listing 9	ZXYZ.	157

ACRONYMS (IN ORDER OF APPEARANCE)

TSV	Through-silicon via
FPGA	Field Programmable Gate Array
EU	European union
GALS	globally asynchronous locally synchronous
NoC	Network on chip
SoC	System on chip
IC	integrated circuit
3D VSoC	3D Vision System on chip
HMC	Hybrid memory cube
HBM	High bandwidth memory
CMOS	Complementary metal-oxide-semiconductor
KOZ	Keep-out zone
FEOL	front-end-of-line
BEOL	back-end-of-line
CMP	chemical mechanical polishing and planarization
DRAM	Dynamic Random Access Memory
DiRAM	Disintegrated Random Access Memory
CPU	Central Processing Unit
PE	Processing Element
NI	Network Interface
QoS	Quality of Service
VC	Virtual Channel
FiFo	First-in-First-out (buffer)
RC	Routing calculation

SA	switch allocation
ST	switch traversal
LT	link traversal
DOR	Dimension ordered routing
MILP	mixed integer linear program
ILP	integer linear program
QoR	Quality of results
PPA	Performance, power, area
CNN	Convolutional neural network
PDE	Partial differential equation
ODE	Ordinary differential equation
SIMD	Single instruction multiple data
TLM	Transaction level model
CA	Cycle accurate
RTL	Register transfer level
UMTS	Universal Mobile Telecommunication System
PARSEC	Princeton Application Repository for Shared-Memory Computers
VHDL	Very High Speed Integrated Circuit Hardware Description Language
GALS	Globally Asynchronous Locally Synchronous
RD	Redistribution
VLSI	Very Large Scale Integration
PSO	Particle Swarm Optimization
ULV	Ultra Low Power
GP	General Purpose
ADC	Analog Digital Converter
CIS	CMOS Image Sensor

- [1] X. Chen and N. K. Jha. "A 3-D CPU-FPGA-DRAM Hybrid Architecture for Low-Power Computation." In: *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 24.5 (2016), pp. 1649–1662. ISSN: 1063-8210. DOI: 10.1109/TVLSI.2015.2483525.
- [2] X. Dong and Y. Xie. "System-level cost analysis and design exploration for three-dimensional integrated circuits (3D ICs)." In: *Asia and South Pacific Design Automation Conference (2009)*. DOI: 10.1109/ASPAC.2009.4796486.
- [3] F. Lemonnier et al. "Towards future adaptive multiprocessor systems-on-chip: An innovative approach for flexible architectures." In: *2012 International Conference on Embedded Computer Systems*. 2012, pp. 228–235. DOI: 10.1109/SAMOS.2012.6404179.
- [4] L. Benini and G. de Micheli. "Networks on chips: A new SoC paradigm." In: *Computer* 35.1 (2002), pp. 70–78. DOI: 10.1109/2.976921.
- [5] M. O. Agyeman, A. Ahmadinia, and A. Shahrabi. "Low power heterogeneous 3D Networks-on-Chip architectures." In: *2011 International Conference on High Performance Computing and Simulation (HPCS)* (2011). DOI: 10.1109/HPCSim.2011.5999871.
- [6] S. H. Seyyedaghaei Rezaei, A. Mazloumi, M. Modarressi, and P. Lotfi-Kamran. "Dynamic Resource Sharing for High-performance 3-D Networks-on-Chip." In: *IEEE Computer Architecture Letters* 15.1 (2015). ISSN: 1556-6056. DOI: 10.1109/LCA.2015.2448532.
- [7] E. Sotiriou-Xanthopoulos, D. Diamantopoulos, K. Siozios, G. Economakos, and D. Soudris. "A framework for rapid evaluation of heterogeneous 3-D NoC architectures." In: *Microprocessors and Microsystems* 38.4 (2014), pp. 292–303. ISSN: 0141-9331. DOI: 10.1016/j.micpro.2013.09.003.
- [8] A. M. Rahmani et al. "High-Performance and Fault-Tolerant 3D NoC-Bus Hybrid Architecture Using ARB-NET-Based Adaptive Monitoring Platform." In: 63 (2014), pp. 734–747.
- [9] R. Sunkam Ramanujam and B. Lin. "A Layer-Multiplexed 3D On-Chip Network Architecture." In: *IEEE Embedded Systems Letters* 1.2 (2009), pp. 50–55. ISSN: 1943-0663. DOI: 10.1109/LES.2009.2034710.
- [10] W. R. Davis et al. "Demystifying 3D ICs: The Pros and Cons of Going Vertical." In: *IEEE Design and Test* (2005), pp. 498–510. DOI: 10.1109/MDT.2005.136.
- [11] Á. Zarándy. *Focal-plane sensor-processor chips*. Springer, 2011. ISBN: 9781441964755.
- [12] V. F. Pavlidis. *Three-dimensional Integrated Circuit Design*. Morgan Kaufmann, 2009. ISBN: 978-0-123-74343-5.
- [13] M. Lee, J. S. Pak, and J. Kim. *Electrical Design of Through Silicon Via*. Dordrecht: Springer, 2014. ISBN: 978-94-017-9037-6.
- [14] K. Salah, Y. I. Ismail, and A. El-Rouby. *Arbitrary Modeling of TSVs for 3D Integrated Circuits*. Analog Circuits and Signal Processing. Springer, 2015. ISBN: 978-3-319-07610-2.

