Fakultät für Elektrotechnik und Informationstechnik
Technische Universität München

TUM

Through Silicon Via Virtualization for Fault-Tolerant Multi-Protocol Interconnect in 3D-ICs

Felix Reinhard Miller

Vollständiger Abdruck der von der Fakultät für Elektrotechnik und Informationstechnik der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktor-Ingenieurs (Dr.-Ing.)

genehmigten Dissertation.

Vorsitzender:
Prof. Dr.-Ing. Wolfgang Kellerer

Prüfende der Dissertation:
1. Prof. Dr. sc.techn. Andreas Herkersdorf
2. Prof. Dr.-Ing. Ulf Schlichtmann

Die Dissertation wurde am 14.09.2017 bei der Technischen Universität München eingereicht und durch die Fakultät für Elektrotechnik und Informationstechnik am 31.07.2018 angenommen.

Abstract

Three dimensional integrated circuits (3D or stacked ICs) are manufactured by vertically stacking multiple conventional dies and subsequently creating high-performance communication links between the chip layers thus formed. Through-silicon-vias (TSVs) are the state-of-the-art method for creating such interconnects. Due to their dimensions (short length and large diameter) TSVs place only a small capacitive load on the driving circuitry and have small electrical resistance. Consequently, they can be operated at high clock frequencies. On the other hand, TSVs take up a large amount of chip area and are prone to manufacturing and run-time faults. This can lead to significant reductions in yield and reliability.

In this work, techniques are presented to reduce the number of TSVs required and to fully exploit the capacity of those deployed. To this end, the suitability of both conventional on-chip bus protocols and modern networks-on-chips (NoCs) for efficient vertical continuation is investigated. To fully exploit TSV bandwidth an approach involving multiplexing and the serialization of multiple virtual links is shown. Protocol adapters enable the transparent continuation of well-established on-chip communication standards. The TSV-Hub is presented, a configurable IP-Core with capabilities to terminate such protocols and implements the approach outlined above in order to significantly reduce TSV count. In addition, the TSV-Hub provides capabilities to improve resilience. Results are presented in form of a case study, where multiple links (based on the AXI protocol) are continued using a single TSV-Hub.

With 3D-ICs the number of functional modules in systems-on-chip (SoCs) will further increase compared to 2D chips. Already in 2D systems with high module counts, packet-switched on-chip networks are gradually replacing conventional buses as interconnect backbone since they are better scalable and more flexible. 3D-ICs additionally offer the possibility of manufacturing highly heterogeneous systems. For heterogeneous systems, it is important that the communication subsystem is specifically tailored to the requirements of the individual. IP-cores (application specific NoCs). Due to the very large design space, creating and optimizing such networks is a complex task and not feasible manually without undue effort. A synthesis algorithm is proposed which considers the physical properties of TSV-based links and bundles connections for a transparent continuation over a shared TSV array. A software tool has been implemented that, on the basis of formally

defined communication requirements, can synthesize an optimized 3D network. Such a network consists of conventional NoC components as well as TSV-Hubs for establishing vertical links. By way of example, the synthesis algorithm is outlined for a mobile communication SoC.

Zusammenfassung

Dreidimensionale integrierte Schaltungen (3D-ICs) werden gefertigt, indem ungehäuste konventionelle integrierte Schaltungen gestapelt und durchsatzstarke Kommunikationsverbindungen zwischen den entstandenen Ebenen hergestellt werden. Bei der Herstellung dieser Verbindungen sind Siliziumdurchkontaktierungen, sog. TSVs (Through Silicon Vias), Mittel der Wahl. Aufgrund ihrer geometrischen Proportionen (geringe Länge bei relativ großem Durchmesser) erzeugen sie nur eine geringe kapazitive Last und weisen nur einen geringen elektrischen Widerstand auf. Dadurch lassen sie sich mit hohen Taktraten betreiben. Andererseits benötigen TSVs jedoch relativ viel Grundfläche und sind anfällig für Fertigungs- und Laufzeitfehler, was zu einem drastischen Abfall der Ausbeute und Zuverlässigkeit führen kann.

In der vorliegenden Arbeit werden Mechanismen vorgestellt, um die benötigte TSV-Anzahl zu reduzieren und die Performanz der vorhandenen TSVs maximal auszureizen. Dabei werden sowohl konventionelle Busprotokolle wie auch modernere paketbasierte Netzwerke (Networks-On-Chip / NoCs) hinsichtlich ihrer Fortsetzbarkeit untersucht. Um die vorhandene TSV-Bandbreite optimal auszunutzen wird ein Ansatz vorgestellt, der durch Multiplexen und Serialisieren mehrere virtuelle Verbindungen über eine gemeinsame TSV-Anordnung bereitstellen kann. Protokollumsetzer zu Standardprotokollen ermöglichen damit eine transparente Fortsetzung etablierter Protokolle. Vorgestellt wird der TSV-Hub, ein konfigurierbarer Funktionsblock (IP-Core), welcher verschiedene verbreitete Verbindungsprotokolle anbinden kann und durch o.g. Maßnahmen die benötigte TSV-Anzahl signifikant reduziert. Ferner stellt der TSV-Hub Mechanismen zur Erhöhung der Fehlertoleranz bereit. Ergebnisse werden anhand einer Fallstudie präsentiert, in welcher ein TSV-Hub zur Fortsetzung von mehreren Bus-Verbindungen, basierend auf dem Protokoll AXI eingesetzt wird.

Mit 3D-ICs wird die Anzahl funktionaler Module in Ein-Chip-Systemen (SoCs) weiter zunehmen. Bereits in 2D-Systemen mit vielen Modulen ersetzen paketbasierte Kommunikationsnetzwerke zunehmend konventionelle Bussysteme, da sie Vorteile hinsichtlich Skalierbarkeit und Flexibilität bieten. Darüber hinaus bietet die 3D-Technologie erweiterte Möglichkeiten zur Herstellung heterogener Systeme. In heterogenen Systemen ist es von Bedeutung, das Netzwerk speziell an die Kommunikationsanforderungen der einzelnen Komponenten anzupassen (applikationsspezifisches Netzwerk). Derartige Netzwerke zu erzeugen und zu optimieren ist

aufgrund des großen Suchraumes eine komplexe Aufgabe und manuell nicht mit vertretbarem Aufwand lösbar. Es wird ein automatisiertes Verfahren vorgestellt, das speziell die Eigenschaften von TSV-basierten Verbindungen berücksichtigt und Verbindungen bündelt, um diese über gemeinsame TSV-Anordnungen fortzusetzen. Mit einer im Rahmen dieser Arbeit entwickelten Software wird auf Grundlage formal definierter Kommunikationsanforderungen ein applikationsspezifisches Netzwerk synthetisiert. Ein solches Netzwerk besteht aus herkömmlichen NoC-Komponenten und aus TSV-Hubs zur Herstellung vertikaler Verbindungen. Exemplarisch wird eine Netzwerksynthese für ein SoC aus dem Bereich der mobilen Kommunikation beschrieben.

Acknowledgements

First of all, I would like to express my sincere gratitude to my doctoral advisor and head of the Chair for Integrated Systems, Prof. Dr. Andreas Herkersdorf, for his guidance and for giving me the opportunity to work in such an interesting field. His scientific input and mentorship have been highly valued and have been essential for the successful completion of my research project and this thesis. I would also like to thank him for the great time at Nanyang Technological University, Singapore. Teaching there has been a fantastic experience for me.

I thank Prof. Dr. Ulf Schlichtmann for acting as co-examiner on the doctoral examination commission of this work.

Special thanks go to Dr. Thomas Wild for supporting me in many respects. Not only did he significantly contribute to my research project but he was also the first researcher from the institute I had the chance to work with in a student job. Back then and still today I was and still am impressed by his immense knowledge and his high-quality feedback.

Furthermore, I like to thank Dr. Daniel Llorente, my former master thesis advisor, for the great work we did together and for opening doors for me at the institute.

I would like to thank Prof. Dr. Walter Stechele for giving me the opportunity to work with him in the field of university teaching. This was a very valuable experience for me, both personally and professionally. I would also like to thank him for his scientific input and advice.

I thank Dr. Daniel Müller-Gritschneder and Dr. Vladimir Todorov from the chair of Electronic Design Automation for the great and fruitful discussions, the successful collaboration and for allowing me to built up on their work.

I am grateful to all partners of the NEEDS project for successful and productive meetings and the great work in general. I especially thank Artur Quiring from the Institute of Microelectronic Systems of the University of Hannover for the successful collaborations.

I thank all students who have contributed to the project, be it in form of bachelor and master theses, student jobs or seminar papers.

It is a pleasure for me to thank all my colleagues at the institute, especially the "Gschnack" group: David May, Gregor Walla, Michael Feilen, Michael Vonbun and Stefan Wallentowitz. Thanks for the fun and the great time we had together.

I like to thank Dr. Manuel Kuehner for his LaTeX support and valuable input in general.

Last but not-least I would like to thank my family: My parents and my sisters for their support. I especially thank my sister Rosi for spending so many hours on proofreading the manuscript.

Contents

List of Figures xiii

List of Tables xvii

1. Introduction 1
- 1.1. Problem Statement . . . 5
- 1.2. Thesis Contributions . . . 6
- 1.3. Thesis Structure . . . 8

2. State of the Art 9
- 2.1. 3D-Integrated Circuits . . . 9
  - 2.1.1. Terminology . . . 12
  - 2.1.2. Driving forces behind 3D-ICs . . . 13
  - 2.1.3. Manufacturing . . . 15
    - 2.1.3.1. TSV Manufacturing . . . 15
    - 2.1.3.2. Wafer thinning and Chip Stacking . . . 16
  - 2.1.4. Interposer based 3D-ICs (2.5D-Integration) . . . 16
  - 2.1.5. Commercially available 3D-ICs . . . 17
- 2.2. TSV-based on-chip interconnect . . . 21
  - 2.2.1. TSV Delay Model . . . 21
  - 2.2.2. TSV Dependability . . . 27
  - 2.2.3. Efficient use of TSVs . . . 31
- 2.3. 3D-Networks-on-Chip . . . 36
  - 2.3.1. NoC Building Blocks and Structure . . . 38
    - 2.3.1.1. Topology . . . 38
    - 2.3.1.2. Routing and Deadlock Avoidance . . . 39
    - 2.3.1.3. Network Interface . . . 40
    - 2.3.1.4. Router Architecture . . . 41
  - 2.3.2. Application Specific Networks-on-Chip . . . 43

3. TSV-Hub for Virtualization of Inter-Layer-Links 49
- 3.1. Efficient Use of TSV Resources . . . 51
  - 3.1.1. Serialization and Multiplexing of Data Streams over TSV Arrays 51

3.1.2. Vertical Continuation of On-Chip Bus Protocols 55
3.1.2.1. Shared Media Buses 56
3.1.2.2. Multiplexed Buses 56
3.1.3. Channel based interconnects 66
3.2. TSV-Hub Architecture and Implementation 70
3.2.1. Concept 70
3.2.1.1. Clock Domains and Clock Distribution 70
3.2.1.2. Fault Tolerance 71
3.2.2. TSV-Hub Protocol Stack 72
3.2.2.1. Physical Layer 72
3.2.2.2. Virtual Link Layer 74
3.2.2.3. Interconnect layer 81
3.3. Experimental Results and Discussion 83
3.3.1. Physical Layer 83
3.3.1.1. Switch Boxes 83
3.3.2. Virtual Link Layer 90
3.3.2.1. Serialization 90
3.3.3. Protocol Layer 90
3.3.3.1. AXI Case Study 90

4. TSV Property Aware 3D-Network-on-Chip Synthesis 97
4.1. NoC Synthesis process 99
4.1.1. Problem formulation 99
4.1.2. Overview 100
4.1.3. Clustering and Router Insertion 101
4.1.4. Generation of NoC-Links 103
4.1.5. Vertical Link Bundling and TSV-Hub-Insertion 104
4.1.5.1. Bandwidth correlation 105
4.1.5.2. Spatial Distance of Link Crossing Points 107
4.1.5.3. Link Affinity 108
4.1.5.4. TSV-Hub Preliminary Placement 109
4.1.5.5. Determining Optimal Number of Vertical Link Bundles 109
4.1.5.6. TSV-Hub Configuration 111
4.1.5.7. Algorithm Summary 112
4.2. Synthesis Output 114
4.3. Experimental Results 116
4.3.1. Complexity and Costs of Vertical Link Bundling 116
4.3.2. Latency Analysis 122
4.3.3. Case Study 127

5. Conclusion and Outlook 131
5.1. Summary of Contributions 131

5.2. Future Work 133
5.3. Outlook 134

A. Appendix 137

A.1. TSV-Array Yield per Area Investment 137
A.2. TSV Sidewall Depletion Area 139
A.3. Markov Modeled on-off Traffic Generation 141

List of Figures

1.1. Evolution of transistor count and process nodes 1
1.2. More Moore and More than Moore 3
1.3. 3D-Packaging vs. 3D-Integration 4

2.1. Through Silicon Via (TSV) Timeline 10
2.2. 2.5D and 3D-IC 11
2.3. Scaling of maximum interconnect length in a 3D-IC 14
2.4. Different approaches for TSV insertion in the fabrication flow 15
2.5. Interposer based FPGA design by Xilinx 18
2.6. Hybrid Memory Cube memory stack 20
2.7. High Bandwidth Memory package arrangement 20
2.8. TSV cross sections 22
2.9. Lumped RC circuit to model TSV behavior 23
2.10. R_{tsv} and C_{tsv} with respect to TSV proportions 27
2.11. System yield with respect to TSV yield 28
2.12. Comparison of methods for fault tolerance in TSV arrays 30
2.13. (De)serializer design of Pasricha et al. 33
2.14. QDI timing 34
2.15. Synchronous serialization methodology of Sun et al. 34
2.16. 2D- and 3D-Mesh NoC topology 38
2.17. Routing schemes in a 2D-mesh Network-on-Chip (NoC) 40
2.18. NoC Router 41
2.19. Scale preserving outline of a 3x3 3D mesh NoC 43
2.20. Communication Graphs 44

3.1. TSV-Hub (Schematic Illustration) 49
3.2. Inter-layer continuation of a 3D-mesh NoC 52
3.3. Pass-through continuation 53
3.4. Serialization with increased TSV clock rate 53
3.5. Multiplexing with increased TSV clock rate 54
3.6. 4-port 8-bit crossbar 55
3.7. 3D Single Master Bus 57
3.8. TSV count for a generic single master bus and four slaves 58

3.9. Different stages of bus multiplexer tree generation for 3D-ICs 60
3.10. Address and data bus dependencies 62
3.11. Bus transaction duration with different pipeline depths 64
3.12. Multi-layer bus with two masters and two slaves (fully connected) . . 65
3.13. Multi-master bus with two masters and 3 slaves 66
3.14. Axi protocol read channels 67
3.15. Axi protocol write channels 67
3.16. Vertical continuation of interconnects with a TSV-Hub 70
3.17. Multiple TSV-Hubs in a GALS-Architecture 71
3.18. Abstraction layers in 3D on-chip-interconnect 73
3.19. Principle of switch boxes (sparing defective TSVs) 74
3.20. Switchbox architecture 75
3.21. Multiple virtual links transported with the TSV-Hub 76
3.22. TSV-hubs in series 77
3.23. Hybrid TDMA and dTDMA Scheduler 79
3.24. Link performance as a function of the number of faulty TSVs 80
3.25. Link performance as a function of the number of faulty TSVs 81
3.26. Switch box size complexity w.r.t. maximum number of faulty TSVs . 84
3.27. Switch box area as a function of k_{max} for an array with 24 TSVs . . . 84
3.28. Switch box area with $k_{max} = 0.5n$ 85
3.29. Yield for switch box TSVs with respect to TSV count ($p_{tsv} = 0.9$) . . . 86
3.30. Effective yield with respect to allowable faulty TSVs 87
3.31. TSV array with 4 *reinforced TSVs* among eight total vertical signals . 88
3.32. Yield for TSV arrays with reinforced TSVs ($p_{tsv} = 0.9$) 88
3.33. Yield per area investment 89
3.34. Area consumption of serializer implementations (64-bit link) 90
3.35. Continuation of two AXI links 91
3.36. Performance with respect to TSV count for upstream and downstream 93
3.37. Performance degradation of TSV continued AXI links w.r.t. burst size 95

4.1. Communication graph example 97
4.2. Synthesis Flow of Todorov et al. 98
4.3. Outcome of using 2D-Synthesis flow for 3D-Design 99
4.5. Intermediate stages of the 3D-NoC synthesis flow 102
4.6. TSV bundling order and layer cross points 105
4.7. Simple flow mapping example 105
4.8. Successive preliminary TSV-Hub placement 108
4.9. Preliminary TSV-Hub placement 109
4.10. Markov modeled IPP 114
4.11. Basic system with 32 PEs forming 16 independent pairs 116
4.12. Synthesis output for different cluster counts 117
4.13. Weighted costs with respect to cluster count 118

4.14. Weighted costs w.r.t number of use cases (1 to 8 clusters) 120
4.15. Weighted costs w.r.t. number of use cases (9 to 16 clusters) 121
4.16. Example of auto-generated CCG an 3D network graph 124
4.17. Average delay 4 layer system with infinite buffer space 125
4.18. Average delay in 4 layer system with 30 words deep queues 125
4.19. Average delay in 4 layer system with 64 cores 126
4.20. CCG, synthesized on-chip network and 3D-continued network 128
4.21. TSV count and area comparison 129

A.1. TSV capacitance with respect to applied voltage [85] 139
A.2. Markov modeled IPP 141

List of Tables

2.1. Comparison of stacking methods 17
2.2. TSV Models in literature 24
2.3. Input parameters for basic TSV model 26
2.4. Comparison of conventional wires and TSVs 31
2.5. Comparison synthesis solutions for Application Specific NoCs 48

3.1. Required AXI channel wires for different bus sizes 67
3.2. VLink Termination Methods 77
3.3. Area consumption of the mesochronous FIFOs 92
3.4. Area consumption of the mesochronous synchronizers 93
3.5. Area consumption of full TSV-Hub 94
3.6. Percentage of original TSV induced area 94

4.1. Case study key parameters 127
4.2. Case study results 127

Acronyms

3D-IC Three dimensional integrated circuit

AC Alternating Current

AMBA Advanced Microcontroller Bus Architecture

APCG Application Characterization Graph

AXI Advanced eXtensible Interface

BEOL Back End of Line

BIST Built In Self Test

CCG Core Communication Graph

CG Core Graph

CMOS Complementary Metal–Oxide–Semiconductor

CMP Chip-Multi-Processor

CoWoS Chip-on-Wafer-on-Substrate

CPU Central Processing Unit

CTE Coefficient of thermal expansion

CTG Communication Task Graph

D2D Die-to-Die

D2W Die-to-Wafer

DAP Duplicate Add Parity

DC Direct Current

DDR Double Data Rate

DMA Direct Memory Access

DOR Dimension Ordered Routing

DP Dynamic Programming

DRAM Dynamic Random Access Memory

dTDMA dynamic Time Division Multiple Access

ECC Error Correcting Code

EDA Electronic Design Automation

EMC Electromagnetic Compatibility

F2B Face-to-Back

F2F Face-to-Face

FEM Finite Element Method

FEOL Front End of Line

FIFO First-In-First-Out

FPGA Field Programmable Gate Array

GALS Globally Asynchronous Locally Synchronous

GMLS Globally Mesochronous Locally Synchronous

GPS Global Positioning System

GPU Graphics Processing Unit

HBM High Bandwidth Memory

HDL Hardware Description Language

HF High Frequency

HMC Hybrid Memory Cube

HoL Head of Line

IC Integrated Circuit

ILL Inter Layer Link

ILP Integer Linear Programming

IPP Interrupted Poisson Process

ITRS International Technology Roadmap for Semiconductors

JEDEC JEDEC Solid State Technology Association (formerly: Joint Electron Device Engineering Council)

KGD Known Good Die

KGD Known Good Die

KoZ Keep-out-Zone

LPDDR Low Power Double Data Rate (a low power double data rate synchronous DRAM standard)

LP Linear Programming

MEMS Microelectromechanical System

MOS Metal–Oxide–Semiconductor

MPSoC Multi-Processor System-on-Chip

MTTF Mean Time To Failure

NEEDS German: Nanoelektronik-Entwurf fuer 3D-Systeme

NI Network Interface

NoC Network-on-Chip

NP nondeterministic polynomial time (a complexity class)

OCP Open Core Protocol

OO Out-of-Order

PCB Printed Circuit Board

PE Processing Element

PoP Package-on-Package

PSO Particle Swarm Optimization

QDI Quasi Delay Insensitive

QDI Quasi-Delay-Insensitive

QoS Quality of Service

RDL Redistribution Layer

RF Radio Frequency

RPA Routing Path Allocation

RTL Register Transfer Level

RTL Register Transfer Level

SCG Switch Communication Graph

SEE Single Event Effect

SiO_2 Silicon dioxide

SiP System in Package

SoC System-on-Chip

TDMA Time Division Multiple Access

TSV Through Silicon Via

UART Universal Asynchronous Receiver Transmitter

UHD Ultra High Definition

VC Virtual Channel

VHDL Very High Speed Integrated Circuit Hardware Description Language

vILL virtual Inter Layer Link

W2W Wafer-to-Wafer

1 Introduction

Integrated Circuits (ICs) have seen an incredible evolution over the past 50 years. This development has revolutionized our everyday life, it has affected how we consume, exchange and store information, how we control machines, how we travel and many other things. Across all domains of science and technology new applications were created, which again were enhanced, and as people adapted to new technologies a continuous demand for new and even more powerful devices was triggered.

The key enabler for this process is the still ongoing down scaling of the minimal feature size in semiconductor devices. This has allowed to integrate more and more transistors into the same area on a single die. The results are more computation resources, higher speed (clock frequencies) and lower power consumption (per transistor). Today, the highest transistor counts are found in the domain of massively parallel computing applications dominated by Graphics Processing Unit (GPU) and Field Programmable Gate Array (FPGA) devices. At the time of writing of this thesis, chips integrating more than 20 billion transistors are available on the market [8]. Besides, also heterogeneous platforms are highly profiting from the shrinking process. Today's high integration density allows to build systems integrating many different and specialized modules on a single chip instead of connecting individual ICs. This is what we call a System-on-Chip (SoC). SoCs power smartphones, smart TVs, navigation systems, Internet routers and many other devices.

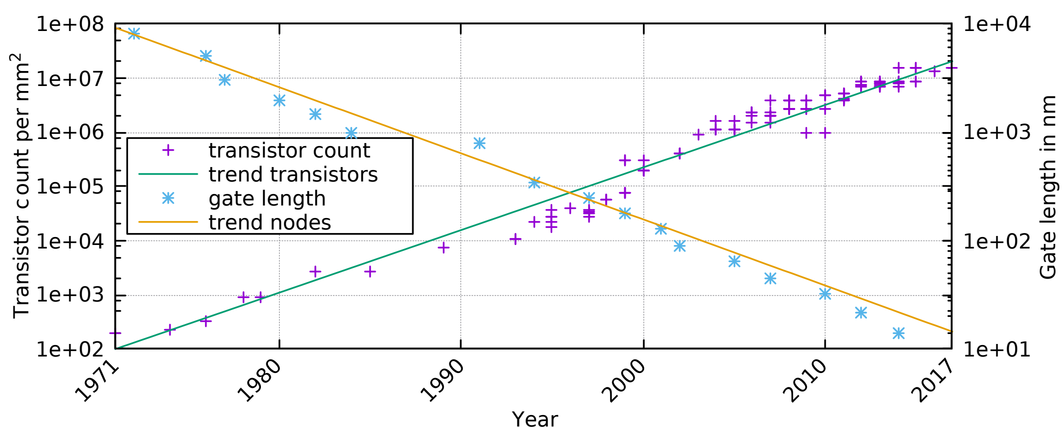


Fig. 1.1.: Evolution of transistor count per area and gate length (process nodes). The data points for the transistor count show the number of high end Intel server and desktop processors (data source [9]). The process node data points show the minimal gate length at the time of their introduction (data source [10]).

The shrinking process was first quantified by Gordon Moore in 1965 [11] as an increase of transistor count per area by a factor of two every year. He later corrected this to a factor of two every two years [12], and shortly later the term *Moore's Law* was born as a name for the process [13]. Looking back from today, we have to correct the number for the doubling to about 20 months, but it still follows very closely an exponential curve.

Fig. 1.1 visualizes the shrinking process and shows the evolution of both transistor count per mm^2 and the process nodes (represented by the gate length) for a time frame from 1971 to 2017. It shows that the process is still ongoing, although its end has often been predicted. However, it will eventually come to its natural end once the scaling reaches the atomic level of silicon [14].

Moore's Law is often illustrated as a continuous line in diagrams or is even characterized by the formula for exponential growth as a continuous process. However, we should be aware that it is in fact a sequence of discrete points of *Process Nodes*, and that a story lies behind each individual point in the graph. Researchers and engineers in the semiconductor industry have worked hard to achieve the individual technological leaps to reach the next node. Process technology has been continuously improved and new processing steps were researched, implemented and improved. In recent years, the shrinking could only be achieved through hard efforts and very sophisticated manufacturing processes, like double patterning lithography [15] or the introduction of *FinFet* transistors to mass production [16]. As a consequence, to this day, only a few companies have been able to keep pace with this development, and worldwide only a few semiconductor fabrication plants (*Fabs*) have the financial resources, the necessary equipment, the required know-how and accordingly well calibrated processes to produce the latest technology. It is clear that this process cannot continue forever, at least regarding mass production, since this would make modern chips too expensive for the normal consumer. However, the demand for more advanced SoCs with even more computational power and more building blocks is still unbroken. Lately, researchers and semiconductor manufacturers have therefore been looking in other directions and evaluating improved integration techniques that not solely relying on downsizing. The term *More than Moore* was coined [17] and can be understood as a process orthogonal to further scaling which is now labeled as *More Moore*.

While transistor scaling is ideal for further improving computation power, More than Moore provides diversification beyond the digital level. It enables the chip to incorperate modules for interacting with the environment like sensors, power electronics, RF components, etc. Fig. 1.2 shows the two dimensions as illustrated in an International Technology Roadmap for Semiconductors (ITRS) White Paper from 2010 [17].