- [15] E. Azarkhish, I. Loi, and L. Benini. "A case for three-dimensional stacking of tightly coupled data memories over multi-core clusters using low-latency interconnects." In: *Computers Digital Techniques* (2013), pp. 191–199. DOI: 10.1049/iet-cdt.2013.0031.
- [16] C. Weis, I. Loi, L. Benini, and N. Wehn. "Exploration and Optimization of 3-D Integrated DRAM Subsystems." In: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* (2013), pp. 597–610. ISSN: 0278-0070. DOI: 10.1109/TCAD.2012.2235125.
- [17] P. E. Garrou, M. Koyanagi, and P. Ramm. *3D process technology: Robust circuit and physical design for sub-65 nm technology nodes*. First edition. Vol. volume 3. Handbook of 3D integration. Hoboken, NJ: Wiley, 2009. ISBN: 978-3-527-32034-9.
- [18] A. Heittmann and U. Ramacher. *Electrical Performance of 3D Circuits*. First edition. Vol. volume 3. Handbook of 3D integration. Hoboken, NJ: Wiley, 2009. ISBN: 978-3-527-32034-9.
- [19] J. W. Joyner, P. Zarkesh-Ha, J. A. Davis, and J. D. Meindl. "A three-dimensional stochastic wire-length distribution for variable separation of strata." In: *Proceedings of the IEEE Electron Devices Society*. 2000, pp. 126–128. DOI: 10.1109/IITC.2000.854301.
- [20] P. E. Garrou, C. A. Bower, and P. Ramm. *Handbook of 3D integration. Volume 1 and 2, Technology and applications of 3D integrated circuits*. Wiley, 2012. ISBN: 978-3527332656.
- [21] Samsung. *Samsung Electronics Develops 16-Chip Multi-Stack Package Technology*. Nov 01, 2006. URL: <http://www.samsung.com/semiconductor/insights/news/4212>.
- [22] V. H. Nguyen and P. Christie. "The impact of interstratal interconnect density on the performance of three-dimensional integrated circuits." In: *Proceedings of the 2005 international workshop on System level interconnect prediction*. 2005, p. 73. ISBN: 1595930337. DOI: 10.1145/1053355.1053372.
- [23] H.-H. S. Lee and K. Chakrabarty. "Test Challenges for 3D Integrated Circuits." In: *IEEE Design and Test* (2009). DOI: 10.1109/MDT.2009.125.
- [24] Y.-J. Huang et al. "A built-in self-test scheme for the post-bond test of TSVs in 3D ICs." In: *IEEE 29th VLSI Test Symposium (VTS)*. 2011, pp. 20–25. DOI: 10.1109/VTS.2011.5783749.
- [25] J. Fu, L. Hou, B. Lu, and J. Wang. "Thermal analysis and thermal optimization of through silicon via in 3D IC." In: *12th IEEE International Conference on Solid-State and Integrated Circuit Technology*. 2014. ISBN: 978-1-4799-3282-5. DOI: 10.1109/ICSICT.2014.7021445.
- [26] J. Cong, J. Wei, and Y. Zhang. "A thermal-driven floorplanning algorithm for 3D ICs." In: *International Conference on Computer Aided Design*. IEEE, 2004. ISBN: 0-7803-8702-3. DOI: 10.1109/ICCAD.2004.1382591.
- [27] B. Goplen and S. Sapatnekar. "Efficient thermal placement of standard cells in 3D ICs using a force directed approach." In: *International conference on computer aided design*. IEEE, 2003. ISBN: 1-58113-762-1. DOI: 10.1109/ICCAD.2003.1257591.
- [28] J. Cong and Y. Zhang. "Thermal via planning for 3-D ICs." In: *International Conference on Computer-Aided Design*. IEEE, 2005. ISBN: 0-7803-9254-X. DOI: 10.1109/ICCAD.2005.1560164.

- [29] B. Goplen and S. S. Sapatnekar. "Placement of thermal vias in 3-D ICs using various thermal objectives." In: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* (2006). ISSN: 0278-0070. DOI: 10.1109/TCAD.2006.870069.
- [30] A. Jain, R. E. Jones, R. Chatterjee, and S. Pozder. "Analytical and Numerical Modeling of the Thermal Performance of Three-Dimensional Integrated Circuits." In: *IEEE Transactions on Components and Packaging Technologies* 33.1 (2010), pp. 56–63. ISSN: 1521-3331. DOI: 10.1109/TCAPT.2009.2020916.
- [31] I. Savidis and E. G. Friedman. "Electrical modeling and characterization of 3-D vias." In: *IEEE International Symposium on Circuits and Systems*. 2008. ISBN: 978-1-4244-1683-7. DOI: 10.1109/ISCAS.2008.4541535.
- [32] L. Bamberg and A. García-Ortiz. "High-Level Energy Estimation for Submicrometric TSV Arrays." In: *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 25.10 (2017), pp. 2856–2866. ISSN: 1063-8210. DOI: 10.1109/TVLSI.2017.2713601.
- [33] Z. Li et al. "Hierarchical 3-D Floorplanning Algorithm for Wire-length Optimization." In: *IEEE Transactions on Circuits and Systems I: Regular Papers* 53.12 (2006), pp. 2637–2646. DOI: 10.1109/TCSI.2006.883857.
- [34] N. A. Sherwani. *Algorithms for VLSI Physical Design Automation*. Third Edition. Boston, MA: Kluwer Academic Publishers, 2002. ISBN: 9780792383932. DOI: 10.1007/b116436.
- [35] M. Healy et al. "Multiobjective Microarchitectural Floorplanning for 2-D and 3-D ICs." In: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 26.1 (2007), pp. 38–52. ISSN: 0278-0070. DOI: 10.1109/TCAD.2006.883925.
- [36] M. Ohmura. "An initial placement algorithm for 3-D VLSI." In: *IEEE International Symposium on Circuits and Systems*. 1998. ISBN: 0-7803-4455-3. DOI: 10.1109/ISCAS.1998.705245.
- [37] T. Tanprasert. "An analytical 3-D placement that reserves routing space." In: *IEEE International Symposium on Circuits and Systems*. 2000. ISBN: 0-7803-5482-6. DOI: 10.1109/ISCAS.2000.855998.
- [38] R. Hentschke and R.A.L. Reis. "A 3D-Via Legalization Algorithm for 3D VLSI Circuits and its Impact on Wire Length." In: *IEEE International Symposium on Circuits and Systems*. IEEE, 2007. ISBN: 1-4244-0920-9. DOI: 10.1109/ISCAS.2007.378497.
- [39] D. H. Kim, K. Athikulwongse, and S. K. Lim. "Study of Through-Silicon-Via Impact on the 3-D Stacked IC Layout." In: *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 21.5 (2013), pp. 862–874. ISSN: 1063-8210. DOI: 10.1109/TVLSI.2012.2201760.
- [40] E. Wong, J. Minz, and S. K. Lim. "Power supply noise-aware 3D floorplanning for system-on-package." In: *IEEE 14th topical meeting on electrical performance of electronic packaging*. IEEE, 2004. ISBN: 0-7803-9220-5. DOI: 10.1109/EPEP.2005.1563753.
- [41] M. Pathak, Y.-J. Lee, T. Moon, and S. K. Lim. "Through-silicon-via management during 3D physical design: When to add and how many?" In: *International Conference on Computer-Aided Design*. IEEE, 2010. ISBN: 978-1-4244-8193-4. DOI: 10.1109/ICCAD.2010.5653703.
- [42] A. Hashimoto and J. Stevens. "Wire routing by optimizing channel assignment within large apertures." In: *Proceedings of the 1971*

- design automation workshop on design automation*. ACM Press, 1971. DOI: 10.1145/800158.805069.
- [43] R. J. Enbody, G. Lynn, and K. H. Tan. "Routing the 3-D chip." In: *Design Automation Conference*. ACM Press, 1991. ISBN: 0897913957. DOI: 10.1145/127601.127644.
- [44] S. Tayu and S. Ueno. "On the Complexity of Three-Dimensional Channel Routing." In: *IEEE International Symposium on Circuits and Systems*. IEEE, 2007. ISBN: 1-4244-0920-9. DOI: 10.1109/ISCAS.2007.378297.
- [45] J. Minz and S. K. Lim. "Block-level 3-D Global Routing With an Application to 3-D Packaging." In: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 25.10 (2006), pp. 2248–2257. ISSN: 0278-0070. DOI: 10.1109/TCAD.2005.860952.
- [46] Cadence Design Systems Inc. *3D-IC Design Solutions*. 2017. URL: https://www.cadence.com/content/cadence-www/global/en_US/home/solutions/3dic-design-solutions.html.
- [47] S. Panth, S. K. Samal, K. Samadi, Y. Du, and S. K. Lim. "Tier Degradation of Monolithic 3-D ICs: A Power Performance Study at Different Technology Nodes." In: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 36.8 (2017), pp. 1265–1273. ISSN: 0278-0070. DOI: 10.1109/TCAD.2017.2681064.
- [48] J. T. Pawlowski. "Hybrid memory cube (HMC)." In: *Hot Chips 23 Symposium*. IEEE, 2011, pp. 1–24. ISBN: 978-1-4673-8877-1. DOI: 10.1109/HOTCHIPS.2011.7477494.
- [49] W. Shockley. "Semiconductive wafer and method of making the same." Pat. US3044909. 1962.
- [50] F. Laermer and A. Schilp. "Method of anisotropically etching silicon." Pat. US5501893. 1996.
- [51] G. Feng, Xiao Peng, J. Cai, and S. Wang. "Through wafer via technology for 3-D packaging." In: *6th International Conference on Electronic Packaging Technology*. IEEE, 2005. ISBN: 0-7803-9449-6. DOI: 10.1109/ICEPT.2005.1564661.
- [52] T. C. Tsai et al. "CMP Process Development for the Via-Middle 3D TSV Applications at 28nm Technology Node." In: *Advanced Metallization Conference*. 2010.
- [53] Thuy Dao, Dina H. Triyoso, Mike Petras, and Michael Canonico. "Through silicon via stress characterization." In: *IEEE International Conference on IC Design and Technology*. 2009, pp. 1–3.
- [54] M. Said, M. El-Sayed, F. Mehdipour, and N. Miyakawa. "Keep-Out-Zone analysis for three-dimensional ICs." In: *International Symposium on VLSI Design, Automation, and Test*. IEEE, 2014. ISBN: 978-1-4799-2776-0. DOI: 10.1109/VLSI-DAT.2014.6834862.
- [55] K. H. Lu et al. "Thermo-mechanical reliability of 3-D ICs containing through silicon vias." In: *Electronic components and Technology Conference*. IEEE, 2009. ISBN: 978-1-4244-4475-5. DOI: 10.1109/ECTC.2009.5074079.
- [56] C. S. Selvanayagam et al. "Nonlinear Thermal Stress/Strain Analyses of Copper Filled TSV (Through Silicon Via) and Their Flip-Chip Microbumps." In: *IEEE Transactions on Advanced Packaging* 32.4 (2009), pp. 720–728. ISSN: 1521-3323. DOI: 10.1109/TADVP.2009.2021661.