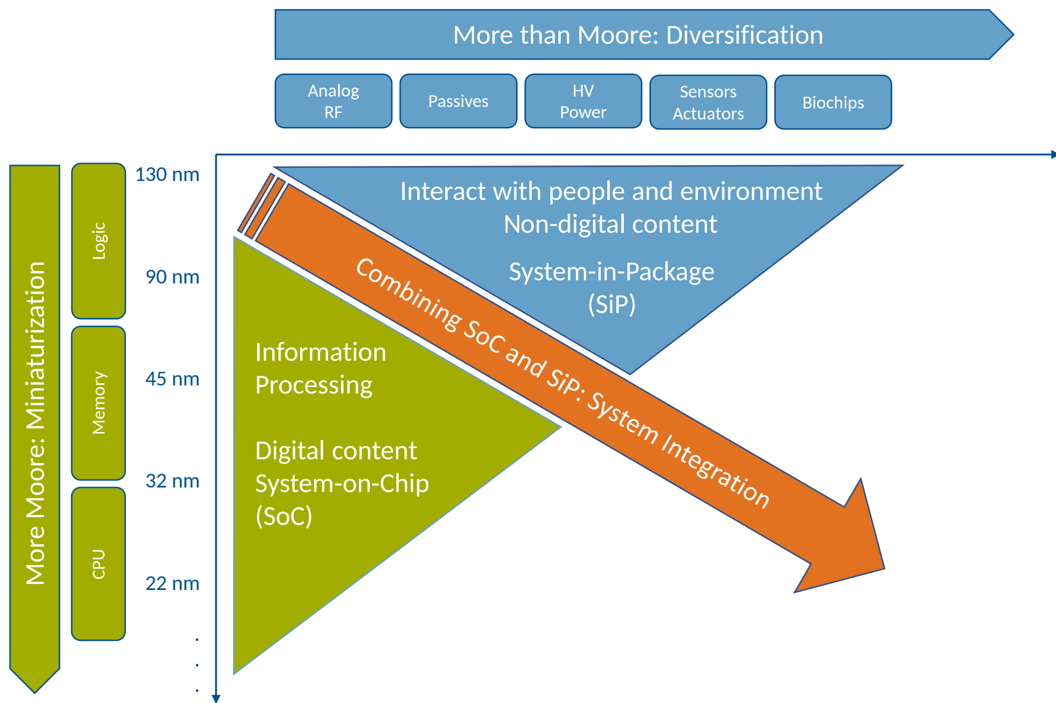
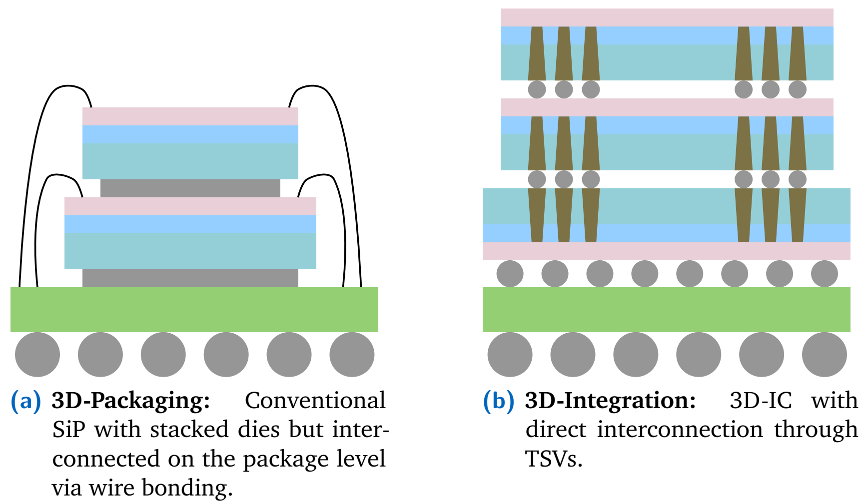


Fig. 1.2.: *More Moore* and *More than Moore* (adapted from [17])

A prominent approach of the *Moore than Moore* category is the System in Package (SiP) technology which has been around for many years now and labels the integration of multiple dies in a single package. Such dies can be aligned next to each other (horizontally) or stacked on top of each other (vertically, Fig. 1.3a) or both. The reasoning for building a SiP is generally due to *Heterogeneous Integration* demands i. e. to combine dies that are manufactured with different integration technologies in order to use the individually best suited process for each one. A famous example is the combination of sensoring or Microelectromechanical System (MEMS) modules with a digital processing platform.

Combining both of the orthogonal evolutions shown in Fig. 1.2 results in a new category of systems that is both SoC and SiP and referred to as *System Integration* [6]. In other words, such are systems designed for the latest integration processes to create highly integrated SoC designs with a high module and transistor count but also to distribute those modules on several dies, possibly created with different integration processes. So-called *3D-ICs* generally fall into this category. Another important differentiator between conventional SiPs and 3D-ICs is the internal interconnection technology. In conventional SiPs wirebonding is the standard, and the individual dies are interconnected on package level. In contrast, in 3D-ICs, high performance direct interconnects are required (Fig. 1.3b) to fully exploit the performance offered by the submodules and to supply sufficient communication bandwidth. While different solutions for realizing such interconnects are investigated, the most promising by far is the so-called TSV technology.



(a) **3D-Packaging:** Conventional SiP with stacked dies but interconnected on the package level via wire bonding.

(b) **3D-Integration:** 3D-IC with direct interconnection through TSVs.

Fig. 1.3.: 3D-Packaging vs. 3D-Integration

At the level of the on-chip communication subsystem a higher SoC complexity poses problems for conventional interconnection designs. Schemes like buses, rings and crossbars do not scale properly for a larger number of nodes. Therefore, the concept and paradigm of NoC was introduced [18], [19], researched and integrated into first consumer products [20], [21]. It is clear that in complex 3D-IC designs, which are generally also complex SoCs, this issue is even more evident, and NoCs will clearly be required in most 3D-ICs and will be the communication subsystem of choice in such designs [22].

It can be concluded that TSV technology will play an important role in physically realizing NoC links in future 3D-SoCs. This work focuses on such 3D-ICs and in particular on the optimization of the communication subsystems for the use of TSV-based interconnects.

1.1 Problem Statement

With TSV-based 3D-ICs, the design of the on-chip interconnect backbone gets more challenging since TSVs differ from conventional wires of planar systems with respect to physical parameters, cost and reliability. Therefore, vertical connections between the individual active layers in 3D-ICs have to be treated differently compared to conventional wires, and conventional design tools as well as IP-Cores for interfacing cannot be directly used without modification and adaption.

Most existing research in the domain of 3D-NoCs considers TSVs in the scope of network topology and routing implications. This approach has clearly its justification, since, as a result of the added degree of freedom, novel topology layouts are possible and new challenges arise with this. However, the physical implications that TSVs impose even on higher levels of abstractions have to be considered as well.

SoCs are usually composed of a multitude of functional building blocks (*IP-Cores*). Such IP-Cores are equipped with standardized interfaces that can connect to on-chip communication subsystems like buses or on-chip networks. Design re-use is crucial in the field of SoC design in order to reduce costs and meet time-to-market requirements. For future 3D-ICs this principle will not change, and designers will still have to rely on existing libraries of building blocks. One challenge is therefore to stretch conventional and popular on-chip interconnection protocols over multiple layers in 3D-ICs. Hence, it is important for the industry to know beforehand which conventional protocols are well suited to be used in a 3D environment and which are not.

The number of available metal layers in ICs has been constantly increased over the past years. This is reflected in modern interconnect protocols which are using many wires per link. Systems like AMBA AXI [23] require hundreds of wires for a single connection. Since wiring resources are plenty in conventional ICs, this is not critical here [24]. Also for 3D-ICs, highly parallel connections have been standardized (e. g. Wide I/O [25]), however those standards are intended for stacked DRAM designs, hence there is usually only a single link present in the system connecting a DRAM die to a CPU die. In a 3D-SoC, however, multiple links can be expected to traverse layer boundaries. If these links exhibit a high wire count, this will result in a total of thousands of TSVs when no countermeasures are taken. This is problematic due to large TSV footprint and reliability concerns. A one to one mapping of such wide connections is next to impossible because it would make the chip both very costly and very likely to fail [26]. Therefore, novel smart mapping mechanisms have to be found that efficiently use TSVs and exploit their physical properties by maintaining the interface compatibility and timing requirements.

NoCs usually use a rather wide interface, and due to their distributed layout they are constructed of many point-to-point connections between the individual routers. For 3D-ICs, this presents the designers with the same challenges as with bus-based systems: a very high expected TSV count.

On top of this, designing an application specific network on chip for a SoC is, due to the large search space and many degrees of freedom, a complex task. Generating such NoCs has to be done in automated fashion and is a research topic on its own. However, for 3D-ICs the step of mapping the on-chip connections on the TSV arrays has to be included in the NoC generation flow in order to enable an efficient and economic use of TSV resources.

Another problem with TSVs is the to this day still rather low reliability. TSVs can turn faulty due to mechanical stress already in the manufacturing process, and there is a significant probability for them to turn defective when the chip is operational. Thus, mechanisms have to be found to cope with such TSV faults.

1.2 Thesis Contributions

The solutions proposed in this thesis aim to leverage the physical properties of TSVs and operate the TSVs on the edge of their physical possibilities as well as aim at making TSV-based connections more resilient.

The three contributions of this work can be summarized as follows:

- The first contribution addresses the vertical continuation of conventional on-chip interconnect protocols in 3D-ICs. The main objective is to reduce the total number of TSVs in order to save costs, reduce footprint and improve yield. This is achieved with multiplexing and serialization techniques in combination with operating TSVs at an increased clock speed. Pure serialization for a single inter-layer link has been proposed before [27], [28]. Link multiplexing and an evaluation when it is advantageous compared to pure serialization has been, to the best of our knowledge, first published by us in [1]. Distributing an on-chip bus structure to several chip-layers solutions has implications for the bus architecture and its timing. These implications are studied and solutions are presented how a bus-matrix can be optimally split on to multiple layers.

- The second contribution is presented in form of an IP-Core design, the so-called TSV-Hub. It is a configurable module that wraps a TSV array which is virtualized by implementing the principles of serialization and multiplexing. Thereby it provides generic virtual links. These links are used by protocol adapters which implement the interfaces of established on-chip communication protocols and thereby allow to transparently continue such links to other chip layers.

The TSV-Hub also provides means to cope with defective TSVs. Existing TSV redundancy techniques [29]–[32] make use of spare TSVs which are initially unused. In contrast, the proposed solution always uses the complete set of TSVs but spares defective ones and adjusts the link bandwidth dynamically. To the best of our knowledge, this approach was first published by us in [2].

- The third contribution is a synthesis flow for 3D application specific NoCs. It extends an existing 2D synthesis algorithm [33]–[35] for the 3D-scenario. The algorithm considers formally defined communication requirements and the spatial distances of IP-Cores in a 3D floorplan. NoC-links traversing layer-boundaries are analyzed and potentially continued over a shared TSV array depending on their timing relations and spatial distances. Compared to other 3D-NoC synthesis algorithms, this is, to the best of our knowledge, the first one that includes multiplexing and serialization techniques and a clock speed-up at TSV level. The new synthesis flow has been developed in collaboration with TUM colleagues from the Chair of Electronic Design Automation (EDA) and was published in [4].

Improvements on the physical level of TSV technology are not within the scope of this thesis. TSVs are, in this work, abstracted on the level of geometry, reliability and basic electrical properties such as capacitance and resistance. The proposed solutions are given as examples on Register Transfer Level (RTL), such that, together with the physical parameters of TSVs and those of targeted standard cell libraries, timing and power simulations can be carried out that show the achieved improvements and underpin presented evaluations.

The physical TSV properties are gathered from scientific publications based on actual measurements or simulations or by means of published calculations models and are referenced accordingly.

Next to the TSV-Hub other NoC building blocks are re-used which are based on state of the art NoC designs, mainly on the LISNOC [36] design.

The work presented in this dissertation is based on the research carried out as part of the NEEDS project, which was funded by the German Federal Ministry of Education and Research, funding label 01M3090.

The ideas presented in this dissertation have been published in the following works: [1],[2],[4],[6],[5].

1.3 Thesis Structure

This thesis is structured as follows:

Chapter 2 gives an overview of the state of the art for the relevant topics and discusses related work to the different subjects discussed in this thesis. This covers 3D integrated circuits in general but with a special focus on Through Silicon Vias (TSV), including surveys on TSV modeling and resilience. Another large part of this chapter concerns the topic of Network-on-Chip (NoC). NoC basics are introduced, and challenges that arise with integrating NoC-based communication structures into 3D-ICs are discussed. The chapter concludes with covering synthesis flows for application specific NoCs and also taking into consideration the impact of a stacked design.

Chapter 3 presents the main contribution of this thesis, i. e.is the TSV-Hub, an IP-Core for efficiently using TSV-based vertical interconnects in 3D-ICs. It starts off with a study on the continuation of conventional interconnect protocols over TSV-based inter-layer connections. Then, the design of the TSV-Hub is introduced and its architecture is described in detail. Finally, experimental results are given in the form of synthesis results, simulations and a case study.

Chapter 4 outlines a TSV property aware 3D-Network-on-Chip synthesis flow for application specific NoCs. The given flow is based on an existing 2D synthesis flow, developed by TUM colleagues from the Chair of Electronic Design Automation (EDA). In collaboration with the authors of this flow, it has been extended to cover the requirements of stacked designs. First, the provision for the impact of TSV-based intra-layer links at an early stage in the flow is described. Later, a novel mapping technique for NoC links onto instantiated TSV arrays is given. The chapter concludes with experimental results for a real-world mobile SoC example.

Chapter 5 concludes this thesis and presents an outlook.

State of the Art

2

This work covers the optimization of TSV based vertical interconnects in 3D-ICs and presents an approach to include the optimizations in an automated NoC synthesis process. Therefore, research related to this work can be found in both, the domain of 3D-ICs and Network-on-Chip

The chapter starts off by an overview of 3D-IC related research and the driving forces behind it and outlines manufacturing methods and challenges. Next, the main aspects of TSVs are covered. This includes the electrical modeling, TSV reliability and throughput optimizations. Finally, on-chip networks and methods for the automatic generation of application specific NoCs are covered. This includes both general and 3D specific aspects.

2.1 3D-Integrated Circuits

The idea of building integrated circuits that extend into the third spatial dimension is nothing particularly new and has been around for many years. Looking at other fields of engineering we see that stacking is an obvious consequence if for instance ground space is limited, distances have to be kept short or just a denser package in general is desired. We stack stories in buildings, containers on ships, build bilevel rail cars and double decker buses. Therefore, also for integrated circuits the idea suggests itself. And in fact it is rather old. The history of silicon through-connections, so-called *TSVs*, dates back to the 1960ies. A patent [37] by IBM from 1967 is usually considered as the invention of TSV technology. However, transistor inventor William Shockley also mentioned TSV-like structures in one of his patents [38] as early as in 1962. Therefore, the question arises as to why 3D-ICs suddenly are becoming relevant more than 50 years after the idea was first described. Again, looking in other domains, there is one factor that often keeps engineers from performing stacking: cost. If there is enough ground available in and around a city, there is no reason to build higher, since this would drive up the costs.

For integrated circuits, reducing the costs per unit is one of the most important design goals, at least for mass production. Apart from a few exceptions (e. g. chips produced in very small volumes for space applications), it is very important to keep the costs as low as possible for a given performance and power requirement. And as long as manufacturers can fulfill these requirements, they will always opt for the technology with the lowest costs. When it comes to 3D Integration, the situation is similar. Until the recently the ongoing downscaling of feature sizes was fast

enough to satisfy the demand for increasing transistor count. Technologies like *System in Package (SiP)* and *Package-on-Package (PoP)* already provided options for heterogeneous integration, and actual die footprint was barely an issue. Looking at the history of integrated circuits, we observe that individual single technological upgrades (e. g. FinFETs) often enabled a new production process with a smaller feature size. These *upgrades*, however, were often not deployed right after their invention but have been kept in the drawer by the semiconductor companies until the time was right, meaning in the most cases, the increased cost for introducing the upgrade was now justifiable because further shrinking could not be achieved by other means. Although 3D-Integration technically does not contribute to die shrinking, it still allows moving forward to house more and diverse components, which are efficiently interconnected in a single package. Current demand for such applications has lead to the first commercially available 3D ICs in recent years.

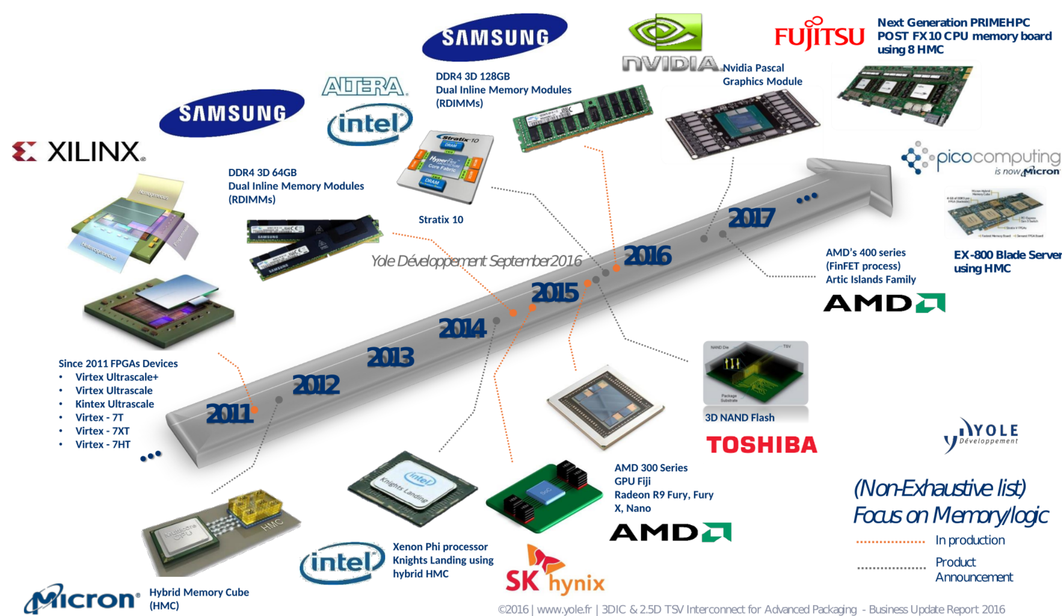


Fig. 2.1.: Timeline for TSVs based commercial 3D-ICs by Yole Développement [39]. Grey lines: announced products. Orange lines: commercially available products.

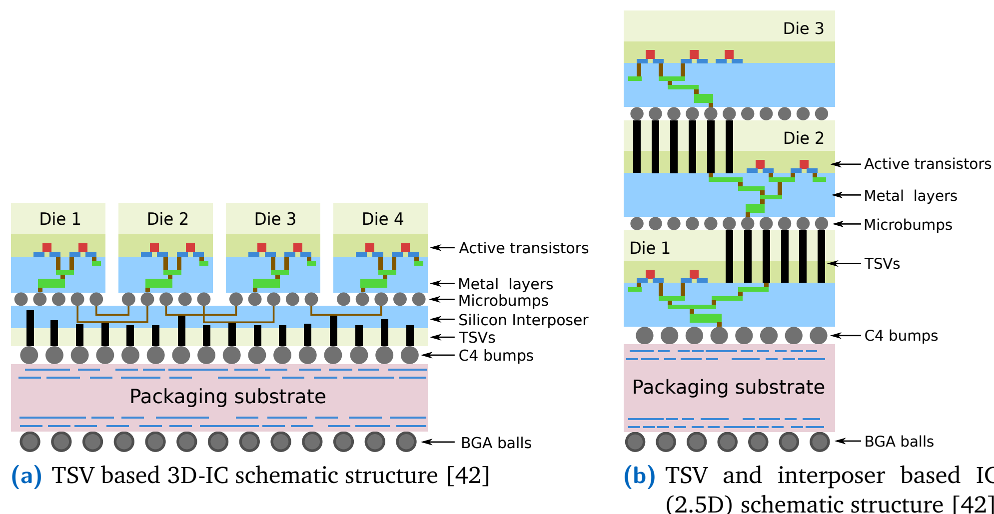
At the time of writing this thesis the first 3D-ICs (with a considerable amount of TSVs) are commercially available [14] in the form of highest-end FPGAs and memory products. Fig. 2.1 shows a time line of the past few years. Note that most listed products are announcements only (grey lines). The only commercial products that are somehow heterogeneous are the Xilinx FPGAs since they incorporate some transceiver chips next to the FPGA fabric [40]. However, chips are manufactured in very low volumes, and the price of a single chip is exorbitantly high (up to multiple tens of thousands of dollars). Nevertheless, they use TSVs in a significant quantity, which means that the first TSV-based applications have broken the cost barrier.

We see that 3D-IC have not yet hit the market in high volumes. The breakthrough is expected in the coming years when costs for the manufacturing process will be

lowered and performance demand will have increased. Nevertheless, there are many Chip prototypes, and the research community is highly active in both industry and academia.

At this point it is worth to discuss what a 3D-IC actually is and what it is not and why the above mentioned chips qualify as 3D-IC and others do not. In fact, it is not absolutely clear and some experts would disagree and not consider most of the currently commercially available 3D-ICs as *real* 3D-ICs since they are built with interposer technology (see Chapter 2.1.4) and do not stack active dies. Also the nomenclature is occasionally confusing, and many different terms have been coined.

Conventional integrated circuits (in the context of 3D-IC often referred to as 2D-circuits or planar circuits) are in fact (as every physical structure) already three dimensional objects. However, their vertical extent is small (several tens of μm) compared to their lateral one (in the range of some mm). Also, they represent already a layered structure. While there is only one layer of active silicon, there are many wiring layers (e. g. up to 13 in Intel's 14 nm process [41]).



(a) TSV based 3D-IC schematic structure [42]

(b) TSV and interposer based IC (2.5D) schematic structure [42]

Fig. 2.2.: Different types of 3D-ICs. With the interposer technology (also *2.5D-Integration*) the dies are arranged horizontally and placed on a silicon interposer. *Real* 3D-ICs form an actual stack of active dies.

A 3D-IC is formed when several of such conventional ICs are stacked on top of each other, forming a layered structure of layered structures. This is why some researchers avoid the term *layer* for the individual levels of a 3D-IC but call them *tier* or *strata* instead. Throughout this work the term *layer* is used to denote an active layer of a 3D-IC. If a metal layer is meant, the term *metal layer* is used.

Still this is nothing new, using a single package for enclosing multiple dies is known for many years as *System in Package (SiP)*, where dies are either stacked on top of each other or arranged next to each other or a combination of both.

However, there is one important difference that qualifies a package to be labeled as *3D-IC*: The chip can communicate internally, meaning there are direct die-to-die connections. The most prominent technique to realize such connections is the TSV technology. Next to TSVs there are also other concepts which rely on inductive or capacitive coupling [43] or even make use of optical connections [44]. Compared to TSV technology, however, such solutions are more exotic and have not been commercialized yet. They face their own challenges like Electromagnetic Compatibility (EMC) issues and also space consuming and complex transceiver circuitry. In general, TSV technology offers the highest integration density of all direct die-to-die communication solutions [42].

This work is focused on TSV-based die-to-die connections. Therefore, a 3D-IC in the context of this work fulfills the following requirements:

- The chip is built by stacking multiple dies
- TSVs are used for vertical interconnections
- There is a significant quantity of TSVs

However, there are other definitions, therefore it is worth to look at the 3D-IC terminology in general, which is done in the next section.

2.1.1 Terminology

As mentioned above, the terminology regarding 3D-ICs can be confusing, and different terms are used in both research and business. For example it is comprehensible that a chip manufacturing company advertises its interposer based product as a *3D-IC* for marketing reasons whereas a researcher would refer to the technology as *2.5D-Integration* to distinguish it from the *real* 3D-Integration with multiple active layers.

The following list shows some terms that have become popular in recent years:

3D-IC Usually refers to an IC built of multiple stacked dies that communicate internally (e.g. by using TSVs or wireless interconnects).

Stacked IC The same as above but some researchers dislike the term *3D-IC* since the third dimension is not a full degree of freedom (transistors can be placed on one of the available layers but not at any arbitrary z-position) and currently only small integer numbers of active layers (Tiers) are realistic (due to manufacturing challenges and heat dissipation issues).

Monolithic 3D-IC This term refers to a technology that does not rely on stacking dies or wafers to create a chip with a vertical extent of its active area, but integrates all active components on a single wafer.

2.5D-IC This is a term often used in research for setups which are TSV based but consist only of two layers: a signal redistribution layer and one active layer. The active layer, however, can comprise multiple dies which are arranged horizontally.

2.75D-IC This is a term coined by Reisinger [45] used in the same way as *Stacked IC* to avoid the term *3D-IC* and emphasize the fact that the third dimension is not a full degree of freedom [45].

Another possibility of confusion was added since FinFet technology has been rolled-out in commercial products. The technology is sometimes marketed as *3D-Transistor technology*[46]. However, this is not related to chip stacking at all.

2.1.2 Driving forces behind 3D-ICs

There is not a single motivator behind 3D-ICs but multiple ones. The following list gives the main driving forces:

- Footprint
- Heterogeneous Integration
- Performance and Power
- Yield and Cost

The most obvious of these drivers, but clearly not the dominant one, is reducing footprint. Naturally the 2D ground area is reduced if the same active silicon area is split into several vertically stacked layers. However, even for conventional planar ICs the actual area used by the die in the package is usually small compared to the full area consumed by the package. Even for applications where PCB space is limited (e. g. mobile phones) this is not critical, and for applications where chips with large sized dies are used, the consumed overall area of the package is mostly uncritical (e. g. high-end FPGAs or high-end graphic chips in work stations).

Heterogeneous integration to a certain extent overlaps with the footprint motivator since with heterogeneous integration it is possible to provide a single package containing multiple different dies manufactured with different processes. However, this is also possible with a SiP. The differentiator here is that a 3D-IC provides, on top of the reduced footprint, also a high direct interconnect throughput compared to the situation where the two dies are interconnected on package level via wire bonding or are even placed in different packages on the PCB.