- [57] C. Okoro et al. "Analysis of the Induced Stresses in Silicon During Thermocompression Cu-Cu Bonding of Cu-Through-Vias in 3D-SiC Architecture." In: *Electronic Components and Technology Conference*. IEEE, 2007. ISBN: 1-4244-0984-5. DOI: 10.1109/ECTC.2007.373805.
- [58] T. C. Xu, P. Liljeberg, and H. Tenhunen. "Optimal number and placement of Through Silicon Vias in 3D Network-on-Chip." In: *International Symposium on Design and Diagnostics of Electronic Circuits and Systems*. IEEE, 2011. ISBN: 978-1-4244-9755-3. DOI: 10.1109/DDECS.2011.5783057.
- [59] D. Park et al. "MIRA: A Multi-layered On-Chip Interconnect Router Architecture." In: *35th International Symposium on Computer Architecture*. IEEE, 2008. DOI: 10.1109/ISCA.2008.13.
- [60] C. Duan, B. J. LaMeres, and S. P. Khatri. *On and off-chip crosstalk avoidance in VLSI design*. Springer, 2010. ISBN: 9781441909473.
- [61] G. Katti, M. Stucchi, K. de Meyer, and W. Dehaene. "Electrical modeling and characterization of through silicon via for three-dimensional ICs." In: *IEEE Transactions on Electron Devices* 57.1 (2010), pp. 256–262. DOI: 10.1109/TED.2009.2034508.
- [62] T. Kgil et al. "PicoServer: using 3D stacking technology to enable a compact energy efficient chip multiprocessor." In: *ACM SIGARCH Computing Architecture News* (2006). DOI: 10.1145/1168857.1168873.
- [63] P. Jacob et al. "Mitigating Memory Wall Effects in High-Clock-Rate and Multicore CMOS 3-D Processor Memory Stacks." In: *Proceedings of the IEEE* 97.1 (2009), pp. 108–122. DOI: 10.1109/JPROC.2008.2007472.
- [64] X. Yu, L. Li, Y. Zhang, H. Pan, and S. He. "Performance and power consumption analysis of memory efficient 3D network-on-chip architecture." In: *International Conference on Control and Automation* (2013). DOI: 10.1109/ICCA.2013.6565107.
- [65] H. Sun et al. "Design of 3D DRAM and Its Application in 3D Integrated Multi-Core Computing Systems." In: *IEEE Design and Test* (2013). DOI: 10.1109/MDT.2009.93.
- [66] Y. Kikuchi et al. "A 40 nm 222 mW H.264 Full-HD Decoding, 25 Power Domains, 14-Core Application Processor With x512b Stacked DRAM." In: *IEEE Journal of Solid-State Circuits* 46.1 (2011), pp. 32–41. ISSN: 0018-9200. DOI: 10.1109/JSSC.2010.2079370.
- [67] K. Abe et al. "Ultra-high bandwidth memory with 3D-stacked emerging memory cells." In: *IEEE International Conference on Integrated Circuit Design and Technology and Tutorial*. 2008. DOI: 10.1109/ICICDT.2008.4567279.
- [68] D. H. Kim et al. "Design and Analysis of 3D-MAPS (3D Massively Parallel Processor with Stacked Memory)." In: *IEEE Transactions on Computers* 64.1 (2015), pp. 112–125. ISSN: 0018-9340. DOI: 10.1109/TC.2013.192.
- [69] Hybrid Memory Cube Consortium. *Hybrid Memory Cube Specification 1.0*. 2013. URL: http://hybridmemorycube.org/files/SiteDownloads/HMC_Specification%201_0.pdf.
- [70] Tezzaron Semiconductor. *Our Technology 101*. 2015. URL: <http://www.tezzaron.com/about-us/our-technology-101/>.
- [71] K.-W. Lee et al. "Highly dependable 3-D stacked multicore processor system module fabricated using reconfigured multichip-on-

- wafer 3-D integration technology." In: *IEEE International Electron Devices Meeting*. 2014. doi: 10.1109/IEDM.2014.7047128.
- [72] M. Koyanagi, H. Kobayashi, T. Aoki, T. Sueyoshi, and T. Kamada. "A 3D-VLSI Architecture for Future Automotive Visual Recognition." In: *VLSI Design and Test for Systems Dependability*. Ed. by S. Asai. Tokyo: Springer Japan, 2019, pp. 719–733.
- [73] K. Kim, S. Lee, J. Y. Kim, M. Kim, and H. J. Yoo. "A 125 GOPS 583 mW Network-on-Chip Based Parallel Processor With Bio-Inspired Visual Attention Engine." In: *IEEE Journal of Solid-State Circuits* 44.1 (2009), pp. 136–147. issn: 0018-9200. doi: 10.1109/JSSC.2008.2007157.
- [74] K. Jia et al. "AICNN: Implementing Typical CNN Algorithms with Analog-to-Information Conversion Architecture." In: *IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*. IEEE, 2017. isbn: 978-1-5090-6762-6. doi: 10.1109/ISVLSI.2017.23.
- [75] V. S. Ghaderi, D. Song, J. Choma, and T. W. Berger. "Nonlinear Cognitive Signal Processing in Ultralow-Power Programmable Analog Hardware." In: *IEEE Transactions on Circuits and Systems II: Express Briefs* 62.2 (2015), pp. 124–128. doi: 10.1109/TCSII.2014.2387693.
- [76] W. J. Dally and B. Towles. "Route packets, not wires: On-chip interconnection networks." In: *Proceedings of the Design Automation Conference*. IEEE, 2001.
- [77] P. P. Pande, C. Grecu, M. Jones, A. Ivanov, and R. Saleh. "Performance evaluation and design trade-offs for network-on-chip interconnect architectures." In: *IEEE Transactions on Computers* 54.8 (2005), pp. 1025–1040. issn: 0018-9340.
- [78] Y. Cai, K. Mai, and O. Mutlu. "Comparative evaluation of FPGA and ASIC implementations of bufferless and buffered routing algorithms for on-chip networks." In: *16th International Symposium on Quality Electronic Design*. IEEE, 2015. isbn: 978-1-4799-7581-5. doi: 10.1109/ISQED.2015.7085472.
- [79] M. Millberg, E. Nilsson, R. Thid, and A. Jantsch. "Guaranteed bandwidth using looped containers in temporally disjoint networks within the nostrum network on chip." In: *Design, Automation and Test in Europe Conference and Exhibition*. IEEE, 2004. isbn: 0-7695-2085-5. doi: 10.1109/DATE.2004.1269001.
- [80] W. J. Dally and C. L. Seitz. "The torus routing chip." In: *Distributed Computing* 1.4 (1986), pp. 187–196. doi: 10.1007/BF01660031.
- [81] N. Banerjee, P. Vellanki, and K. S. Chatha. "A power and performance model for network-on-chip architectures." In: *Design, Automation and Test in Europe Conference and Exhibition*. IEEE, 2004. isbn: 0-7695-2085-5. doi: 10.1109/DATE.2004.1269067.
- [82] Y. Hoskote, S. Vangal, A. Singh, N. Borkar, and S. Borkar. "A 5-GHz Mesh Interconnect for a Teraflops Processor." In: *IEEE micro* 27.5 (2007), pp. 51–61. doi: 10.1109/MM.2007.4378783.
- [83] T. W. Ainsworth and T. M. Pinkston. "Characterizing the Cell EIB On-Chip Network." In: *IEEE Micro* 27.5 (2007), pp. 6–14. doi: 10.1109/MM.2007.4378779.
- [84] P. Gratz et al. "On-Chip Interconnection Networks of the TRIPS Chip." In: *IEEE Micro* 27.5 (2007), pp. 41–50. doi: 10.1109/MM.2007.4378782.