Interconnect length has a significant impact on both power and performance. Shorter wires lead to less resistance and lower capacitive load, hence smaller delays. The maximum wire length of global wires can be drastically reduced with 3D-Integration. Fig. 2.3 shows an example which has been adapted from [47]. The maximum (meaningful) global wire length in a die is the distance from one corner to the one

diagonally opposed. For a planar chip with a silicon area of A this gives a maximum global wire length of $L_{max} = 2\sqrt{A}$. Note that the wire itself cannot be diagonally oriented for most standard cell processes, and its path is always aligned parallel to the X or Y direction, hence the total length of two times the side length. When the silicon area A is distributed onto n layers by using 3D-Integration, the per layer area is reduced to $A_l = A/n$ and thus the side length is reduced to $l = \sqrt{A/n}$. Therefore, it can be concluded that the maximum length of global interconnects roughly scales with a factor of $\sqrt{(n)}$ when silicon area is distributed to multiple layers [47] and overall silicon area is kept constant.

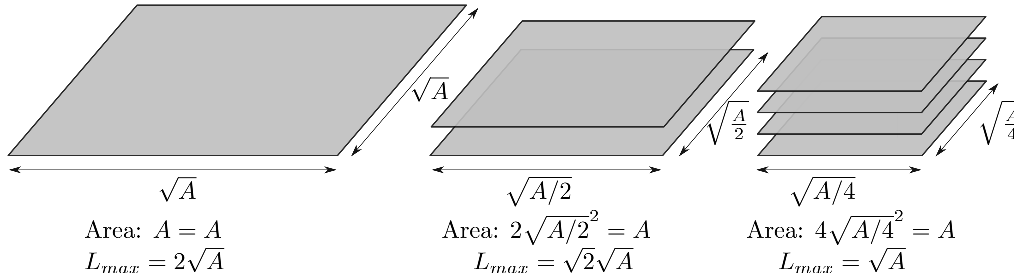


Fig. 2.3.: Scaling of the maximum interconnect length in a 3D-IC with the number of layers while keeping overall silicon area constant (adapted from [47]).

Another strong force demanding vertical integration are the cost and yield factors, which have to be seen intertwined. At first, it seems counterintuitive that with 3D-Integration yield can be increased since TSVs themselves have a rather low yield and in principal reduce the overall system yield. However, with 3D-Integration it is possible to split a large die into smaller portions. The probability that a defect is found on a die depends on its size. When splitting a chip into n dies the probability that a defect is found on any of the n smaller dies is still the same. However, each die can be tested separately, and it can be ensured that only functional dies are interconnected by a Known Good Die (KGD) approach. To this end, the dies do not necessarily have to be vertically stacked. They could also be arranged next to each other horizontally. Therefore, the same yield improvement can also be achieved by building a SiP. However, the requirements on interconnect performance might be too high for wire bonded connections on the package level but not that high that TSV-based connections are needed. For such systems an interposer based integration technology (2.5D-Integration) is the method of choice.

2.1.3 Manufacturing

The manufacturing process of 3D-ICs requires additional steps compared to the one for classical planar ICs. For TSV-based 3D-ICs at least the following three major steps have to be added to the process:

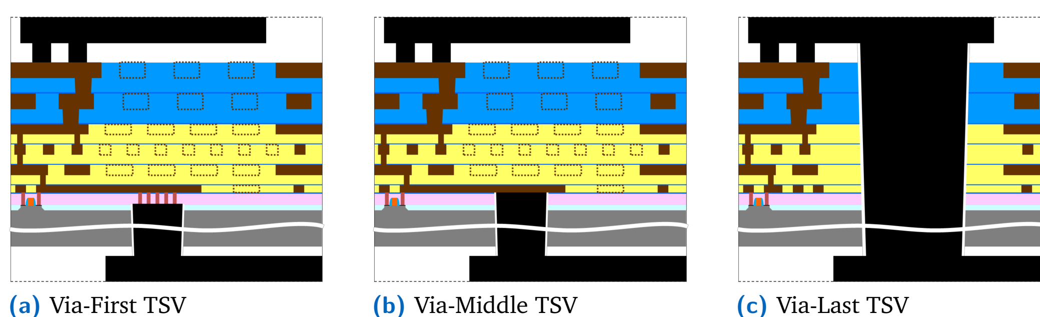
- TSV manufacturing (etching and filling)
- Wafer thinning
- Die Stacking and bonding

2.1.3.1 TSV Manufacturing

TSVs are vias that punch through the bulk silicon. To create such vias, holes have to be etched and then these holes have to be filled with conducting material (usually copper or tungsten). Different etching processes are available [48].

A major impact on TSV geometry and interconnection is governed by the point in time, relative to the other (2D) process steps, when TSV manufacturing is done. The three prominent approaches for manufacturing TSV based 3D-ICs are the following [49]:

- **Via First TSVs** are manufactured before Front End of Line (FEOL) processing (Fig. 2.4a), i. e. TSVs are fabricated even before transistors are patterned.
- **Via Middle TSVs** are manufactured in between FEOL processing and Back End of Line (BEOL) processing (Fig. 2.4b), i. e. TSVs are fabricated after transistors but before metalization.
- **Via Last TSVs** are manufactured after BEOL processing (Fig. 2.4c), i. e. TSVs are fabricated after metalization and therefore have to punch through all the metal layers additionally.



(a) Via-First TSV (b) Via-Middle TSV (c) Via-Last TSV

Fig. 2.4.: Different approaches for TSV insertion in the fabrication flow [50]

In general, short TSVs are desirable since, due to a limited geometric aspect ratio, short TSVs also achieve small diameters. The shortest TSVs can be created with the Via-First approach. However, in this flow, the TSVs have to withstand the high temperatures of up to 1000 °C during transistor formation of the FEOL process [42].

For this reason, copper or tungsten cannot be used as filling material due to their high Coefficients of thermal expansion (CTEs). The only choice remaining, is polysilicon which in turn has a much higher resistance and limits the TSV performance. Therefore, the Via-First approach is not widely used [42]. With the Via-Middle approach the TSVs only have to withstand a temperature of roughly $400\,^{\circ}\mathrm{C}$ encountered during the metalization process. This is feasible with the CTE of tungsten but still problematic with the one of copper [42]. With the Via-Last approach no high temperatures are encountered anymore after TSV fabrication. Therefore, this process is ideal for materials with high CTEs like copper. However, here the TSVs are longer since they also traverse the metalization stack.

2.1.3.2 Wafer thinning and Chip Stacking

When excluding the monolithic approach, the process to manufacture 3D-ICs always involves some sort of stacking. However, there are different possible points in time during the manufacturing process when the stacking can be performed. The main difference is wether stacking is performed before or after wafer dicing.

The available methods are [49]:

- **Die-to-Die (D2D) Stacking**: single dies are stacked on single dies
- **Die-to-Wafer (D2W) Stacking**: single dies are stacked on undiced wafers
- **Wafer-to-Wafer (W2W) Stacking**: undiced wafers are stacked on undiced wafers

The stacking method has implications for several aspects like testing, wafer handling, alignment and production throughput. Tab. 2.1 shows a comparison. A KGD approach is only possible with D2D and D2W flows. With D2D only functional dies are stacked, and with D2W the defective chips in the waver can be spared if wafer level testing is available. When stacking undiced wafers in a W2W flow, defective chips might be stacked on functional ones or vice versa, leading to a reduced yield. Also for the W2W flow, the handling of thinned wafers (with a height of only a few tens of $\mu\mathrm{m}$) is needed, which is delicate due to possible warpage [51]. However, W2W stacking offers the highest production throughput.

2.1.4 Interposer based 3D-ICs (2.5D-Integration)

Two of the four drivers listed in Chapter 2.1.2 can be addressed without actually performing stacking of active layers, namely heterogeneous integration and yield and cost improvement. To this end, so-called *interposer based* systems (also 2.5D-Integration) can be used. Here, multiple active dies are arranged next to each other (horizontally) and are vertically stacked on a larger interposer die using a Face-to-Face (F2F) approach (see Fig. 2.2a). With such a F2F orientation no TSVs have to be placed in the active dies but only in the interposer. This removes all impact of TSV

Tab. 2.1.: Comparison of stacking methods

	D2D	D2W	W2W
Production throughput	low	medium	high
Test approach	KGD possible die level testing	KGD possible die + wafer level testing	KGD not possible wafer level testing
Alignment results	critical	critical	good
Wafer handling	manageable	manageable	delicate

manufacturing for active silicon and vice versa. Also, no silicon real estate is lost due to TSV footprint or Keep-out-Zones (KoZs). Looking at the list of drivers again, the issues not always addressed with 2.5D designs are footprint and performance. When splitting an otherwise large die into smaller pieces to create a 2.5D design, footprint cannot be reduced due to the horizontal arrangement. Performance is also not increased in this case, since wire length is not reduced. Every inter-die signal has to travel through the interposer and therefore, compared to a large planar chip, the average length of global wires is even increased. However, such a large planar chip might not be feasible due to low yield and resulting high costs. Therefore, the interposer design rather competes with the SiP approach or even a multi-package alternative interconnected on PCB level. Compared to such alternatives, a TSV-based interposer design clearly offers higher performance.

With heterogeneous 2.5D designs the remaining aspects (footprint and performance) are also addressed since here, the reference is a multi-package configuration on a PCB (or a SiP) and not a single larger die.

Currently silicon is used as bulk material for the interposer even though there are no active components realized on this layer. However, the process flow is well mastered and therefore the obvious choice for integrating a Redistribution Layer (RDL). In the future, glass might be used as an alternative to save costs [52].

2.1.5 Commercially available 3D-ICs

At the time of this writing only few commercial TSV-based products had reached the market. This is assumed to be due to the fact that TSV production costs are still high and yield is assumed to be low. However, the semiconductor foundries usually do not share much information on actual quantities regarding these figures.

In Fig. 2.1 a timeline of already available products is given. The list of available products is dominated by two applications:

- Interposer based high-end FPGAs
- Stacked DRAM

The objective of the interposer based systems is (as already discussed in 2.1.4) mainly to improve the overall yield for chips with large silicon area. Therefore, it is no surprise that the only commercial representatives are high-end FPGAs with a very large die area, namely Xilinx' UltraScale FPGAs.

Xilinx 3D FPGAs are available since 2011 and use TSMC's Chip-on-Wafer-on-Substrate (CoWoS) platform [53], [54]. The first representative within this family was the Virtex-7 2000T FPGA made from four identical FPGA dies at the 28 nm node and a silicon interposer at the 68 nm node. Its structure is shown in Fig. 2.5. Roughly 10000 inter-die connections are used [55]. I/O connections are routed via TSVs to the package substrate. The TSVs are placed with a pitch of 180 μm [56]. In 2012, Xilinx added the *Virtex-7 HT* family to their portfolio with some of its members including dedicated serial transceiver dies. This was advertised as the first *Heterogeneous 3D-IC* [57].

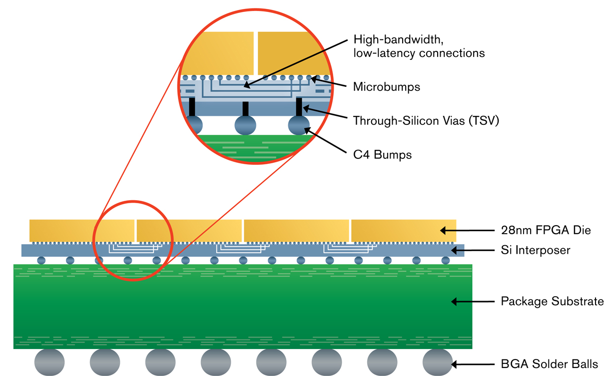


Fig. 2.5.: Interposer based FPGA design by Xilinx [55]

The remaining bulk of commercially available 3D-ICs is formed by stacked DRAM systems. In the realm of 3D DRAM different standards targeting three different application domains have been published: Mobile Memory, high-end server and datacenter memory and graphics memory. However, only for the latter two products can be purchased.

Currently products supporting the following memory standards are available:

DDR4 is a conventional SDRAM standard, implemented also in a few 3D products. From an interface standpoint, such modules are not different compared to conventional DDR4 modules. The protocol as it is defined in the DDR4 standard

[58] is used for communication with the memory controller. The motivation behind such products is the denser package possible with stacking multiple DRAM dies. The technology was introduced by Samsung in 2014 with a 64 GB module. In 2014, a 128 GB module was announced by SK Hynix [59]. In 2015, Samsung also added a 128 GB module [60]. No information is publicly available regarding the underlying TSV technology like geometry, pitch and clocking.

Hybrid Memory Cube (HMC) HMC is a high bandwidth memory standard for stacked DRAM. The first version (1.0) of the HMC standard was published in 2013 [61], later versions 2.0 (2014, [62]) and 2.1 (2015, [63]) were released. The standard is maintained by the HMC Consortium, which is backed by major semiconductor and memory vendors including Micron, ARM, Samsung, Altera, Xilinx. Fig. 2.6 shows the basic setup. An HMC chip stack consists of the controller logic, which always takes up the base layer. On top of the controller up to eight DRAM layers are stacked vertically and interconnected by TSVs. The individual DRAM layers are divided into 16 or 32 partitions (depending on the standard version). A so-called *Vault* is formed by interconnecting all partitions which are located at the same XY position but in different layers. Such a Vault is then controlled by the respective *Vault Controller* on the base layer. TSVs are fabricated with a pitch of $60\,\mu\text{m}$ in the Micron module [64]. HMC is incompatible with existing DRAM memory standards like DDRx or LPDDR. A chip stack (also called *cube*) is interfaced with up to 4 links, each link comprising up to 16 differential serial lanes with a bandwidth of up to 30 Gbit/s per lane. Hence an aggregated (full-duplex) bandwidth of up to 480 GB/s is possible [63]. A packet based protocol is used on top of the serial links. With this protocol multiple cubes can be arranged in different topologies (described in the standard) and connected to one ore more processing cores. HMC is an off-chip memory system, and so far, the Micron cube is the only available memory module incorporating the standard on the memory side. However, multiple vendors have equipped their chips with the host interface in order to connect to an HMC.

High Bandwidth Memory (HBM) is another standard for stacked DRAM, but which is targeting graphics applications. In contrast to HMC, it is not maintained by an industry consortium but by the independent JEDEC organization, which is also responsible for the conventional DDR1-5 standards. The standard was first published in 2013 and revised in 2015 [65]. The basic setup of an HBM system is shown in Fig. 2.7. In contrast to HMC the memory stack is included in a package together with the GPU. Both GPU and memory stacked are placed next to each other on a silicon interposer. On the host interface level, HBM implements a wide parallel interface consisting of 8 independent channels. An aggregated bandwidth (full-duplex) of 256 GB/s is provided [66]. HBM

is currently available in the AMD Fiji GPU [67] and the Nvidia Tesla P100 GPU [68].

The different stacked memory standards do not directly compete since they serve different target applications (server/supercomputer applications for HMC and GPU applications for HBM) [66]. Also a major difference between the two is that HMC uses a discrete package for the memory stack, whereas for HBM the memory is integrated in the SoC. With *Wide I/O* [69] another standard for stacked DRAM is available, which is, however, targeting mobile applications. No products have hit the market yet, though, since with advanced LPDDR standards the same bandwidth could be achieved at a much lower cost [14].

Note that also a multitude of non-TSV stacked memory solutions exist. Those are out of the scope of this work, however.

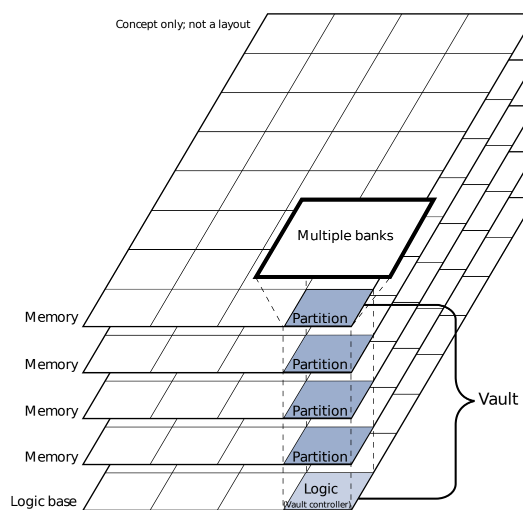


Fig. 2.6.: HMC die stack scheme from the HMC specification rev. 2.0 [62]. The base layer houses the control logic, and up to 8 DRAM layers are stacked on top of it.

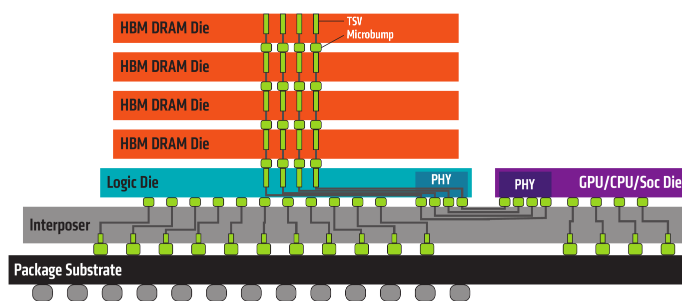


Fig. 2.7.: HBM setup [67]. The DRAM stack is arranged next to a GPU on a silicon interposer. In the RAM stack the base layer houses the control logic. 4 or 8 DRAM layers are stacked on top of it (depending on the standard version).

2.2 TSV-based on-chip interconnect

The by far most promising interconnection technology for 3D-ICs is the TSV-based one. In both, the research community and industry, the vast majority of publications in the realm of 3D-ICs covers TSV-based systems.

TSVs offer high throughput but are costly with regards to the following aspects:

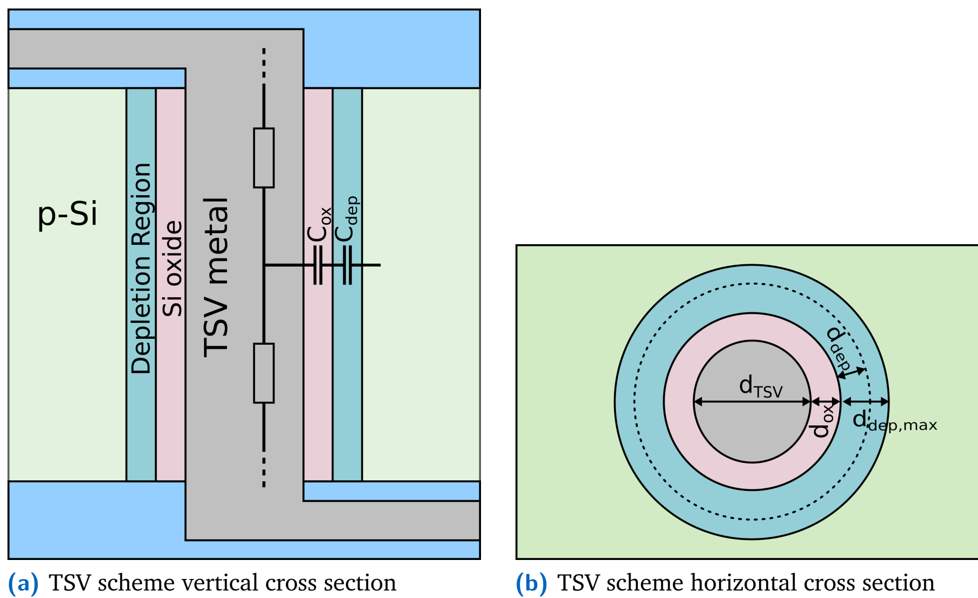
- **Area consumption**: Despite being very short (compared to intra-layer interconnects) TSVs are wide (in the range of several (tens of μm) and thus consume a high amount of chip area. Additionally, most TSVs require surrounding Keep-out-Zones that further use up valuable silicon area.
- **Yield drop**: Even though the yield of a single TSV is high and close to 100%, when many TSVs are inserted in a system, the individual yield factors have to be multiplied, and the overall yield drops exponentially with the number of used TSVs.
- **Manufacturing**: Additional process steps have to be included when manufacturing TSVs (e.g. etching, filling and bonding).

The last one of the above points is mostly independent from the number of TSVs used in the system. The area consumption and yield drop, however, correlate with the number of used TSVs and are the reason why TSVs are a valuable resource that needs to be budgeted efficiently.

2.2.1 TSV Delay Model

One objective of this work is to provide means to fully exploit TSV performance. To this end, relevant electrical parameters are studied in order to characterize TSVs from a more abstract viewpoint. Also, such models are needed to extract realistic parameter values which are then used in simulations and calculations on higher abstraction layers for calculating delay and power figures. To simulate the electric behavior with maximized accuracy so-called *field solvers* are required. Field solvers are software programs applying Finite Element Methods (FEMs) on user-entered geometric structures. However, entering the structure is a time-intensive task. This is why more compact models, so-called *lumped models*, are desired, which characterize TSV behavior by using an equivalent circuit network with a manageable number of electrical components like inductances, capacitances and resistances. The actual quantities for the model are then derived using closed-form expression based on geometrical and material parameters. Fig. 2.9 shows such a lumped model for characterizing a single TSV. Finally, a model can be verified for a set of input parameters by comparing the results to those of field solvers or measurement data with corresponding input data.

Several of such models and corresponding closed-form expressions have been published (see Tab. 2.2) in the recent years, with different levels of complexity, accuracy and set of input parameters and output values. Common to all of these models is that they consider at least in parts the geometric dimensions of a single TSV. Therefore, at first we have to understand that TSVs are in fact cylindrical structures, despite being often depicted with rectangle or cuboid like shapes in illustrations (Fig. 2.8 shows cross section views). Another property of TSV structures is that they form a Metal–Oxide–Semiconductor (MOS) setup with the surrounding bulk silicon and the Silicon dioxide (SiO_2) as an insulator. Therefore, TSVs show behavior similar to MOS transistors.



(a) TSV scheme vertical cross section

(b) TSV scheme horizontal cross section

Fig. 2.8.: TSV cross sections. The TSV forms a cylindrical MOS structure within the bulk silicon.

The first electrical model for TSV behavior was presented by Alam et al. in 2007 [70]. It does not yet use the term TSV but *Through-Si Via*. It is a very basic model relying solely on the standard formulas to calculate the resistance and capacitance of a cylindrical structure, also TSV inductance is neglected.

With respect to modeling the **resistance** we see two more steps of refinement. One is representing the geometric proportions more closely in the model. TSVs are strictly speaking not of pure cylindrical shape but resemble a tapered cylinder as a result of the etching process. This is accounted for in the model of [71] with an additional parameter for the *slope angle*. The other possible enhancement is considering the *Skin Effect*, which is obviously only relevant for higher frequencies. The impacts of the Skin Effect are included in [71], [72] and [73] by providing an additional formula for AC resistance.

With respect to modeling the **capacitance**, the models can be enhanced by including the depletion area that reaches into the bulk silicon depending on the applied voltage level. The initial model of [70] (and also [74]) only model the C_{ox} capacitance (see Fig. 2.8a) between the TSV metal and the silicon substrate and neglect the depletion region formed depending on the voltage level. More comprehensive models, where the depletion region is considered via an additional capacitance (C_{dep} in Fig. 2.8), have been developed from 2010 on [72], [74], [75].

Inductance is not considered in all models, [70] and [74] omit it. Where it is considered, it is based on the self inductance for a cylindrical structure (*Rosa Expressions* [76]). In [75] the basic form is used. In [73] two fitting parameters are added to account for low aspect ratios and high frequencies. In [71], which takes the tapered shape into account, the basic Rosa expression for the cylindrical structure is extended by a parameter for the slope angle.

Additional structures are taken into account in [77] and [72] by modeling landing bumps and the impact of RDL respectively.

Most of the newer models take neighboring TSVs into account [78], focus on crosstalk modeling [79] or are full matrix models, which consider all coupling between any TSV combination [42], [80].

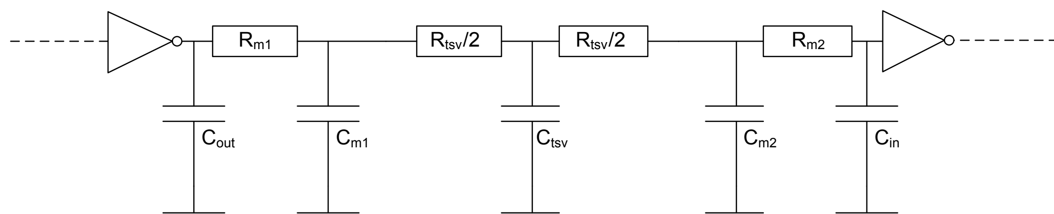


Fig. 2.9.: Lumped RC circuit to model TSV behavior [75]

Below, the basic equations, forming the foundation of most of the models above, are given. Assuming a cylindrical TSV shape but, considering the effect of the depletion region, gives a fairly accurate model [75]. The authors of [75] also compare their models with actual measurements and results from field solvers. The errors for the capacitance and resistance are always below 5% when compared to the results from field solver simulations. Solely the inductance shows a higher deviation (up to 18%). However, the authors also state that the inductance only becomes predominant at clock frequencies higher than $3\,\mathrm{GHz}$. For lower frequencies, they suggest to neglect the impact of the indcatance and use a lumped model as the one shown in Fig. 2.9.

With such a lumped model, the propagation delay and power consumption can be estimated. A conservative estimate on the propagation delay of a lumped circuit can be computed with the *Elmore Delay* [82], [83].