- [85] K. Goossens, J. Dielissen, and A. Radulescu. "AEthereal Network on Chip: Concepts, Architectures, and Implementations." In: *IEEE Design and Test* (2005). DOI: 10.1109/MDT.2005.99.
- [86] J. Flich and D. Bertozzi. *Designing Network On-Chip Architectures in the Nanoscale Era*. Taylor and Francis, 2010. ISBN: 9781439837108.
- [87] M. Langar, R. Bourguiba, and J. Mouine. "Virtual channel router architecture for Network on Chip with adaptive inter-port buffers sharing." In: *13th International Multi-Conference on Systems, Signals & Devices*. IEEE, 2016. ISBN: 978-1-5090-1291-6. DOI: 10.1109/SSD.2016.7473771.
- [88] S. H. S. Rezaei, M. Modarressi, M. Daneshtalab, and S. Roshanise-fat. "A Three-Dimensional Networks-on-Chip Architecture with Dynamic Buffer Sharing." In: *24th Euromicro International Conference on Parallel, Distributed, and Network-Based Processing*. IEEE, 2016. ISBN: 978-1-4673-8776-7. DOI: 10.1109/PDP.2016.124.
- [89] G. Dimitrakopoulos, A. Psarras, and I. Seitanidis. *Microarchitecture of Network-on-Chip Routers: A Designer's Perspective*. Springer, 2015. ISBN: 978-1-4614-4301-8.
- [90] Daniel U. Becker and William J. Dally. "Allocator Implementations for Network-on-chip Routers." In: *Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis*. ACM, 2009, pp. 1–12. DOI: 10.1145/1654059.1654112.
- [91] W. J. Dally. "Virtual-channel flow control." In: *IEEE Transactions on Parallel and Distributed Systems* 3.2 (1992), pp. 194–205.
- [92] J. Duato, S. Yalamanchili, and L. M. Ni. *Interconnection networks: An engineering approach*. Morgan Kaufmann, 2003. ISBN: 9781558608528.
- [93] W. J. Dally and B. Towles. *Principles and Practices of Interconnection Networks*. Elsevier, 2004.
- [94] Y. Hoskote, S. Vangal, S. Dighe, N. Borkar, and S. Borkar. "Teraflops prototype processor with 80 cores." In: *Hot Chips 19 Symposium*. IEEE, 2007. ISBN: 978-1-4673-8869-6. DOI: 10.1109/HOTCHIPS.2007.7482494.
- [95] S. Kundu. *Network-on-Chip: The next generation of system-on-chip integration*. CRC PRESS, 2017. ISBN: 9781138749351.
- [96] Y. Ye et al. "Holistic comparison of optical routers for chip multi-processors." In: *Anti-counterfeiting, Security, and Identification*. 2012, pp. 1–5. DOI: 10.1109/ICASID.2012.6325348.
- [97] V. F. Pavlidis and E. G. Friedman. "3-D Topologies for Networks-on-Chip." In: *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 15.10 (2007), pp. 1081–1090. ISSN: 1063-8210. DOI: 10.1109/TVLSI.2007.893649.
- [98] M. Coppola. "Spidergon STNoC: The technology that adds value to your System." In: *2010 IEEE Hot Chips 22 Symposium (HCS)*. 2010, pp. 1–39. DOI: 10.1109/HOTCHIPS.2010.7480082.
- [99] J. Duato. "A new theory of deadlock-free adaptive routing in worm-hole networks." In: *IEEE Transactions on Parallel and Distributed Systems* 4.12 (1993), pp. 1320–1331. DOI: 10.1109/71.250114.
- [100] M. Ebrahimi et al. "DyXYZ: Fully Adaptive Routing Algorithm for 3D NoCs." In: *Euromicro International Conference on Parallel, Distributed, and Network-Based Processing*. 2013, pp. 499–503. DOI: 10.1109/PDP.2013.80.