Tab. 2.2.: TSV Models in literature

	Reference	Year	Parameters			
			Resistance	Capacitance[1]	Inductance	additional effects
[70]	Alam et al.	2007	TSV geometry, metal resistivity	TSV geometry, oxide width, permittivities	not considered	none
[74]	Savidis et al.	2009	TSV geometry, metal resistivity	TSV geometry, oxide width, permittivities	not considered	none
[75]	Katti et al.	2010	TSV geometry, metal resistivity	TSV geometry, oxide width, permittivities, doping level, voltage[2]	TSV geometry, permeability	none
[73]	Savidis et al.	2010	TSV geometry, metal resistivity, frequency	TSV geometry, oxide width, permittivities, doping level, voltage[2]	TSV geometry, permeability, frequency[3]	skin effect, capacitance to ground plane
[77]	Kim et al.	2010	TSV geometry, metal resistivity, frequency, TSV pitch, bump geometry	TSV geometry, oxide width, permittivities	TSV geometry, permeability	skin effect, landing bumps, neighbor TSVs
[71]	Liang et al.	2011	TSV geometry, metal resistivity, slope angle	not considered	TSV geometry, permeability, slope angle	tapered shape
[72]	Kim et al.	2011	TSV geometry, metal resistivity, frequency, TSV pitch, bump geometry, RDL[4] geometry	TSV geometry, oxide width, permittivities, RDL[4] geometry	TSV geometry, permeability	skin effect, landing bumps, neighbor TSVs, RDL[4] impact
[79]	Engin et al.	2013	not covered ref. to literature	TSV geometry, oxide width, permittivities	not covered ref. to literature	focus on crosstalk modeling
[78]	Ndip et al.	2014	TSV geometry, metal resistivity, frequency, depletion area, TSV pitch	TSV geometry, oxide width, permittivities, frequency	TSV geometry, permeability, frequency, TSV pitch	skin effect[5], landing bumps, neighborTSVs
[80]	Lim et al.	2015	not covered	TSV geometry, oxide width, permittivities, depletion area, distance to all TSVs	TSV geometry, permeability, dist. to all TSVs	matrix model[6]
[42]	Kaushik et al.	2016	TSV geometry, metal resistivity, frequency, dist. to allTSVs, bump geometry	TSV geometry, oxide width, permittivities, dist. to allTSVs	TSV geometry, permeability, distance to other TSVs	skin effect, landing bumps, matrix model[6]
[81]	Kim et al.	2017	not covered	TSV geometry, oxide width, permittivities, bias voltage, doping level	not covered	models hysteresis

[1] Some models also include a conductance expression along with capacitance. Conductance is not covered in this table.
[2] Via depletion area
[3] By means of a second formula for higher frequencies.
[4] Redistribution Layer
[5] Impact of TSV pitch on skin effect is considered.
[6] Considers all distances

In [75] the Elmore Delay reads as follows:

$$
\begin{aligned}
t_p = {} & 0.69 R_{dr} C_{ext} \\
& + 0.69 \left(R_{dr} + R_{m1} \right) C_{tsv} \\
& + 0.69 \left(R_{dr} + R_{m1} + 0.5 R_{tsv} \right) C_{m1} \\
& + 0.69 \left(R_{dr} + R_{m1} + R_{tsv} + R_{m2} \right) \left(C_{m2} + C_{int} \right)
\end{aligned}
\tag{2.1}
$$

In (2.1) C_{out} and C_{in} denote the output and input capacitance of the inverters, R_{dr} is the driving resistance of the inverter and R_{m1}, and R_{m2} denote the resistance of the conventional metal layer wires (on die 1 and die 2).

For retrieving the TSV related quantities, namely the TSV capacitance C_{tsv} and the resistance R_{tsv} the TSV model is derived as follows:

The standard formula for a cylindrical shaped ohmic conductor yields for the overall TSV resistance R_{tsv}:

$$
R_{tsv} = \frac{4 \rho_m h_{tsv}}{\pi {d_{tsv}}^2} \tag{2.2}
$$

with the geometrical parameters of the TSV (h_{tsv}: height; d_{tsv}: diameter) and the resistivity of the metal fill ρ_m.

Also, for the TSV sidewall capacitance C_{ox} the standard formula for cylindrical structures can be applied:

$$
C_{ox} = \frac{2 \pi \varepsilon_r \varepsilon_0 h_{tsv}}{\ln \frac{d_{ox}}{d_{tsv}}} \tag{2.3}
$$

With the permittivity for empty space ε_0 and SiO_2 ε_r.

However, a non negligible impact is induced by the MOS structure that is formed by the TSV-SiO_2-silicon structure next to the TSV sidewalls. Assuming a p-doped substrate and positive voltage being applied to the TSV, the MOS-structure will be operated in the depletion mode, and a depletion region will form at the sidewalls, which affects (reduces) TSV capacitance.

This depletion region can be seen as connected in series with the oxide capacitance (see also Fig. 2.8). Hence the overall TSV capacitance results in:

$$
C_{tsv} = \frac{C_{ox} C_{dep}}{C_{ox} + C_{dep}} \tag{2.4}
$$

To calculate the depletion capacitance C_{dep} in (2.4), again the formula for cyilindrical capacitance is used as already in (2.3), just with different radii and the permittivity for silicon:

$$
C_{dep} = \frac{2 \pi \varepsilon_{si} h_{tsv}}{\ln \frac{0.5 d_{tsv} + d_{ox} + d_{dep}}{0.5 d_{tsv} + d_{ox}}} \tag{2.5}
$$

The unknown in (2.5) now is the width of the depletion zone d_{dep}. This parameter depends (among others) on the voltage applied to the TSV.

The C-V behavior of a MOS structure exhibits a minimum capacitance C_{min} for the AC realm. When moving into the inversion region at the Threshold Voltage V_{th}, there is no further increase of the overall capacitance with further increasing TSV voltage [84]. At this point, the depletion area has reached its maximum width $d_{dep,max}$. Considering that for the low doping levels of the bulk substrate, V_{th} is negative even for a p-type substrate [85], the width of the depletion layer is always at its maximum width for the normal operating conditions of a TSV. Consequently the capacitance becomes independent again from the applied TSV voltage. However, there is no analytic expression for calculating $d_{dep,max}$ for the cylindrical case, and a non-linear equation must be solved numerically for obtaining $d_{dep,max}$ (see Appendix A.2 for a more in-depth discussion). Besides the geometrical parameters of the TSV and some natural constants, $_{dep,max}$ depends on the substrate doping level N_a. Tab. 2.3 summarizes all the parameters needed for calculating the full TSV RC-model.

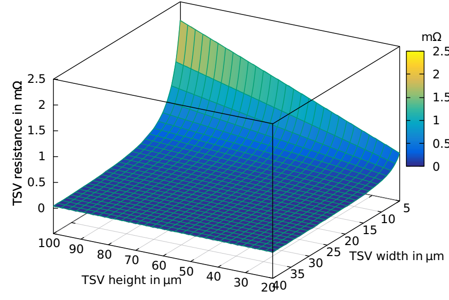
Fig. 2.10 shows the TSV capacitance, resistance and RC-delay ($t_p = ln(2)RC$) with respect to the TSV proportions. Capacitance is highest for long and wide TSVs since this maximizes the sidewall area. Resistance, however, is highest for long and thin TSVs. Regarding the RC delay, it is important to note that this is just the delay for the isolated TSV without any driving or receiving circuitry and without wiring. According to [75], the impact of conventional wiring will have a much higher impact on the total resistance of a TSV based link.

Tab. 2.3.: Input parameters for basic TSV model

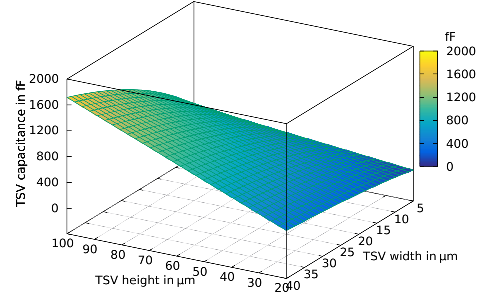
Symbol	Parameter
h_{tsv}	TSVs height
d_{tsv}	TSVs diameter
d_{ox}	Width of SiO_2 layer at TSV sidewall
ρ_m	Resistivity of the TSV metal fill
ε_0	Permittivity of free space
ε_r	Permittivity of SiO_2
ε_{si}	Permittivity of silicon
n_i	Intrinsic carrier concentration of the substrate
N_a	Substrate doping level
T	Temperature

2.2.2 TSV Dependability

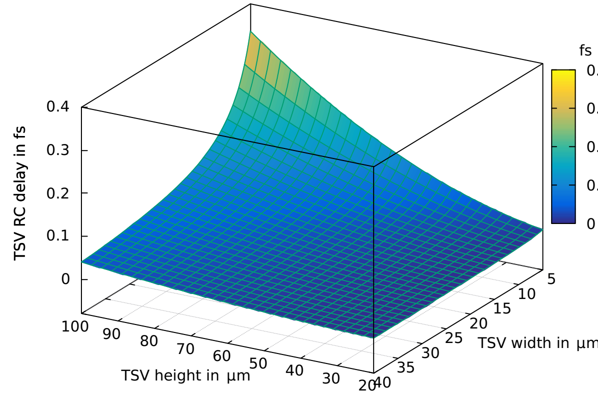
A major challenge in the fabrication of 3D-ICs is the still rather low yield of TSVs. Yield figures and statistics from the manufacturers are sketchy, and real numbers are hard to obtain since they are usually considered a business secret. Published numbers in literature vary widely. In [86], numbers from as low as 24% up to



(a) Resistance



(b) Capacitance



(c) RC delay in femto seconds

Fig. 2.10.: R_{tsv} and C_{tsv} with respect to TSV proportions (TSV material: cu, Oxide thickness: $t_{ox} = 250\,\mathrm{nm}$, Temperature: $T = 300\,\mathrm{K}$, Substrate doping: $N_a = 1.25 \times 10^{15}\,\mathrm{cm}^{-2}$)

100%, depending on the individual position of the TSV, are given. In [87], a single TSV yield figure of 99.2% is stated. The yield of a single TSV massively depends on manufacturing parameters, especially the TSV aspect ratio, TSV pitch and surrounding keep-out zones [86]. What can be said is that the fault rates of TSVs are orders of magnitudes higher compared to conventional interconnect failures in 2D designs. Therefore, a 3D-IC with a considerable TSV count either has to have non-critical parameters (low aspect rations, keep out zones), or means to tackle defective TSVs have to be integrated, otherwise it will most likely fail and the yield of the associated fabrication process will be close to zero. Fig. 2.11 illustrates this. When we assume a constant TSV yield of p_{tsv} the system yield drops to $p_{sys} = {p_{tsv}}^n$ for a system with n TSVs. As an example, even with a very high TSV yield of $p_{tsv} = 0.999$, the system yield would drop to $p_{sys} \approx 0.36$ for a system with $n = 1024$ TSVs. Therefore, most researchers treat the TSV yield as an input parameter. Resilient designs are then built in a scalable fashion, such that the added

effort to cope with faulty TSVs depends on both the TSV yield of the process and the targeted system yield.

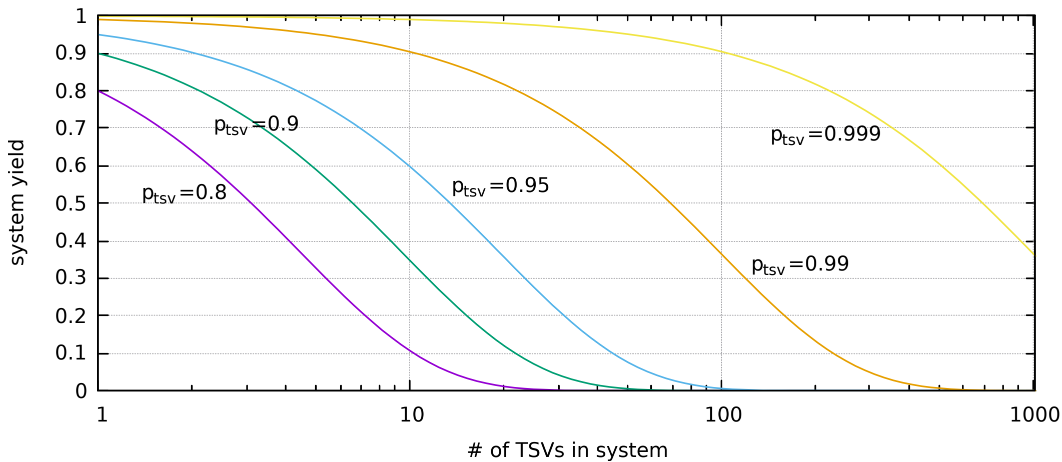


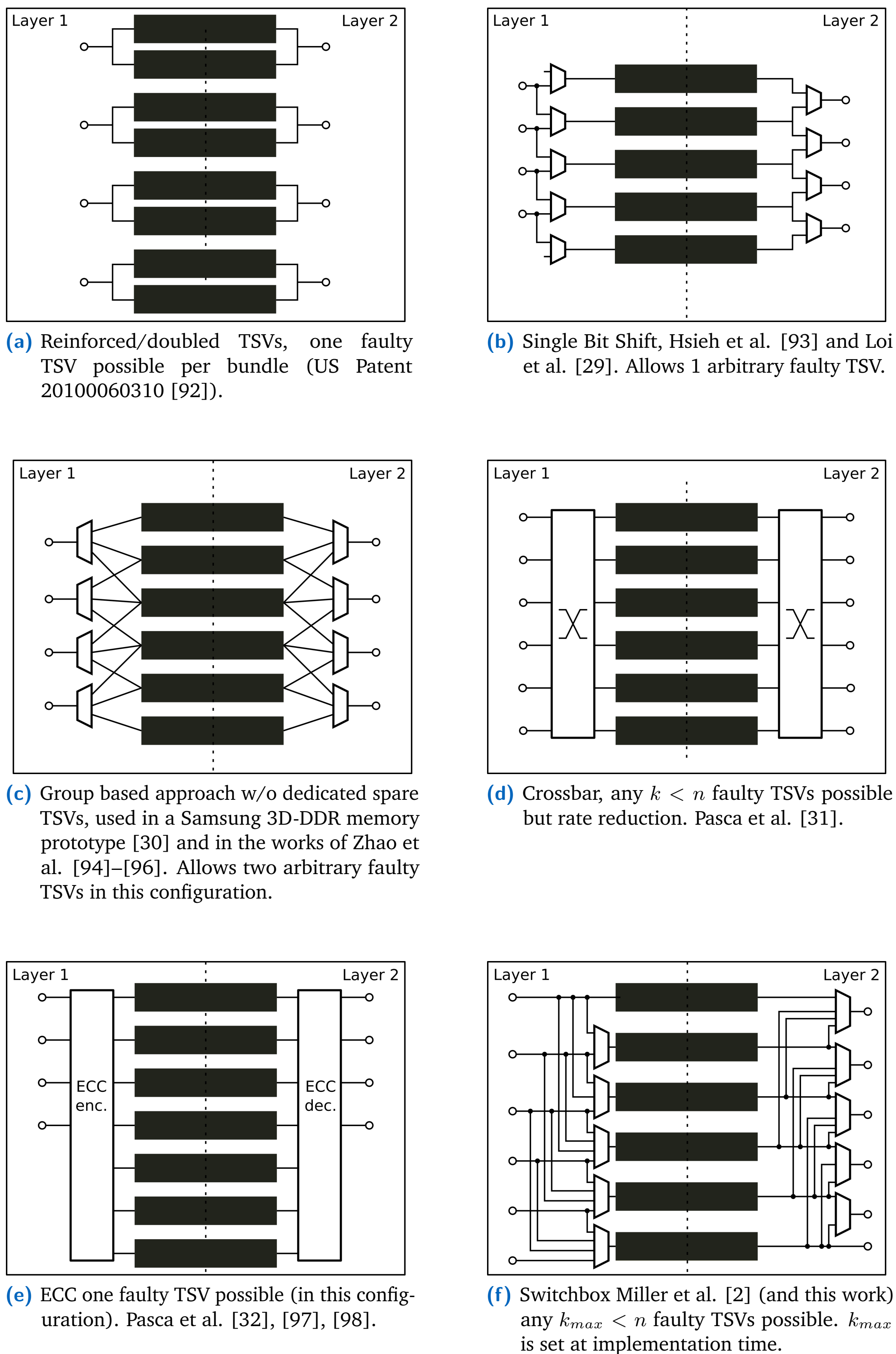
Fig. 2.11.: System yield with respect to TSV yield

A cross domain solution to build resilient interconnects, also in conventional planar ICs, is using Error Correcting Codes (ECCs). The construction and optimization of such codes forms its own research domain, dating back to the development of the famous Hamming Codes [88]. However, for 3D-ICs it is problematic to apply ECCs due to their requirements regarding additional signal wires. In modern planar circuits, wires are rather cheap due to many available metal layers [24], but in the case of 3D-ICs, additional TSVs are needed to carry the signals. For instance, for transmitting a Hamming encoded 4-bit data vector that can correct a single bit, 7 data bits are required. What is more, the encoding and decoding modules require a considerable amount of logic area. To reduce the area overhead induced by ECC, codes exist that require even more additional wires for the sake of reduced area costs, e.g. Duplicate Add Parity (DAP) [89] requiring 9 signals for a 4 bit wide vector.

TSV defects during production can occur due to a multitude of physical reasons, e.g. voids in TSVs, TSV pinch-off, oxide-defects, thermo-mechanical stress, chip warpage [90]. But also run-time errors occur and can render 3D-ICs faulty. Responsible for such in-field defects are mainly mechanical stress and warpage, thermal issues, but also Single Event Effect (SEE) issues and electromigration [91]. This further encourages researchers to see the TSV fault rate as a generic parameter and to also provide means against run-time failures for enabling extended product life cycles.

For the reasons above, most researchers focus on redundancy by some form of re-routing to build resilient TSV-based interconnects. Figure 2.12 shows an overview of published implementations.

The most simple and straightforward solution without any logical overhead is the approach based on what we call in this thesis *Reinforced TSVs* - sometimes referred to as *Double TSV* [99] or *TSV doubling* [50]. However, theoretically also triples,



(a) Reinforced/doubled TSVs, one faulty TSV possible per bundle (US Patent 20100060310 [92]).

(b) Single Bit Shift, Hsieh et al. [93] and Loi et al. [29]. Allows 1 arbitrary faulty TSV.

(c) Group based approach w/o dedicated spare TSVs, used in a Samsung 3D-DDR memory prototype [30] and in the works of Zhao et al. [94]–[96]. Allows two arbitrary faulty TSVs in this configuration.

(d) Crossbar, any $k < n$ faulty TSVs possible but rate reduction. Pasca et al. [31].

(e) ECC one faulty TSV possible (in this configuration). Pasca et al. [32], [97], [98].

(f) Switchbox Miller et al. [2] (and this work) any $k_{max} < n$ faulty TSVs possible. k_{max} is set at implementation time.

Fig. 2.12.: Comparison of methods for fault tolerance in TSV arrays

quadruples or n-tuples of TSVs are possible. The principle is initially described in an US patent [92] and is depicted in Figure 2.12a. It is based on the idea that it is less likely that two TSVs fail at the same time. Therefore, for each signal, two TSVs are connected in parallel to form a *Wired-OR*. The drawback of this solution is that despite many redundant TSVs being present, they cannot be used to replace any arbitrary TSV but only the one they form a bundle with. In Chapter 3.3 this principle is consulted as a baseline reference.

The majority of redundancy patterns perform some multiplexing or switching and reroute a signal blocked by a defective TSV through one of the available spare TSVs. The minimal form of this principle is built by placing a 2-to-1-Mux at every TSV input connecting to a TSV and to its neighboring TSV. One spare TSV is added to the array. This idea was first published by Loi et al [29], [100] but is also applied in the works of Hsieh [93]. The operating principle is as follows: The Multiplexers are set such that the input is directly assigned to the first of the two TSVs it is connected to. When a faulty TSV is encountered, the connected Multiplexer and all succeeding ones are switched to assign the signal to the next TSV to spare the faulty one. The last input is then mapped on the spare TSV. If no defective TSV is present, the spare TSV lays idle. The area overhead of this solution is relatively small, however only a single TSV can be repaired, making it only applicable for processes with high single TSV yield figures.

The next evolutionary step are re-routing systems that can cope with more than one faulty TSV (Figure 2.12c). The idea is presented in the works of [94]–[96]. A similar approach is also used in a Samsung's DDR memory prototype [30]. The works of Roy et al. [101], [102] are also based on this principle. With the Grouping approach, the TSVs of an array are divided into groups containing m TSVs. In each group k spare TSVs are added. Every TSV in the set can be mapped on a spare TSV through a Multiplexer structure (partial crossbar). The example in Figure 2.12c shows a single group with $m = 4$ TSVs and $k = 2$ spare TSVs. There are, however, no dedicated spare TSVs. Each of the $m + k = 6$ ones could be denoted as spare TSV. However, even if all TSVs are functional k TSVs lay idle.

In the works of Pasca et al. [32], [97], [98], ECCs are used for detecting and correcting errors. The main advantage of this principle is that no Built In Self Test (BIST) is required to detect exactly which TSVs are faulty. However, on the downside a large area penalty is induced since for every parity bit an additional TSV is required. Hence for a $(7, 4)$-Hamming Code [88] as depicted in Figure 2.12d 3 additional TSVs are required to cope with a single faulty TSV. Osewold et al. [103] also leverage ECCs but also require a *Fault Vector* denoting which TSVs are defective. In this case, only a single parity bit is required (hence one additional TSV) to correct a single error on the receiver side.

A crossbar approach is illustrated in Fig. 2.12d and is presented by Pasca et al. [31]. It maps any of the n inputs to any of the $n + k$ acpTSVs. It thus allows any number of defective TSVs, as long as the reduced size data word can be provided at the input and handled at the output.

The last approach shown in Fig. 2.12f is the Switch Box based approach published by us in [2] and discussed in more detail in Chapter 3.3. It was designed with the objective in mind to combine the advantages of the crossbar and grouping solutions described above, being:

- Usage of only a partial crossbar (of the grouping based solution)
- Allow any number of faulty TSVs (of the crossbar based solution)

On top of this, the maximum number of possible faulty TSVs k_{max} is a design parameter and can be set to any integer number $0 > k_{max} < m$. Setting the value to a lower number results in a smaller area overhead since the Switch Box logic gets less complex. There is no special denotation of spare TSVs. Instead, with increasing faulty TSVs, the capacity of the TSV array is reduced and link traffic is shaped accordingly.

2.2.3 Efficient use of TSVs

In Chapter 2.2.1, models to calculate the electric behavior of TSVs and to extract relevant parameters were discussed. The main objective to research such models in the scope of this work was to get an insight of the possible capacity a set of TSV can offer and on the extent TSVs behave differently when compared with conventional Back End of Line (BEOL) wires, looking at certain aspects one could even say contrary.

Tab. 2.4.: Comparison of conventional wires and TSVs

	global BEOL wire	TSV
Cost of a single wire	low	high
Capacitance	high	low
Resistance	high	lower
Length	high	low
Area	low	high
Max clock speed	medium/low	high/very high
Reliability	high	medium/low

Tab. 2.4 shows that the simple electrical continuation of conventional wires by connecting them directly to dedicated TSVs has its downsides. Such a connection combines the negative aspects of both types since each aspect in Tab. 2.4 will be dominated by the respective low performer. The TSV part will induce a high area due to a large footprint, and it will have a low reliability, governed by the low

TSV reliability. Overall length, capacitance and resistance, which determine the maximum clock speed, will be limited by the BEOL part.

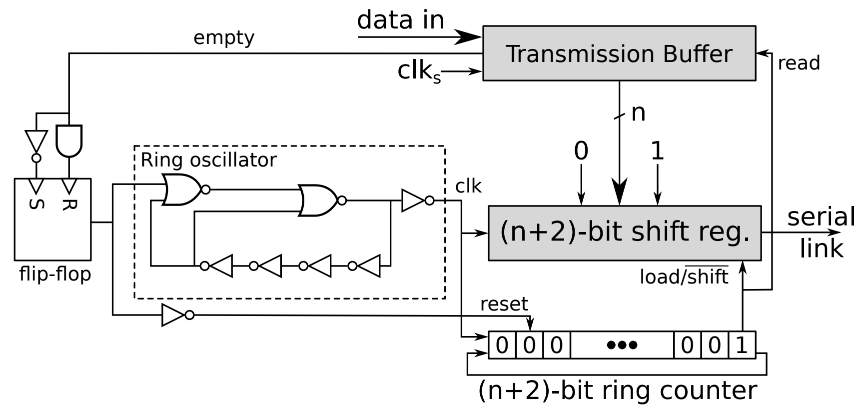
Therefore, using TSVs to stretch conventional links to other layers in 3D-ICs has negative side effects since the TSVs are not working to their full potential. The large area consumption makes wide parallel TSV array based inter layer links next to impossible. In [28], the authors show that a single 32 bit link when used in a conventional synchronous fashion leads to an area overhead of roughly half of the size of a modern processing core. The authors assume a $90\,\mathrm{nm}$ process but also a high area consumption per single TSV. While minimal feature size has been further scaled down since the publication of [28], the evolution of TSV diameters did not keep pace with this trend. Hence, taking more up-to-date numbers for both TSV area and node size will even show a more dramatic discrepancy of logic vs. TSV area.

One solution to better exploit TSV capabilities is to use serialization and to make TSV-based interconnects narrower but faster (in terms of used clock speed). Serialization mechanisms for TSV based interconnects have been discussed in several publications [27], [28], [104], [105].

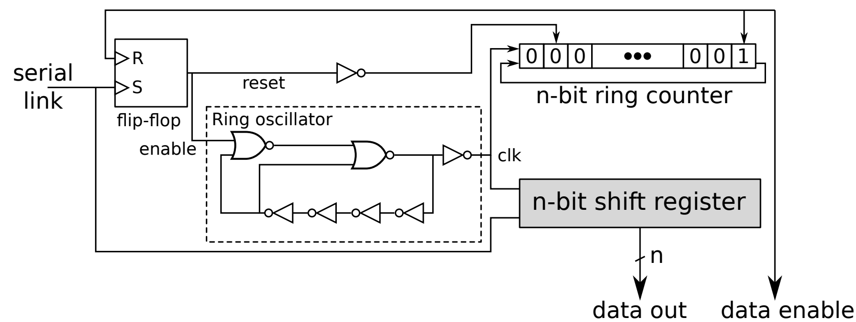
Pasricha et al. first proposed the idea of serializing inter-layer links in [27]. The serializer design is based on existing works for serial links in conventional circuits [106]–[108] and forms a UART-like setting with no dedicated clock signal. Instead, the clock signal is (re)created on both layers by means of an internal Ring Oscillator. Fig. 2.13 shows the receiver and transmitter design. The authors synthesized their design for several serialization rates, technology nodes and TSV pitch figures. They show area savings of 55 to 70% for a 4:1 serialization.