- [101] A. Charif, N. E. Zergainoh, and M. Nicolaidis. "A new approach to deadlock-free fully adaptive routing for high-performance fault-tolerant NoCs." In: *2016 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT)*. 2016, pp. 121–126. doi: 10.1109/DFT.2016.7684082.
- [102] M. Ebrahimi, M. Daneshtalab, J. Plosila, and H. Tenhunen. "MAFA: Adaptive Fault-Tolerant Routing Algorithm for Networks-on-Chip." In: *2012 15th Euromicro Conference on Digital System Design*. 2012, pp. 201–207. doi: 10.1109/DSD.2012.82.
- [103] C. J. Glass and L. M. Ni. "The Turn Model for Adaptive Routing." In: *Proceedings the 19th Annual International Symposium on Computer Architecture*. 1992. doi: 10.1109/ISCA.1992.753324.
- [104] W. J. Dally and C. L. Seitz. "Deadlock-free message routing in multiprocessor interconnection networks." In: *IEEE Transactions on Computers* C-36.5 (1987). issn: 0018-9340. doi: 10.1109/TC.1987.1676939.
- [105] R. Pop and S. Kumar. "A survey of techniques for mapping and scheduling applications to network on chip systems." In: *School of Engineering, Jonkoping University, Research Report* (2004).
- [106] P. K. Sahu and S. Chattopadhyay. "A survey on application mapping strategies for Network-on-Chip design." In: *Journal of systems architecture* (2013). doi: 10.1016/j.sysarc.2012.10.004.
- [107] A. Bender. "MILP Based Task Mapping for Heterogeneous Multiprocessor Systems." In: *Proceedings of the Conference on European Design Automation*. IEEE, 1996.
- [108] S. K. Mandal et al. "NoCBench: a benchmarking platform for network on chip." In: *Workshop on Unique Chips and Systems*. 2009.
- [109] G. Kahn. "The semantics of a simple language for parallel programming." In: *Proceedings of the IFIP Congress on Information Processing*. 1974.
- [110] E. Pekkarinen, L. Lehtonen, E. Salminen, and T. D. Hamalainen. "A set of traffic models for Network-on-Chip benchmarking." In: *International Symposium on System on Chip*. 2011.
- [111] W. Liu et al. "A NoC Traffic Suite Based on Real Applications." In: *2011 IEEE Computer Society Annual* (2011), pp. 66–71. doi: 10.1109/ISVLSI.2011.49.
- [112] N. Binkert et al. "The Gem5 Simulator." In: *SIGARCH Computer Architecture News* 39.2 (2011), pp. 1–7. doi: 10.1145/2024716.2024718.
- [113] C. Bienia. "Benchmarking Modern Multiprocessors." PhD thesis. Princeton University, 1.01.2011.
- [114] J. Hestness and S. W. Keckler. *Netrace: Dependency-Tracking Traces for Efficient Network-on-Chip Experimentation*.
- [115] Synopsys. *RTL Synthesis and Test*. 2017. URL: <https://www.synopsys.com/implementation-and-signoff/rtl-synthesis-test.html>.
- [116] Inc. Cadence Design Systems. *Genus Synthesis Solution*. 2016. URL: https://www.cadence.com/content/cadence-www/global/en_US/home/tools/digital-design-and-signoff/synthesis/genus-synthesis-solution.html.
- [117] V. Catania, A. Mineo, S. Monteleone, M. Palesi, and D. Patti. "Noxim: An open, extensible and cycle-accurate network on chip simulator." In: *International Conference on Application-specific Systems*,

- Architectures and Processors*. IEEE, 2015. DOI: 10.1109/ASAP.2015.7245728.
- [118] A. García-Ortiz, L. Indrusiak, T. Murgan, and M. Glesner. "Low-Power Coding for Networks-on-Chip with Virtual Channels." In: *Journal of Low-Power Electronics* 5 (2009), pp. 1–8. DOI: 10.1166/jolpe.2009.1006.
- [119] Z. Lu, R. Thid, M. Millberg, E. Nilsson, and A. Jantsch. "NNSE: Nostrum network-on-chip simulation environment." In: *Proceedings of SSoCC* (2005).
- [120] NoC Blog. *Top 5 most popular NoC simulators*. 2012. URL: <https://networkchip.wordpress.com/2015/11/02/what-is-the-most-popular-full-system-simulator/>.
- [121] Nan Jiang et al. "A detailed and flexible cycle-accurate Network-on-Chip simulator." In: *International Symposium on Performance Analysis of Systems and Software*. IEEE, 2013, pp. 86–96. DOI: 10.1109/ISPASS.2013.6557149.
- [122] V. Catania, A. Mineo, S. Monteleone, M. Palesi, and D. Patti. "Cycle-Accurate Network on Chip Simulation with Noxim." In: *ACM Transactions on Modeling and Computer Simulation* 27.1 (2016), pp. 1–25. DOI: 10.1145/2953878.
- [123] B. S. Feero and P. P. Pande. "Networks-on-Chip in a Three-Dimensional Environment: A Performance Evaluation." In: *IEEE Transactions on Computers* 58.1 (2009), pp. 32–45. ISSN: 0018-9340. DOI: 10.1109/TC.2008.142.
- [124] K. Tatas, K. Siozios, D. Soudris, and A. Jantsch. *Designing 2D and 3D Network-on-Chip Architectures*. Springer, 2014.
- [125] M. O. Agyeman and A. Ahmadinia. "Optimising Heterogeneous 3D Networks-on-Chip." In: *Parallel Computing in Electrical Engineering*. 2011. DOI: 10.1109/PARELEC.2011.40.
- [126] A. B. Ahmed and A. B. Abdallah. "LA-XYZ: Low Latency, High Throughput Look-Ahead Routing Algorithm for 3D Network-on-Chip (3D-NoC) Architecture." In: *International Symposium on Embedded Multicore SoCs* (2012). DOI: 10.1109/MCSoc.2012.24.
- [127] R. Mullins, A. West, and S. Moore. "Low-latency virtual-channel routers for on-chip networks." In: *ACM SIGARCH Computer Architecture News*. 2004.
- [128] S. Foroutan, A. Sheibanyrad, and F. Petrot. "Assignment of Vertical-Links to Routers in Vertically-Partially-Connected 3-D-NoCs." In: *IEEE Transactions on Computer Aided Design of Integrated Circuits and Systems* 33.8 (2014), pp. 1208–1218. DOI: 10.1109/TCAD.2014.2323219.
- [129] M. Bahmani, A. Sheibanyrad, F. Petrot, F. Dubois, and P. Durante. "A 3D-NoC Router Implementation Exploiting Vertically-Partially-Connected Topologies." In: *IEEE Computer Society Annual* (2012). DOI: 10.1109/ISVLSI.2012.19.
- [130] Y. Ying et al. "Economizing TSV Resources in 3-D Network-on-Chip Design." In: *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 23.3 (2015), pp. 493–506. ISSN: 1063-8210. DOI: 10.1109/TVLSI.2014.2311835.
- [131] T. Webber et al. "Tiny – optimised 3D mesh NoC for area and latency minimisation." In: *Electronics Letters* 50.3 (2014), pp. 165–166. ISSN: 0013-5194. DOI: 10.1049/el.2013.2557.