A significant portion of proposed serialization solutions use Quasi-Delay-Insensitive (QDI) signaling [109] on the TSV-based link between serializer and deserializer [26], [104], [105], [110]–[112]. QDI signaling was already previously suggested for conventional 2D-NoC-links [113], [114]. The standard way to build a QDI-link is by using *1-of-m* signaling, with a dedicated signal being used for each possible data word and a single acknowledge signal. Therefore, to replace an n-bit wide synchronous link $m = 2^n + 1$ signals is required. Fig. 2.14 shows the signal sequence for a *1-of-4* signaling (as it is used in all works cited above) when transmitting the sequence $[00\ 01\ 10\ 11]$.

The main advantage of QDI is, as the name suggests, that it is delay insensitive. This means not only that the individual wires are always operated at the edge of their switching performance, but also that signal skew is fully irrelevant. Also no clock has to be conveyed, meaning that the clock domain crossing comes for free. The only required synchronization is located in the sender and receiver to attach to other synchronous parts of the circuit.



(a) Transmitter



(b) Receiver

Fig. 2.13.: (De)serializer design of [27]. Data is loaded into a Shift Register when data is available and the serial link is available. The individual bits are then put on the single bit transmission line sequentially, framed by a start bit and a stop bit and clocked by an internal ring oscillator. On the receiver side, the process is reversed.

On the downside, however, QDI links require a significant signal overhead. From all 1-of-m variants, 1-of-4 shows the greatest efficiency with respect to signals per data bit (with 2.5 signals per bit), which explains why it is widely used. Both 1-of-2 and 1-of-8 signaling require 3 signals per bit, and with higher m-values efficiency decreases further. Therefore, wider signals are usually divided in groups of 4. The authors of [104] state that the solution requires 184 signals to establish an asynchronous full-duplex 32-bit link (this includes data for up- and downstream direction and additional control signals). In [104], the authors do not show area savings yet. In a follow-up work [105], however, they show significant area savings, dependent on serialization rate and used TSV and conventional integration technology.

In [28], the authors present a synchronous serialization scheme. A dedicated clock TSV is used to convey the clock signal to the receiving layer. The operating principle is shown in Fig. 2.15. The impact of different serialization ratios with respect to its savings regarding area is studied. The authors show area savings of up to 75% for a $20\,\mu\text{m}$ TSV pitch (compared to a reference system where no serialization mechanisms are applied). A similar approach is used in the works of Beanato et al. [115]–[117].

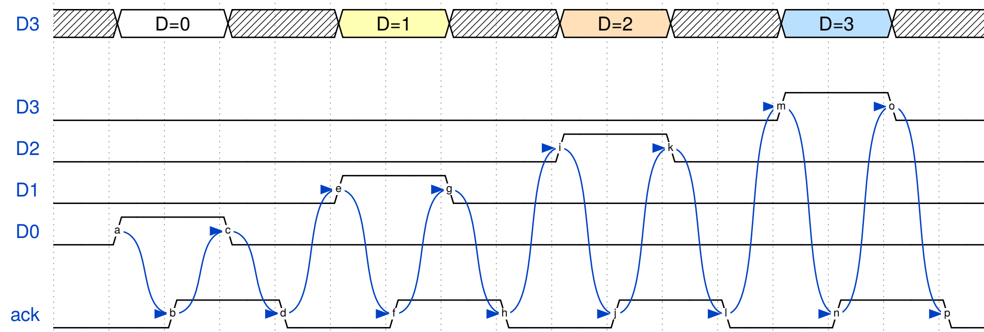


Fig. 2.14.: Timing for a QDI link with *1-of4* signaling as it is used in [26], [104], [105], [110]–[112] to replace a 2-bit synchronous link. In this example, the sequence $[00\ 01\ 10\ 11]$ is transmitted. A dedicated signal is assigned to each possible link state (combination of single bits). When a data word is to be transmitted, the assigned signal is set to high by the sender. As soon as the signal transition is recognized by the receiver, this is indicated by asserting the *acknowledge* signal. This is in turn acknowledged by the sender by setting the data signal to low again. This is one last time answered by the receiver by deasserting the acknowledge signal. Consequently, at the end of this cycle, all signals are low again (*All-Zero-Wavefront*), and a new word can be transmitted. From another perspective, each data signal, when used, executes a *4-Phase-Handshake* process together with the acknowledge signal.

The authors show area savings between 75% and 95%, subject to serialization rate, TSV diameter and conventional feature size.

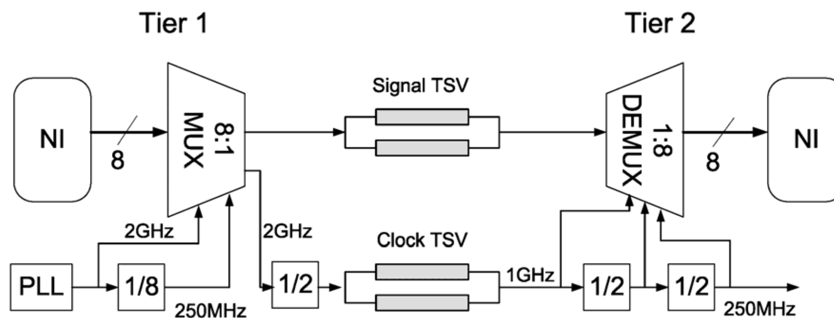


Fig. 2.15.: Synchronous serialization methodology of [28]. The clock is forwarded to the receiving layer with dedicated clock TSVs. To improve resilience, TSVs are redundant.

Besides the impact of different serializer complexities the area savings for a given serialization rate mainly depends on the ratio of TSV pitch to the feature size of the used 2D integration technology. In general, we can observe that since the first 3D-ICs emerged, the scaling of the TSV pitch has not been able to keep pace with the traditional *More More* downscaling of conventional integration. As a result, the area and power overhead introduced with serialization techniques has reduced impact. This can be seen on the recently published works Beanato et al. [115]–[117]. The highest area savings are reported there for the case where a $40\,\mathrm{nm}$ node is used together with a $40\,\mu\mathrm{m}$ TSV pitch.

A serialization solution has also been published by us [1], [2]. When solely comparing the serialization portion to the discussed variants above, it is related most closely to the synchronous solutions, hence the works of Sun [28] and Beanato [115]. However, a logarithmic shifter is used for serializing. This allows for dynamically adapting the serialized word size and coping with faulty TSVs. Also, it incorporates the multiplexing of multiple independent links on a TSV array. To the best of our knowledge, this is the first work that presents a solution where the bandwidth of a high speed TSV array can be dynamically assigned to individual links. This solution is presented in Chapter 3 in more detail.

2.3 3D-Networks-on-Chip

In the last 10 to 15 years, the paradigm of *Network-on-Chip* has been pushed by research and industry to replace conventional interconnect structures like buses, rings, etc. by more sophisticated networks similar to computer networks. The principle was introduced in two publications by Benini and Micheli in 2002 [19] and by Dally and Towles in 2001 [18], respectively. Dally and Towles did not use the term *NoC* but described the same idea, i. e., replacing the previously bus based backbone of SoCs by packet switched networks.

The key motivator behind this development is the fact that conventional interconnect structures like buses and crossbars do not scale well with increasing numbers of nodes. Shared media buses do not allow concurrent transactions by nature, and systems that do, like crossbars, are not feasible for a larger number of ports due to their N^2 circuit complexity (with N being the number of ports) [118].

Another factor is the so-called *Interconnect Bottleneck* [119] which denotes the increasing gap between wire and gate delays. Gate delays have been constantly reduced with the ongoing shrink of feature sizes. However, overall chip area has stayed constant or even increased, and so did the delay of longer wires. In the pre-NoC era, it was possible for electrical signals to cross the entire chip within a single clock cycle by using global wires [120]. With today's clock frequencies this is not feasible anymore, and therefore shared media buses are not viable as main interconnect backbone anymore.

The above problems have also been addressed by introducing segmented buses even before the term *NoC* was coined [121]. Today, besides NoCs, sophisticated interconnect architectures like Advanced eXtensible Interface (AXI) or Multi-Layer-AHB [122] exist, which are often also combined with a multi-hop interconnect fabric and also allow concurrent transactions to a certain extent [27].

However, there is another key differentiator which makes up a NoC: The packet switched nature, hence the fact that data is packetized and sent sequentially. NoC packets are further divisible into so-called *Flits*, which sequentially traverse through the network. In contrast to bus-like interconnects, where the address, control and payload data is sent in parallel, in a NoC, address and control information is contained in a *Header Flit* preceding the payload data Flits. Consequently, NoC links usually also require less wiring resources when compared with traditional interconnects.

For designs forming a regular pattern, meaning that the same type of core is instantiated repetitively, also the on-chip network will form a regular structure. A good example for this category are Chip-Multi-Processors (CMPs) [123] usually backed by a mesh structured NoC [24], [124], [125]. Such systems are designed as general

purpose system, hence the concrete application is not known during the hardware design.

This is different however for most SoCs with their many specialized modules, like hardware accelerators, memories or interface circuits. For such systems, much more information on the potential use cases is known at hardware design time. Therefore, it makes sense to specifically tailor the NoC to the communication requirements for optimally supporting the use cases. Such NoCs are then called *Application Specific NoCs* [126].

The reasons given above already justify the usage of NoCs for conventional planar designs. However, in 3D-ICs the factors favoring the application of NoCs are emphasized:

- **Number of Nodes:** In 3D-ICs, a large amount of silicon area is available allowing a large number of modules, hence the node count will be increased.
- **Heterogeneity:** The possibility to integrate dies, manufactured with different integration processes, enables heterogeneous integration. Such highly heterogeneous systems require an Application Specific NoC tailored to the specific communication requirements of the individual components.
- **Wires:** Wires are not necessarily longer for 3D-ICs. In fact, 3D-Integration can even be used to reduce the average wire length [127]. However, continuing shared media buses with TSV based interconnects without terminating the protocol with adapters to use an underlying transport protocol is not economical due to the large area requirements of TSVs.

The authors of [19] also point out the hierarchy of layers representing the different abstraction layers that define a NoC much like in the OSI model [128]. This is of particular interest in the context of this work, since later a stack is presented that adds an additional layer for TSV virtualization.

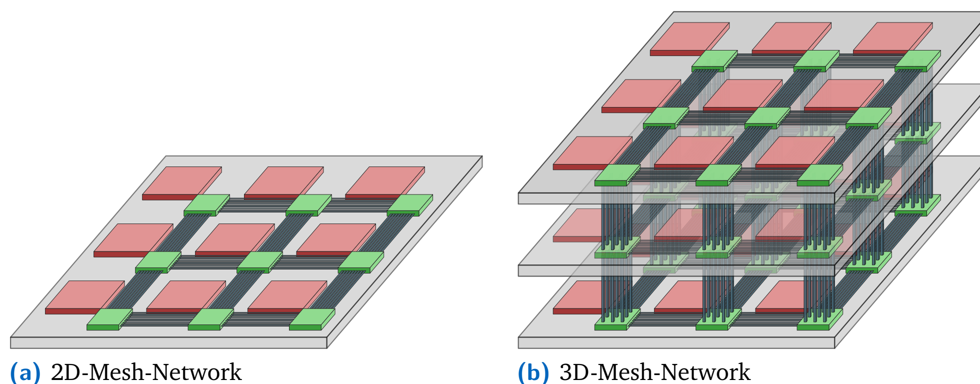
- The **Physical Layer** is the lowest layer in the stack, which characterizes the physical point-to-point links for interconnecting individual routers.
- The **Data Link Layer** ensures reliable communication over the physical links. It provides error detection/correction, flow control and synchronization.
- The **Network Layer** is the layer where the topology, router architecture, buffering scheme and routing scheme are defined.
- On the **Transport Layer**, the highest layer in the stack, packetization and depacketization as well as end-to-end flow-control and Quality of Service (QoS) are defined.

2.3.1 NoC Building Blocks and Structure

A full on-chip network is composed of smaller building blocks. Routers and Network Interfaces (NIs) are interconnected to form a topology, the structure of the network. A *routing protocol* is applied to conduct the flow of packets through this network.

2.3.1.1 Topology

The topology having by far received the most attention in research is the regular mesh topology. Fig. 2.16 shows an example of the layout of such a structure for both a 2D and a 3D version. The fully interconnected mesh topology is the solution of choice for providing a high capacity NoC fabric for regular systems like CMPs and is already used in commercial products [129], [130].



(a) 2D-Mesh-Network (b) 3D-Mesh-Network

Fig. 2.16.: 2D- and 3D-Mesh Network-on-Chip topology

Another popular and widely used regular topology is the ring topology which is less complex than a mesh setup with respect to several aspects like routing scheme, router architecture and wiring. Ring topologies can also be found in commercial products [129].

Many other regular topologies exist, which can be variants of a mesh structure (sparsely interconnected mesh networks or torus networks), tree-like structures, spidergon networks or butterfly topologies [129], [131], [132]. Also mixed versions of the aforementioned structures are possible [132].

Regular topologies can be extended for the 3D scenario. The added degree of freedom further increases the possible variants and combinations of such structures [133]–[137]. The natural extension of the standard fully interconnected 2D-mesh is the fully interconnected 3D-mesh as illustrated in Fig. 2.16b. However, such a topology is of rather academic nature since the vast amount of TSVs that would be required to realize such a structure is unrealistic from a manufacturing standpoint regarding reliability as well as area consumption [138]. Therefore, more sparsely interconnected topologies are used [135]–[137], or TSVs are shared by

using them as a bus-based shared medium [139] or by using serialization and multiplexing which is the approach followed by the methods presented in this work [1], [2].

2.3.1.2 Routing and Deadlock Avoidance

The routing algorithm decides the path a packet follows when it traverses the network. Hence, it determines the sequence of routers from source to destination. The routing algorithm is an aspect where most NoCs substantially differ from computer networks as NoC routing schemes are designed to be area and power saving.

The aspect of routing has to be seen in conjunction with the deadlock problem. A routing algorithm is deadlock free, regardless of the topology, when the *Channel Dependency Graph* is acyclic [140].

Routing algorithms are categorized regarding several aspects. On a high level, the routing schemes can be grouped according to their level of adaptivity [129]:

- **Adaptive routing** schemes consider the current status of the network (e. g. load level) for the routing decisions. Such algorithms offer a higher performance since they aim to balance the load throughout the network. However, they are more costly in terms of area and power requirements.
- **Oblivious routing** schemes do not consider the state of the NoC regarding e. g. the current load level.

Most NoC routing algorithms fall into the second category and are not adaptive. This is mostly due to restrictions regarding router complexity. Within the category of oblivious routing, the subcategory of *deterministic routing* can be defined. Such algorithms are of even more simplicity, and here the full path of a message through the network is defined once source and destination is known.

An important and widely used group in the category of deterministic routing are *Dimension Ordered Routing (DOR)* algorithms. Here, one dimension is traversed after the other. Hence, in a 2D network, there will always be at most one turn, in a 3D network at most two turns. With a DOR algorithm two packets originating from the same source and destined to the same sink will always take the same path through the network. DOR algorithms are inherently deadlock free since they only allow a reduced set of possible turns, cyclic dependencies cannot be created.

Fig. 2.17 illustrates the categories exemplary for a 2D-mesh network.

In a 2D-mesh network, the sufficient requirement for deadlock freedom is that at least one of the possible turns is forbidden for both clockwise and counter-clockwise direction. Therefore, routing algorithms have been constructed that allow more flexibility than DOR but are still deadlock free. Such routing algorithms form the

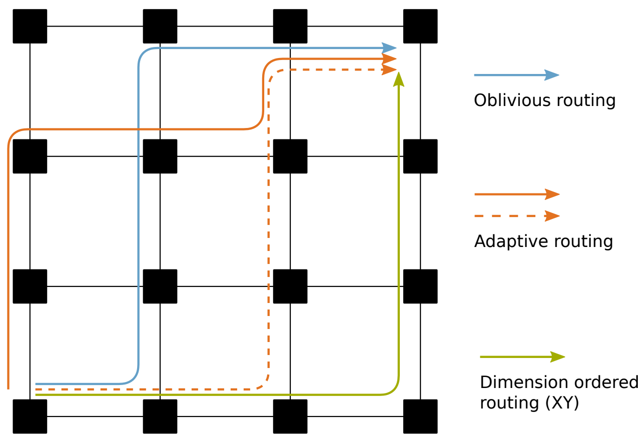


Fig. 2.17.: Different routing categories illustrated for a 2D-mesh NoC. All packets are originating form router (1,1) and are destined to (4,4). (Adapted from [129])

.

group of turn model based routing and carry names like *west-first*, *north-last* [141] or *odd-even* [142].

The 2D-mesh requirements for deadlock freedom can also be extended for 3D-meshes. DOR algorithms like *XYZ* routing have been proposed [143] as well as turn model based algorithms like *Elevator first* [144].

2.3.1.3 Network Interface

The NI forms the adapter between a Processing Element (PE) and the actual network. The basic task performed by the NI is the packetization and depacketization of data. Hence, the NI provides two interfaces, one towards the NoC fabric and one towards the PE. The interface between the network and the NI clearly is the NoC's specific link interface. The one between PE and the NI might also be NoC/implementation specific, but often one of the socket-based on-chip interface standards [27] like *AXI*, *Whishbone* or *Open Core Protocol (OCP)* is used here since many IP-Cores are equipped with such interfaces.

In addition to the mandatory functionality of (de)packetization, many other potential tasks are optionally performed by the NI depending on the NoC complexity and feature set. This includes the following:

- End-to-end flow control
- Packet/flit reordering
- QoS handling
- Clock domain crossing

2.3.1.4 Router Architecture

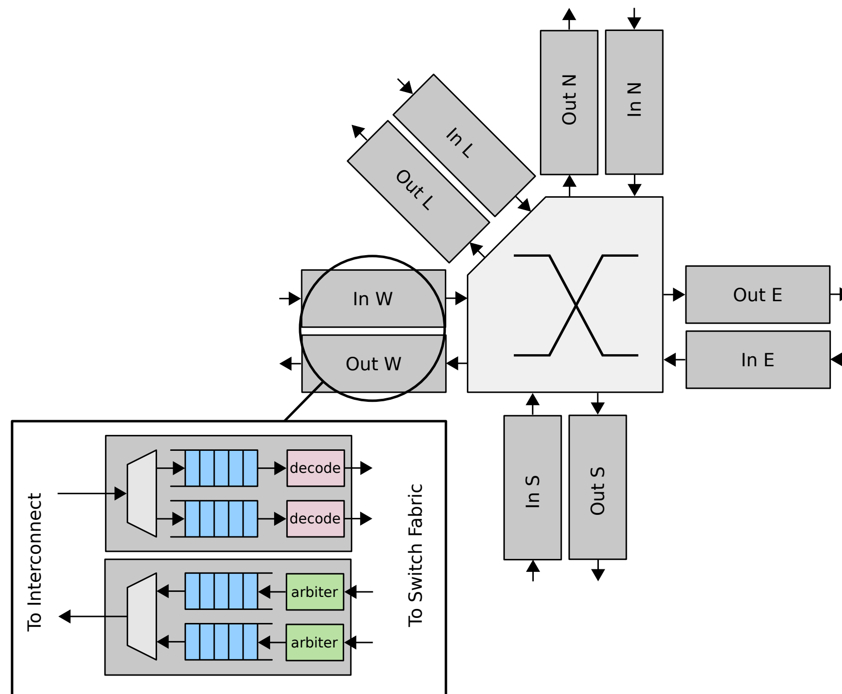


Fig. 2.18.: NoC router with 5 ports and 2 Virtual Channels (VCs) per port to be used in a regular mesh network.

The key functionality of the NoC router is to forward packets to the appropriate output port. A typical router setup is depicted in Fig. 2.18, showing an inner router of a 2D-mesh topology. In many cases, like the traditional baseline fully interconnect 2D-mesh network, a router has four ports to connect to neighboring routers and one local port. Obviously, corner routers and the ones at the edges have fewer ports. Those ports are often denoted with the names of the cardinal directions, hence they are called *north*, *east*, *south* and *west*. When extending the mesh to a fully interconnected 3D grid, two more ports have to be added, which are then denoted as *up* and *down* port. Hence, a router in a 3D-mesh NoC is equipped with at most seven ports.

The port count is also what generally is referred to as the router's *Radix*. Obviously, the Radix depends on the network topology and the position of the router in this topology. The Radix has a significant impact on the router's complexity, its area footprint and power consumption.

In most cases, the PE is connected via a single local port to the router. Sometimes multiple such elements share a port and form a so-called *Tile*, which has independent means to communicate internally, e. g. a local bus or crossbar. Also, the router's switching matrix can be leveraged to provide the communication between the elements of a Tile, by using routers with more than one local port.

A router in turn is again constituted by various building blocks including:

- **Switching Matrix:** The Switching Matrix is a building block that can forward multiple Flits at the same time to a respective output port. It is usually build as a fully interconnected crossbar. However, in certain situations, for instance if it is a priori known that packets from a certain input are never forwarded to a certain output, it might only be partially interconnected to reduce complexity.
- **Buffers:** Buffers serve the task to temporarily store packets or flits if they cannot be instantly forwarded to the desired output port (e. g. because of arbitration or back pressure). Buffers are located at the input or output side of the Switching Matrix or both. Classical challenges like Head of Line (HoL) blocking for input buffered routers or the requirement for crossbar speedup for output only buffered routers, which are already known from classical interconnection networks [145], live on in the NoC world.
- **Virtual channels:** A virtual channel is actually not an independent building block but is realized by adding additional buffers to an input. This is also shown in the example in Fig. 2.18. Multiple virtual channels share the same physical link between two routers. This allows still using the physical link even if some queues are blocked. Virtual channels can reduce the impact of HoL blocking, allow QoS handling and are helpful to prevent deadlocks.
- **Arbiters and allocators:** An arbiter selects one of multiple input ports contending for the same output port and assigns it to this output for one cycle. An allocator covers multiple output ports and assigns an input port to each (if data is available). Wheter arbiters or allocators are used or not depends on the router architecture. If each input queue is directly connected to the crossbar (has its dedicated port), usually one arbiter per output port is used [129]. In this case, it is always guaranteed that the output will be served by the selected input. If, however, multiple input queues are contending for an input port to the crossbar, the task gets more complex and forms an allocation problem [145]. The allocation problem is much more complex than the arbitration one, since finding an optimal mapping is a challenging task. Also, the optimal mapping is unclear. The allocation logic can be optimized for different objectives, e. g. maximize throughput or fairness [145].
- **Decoders:** The decoder logic (usually one block per input queue) is responsible for looking up the output port the packet has to be forwarded to. Hence, it probes the header Flit and sets the appropriate control signals to signalize that a packet for a specific port is waiting.

When adapting NoC router designs for the 3D realm, the required modifications seem straight-forward at first glance. Simply two ports have to be added. However, there are several reasons why a fully-fledged 3D-mesh NoC is not a practical solution. Firstly, as already mentioned, the vast amount of required TSVs is problematic. Secondly, due to the quadratic circuit complexity measures, a 7-port crossbar is

overly more complex than a 5-port one [146]. There is one more reason: The 3D-mesh as depicted in Fig. 2.16b is only a very schematic illustration showing the mesh as an almost cube-like structure. However, when looking at the real dimensions, it can be seen that the vertical extent is very small compared to the horizontal one. The thickness of a thinned waver is in the region of $50\,\mu\mathrm{m}$. The horizontal distances depend on the core size but can easily reside in the mm range. A more scale preserving outline is shown in Fig. 2.19. Hence, the routers in a 3D-mesh are in very close proximity to their vertical neighbors. And what is more, in between resides only a very short, very high capacity TSV-based vertical link, that consumes however a full hop [146].

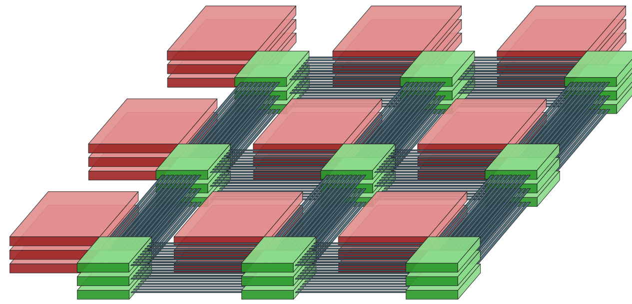


Fig. 2.19.: Scale preserving scheme of a 3D-mesh NoC. The horizontal distances between the routers are in the dimension of mm, whereas the vertical distances are in the dimension of tens of $\mu\mathrm{m}$.

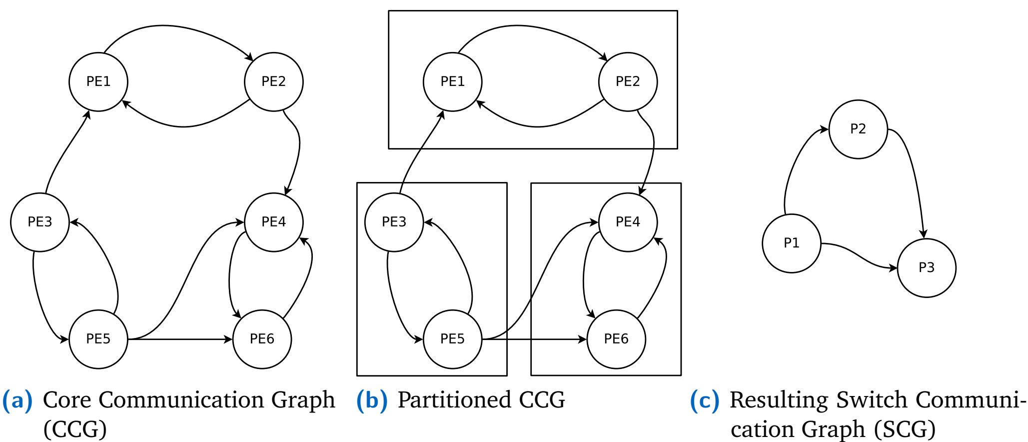
The aforementioned issues of 3D-mesh networks have led to a situation where the fully interconnected mesh normally is not considered as the standard baseline architecture optimized and considered a valid practical solution, as this is the case for 2D designs. Instead, many alternative router concepts have been researched (see [147] for a survey). This includes for example bus based routers [148], [149] or routers extending to multiple layers [150]. In [146], a router design is proposed that decouples inter-layer and intra-layer communication and provides a separate crossbar for each. In [151], a 3D-crossbar is proposed. Nevertheless, also the 7-port 3D-mesh router is outlined (e. g.in [152]), although it consumes, as expected, a high amount of silicon area.

2.3.2 Application Specific Networks-on-Chip

For designs with many specialized modules the communicating patterns are already (at least partly) known at design time. The communication requirements between the individual modules (PEs) can vary vastly. Think of a GPS sensor that only needs to inject a few bytes or kilobytes into the network every few Milliseconds and a high-resolution image sensor used to record a UHD video, where Gigabits of data have to be transferred per second. A mesh network would either be an underdesign for the one or an overdesign for the other. Therefore, if information on the communication patterns is available, it makes sense to specifically tailor the on-chip network to

the requirements. However, with many participating PEs, the task of designing the optimized NoC gets utterly complex and a huge design space has to be covered. Many research papers show that the problem is in fact NP-hard [34], [153]–[155]. Hence, closed form solutions do not exist and the problem has to be tackled by heuristics and optimization algorithms. The topic of application specific NoC design is therefore closely coupled to the topic of the automated synthesis of such networks, and both topics are usually covered together in research papers. An initial problem formulation can be found in [156].

The scientific works on application specific NoC synthesis are diverse in the sense that different sets of information are used as initial starting point, and different aspects of the design process are given already. Regarding the full system design process, communication is only one aspect of many that have to be considered. And clearly, it would lead to an improperly balanced system if the synthesis process was only based on communication. Also, the objectives of these aspects can be very different. A good example for this is heat dissipation vs. communication. Cores that communicate heavily and therefore are usually active at the same time are grouped closely together if the synthesis process focuses on optimizing communication to limit wiring resources. On the other hand, in a process where the focus is optimized heat dissipation those two cores would be placed with high spatial distance to prevent thermal hot spots.



(a) Core Communication Graph (CCG) (b) Partitioned CCG (c) Resulting Switch Communication Graph (SCG)

Fig. 2.20.: A Core Communication Graph (CCG) with 6 PEs, a partitioned version of this CCG and a corresponding Switch Communication Graph (SCG) with a node representing each partition.

The starting point and problem formulation for an automated generation of a specific network (whether 2D or 3D) is usually a directed graph, which formally specifies the communication requirements of the system. In some works this graph is called *Core Communication Graph (CCG)* [34], [157], others call it *Core Graph (CG)* [153] or *Communication Task Graph (CTG)* [156]. Fig. 2.20a depicts a very basic example of such a graph. The vertices represent the communicating entities, and the edges

represent the communication requirements between two elements. The graph itself does not say anything about the spatial dimensioning, underlying layout or floorplan of a design. It might already exist, or creating this information might be part of the synthesis process. Also, the type of entity which the nodes of the graph represent and the level of granularity might be different for different scenarios. It could be cores of an already floorplanned system, different concurrent processes on a core, or it can even be tasks that still have to be mapped on compute elements. The communication requirements between the nodes that are represented by the edges can have multiple constraints, assigned depending on the type of algorithm, scenario, synthesis tool, design goal, etc. The most prominent constraint is clearly the bandwidth requirement in bytes per second, but also others like latency or burstiness can play an important role. It is realistic for the communication between two nodes to be asymmetric (e. g. a processor polling an image sensor), hence the use of directed graphs. This applies not only to bandwidth but also to other constraints (e. g. a latency requirement can be different for upstream and downstream direction). Also, having multiple flows between the same nodes in the same direction is realistic and can be legitimate for some synthesis algorithms. This might be the case if multiple channels exist between two nodes (e. g. for differently prioritized information) with different constraints for each of the channels.

Some algorithms do not rely on a single CCG but multiple of such graphs as input. The different graphs are then denoted as *Use Cases* [34] since they represent the different application scenarios a SoC can be operated in. The requirements for the individual use cases might be very different, e. g. recording a video and browsing the web will generate different graphs for a mobile SoC, where some connections may lay idle and some may create high requirements [35]. The synthesis tool then has to create a network satisfying all of the Use Cases.

An overview description of the NoC synthesis problem can be found in the works of Marculescu et al. [156], [158]. The authors break the problem of NoC synthesis down into four smaller problems:

- Topology Synthesis Problem
- Channel Width Problem
- Buffer Sizing Problem
- Floorplanning Problem

In most research, the term *Application Specific NOC-Synthesis* refers to the problem of *topology synthesis* only, i. e. to finding an optimal NoC topology for a given CCG. However, from a holistic point of view, the synthesis process cannot be seen as a simple top-down task in a chain of subsequently executed tools. Topology Synthesis is rather a step in a cyclic process since the outcome might not be satisfactory. Also, the parameters and optimization goals used and set in a synthesis flow are of high importance. It is clear that in a scenario with unlimited resources the synthesis

flow is very simple. One could implement a huge crossbar, with the throughput of the channel with the highest communication demand. However, since resources are usually limited and there are different kinds of resources (e.g. area, power, interconnect, heat dissipation), it is clear that a synthesis tool must have some information on the individual design objective.

Many researchers have studied the topic of automated topology creation for Application Specific NoCs in the last 10 years. The approaches of early works and tools were not aimed at a fully customized topology creation but rather used a mapping process to map a given design on existing modifications of mesh-like structures [126], [159]–[162].

The next evolutionary step are fully-fledged topology synthesis processes. In contrast to the mapping algorithms, these tools follow a generative philosophy and build the network from scratch from a set of configurable building blocks.

Tab. 2.5 compares published NoC synthesis flows. Most approaches further divide the problem of topology synthesis into the following steps:

- Determining the number of clusters
- Clustering
- Router insertion
- Routing path allocation

Some algorithms combine the first and the second step in the list above, and some combine the second and the third. The initial numbers of clusters is not straight forward and has to be determined. Some flows require this as user input, some sweep over all possible cluster counts, perform the full synthesis for each and select the one that best satisfies the design objective. Others run a heuristic or algorithm to determine the number of clusters. The approach for cluster count selection is listed in column *Cluster Size* of Tab. 2.5 for the compared solutions. For the actual clustering phase optimization algorithms are used (e. g. Min-Cut, Linear Programming) or heuristics (see column *Clustering Algorithm* in the table). The router insertion step is straight-forward in most flows and a router is inserted for each cluster. The outcome of the clustering process is depicted in Fig. 2.20b as the partitioned CCG.

The step of *routing path allocation* is always treated separately since this covers an independent problem. It denotes the construction of links between the inserted routers. Many flows use a greedy algorithm for this step (see column *Path Allocation*). The outcome of the Routing Path Allocation (RPA) phase is illustrated in Fig. 2.20c as the Switch Communication Graph (SCG), where each node represents a router, and the edges represent the links between the routers.

Another aspect where the proposed flows differ is the set of considered metrics for specifying the *affinity* between two links. The affinity is the final value that is

annotated to the edges of the CCG and is constructed from a weighting function. This weighting function always considers the bandwidth requirements, but sometimes latency and spatial distances are included as well (see column *Affinity*).

In the last three columns in Tab. 2.5 specific features are covered that may be supported by the individual flows. Some algorithm support multiple *Use Cases*, i. e. more than one input CCG. An important feature is deadlock avoidance. Some approaches create deadlock free networks by prohibiting cycles in the Channel Dependency Graph, some by inserting Virtual Channels, and some do not guarantee deadlock freedom at all.

Also, synthesis flows for applications specific NoCs for 3D systems have been researched [155], [163], [164]. Those are all based on existing 2D synthesis flows adapted to create a 3D topology. None of thes incorporate serialization and multiplexing strategies for reducing TSV count. The 2D synthesis process presented in the works of Todorov et al. and Verma et al. [33]–[35] forms the foundation of the flow proposed in this work (Chapter 4) and has been extended in collaboration with the authors of [34], [35] for the 3D scenario.

Tab. 2.5.: Comparison of published synthesis solutions for Application Specific NoCs

	Reference	Year	Cluster Size	Clustering Algorithm	Affinity	Path Allocation	Multiple Use Cases	Deadlock Avoidance	3D-NoC
[153]	Murali et al.	2006	sweep	min cut	bandwidth	greedy algorithm	no	yes	no
[157]	Yu et al.	2010	fixed constraint	min cut	distance bandwidth	DP	no	no	no
[165]	Leary and Chatha	2010	fixed (core count) routers can be merged	heuristic	bandwidth	heuristic	yes	yes	no
[166]	Soumya et al.	2013	fixed (core count)	not applicable	bandwidth	heuristic	no	no	no
[167]	Tosun et al.	2012	minimum router count	heuristic	bandwidth	greedy algorithm	no	yes	no
[168]	Khan and Tino	2012	included in clustering algorithm	Tabu Search	bandwidth	Tabu Search, heuristic	no	yes	no
[35]	Todorov et al.	2014	heuristic	Spectral Clustering	bandwidth	heuristic	yes	yes	no
[155]	Seiculescu et al.	2009	sweep	min cut	bandwidth, latency	greedy algorithm of [153]	no	yes	yes
[163]	Zhong et al.	2011	heuristic	LP	bandwidth, distance	greedy (Dijkstra's algorithm)	no	yes	yes
[164]	Barzinmehr et al.	2017	method of [167]	method of [167]	bandwidth	greedy heuristic	no	no	yes

3 TSV-Hub for Virtualization of Inter-Layer-Links

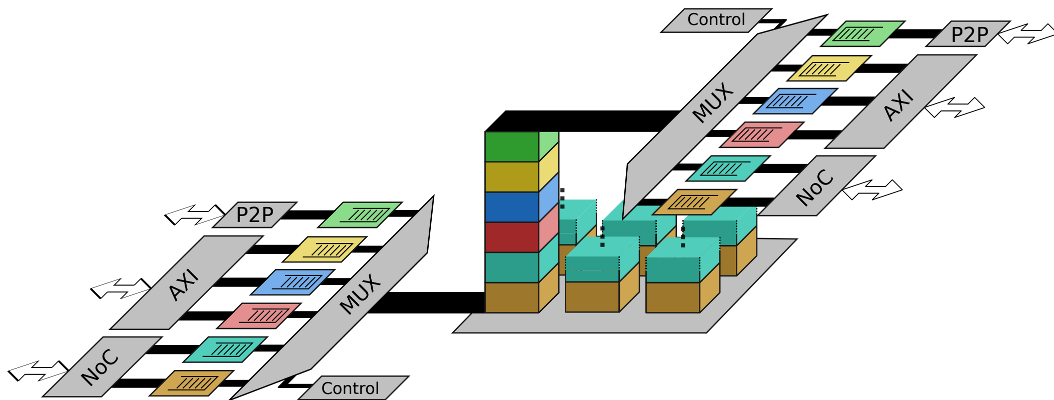


Fig. 3.1.: Schematic illustration of the TSV-Hub for vertical continuation of multiple conventional on-chip interconnect protocols

Through Silicon Vias (TSVs) are the most promising solution for realizing inter-layer connections in 3D-ICs. Although not many commercial applications can be found at the time of writing, it can be foreseen that, once the manufacturing process has matured, TSVs will play an important role in future 3D-ICs due to the high bandwidth provided. The evolution currently seen in the domain of stacked DRAM, where first products are deployed and standards have been defined, will most likely be repeated in the domain of System-on-Chip (SoC).

As already covered in more detail in Chapter 2.2, TSVs are costly with respect to area, yield and manufacturing. Therefore, TSVs should be treated as valuable resources in a 3D-IC, and designers are advised to fully exploit their capacity. This can be achieved by operating TSVs at an increased clock speed and use multiplexing and serialization techniques to supply a continues high-speed data stream.

To this end, the TSV-Hub was developed in the scope of this work. It is a highly configurable IP-Core that efficiently wraps and operates a TSV array in order to keep the required number of TSVs low and to operate them at the edge of their capacity. Additionally, it provides means for coping with TSV defects for yield and reliability improvement. On the protocol level, it can connect to existing on-chip interconnect protocols or can be incorporated in a Network-on-Chip (NoC). Fig. 3.1 shows a schematic illustration of the TSV-Hub.

To conduct experiments and prove the feasibility of the concepts researched in this work, a flexible and configurable IP-Core for supporting inter-layer connec-

tions within 3D-ICs was developed. This so-called *TSV-Hub* was designed with the following objectives:

- Reduce TSV count to optimize cost and footprint
- Provide interfaces to conventional bus protocols to facilitate migration to a 3D environment
- Provide fault tolerance against defective TSVs
- Facilitate creation of 3D-NoCs with economic use of TSV and logic resources

This chapter first covers the efficient use of TSVs with serialization and multiplexing techniques in general. Then it is researched how such methods can be used to vertically continue conventional bus protocols. Next, the architecture of the TSV-Hub is presented together with implementation details. Finally experimental results are given and discussed.

3.1 Efficient Use of TSV Resources

As a consequence of their geometry, TSVs exhibit different electric properties compared to conventional BEOL wiring in the chip's metal layers. The TSV length depends on the thickness of the thinned waver and can be as low as $70\,\mu\text{m}$ [86]. The TSV diameter also depends on the wafer thickness since the TSV's maximum aspect ratio sets the lower bound on TSV width. This aspect ratio is usually below 10, depending on the manufacturing process and the filling material. In Chapter 2.2.1, it was outlined how electric parameters can be extracted by using lumped models. Due to their shortness, TSVs show very low capacitance values (in the range of multiple tens of femto-farad [86]) and also low resistance values (in the range of a few tens of $\text{m}\Omega$ [86]). When using such values as an input to a delay model (see Chapter 2.2.1), this yields very short delays in the range of femto-seconds. Hence, theoretically, TSVs can be operated at much higher clock frequencies than the conventional circuitry, which forms the limiting factor. Thus, fully exploiting the TSV capacity of a single TSV is next to impossible but clock frequencies can be significantly increased (compared to the global interconnect clock) in small islands wrapping TSV arrays.

With an increased TSV clock, the TSV's capacity is also increased. Therefore, to benefit from this capacity a suitable data stream has to be provided. This can be achieved by serialization and multiplexing methods.

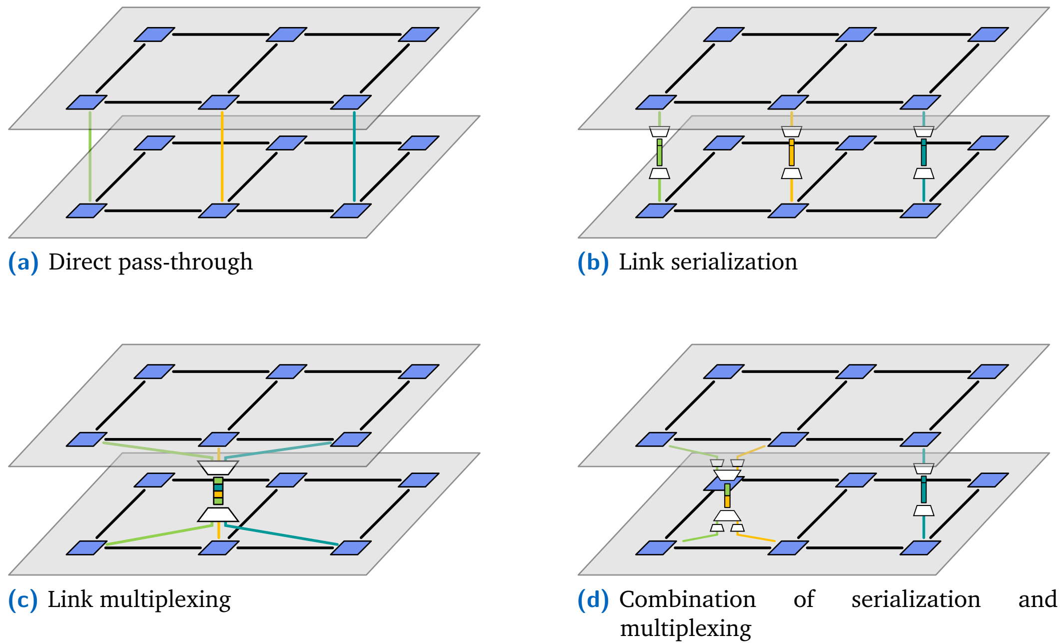
3.1.1 Serialization and Multiplexing of Data Streams over TSV Arrays

When an array of TSVs is operated at a higher clock frequency (compared to the intra-layer interconnect clock), a data stream has to be provided to serve the increased throughput, or the inter-layer-link has to have a reduced bit-width compared to its intra-layer counterpart.

Therefore, the following options are available to support a TSV speed-up:

- **Serialization:** A single link is made narrower (adapted to a smaller word size)
- **Multiplexing:** Multiple links are aggregated, and their individual data words are transferred in TDMA style
- a combination of both methods

For both principles, the required logic elements are multiplexers, and from the perspective of a single TSV there is no difference. It will be connected to a multiplexer in any case and map either wires from the same link or from different links on said TSV following a TDMA schedule. Nevertheless, we will refer to the first as *link serialization* and to the latter as *link multiplexing* below. This difference is illustrated in Fig. 3.2.


(a) Direct pass-through

(b) Link serialization

(c) Link multiplexing

(d) Combination of serialization and multiplexing

Fig. 3.2.: Inter-layer continuation of a 3D-mesh NoC. (a) direct pass through without any TSV clock speed-up; (b) link serialization: data stream for a TSV array is formed from a single link; (c) link multiplexing: data stream for a TSV array is formed from multiple link sources; (d) combination of serialization and multiplexing: data streams from multiple links are first serialized and then multiplexed.

Both methods reduce the number of required TSVs to continue the link(s) but also add additional logic for multiplexing and control. The objective with the proposed method is to achieve a total area (TSVs + logic) that is smaller compared to the TSV area required for direct continuation with a dedicated TSV per wire. The resulting total area depends on both the properties of the used standard cell technology and the TSV properties. With a combination of current technologies an area reduction is achievable (see Chapter 3.3.3.1).

The two main factors are:

- The process node used for the multiplexer logic
- The minimal TSV pitch of the 3D integration technology used

In addition, further constraints of the 3D technology might be relevant, e.g. Keep-out-Zones (KoZs) around TSV(-arrays), and whether placement of logic between TSV arrays is allowed or not.

The next question that arises is whether link multiplexing has an advantage over pure serialization. The problem is illustrated below with a quantitative example. Imagine a two-layer setup with four inter-layer links which are 8-bit-wide each. An intra-layer interconnect clock of $f_w = 500\,\text{MHz}$ is assumed. This results in a required per link bandwidth of $B_l = f_w \cdot m = 4\,\text{Gbit/s}$, hence an overall inter-layer bandwidth of $B = 16\,\text{Gbit/s}$.

The straight-forward solution is to connect each wire to a TSV, thus resulting in 32 TSVs. This pass-through configuration is shown in Figure 3.3. The TSVs in this case are clocked at same speed as the intra-layer interconnects ($f_{tsv} = f_w = 500\,\mathrm{MHz}$), and there is no need for adaptation logic.

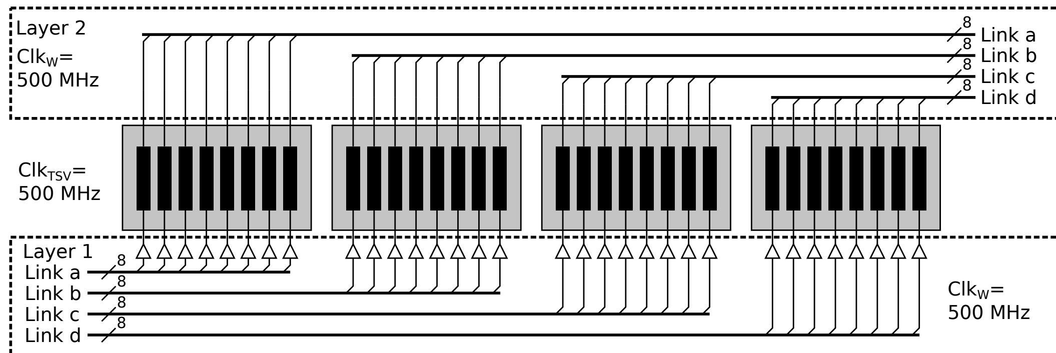


Fig. 3.3.: Simple pass-through continuation of four 8-bit-wide links without speed-up in the TSV layer

Assuming now that a TSV clock of $f_{tsv,max} = 2\,\mathrm{GHz}$ is feasible, the TSV count can be reduced to $n = 8$ to still cope with the bandwidth requirement of $B = 16\,\mathrm{Gbit/s}$. This can be achieved by multiplexing or serialization.

With pure serialization, each link is reduced to just two TSVs with a serialization rate of $S = \frac{f_{tsv,max}}{f_w} = 4$. The logic to support this serialization is formed by two 4-to-1 multiplexers per link, hence a total of eight 4-to-1 multiplexers on the sending side. On the receiving side, eight register cells per link are needed. This is illustrated in Fig. 3.4. Note that all four links operate independently in this case, all wires connected to a specific multiplexer are taken from the same link.

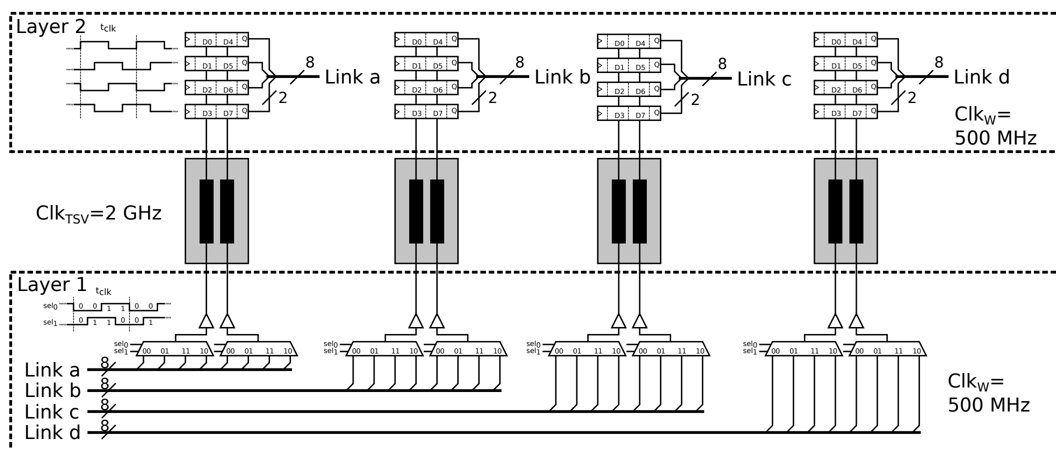


Fig. 3.4.: Serialization of four 8-bit-wide links with increased TSV clock rate

With a pure multiplexing solution, a single TSV array with eight TSVs is created such that the array has the same width as each of the links. The individual links are mapped consecutively on the array, such that a full round is accomplished in one clock cycle of the interconnect clock. To this end, eight 4-to-1 multiplexers are

required and also eight register cells per link on the receiving side. In contrast to the serialization solution, the multiplexers are connected to wires from different links. This is illustrated in Fig. 3.5

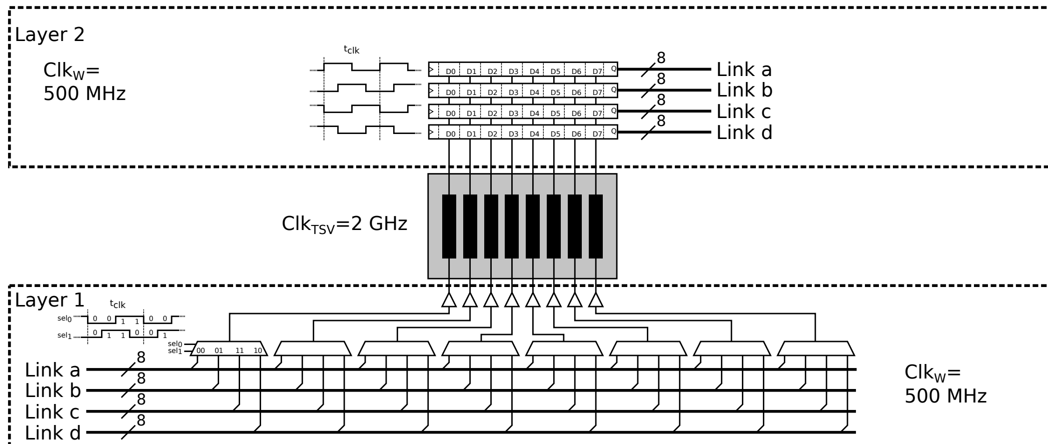


Fig. 3.5.: Multiplexing of four 8-bit-wide links over a single TSV bundle with increased TSV clock rate

In each case we need eight 4-to-1 Multiplexers and 32 register cells, plus additional control logic, which is not shown in the figure but is also similar in complexity for both cases. Therefore, we trade the saved 56 TSVs for this logic. Whether this is beneficial depends on the technology parameters.

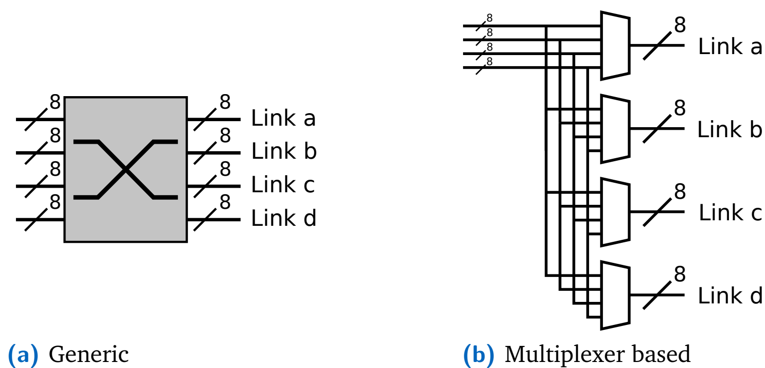
Conclusion: Both link multiplexing and serialization require the same logical overhead for the same level of TSV count reduction. Multiplexing is not advantageous over pure serialization as long as a fixed schedule is used, such that each multiplexed link constantly receives the full bandwidth share.

Since placing the TSV bundles in the serialization case offers more freedom than placing one large bundle and the wiring is better decoupled, one would generally select link serialization in the case of fully independent links. This changes as soon as any knowledge about temporal dependencies on the traffic carried by the individual links is available.