- [132] K. Lee, S.-J. Lee, and H.-J. Yoo. "Low-power network-on-chip for high-performance SoC design." In: *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 14.2 (2006), pp. 148–160. ISSN: 1063-8210. DOI: 10.1109/TVLSI.2005.863753.
- [133] U. Y. Ogras and R. Marculescu. "It's a small world after all: NoC performance optimization via long-range link insertion." In: *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 14.7 (2006), pp. 693–706. ISSN: 1063-8210. DOI: 10.1109/TVLSI.2006.878263.
- [134] J. Lienig. *Layoutsynthese elektronischer Schaltungen - Grundlegende Algorithmen für die Entwurfsautomatisierung*. Springer, 2006. ISBN: 978-3-540-29942-4. DOI: 10.1007/3-540-29942-4.
- [135] J. Cong and G. Luo. "A Multilevel Analytical Placement for 3D ICs." In: *Asia and South Pacific Design Automation Conference. ASP-DAC '09. IEEE, 2009*, pp. 361–366. ISBN: 978-1-4244-2748-2.
- [136] M.-C. Tsai, T.-C. Wang, and T. T. Hwang. "Through-Silicon Via Planning in 3-D Floorplanning." In: *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 19.8 (2011). ISSN: 1063-8210. DOI: 10.1109/TVLSI.2010.2050012.
- [137] M.-K. Hsu, Y.-W. Chang, and V. Balabanov. "TSV-aware Analytical Placement for 3D IC Designs." In: *Design Automation Conference. ACM, 2011*. DOI: 10.1145/2024724.2024875.
- [138] J. Cong, M. Romesis, and J. R. Shinnerl. "Robust Mixed-size Placement Under Tight White-space Constraints." In: *International Conference on Computer-aided Design. ICCAD '05. IEEE, 2005*. ISBN: 0-7803-9254-X.
- [139] J. A. Roy et al. "Capo: Robust and Scalable Open-source Min-cut Floorplacer." In: *International Symposium on Physical Design. ACM, 2005*. DOI: 10.1145/1055137.1055184.
- [140] Thomas Canhao Xu et al. "Optimal placement of vertical connections in 3D Network-on-Chip." In: *Journal of systems architecture* 59.7 (2013), pp. 441–454. DOI: 10.1016/j.sysarc.2013.05.002.
- [141] Kanchan Manna, Santanu Chattopadhyay, and Indranil Sengupta. "Through silicon via placement and mapping strategy for 3D mesh based Network-on-Chip." In: *International Conference on Very Large Scale Integration (VLSI-SoC). IEEE. DOI: 10.1109/VLSI-SoC.2014.7004177*.
- [142] K. Srinivasan, K. S. Chatha, and G. Konjevod. "Linear-programming-based techniques for synthesis of network-on-chip architectures." In: *IEEE Trans. on Very Large Scale Integration (VLSI) Systems* 14.4 (2006), pp. 407–420. ISSN: 1063-8210. DOI: 10.1109/TVLSI.2006.871762.
- [143] W. H. Wolf. "Hardware-software co-design of embedded systems [and prolog]." In: *Proceedings of the IEEE* 82.7 (1994), pp. 967–989. DOI: 10.1109/5.293155.
- [144] B. Niazmand et al. "Logic-based implementation of fault-tolerant routing in 3D network-on-chips." In: *International Symposium on Networks-on-Chip. IEEE, 2016*. ISBN: 978-1-4673-9030-9. DOI: 10.1109/NOCS.2016.7579317.
- [145] B. Korte and J. Vygen. *Combinatorial optimization: Theory and algorithms*. 2nd edition. Springer, 2002. ISBN: 3662560380.
- [146] B. Montreuil. "A Modelling Framework for Integrating Layout Design and flow Network Design." In: (1990).