Such dependencies can be derived from both, either a high-level application view or a low-level architecture view. For the high-level view, consider for instance two CMOS image sensors delivering a constant bitstream each, that is constantly evaluated. Link multiplexing would not be advantageous here over plain link serialization of the individual links. Now consider the two image sensors being the front and rear camera of a smartphone. The data streams of those two could easily be multiplexed. The TSV array would be oversubscribed (in terms of bandwidth) since using both cameras at the same time is not intended. For the low-level example consider multiple slaves in a single master multiplexed bus. By definition, only one slave is active at any given time.

What is more, a correlation factor describing such relations is not limited to 0 and 1 (no temporal relation vs. full temporal relation) but can also express a likelihood that the two links are exchanging data at the same time. Together with additional buffering, statistical multiplexing can then be applied. This is outlined later (Chapter 3.2.2.2.2).

Link multiplexing can also be beneficial if the transported links are not fully independent. This can be the case if the links represent the outputs of a switching structure. In such a case, the TSV multiplexers can partially take over the switching task of the switching structure.



(a) Generic (b) Multiplexer based

Fig. 3.6.: 4-port 8-bit crossbar

Fig. 3.6 shows how an 8-bit 4-port crossbar is constructed from dedicated multiplexers. For each of the four ports an 8-bit-wide 4-to-1 multiplexer is needed. One of these multiplexers again contains eight 4-to-1 single bit multiplexers. When comparing this with the lower part of Figure 3.5, it can be seen that this is exactly the same logic as one of the multiplexers of the crossbar. Considering now the 4-times speed-up of the TSV array, four data words can be transported in a single $500\,\mathrm{MHz}$ clock cycle. The only part missing so far is a more flexible assignment of the receiving logic. This can be easily achieved by dynamically rearranging the clock sequences of the output logic's registers. It can therefore be concluded that a TSV array with a wrapping link multiplexing logic inherently contains a full crossbar that can be enabled by making the control logic more flexible.

3.1.2 Vertical Continuation of On-Chip Bus Protocols

Although the transition from shared media buses to NoCs is often labeled as a paradigm shift, it has been rather a gradual evolution from bus-based solutions with many interconnect schemes residing somewhere in between of the two paradigms.

Because of backwards-compatibility requirements, using the newest and most sophisticated interconnection subsystem available is often not an option since older IP-Cores must still be supported, and IP-cores are often re-used from existing designs

or licensed from third parties. In many applications, we find a combination of different interfaces with adaptation layers in-between. It can be concluded, that because of this aspect, even in future 3D-IC based SoCs, we will find older bus interfaces for enabling subcomponent re-use.

In this section, it is studied how such interconnect structures can be re-used in 3D-ICs, how they can be transparently continued to other chip layers and how the impact of TSVs can be minimized.

3.1.2.1 Shared Media Buses

Most widely used on-chip buses were originally drafted as shared media architectures. Hence, originally, there was a set of wires that was physically *shared*, and an arbiter granted exclusive access to a Master/Slave pairing for the duration of a transaction.

Both multiplexing and serialization are no viable options for the vertical continuation of shared media buses over TSVs. The reason for this is that both require register stages to be inserted in the signal path. If a bus has to be realized as a shared media system in a 3D-ICs, a TSV has to be inserted for every wire of the bus.

However, shared media buses are deprecated in modern SoCs. This is due to several aspects:

- A high capacitive load on driving logic and high power consumption [169]
- Signal delay problem (no possibility to insert register stages)
- Special tri-state gates are needed and are hard to tackle with a standard cell process

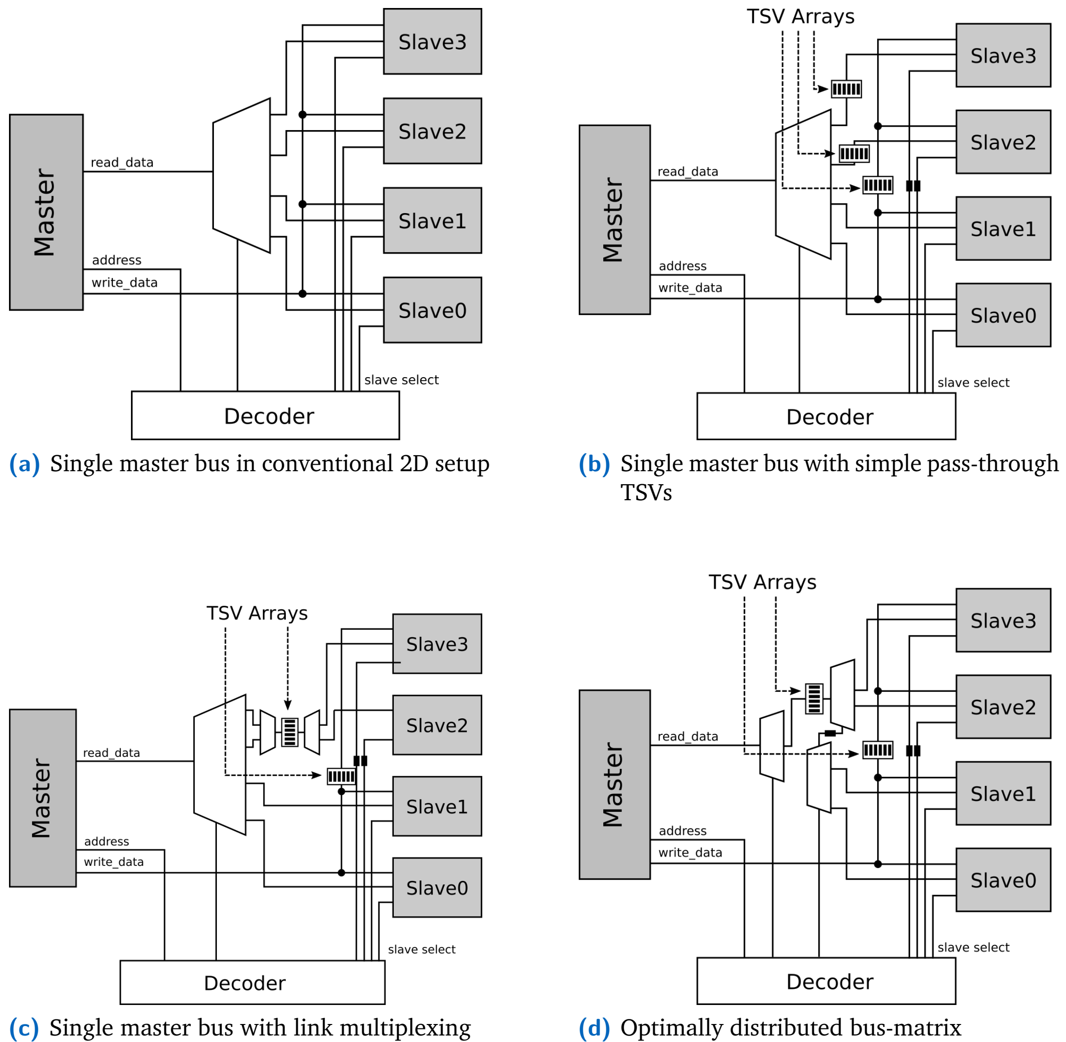
Due to the above reasons, shared media buses are not expected to play a significant role in 3D-ICs. Instead, buses are implemented in a multiplexed fashion in modern SoCs.

3.1.2.2 Multiplexed Buses

Multiplexer based bus-matrices are the state-of-the-art interconnect fabric in modern SoCs and also FPGAs. They avoid problems found in shared media architectures and are well suited for an implementation using a standard cell or FPGA flow.

In a standard single master bus system (e.g. AHB-Lite [122]) the master's write data signal is delivered to every slave. The read data signals from the slaves are aggregated in the bus-matrix which selects and passes the currently valid signal to the master. The multiplexer contained in the bus-matrix is controlled by a *decoder* selecting the currently active slave subject to the current address and predefined address ranges. Fig. 3.7a shows such a setup for a conventional 2D system.

Fig. 3.7b to 3.7d show the same setup but here it is assumed that Slave 2 and Slave 3 are located on a second chip layer in a 3D-IC.



(a) Single master bus in conventional 2D setup

(b) Single master bus with simple pass-through TSVs

(c) Single master bus with link multiplexing

(d) Optimally distributed bus-matrix

Fig. 3.7.: 3D Single Master Bus

The naive approach to adapt this setup for the 3D environment is a pass-through solution where a TSV is inserted for every wire that is supposed to cross the layer boundary. This is depicted in Fig. 3.7b. This solution does not induce a logic overhead but requires a high, possibly infeasible, amount of TSVs, especially for more complicated setups were individual slave links traverse more than one layer boundary. Hence, even more than one TSV would possibly be required per wire of a single link.

The scenario of keeping the full bus-matrix on a single layer seems unrealistic at first but might be the only viable option if the matrix IP is provided as a macro and therefore cannot be split. But even in this case, means can be taken to reduce TSV count since the temporal correlation of the individual links is known. It can be safely assumed that only one slave link is active at a time and all others lay idle. Therefore,

link multiplexing is beneficial in this case since not the full aggregated bandwidth of all links has to be provided by the TSV array. In this case, only the capacity of a single link has to be provided. This setup is shown in Fig. 3.7c. It looks odd at first glance since the function of the multiplexer in the bus-matrix is partly reversed, however, this is the only option if the bus-matrix cannot be split.

Now let us assume that the description of the bus-matrix is available on RTL level and can be split and distributed to multiple chip layers. In the above example, this means that a part of the bus-matrix' multiplexer is moved to the second layer as depicted in Fig. 3.7d. A hierarchical multiplexer structure is formed with TSV arrays in-between the stages. This method achieves the same reduction of TSV count as the *link multiplexing* method of the previous paragraph but does not add logic overhead (or only slightly, depending on the available standard cells).

Note that no serialization is applied to this point in all methods, and there is not yet any clock speed-up on the TSV arrays considered. With serialization, TSV count can be further reduced.

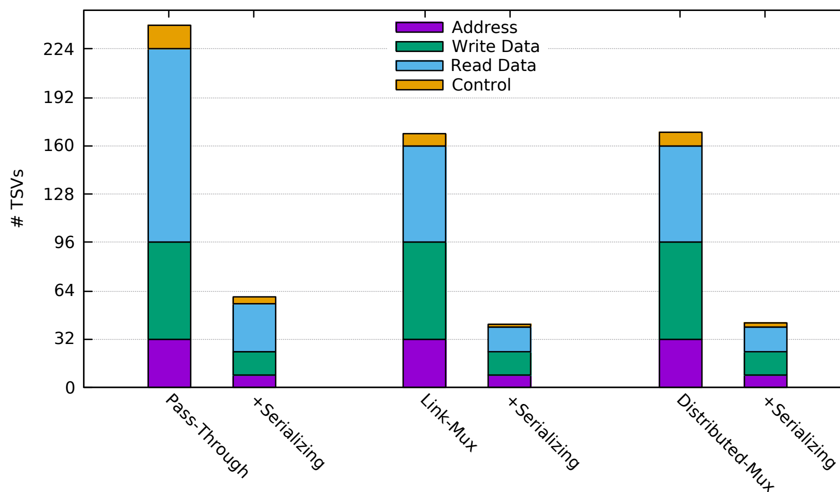


Fig. 3.8.: TSV count for different 3D implementations of a generic single master bus and four slaves

Numerical example: Assume a generic single master bus with an address width of 32 bit and a data width of 64 bit. Furthermore, 4 bit of control signaling in each direction are assumed at the slave interface. For the scenario above, the data TSV count is halved with both *link multiplexing* and *distributed multiplexing*. Assuming a 4-times clock speed-up at TSV level reduces all types of TSVs by a factor of 4, for the cost of additional logic. A diagram comparing the TSV count for the different scenarios is given in Fig. 3.8.

3.1.2.2.1 Automated Generation of 3D Multiplexer Trees

In the previous section we have shown that it is possible to distribute a bus matrix to multiple layers in a 3D-system and thereby reduce TSV count (compared to a concentrated bus matrix). Cutting the bus multiplexer and distributing it to the individual layers gets more challenging for larger systems with multiple layers and many slaves. Here we show calculations for the TSV reduction rate and present an algorithm for the automatic generation of multiplexer trees.

For creating the optimal distribution we make use of the fact that every multiplexer can be divided into a tree of smaller multiplexers (up to the point of a tree of only 2-to-1 multiplexers). We take all slaves (forming the leaves of the tree) that are located on the same layer and connect them to a common multiplexer which is placed on the same layer as the slaves. Next, the multiplexers have to be chained together to reach the master, and a TSV array has to be inserted at every layer crossing. Algorithm 1 shows a greedy algorithm for accomplishing this task. It can be divided into three steps: First, it creates and interconnects the multiplexers and TSV arrays for all layers below the master layer, starting from the lowest layer. Next, it does the same again for all layers above the master layer starting from the highest layer. Finally, the master layer is interconnected.

For the automated generation of such bus trees, the *MuxCutter* software tool was developed in the scope of this work. Fig. 3.9 shows an example that was implemented with this tool. It shows the read data path of a single master bus with 11 slaves, with the slaves being placed on different layers (0 to 3) and the master being placed on layer 1 (the second layer from the bottom). The resulting multiplexer tree is given in Fig. 3.9b. The final result with inserted TSV arrays is shown in Fig. 3.9c.

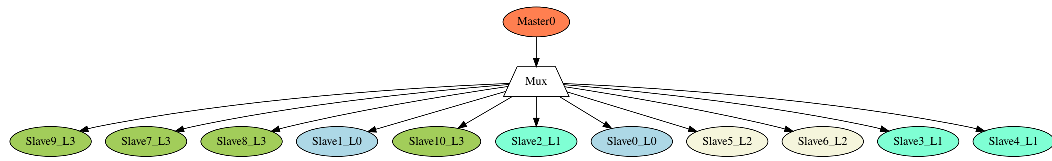
To calculate the required TSV resources first the bus has to be characterized with respect to the following properties:

- write data width w_{wd}
- read data width w_{rd}
- address width w_a
- number of master to slave control signals w_{m2sc}
- number of slave to master control signals w_{s2mc}

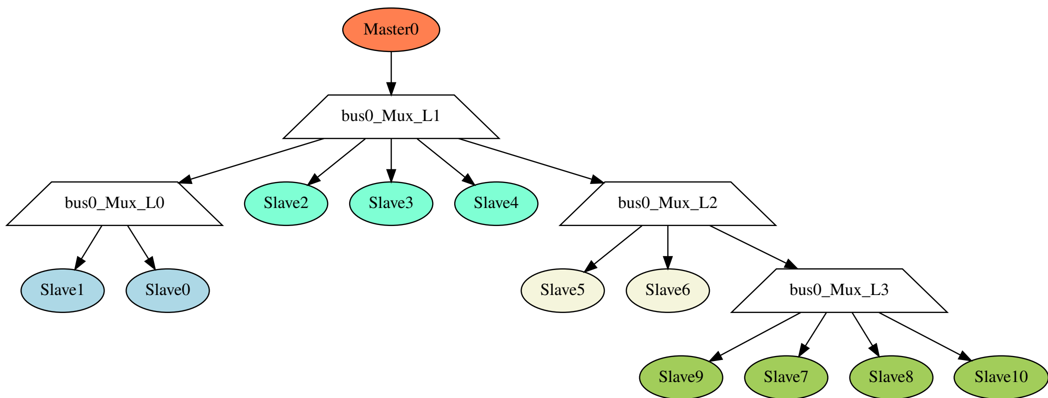
Write data width and read data width are generally of the same size but they are treated separately here since only the read data has to be routed through the multiplexer tree.

Furthermore, the following aspects of the IC stack have to be known:

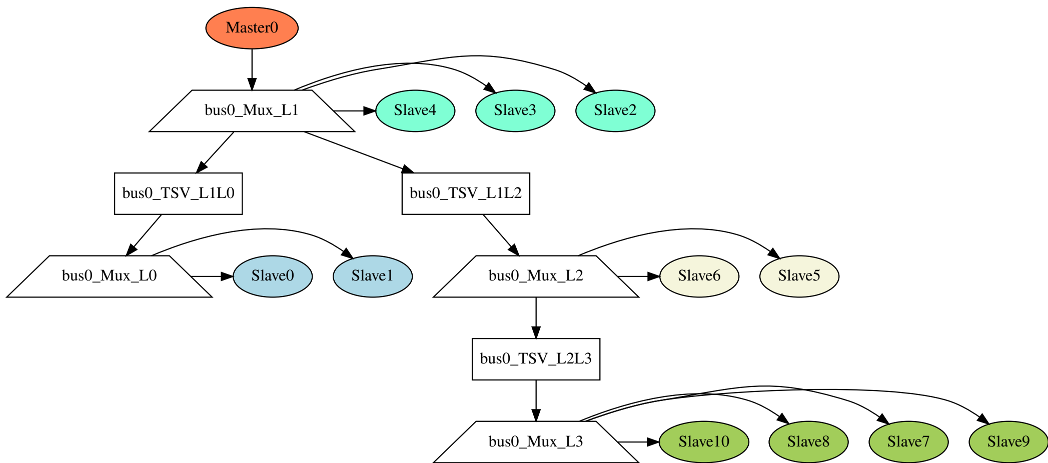
- Lowest layer containing slaves l_l
- Highest layer containing slave l_h
- Layer containing the master l_m



(a) Read Data path of single master bus with 12 slaves



(b) Mux-Tree with dedicated Multiplexer for each layer



(c) Mux-Tree with dedicated Multiplexer for each layer and TSV arrays for inter-layer connections

Fig. 3.9.: Different stages of bus multiplexer tree generation for 3D-ICs

The master-to-slave signals (write data, address, master to slave control) are independent from the selected slave and have to be transported to every layer. Therefore, these signals always create a TSV demand of:

$$n_{m2s} = (l_h - l_l)(w_{wd} + w_a + w_{m2sc}) \tag{3.1}$$

regardless if a central or distributed bus-matrix is used.

The TSV count required to implement the slave-to-master signaling, however, depends on the bus-matrix realization.

For the central matrix, we get:

$$n_{s2m,c} = (w_{s2mc} + w_{rd}) \sum_{s \in \mathcal{S}} |\mathrm{L}(s) - l_m| \tag{3.2}$$

where $\mathcal{S}$ denotes the set of slaves in the system and $\mathrm{L}(s)$ gives the layer on which a specific slave s is located. Also, it is assumed that the bus-matrix is located on the same layer as the master.

For the distributed matrix, the TSV count can be written as:

$$n_{s2m,d} = (l_h - l_l)(w_{s2mc} + w_{rd}) \tag{3.3}$$

With the TSV counts for the different implementation styles, the TSV reduction rate r can be calculated:

$$r = \frac{n_{m2s} + n_{s2m,c}}{n_{m2s} + n_{s2m,d}} = \frac{(l_h - l_l)(w_{wd} + w_a + w_{rd}) + (w_{s2mc} + w_{rd}) \sum_{s \in \mathcal{S}} |\mathrm{L}(s) - l_m|}{(l_h - l_l)(w_{wd} + w_a + w_{rd}) + (l_h - l_l)(w_{s2mc} + w_{rd})} \tag{3.4}$$

A widely used single master on-chip bus system is the AMBA AHB-Lite bus. Its architecture resembles the generic bus illustrated in Fig. 3.7 and uses 32-bit addressing and a configurable data width, which can be any power of 2 between 32 and 1024 bit. We assume a data width of 64 bit below. Hence, $w_{wd} = w_{rd} = 64$ and $w_a = 64$.

The master-to-slave control signals (`HBURST[2:0]`, `HMASTLOCK`, `HPROT[3:0]`, `HSIZE[2:0]`, `HTRANS[1:0]`, `HWRITE`) sum up to 14 bit, the slave-to-master signaling (`HREADYOUT`, `HRESP`) totals 2 bit. Hence, $w_{m2sc} = 14$ and $w_{s2mc} = 2$.

This gives a TSV count of $n_{m2s} = 3 \cdot 110 = 330$ induced by master to slave signaling.

For the slave to master signaling the centralized variant yields a TSV count of $n_{s2m,c} = 66 \cdot 2 \cdot 1 + 66 \cdot 2 \cdot 1 + 66 \cdot 4 \cdot 2 = 792$ and the distributed variant requires $n_{s2m,d} = 3 \cdot 66 = 198$ TSVs.

The TSV reduction factor amounts to $r = \frac{330+792}{330+198} = 2.125$

Note that the above calculations include neither the control signals for the multiplexers nor the slave select signals. The AHB bus uses a centralized address decoder, which is why a dedicated slave select signal has to be delivered from the decoder to every slave. Assuming that the address decoder is located on the same level as the master, this adds another $n_{sel} = \sum_{s \in \mathcal{S}} |\mathrm{L}(s) - l_m|$ TSVs, regardless whether the bus-matrix is distributed or centralized. In the above example $n_{sel} = 2{\cdot}1{+}2{\cdot}1{+}2{\cdot}4{\cdot}2 = 12$ TSVs would be added.

The select signals for the multiplexers are only required in the distributed case and depend on the size of the individual multiplexers. The size of a multiplexer is the number of slaves located on the same layer plus any TSV arrays connected to it. In the case of the above example this adds $n_{sel,mux} = 1 + 1 + 2 \cdot 2 = 6$ TSVs. However, the multiplexer select signals can also be derived from the slave select signals.

3.1.2.2.2 Serialization and Performance

Restructuring and optimizing a bus-matrix for 3D-ICs already reduces TSV count. More savings can be achieved by including clock speed-up at the TSV layer. A speed-up can be added through serialization and can further significantly reduce TSV count. However, there is one serious caveat: Most bus standards do not allow register stages to be added in the individual links, since there are strong timing restrictions on the correlation between address and data bus. Register stages are mandatory at a TSV array where serialization is applied, since the deserialized data has to be stored in a register on the receive side.

The AMBA AHB bus, for instance, incorporates a two-stage pipeline. This means that in the last cycle of a transfer the next address can already be put on the address bus. However, an overlap of more than two transactions is not possible. A slave can insert wait cycles, though, a functionality that was originally intended for slaves to signal to the master that additional cycles are needed to complete the request. Fig. 3.10 shows a simplified timing diagram (address bus and read data bus for transactions) where wait cycles are used. With wait cycles, the complete transaction is prolonged including the address phase.

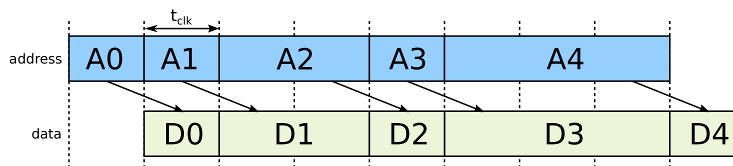


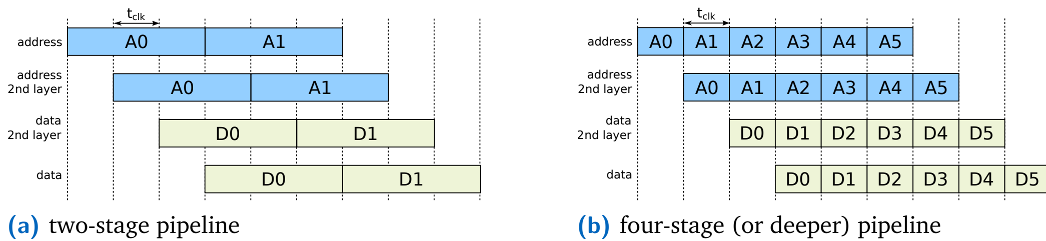
Fig. 3.10.: Address and data bus dependencies

Wait cycles are the only option to cope with additional delays induced by register stages on the address or data bus. In a 3D system applying serialization, a delay of at

Algorithm 1 Create optimal multiplexer tree

```
1: PROCEDURE: CreateMuxTree
2: BEGIN
3: //Layers below master layer:
4: prevElmntsBelow ← 0
5: for l ← 0, l_m − 1 do
6:     n ← S_l + prevElmntsBelow
7:     if n > 0 then
8:         a_l ← CreateTsvArray(l, l − 1)
9:         if n ≥ 2 then
10:            mux_l ← CreateMux(n)
11:            for all s ∈ S_l) do
12:                ConnectSlaveToMux(mux_l, s)
13:            end for
14:            if prevElmntsBelow = 1 then
15:                ConnectTsvArrayToMux(mux_l, a_{l−1})
16:            end if
17:        end if
18:        prevElmntsBelow ← 1
19:    end if
20: end for
21: //Layers above master layer:
22: prevElmntsAbove ← 0
23: for l ← l_max, l_m + 1 do
24:    n ← S_l + prevElmntsAbove
25:    if n > 0 then
26:        a_l ← CreateTsvArray(l, l + 1)
27:        if n ≥ 2 then
28:            mux_l ← CreateMux(n)
29:            for all s ∈ S_l) do
30:                ConnectSlaveToMux(mux_l, s)
31:            end for
32:            if prevElmntsAbove = 1 then
33:                ConnectTsvArrayToMux(mux_l, a_{l+1})
34:            end if
35:        end if
36:        prevElmntsAbove ← 1
37:    end if
38: end for
39: //Master layer:
40: n ← S_l + prevElmntsAbove + prevElmntsBelow
41: if n ≥ 2 then
42:    mux_{l_m} ← CreateMux(n)
43:    for all s ∈ S_{l_m}) do
44:        ConnectSlaveToMux(mux_{l_m}, s)
45:    end for
46:    if prevElmntsAbove = 1 then
47:        ConnectTsvArrayToMux(mux_{l_m}, a_{l_m+1})
48:    end if
49:    if prevElmntsBelow = 1 then
50:        ConnectTsvArrayToMux(mux_{l_m}, a_{l_m−1})
51:    end if
52: end if
53: END
```

least two cycles is inserted when a master queries a slave located on a neighboring layer (one at each deserialization stage, hence on the receive side of the address bus and one on the receive side of the read data bus). Thus, in the resulting timing each bus transaction is stretched to three cycles (Fig. 3.11a). This will degrade the throughput by a factor of three. A desirable timing would be one with a deeper pipeline of more than two stages, like the one depicted in Fig. 3.11b. This is, however, not possible with buses putting fixed timing dependencies between address and data bus. We will see later (section 3.1.3) though, that such a timing is feasible with channel based protocols like the AMBA AXI bus.