- [147] T. A. Lacksonen. "Static and Dynamic Layout Problems with Varying Areas." In: *Journal of the Operational Research Society* 45.1 (1994), pp. 59–69. doi: 10.1057/jors.1994.7.
- [148] B. Korte and J. Vygen. *Combinatorial optimization: Theory and algorithms*. 5th edition. Springer, 2012. ISBN: 3642427677.
- [149] IBM. *Cplex 12.8 User's Manual*. 2017.
- [150] L. Khachiyan. "A polynomial algorithm in linear programming." In: *Doklady Akademii Nauk SSSR* 244 (1979), pp. 1093–1096.
- [151] Mosek ApS. *Mosek*. 2018.
- [152] Kanchan Manna, Shivam Swami, Santanu Chattopadhyay, and Indranil Sengupta. "Integrated Through-Silicon Via Placement and Application Mapping for 3D Mesh-Based NoC Design." In: *ACM Transactions on Embedded Computing Systems* 16.1 (2016). doi: 10.1145/2968446.
- [153] L. Popova-Zeugmann. *Time and Petri nets*. Springer, 2013. ISBN: 978-3-642-41115-1.
- [154] P. J. Haas. *Stochastic Petri nets: Modelling, stability, simulation*. Springer, 2002. ISBN: 978-0-387-21552-5.
- [155] A. García-Ortiz and L. S. Indrusiak. "Practical and Theoretical Considerations on Low-Power Probability-Codes for Networks-on-Chip." In: *International Symposium on Power and Timing Modeling, Optimization and Simulation*. 2011. ISBN: 978-3-642-17752-1.
- [156] *IEEE Standard for Standard SystemC Language Reference Manual*. Piscataway, NJ, USA. doi: 10.1109/IEEESTD.2012.6134619.
- [157] 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." In: *Journal of systems architecture* 56.7 (2010), pp. 256–264. doi: 10.1016/j.sysarc.2010.05.002.
- [158] Erik B. van der Tol and E. G. Jaspers. "Mapping of MPEG-4 decoding on a flexible architecture platform." In: *Media Processors* (2002).
- [159] L. Cai and D. Gajski. "Transaction level modeling: an overview." In: *International Conference on Hardware/Software Codesign and System Synthesis*. ACM, 2003.
- [160] N. Jafarzadeh, M. Palesi, A. Khademzadeh, and A. Afzali-Kusha. "Data Encoding Techniques for Reducing Energy Consumption in Network-on-Chip." In: *IEEE Trans. on Very Large Scale Integration (VLSI) Systems* 22.3 (2014), pp. 675–685. doi: 10.1109/TVLSI.2013.2251020.
- [161] P. Viola and M. Jones. "Rapid object detection using a boosted cascade of simple features." In: *Computer Society Conference on Computer Vision and Pattern Recognition*. IEEE, 2001. ISBN: 0-7695-1272-0. doi: 10.1109/CVPR.2001.990517.
- [162] J. Shi and Tomasi C. "Good features to track." In: *Conference on Computer Vision and Pattern Recognition*. IEEE, 1994. ISBN: 0-8186-5825-8. doi: 10.1109/CVPR.1994.323794.
- [163] Tomasi C. and T. Kanade. "Detection and Tracking of Point Features." In: *International Journal of Computer Vision* (1991).
- [164] C.-H. H. Chen et al. "SMART: A Single-cycle Reconfigurable NoC for SoC Applications." In: *Conference on Design, Automation and Test in Europe*. DATE '13. EDA Consortium, 2013. ISBN: 978-1-4503-2153-2.

- [165] Akram Ben Ahmed and Abderazek Ben Abdallah. "Low-overhead Routing Algorithm for 3D Network-on-Chip." In: *ICNC (2012)*, pp. 23–32. DOI: 10.1109/ICNC.2012.14.
- [166] M. Palesi and M. Daneshtalab. *Routing algorithms in Networks-on-Chip*. Springer, 2014. ISBN: 978-1-4614-8274-1.
- [167] Wolfram Research. *Mathematica Edition: Version 10.4*. Champaign, Illinois: Wolfram Research, Inc., 2016.
- [168] P. H. Starke. "A Memo on Time Constraints in Petri Nets." In: *Informatik-Bericht 46* (1995).

DECLARATION

EHRENERKLÄRUNG

Ich versichere hiermit, dass ich die vorliegende Arbeit ohne unzulässige Hilfe Dritter und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe. Die Hilfe eines kommerziellen Promotionsberaters habe ich nicht in Anspruch genommen. Dritte haben von mir weder unmittelbar noch mittelbar geldwerte Leistungen für Arbeiten erhalten, die im Zusammenhang mit dem Inhalt der vorgelegten Dissertation stehen. Verwendete fremde und eigene Quellen sind als solche kenntlich gemacht. Ich habe insbesondere nicht wesentlich:

- Ergebnisse erfunden oder widersprüchliche Ergebnisse verschwiegen,
- statistische Verfahren absichtlich mißbraucht, um Daten in ungerechtfertigter Weise zu interpretieren,
- fremde Ergebnisse oder Veröffentlichungen plagiiert,
- fremde Forschungsergebnisse verzerrt wiedergegeben

Mir ist bekannt, dass Verstöße gegen das Urheberrecht Unterlassungs- und Schadensersatzansprüche des Urhebers sowie eine strafrechtliche Ahndung durch die Strafverfolgungsbehörden begründen kann.

Ich erkläre mich damit einverstanden, dass die Dissertation ggf. mit Mitteln der elektronischen Datenverarbeitung auf Plagiate überprüft werden kann.

Die Arbeit wurde bisher weder im Inland noch im Ausland in gleicher oder ähnlicher Form als Dissertation eingereicht und ist als Ganzes auch noch nicht veröffentlicht.

DECLARATION OF HONOR

I hereby declare that I produced this thesis without prohibited external assistance and that none other than the listed references and tools have been used. I did not make use of any commercial consultant concerning graduation. A third party did not receive any nonmonetary perquisites neither directly nor indirectly for activities which are connected with the contents of the presented thesis. All sources of information are clearly marked, including my own publications. In particular I have not consciously:

- Fabricated data or rejected undesired results
- Misused statistical methods with the aim of drawing other conclusions than those warranted by the available data
- Plagiarized data or publications

- Presented the results of other researchers in a distorted way

I do know that violations of copyright may lead to injunction and damage claims of the author and also to prosecution by the law enforcement authorities.

I hereby agree that the thesis may need to be reviewed with an electronic data processing for plagiarism.

This work has not yet been submitted as a doctoral thesis in the same or a similar form in Germany or in any other country. It has not yet been published as a whole.

Magdeburg, den 28. Februar 2019

Jan Moritz Joseph

The image shows a musical score in 4/4 time with a key signature of two flats (B-flat and E-flat). The score consists of two staves: a treble clef staff and a bass clef staff. The treble staff features a melodic line with eighth and quarter notes, including slurs and ties. The bass staff provides a harmonic accompaniment with chords and eighth notes. The piece concludes with a final chord in the bass staff.