(a) two-stage pipeline

(b) four-stage (or deeper) pipeline

Fig. 3.11.: Bus transaction duration with different pipeline depths for a master-slave transaction to a slave on a neighboring chip layer

3.1.2.2.3 Multi-layer bus Architectures

First, it is worth mentioning that the term *layer* in the context of multi-layered bus systems is not to be confused with a chip layer of a 3D-IC.

Multi-layered buses are state-of-the-art in 2D ICs and when applied to a 3D-IC a bus layer can spread over multiple chip layers. Therefore, below, we will use the term *bus layer* when referring to a layer of a multi-layered bus.

Nowadays, multi-master on-chip interconnects are often realized as multi-layer bus architectures (e. g. Multi-layer AHB [170]). Such a bus is constructed by combining multiple single master buses, one for each bus layer. The slaves are then connected to multiple layers. In contrast to classical multi-master systems with a central arbiter (e. g. conventional AHB), such buses are more flexible and enable concurrent transactions to a certain extent.

Originally, a *bus* was defined as a communication system where access to a *shared medium* is granted to a single master-slave pair until a transaction is completed. In most on-chip buses, the shared medium is replaced by a multiplexer structure but still only a single master-slave pair is active at the same time. This is different for multi-layered buses, as they allow up to as many parallel connections as there are *layers* in place.

Fig. 3.12 shows the basic principle of a multi-layered bus with two masters (here each having its own bus layer) and two slaves. The depicted example shows a fully

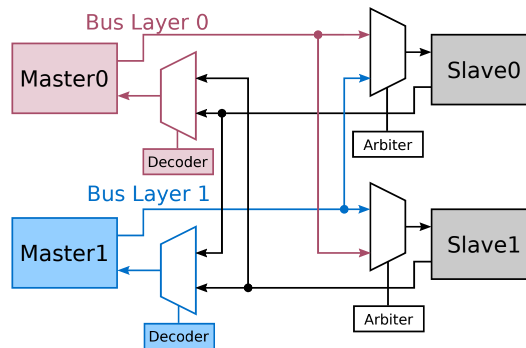


Fig. 3.12.: Multi-layer bus with two masters and two slaves (fully connected)

interconnect configuration. Thus, it resembles a full crossbar, since each of the two masters can communicate with one of the two slaves concurrently (as long as not both masters access the same slave at the same time). Note that there is no central arbiter which controls access to a shared medium, as with classical multi-master systems (see next section), instead arbitration takes place at each slave port separately.

However, crossbars do not scale well with larger port numbers. Therefore, the bus-matrix of a multi-layered bus with high port counts usually is not fully interconnected. Therefore, such a bus ranges between a shared media bus and a full crossbar.

When adapting a multi-layered bus architecture for a 3D scenario, the same approach as with single master buses can be taken, except that the steps have to be repeated for each bus layer.

3.1.2.2.4 Classical Multi-Master Systems

Traditionally, SoCs have used multi-master bus systems like AMBA AHB or IBM's Core Connect as an interconnection backbone. Figure 3.13 shows the principal architecture of such a multi-master multiplexed bus with two masters and four slaves.

Multi-master systems make multiplexed buses more complex with respect to two aspects:

- **More complex multiplexing**: A second multiplexer is required for the write-data path, address and master-to-slave control signals.
- **Arbitration:** An arbiter is required to resolve conflicts when multiple masters try to access the bus at the same time. Arbitration can be complex and not straight forward. There are many arbitration schemes with different objectives regarding overall throughput, fairness, quality of service, delays, etc.

For the 3D scenario, the first of the two points listed above can be addressed in a similar manner as with single-master bus systems. One or two additional multiplexer

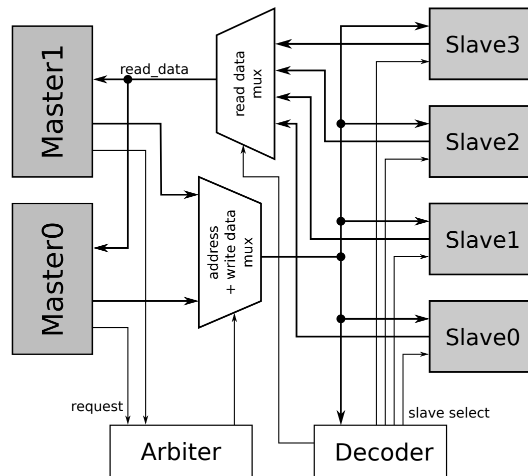


Fig. 3.13.: Multi-master bus with two masters and 3 slaves

trees are built in the same way as the multiplexer tree for the read data, with the difference that the root of the tree is on the level where the arbiter is located and the leaves of the tree are formed by the masters. The second point, however, can be more troublesome. The problem here is that an arbiter usually cannot easily be split-up into smaller chunks and distributed on several layers, since it needs to collect, maintain and compare information on all of the participating masters. In general, among all bus systems, classical multi-master systems are least suited for adaption in 3D-ICs.

Luckily, such bus systems have been mostly replaced by either multi-layer based variants or channel based variants (which are covered in the next section).

3.1.3 Channel based interconnects

The main advantage of such systems is that they allow additional register stages being placed in the channel's path. This makes them a suitable candidate for serialization in 3D-ICs.

Channel based interconnect systems make use of independent channels for executing bus transactions. *Independent* means that there are no strict timing constraints, e.g. that the address must be held valid on the address bus until the transaction has been finished. Channel based protocols already provide many of the advantages of NoC solutions, however, they are not packet based but still circuit switched.

The most famous channel based protocol is the AMBA AXI (Advanced eXtensible Interface) bus protocol [23]. It uses five independent channels as depicted in Fig. 3.14 and Fig. 3.15.

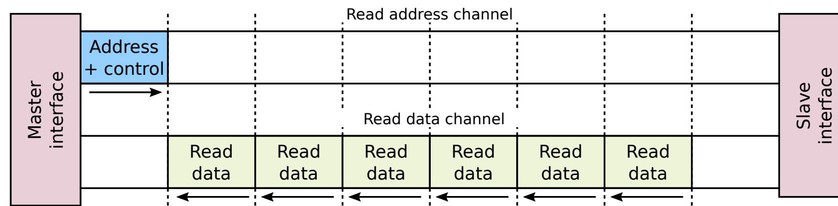


Fig. 3.14.: Axi protocol read channels [23]

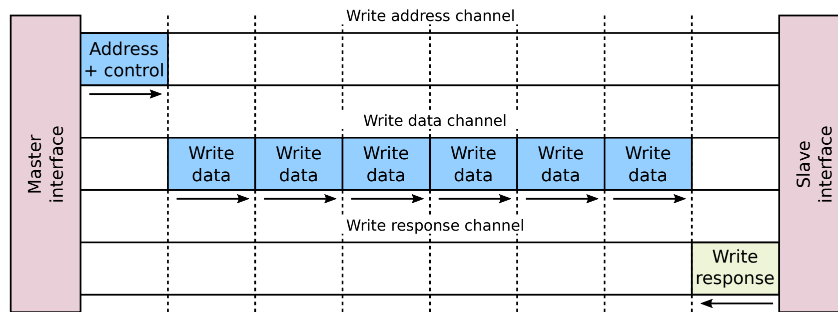


Fig. 3.15.: Axi protocol write channels [23]

Tab. 3.1.: Required AXI channel wires for different bus sizes

Channel	n Bit	32 Bit	64 Bit	128 Bit
Read Address (W_{raddr})	49	49	49	49
Write Address (W_{waddr})	55	55	55	55
Write Data (W_{wdata})	$n + 10$	42	74	138
Read Data (W_{rdata})	$n + 8$	40	72	136
Write Response (W_{wresp})	9	9	9	9
Sum	$2 \cdot n + 131$	195	259	387

The key properties of a channel are:

- **Unidirectional:** All signals of a channel have a common source and common destination and operate only in this direction.
- **Timing invariant:** All channels operate separately, and there are no timing dependencies between the individual channels. Consequently, adding register stages at any point in the channels is possible without violating timing restrictions.

The AXI interface is part of the third generation of the AMBA specification and provides significant improvements with respect to implementation flexibility, but has also some drawbacks compared to conventional buses due to its sheer amount of required wires. This is mainly due to separate read and write address channels, but also to a large number of control signals. Tab. 3.1 shows the general channel widths as a function of the data width n as well as the concrete numbers of required wires for a 32-bit, 64-bit and 128-bit implementation.

In general, when building 3D bus interconnect modules for the AXI protocol, the same principles can be used as with conventional protocols. However, a major advantage is the tolerance towards additional register stages. Therefore, AXI interconnects do not necessarily suffer from performance degradation when adding registers before or after TSV arrays. Of course there will be a latency penalty of a clock cycle for each added register stage, but the overall throughput of an AXI link can reach the original level even when TSV clock speed-up is used.

To further improve the inter-layer transport of AXI links, it is worth to have a closer look on AXI's address channel. Here, we find a major difference compared to conventional buses: For burst transfers, only the start address is sent over the channel. This means, in case of a long burst, the address channel lays idle most of the time. After transferring the initial address word no further traffic is conveyed over the address channel.

Consequently, the net information transferred over the bus depends on the burst size, or, when observing a longer time frame, on the distribution of burst sizes. The highest utilization will be observed for single word random access transactions, the lowest utilization for subsequent transfers with the maximum possible burst size.

This can be generalized with the following formulas. The overall net bandwidth of an AXI link can be calculated as follows for the downstream (master-to-slave / write) direction:

$$B_{axi,down} = f_{ic}\left(W_{wdata} + \sum_{i=1}^{N_w} \frac{W_{waddr}}{i} r_{w,i} + \sum_{i=1}^{N_r} \frac{W_{raddr}}{i} r_{r,i}\right) \tag{3.5}$$

and as follows for the upstream (slave-to-master / read) direction:

$$B_{axi,up} = f_{ic}\left(W_{rdata} + \sum_{i=1}^{N_r} \frac{W_{wresp}}{i} \cdot r_{w,i}\right) \tag{3.6}$$

With the interconnect frequency f_{ic}, the maximum occurring burst sizes N_w and N_r and with $r_{w,i}$ being the probability that a write burst is of length i, and $r_{r,i}$ being the probability that a read burst is of length i.

The numbers calculated with (3.5 and 3.6) are the required bandwidth a TSV array has to provide if only the net information carried on the AXI bus is transferred over the array. This, however, requires a dynamic scheduling where the unused bandwidth of idle channels can be reassigned to other channels. This is not covered by the pure link multiplexing and serialization methods described so far. The TSV-Hub, described in the next section, contains such a scheduler.

In the following section, the *TSV-Hub* is outlined as a general purpose IP-core for adapting bus protocols and providing serialization, multiplexing and fault tolerance. Furthermore, it provides an adaptive scheduling mechanism such that protocols like the AXI bus can be continued efficiently. A case study where TSV-Hubs are used to vertically continue AXI links can be found in Chapter 3.3.3.1.

3.2 TSV-Hub Architecture and Implementation

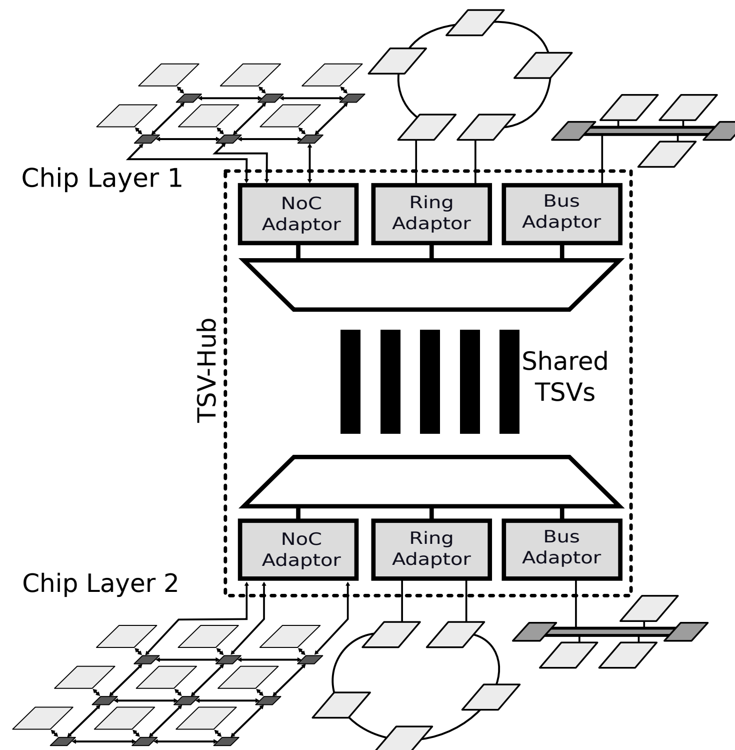


Fig. 3.16.: Vertical continuation of interconnects with a TSV-Hub

3.2.1 Concept

Figure 3.16 shows the basic concept of the TSV-Hub. With a TSV-Hub, multiple (possibly different) interconnect protocols (e.g. bus protocols, NoC protocols) can be transparently continued to other chip layers. To interface with the specific interconnect protocols, protocol adapters are used on the highest level. The protocol adapters use generic abstract virtual links (so-called *VLinks* in the following passages) for transporting the interconnect payload and control data. Such a VLink is a generic synchronous link with a configurable data width and stall/go flow control. The VLinks of a TSV-Hub share the same TSV array for layer crossing. Therefore, the links are multiplexed and possibly serialized before they are transported over the physical array, which is operated at a higher clock frequency compared to the interconnects. This results in significant reduction of footprint and also improves the hub's reliability, since with fewer TSVs, the likelihood of a TSV being faulty is reduced.

3.2.1.1 Clock Domains and Clock Distribution

The TSVs within a TSV-Hub are operated synchronously. However, the hub provides means for clock domain crossing at its interfaces. Therefore, TSV-Hubs are prepared to be used in a *Globally Asynchronous Locally Synchronous (GALS)* [171] environment, where the TSV-Hubs themselves form synchronous islands spanning multiple layers

(Figure 3.17). Synchronizers are required to cross the clock domains at the entry points. The type of required synchronizer depends on the dependencies between the clock signals of both domains. TSV arrays are predestined to form high-speed synchronous parallel links since they can be built in regular patterns and with only small deviations regarding physical parameters. However, a globally synchronous clock distribution in 3D-ICs is not feasible [172].

For fully asynchronous clock signals, dual clock FIFOs are needed, which require a significant amount of logic resources. A promising approach that leads to a simpler synchronizer design is *mesochronous* [173] clocking. In this clocking scheme, all clock signals origin from the same clock source but any amount of clock skew is allowed on a global level. Synchronous design is only enforced within the islands, hence no global H-Tree is required. Consequently, massive phase differences are the result but clock distribution is significantly simplified. This approach is also known as *Globally Mesochronous Locally Synchronous (GMLS)* [174]. Furthermore, we make use of the *clock travels with data approach* where a clock signal is provided alongside the data signals (source synchronous) and serves as a strobe signal on the inputs.

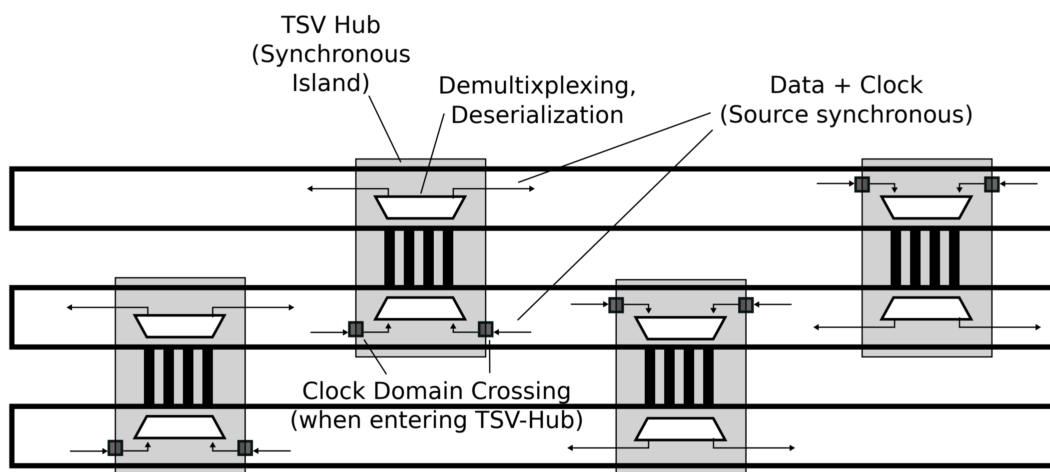


Fig. 3.17.: Multiple TSV-Hubs in a GALS-Architecture

3.2.1.2 Fault Tolerance

Some authors [93], [172], [175], [176] suggest using spare TSVs which lie idle initially and are connected only if a TSV gets faulty. In the proposed approach, additional TSVs are also provided, but the overall TSV array is overdesigned with respect to bandwidth. When TSVs get faulty, the bandwidth will drop, however, it will stay above the lower limit (nominal bandwidth) due to the overdesign.

TSV failures that happen already during manufacturing (e.g. due to mechanical stress) can be "repaired" by rerouting with irreversible techniques, e.g. fuses. To compensate runtime failures, however, active switching logic is needed. Once this switching logic is present, the "spare TSVs" can also be incorporated in the

dynamically provided capacity and do not necessarily have to stay idle until a TSV is faulty. The overall TSV count is split into two parts:

$$n = n_{nominal} + n_{spare} \tag{3.7}$$

with $n_{nominal}$ being the number of TSVs needed to fulfil the nominal bandwidth requirement and n_{spare} denoting additional TSVs leading to an initial overdesign of the array.

If not more than n_{spare} TSVs are faulty, the system can still operate above the nominal bandwidth requirement. When more than n_{spare} TSVs are defective, this capacity is not offered anymore, but the TSV-Hub design can be equipped with a feature that alters the scheduling in such a case, such that guaranteed service links receive priority and are not degraded until even more TSVs get faulty and the sum of all guaranteed service links cannot be provided anymore. In such a case, the system is considered defective. Note that there is no role assignment which defines a specific TSV instance to be spare or nominal. This classification only exists on an abstract level to allow calculations and bandwidth predictions.

3.2.2 TSV-Hub Protocol Stack

To characterize the inter-layer communication in a 3D design we propose three abstraction layers which are named as follows:

- **Physical layer:** The physical TSV connections characterized by TSV geometry and maximum TSV clock frequency. Furthermore, mechanisms to cope with faulty TSVs are incorporated on this level.
- **Virtual Link Layer:** Provides generic virtual links with stall/go flow control transported over the physical TSV array
- **Interconnect Layer:** Interfacing to existing interconnect protocols (e.g. Bus or NoC)

Figure 3.18 illustrates the abstraction levels for the TSV-Hub. In the following sections, a more detailed view on each layer is given.

3.2.2.1 Physical Layer

The physical layer forms the actual electrical inter-layer connection based on TSV technology. For the TSV-Hub, a physical inter-layer link is effectively characterized by four parameters to the next higher level of abstraction:

- The maxim clock frequency (f_{tsv}) the TSV array can be operated at
- The number (n) of TSVs in the array
- The maximum number of faulty TSVs (k_{max}) when switch boxes are used
- The TSV geometry and layout within the array

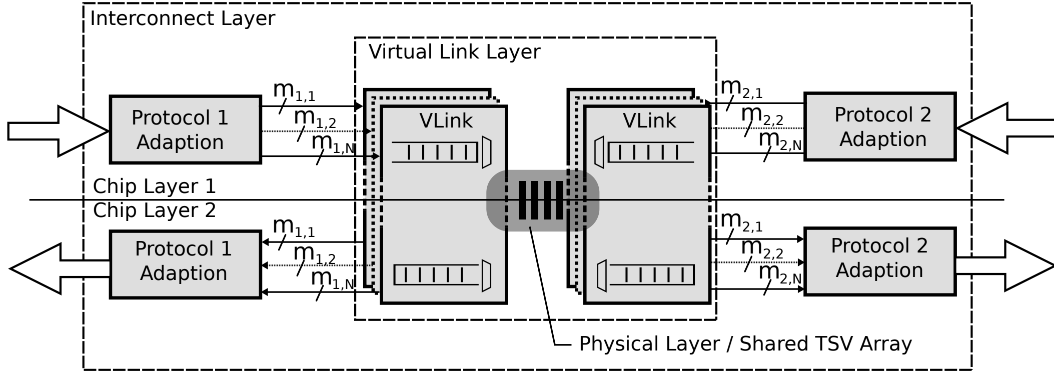


Fig. 3.18.: Abstraction layers in 3D on-chip-interconnect

Effectively many more parameters exist, like material properties, oxide thickness, level of doping of the surrounding substrate and geometric aspects of the individual TSVs. Those parameters, however, serve as input to calculation models, like the lumped model described in Chapter 2.2.1 (based on [75]), and can be abstracted by the resulting capacitance value finally resulting in a propagation delay. This propagation delay again returns the maximum clock frequency.

The raw capacity of a TSV array reads as:

$$B_{tsv} = f_{tsv}\,(n - k) \tag{3.8}$$

with k being the number of faulty TSVs, k_{max} being the maximum number of allowable faulty TSVs and $k \leq k_{max}$.

The nominal capacity reads as:

$$B_{tsv,nom} = f_{tsv}\,(n - k_{max}) \tag{3.9}$$

The total area consumption of a TSV array is a result of multiple parameters, including:

- The diameter of the TSV, usually a result of the maximal aspect ratio and therefore the layer distance
- The diameter of the TSV landing pads (larger pads facilitate the alignment process)
- Surrounding TSV Keep-out-Zones (KoZs) (larger KoZs, hence larger pitch, increases TSV yield [86]).

The maximum number of allowable faulty TSVs k_{max} is relevant for the dimensioning of the switch boxes at both sides of the TSV array. Figure 3.19 shows the principle of the boxes, performing re-mapping of link-signals in case of defects by rerouting $n - k$ signal wires to $n - k$ functional TSVs. Theoretically, it is possible to design switch

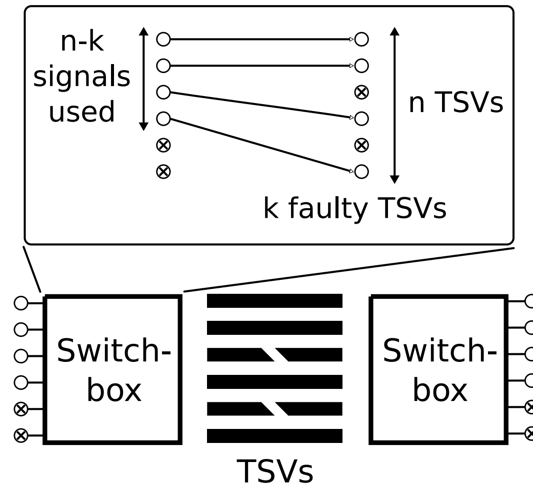


Fig. 3.19.: Principle of switch boxes (sparing defective TSVs)

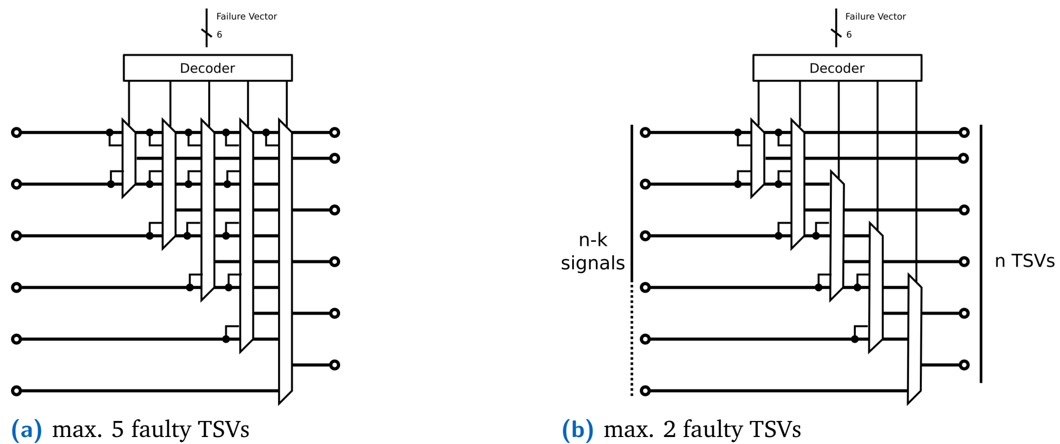
boxes that allow re-routing until only a single remaining data TSV is left, however, this is unrealistic for two reasons: Firstly, such a box (with $kmax = n - 1$) forms a crossbar-like structure and therefore induces a large logic overhead, and secondly, if a large ratio of the TSVs are defective, the remaining capacity would be reduced to an inoperative share of only $1/n$ of the nominal bandwidth.

Therefore, since the number of maximum faulty TSVs will always be limited, we can also limit it already in the hardware implementation making the switch box logic less complex and saving logic resources. To this end, the switch box design requires a parameter k_{max} to be set at implementation time denoting the maximum number of faulty TSVs the switch box instance can cope with. This parameter significantly affects the switch box logic complexity as shown in Fig. 3.20, which gives an example of a switch box design for $n = 6$ TSVs. The multiplexers of the version in Fig. 3.20a with $k_{max} = 5$ have to span more input wires than the ones of the version in Fig. 3.20b which can only cope with $k_{max} = 2$ defective TSVs.

The multiplexers are controlled by a *Failure Bit Vector* $F[0 : n]$, which denotes if a specific TSV is defective (by setting $F[i]$ to 1 for a defective i-th TSV). The gathering of this vector is considered out of the scope of this thesis. It is assumed that appropriate BISTs are in place which check for defects in the product test and regularly during runtime.

3.2.2.2 Virtual Link Layer

On the abstraction level of the Virtual Link Layer, generic virtual links (called *VLinks* in the hereafter) are provided. All *VLinks* of a TSV-Hub are continued through the same TSV array by using a Time Division Multiple Access (TDMA) scheme. Multiple termination types have been implemented and can be used as building blocks to create a TSV-Hub tailored to the requirements of the transported protocols and to cope with the required level of synchronization.



(a) max. 5 faulty TSVs

(b) max. 2 faulty TSVs

Fig. 3.20.: Switchbox architecture

The n TSVs provided by the physical layers are grouped into n_d data TSVs and n_c control TSVs. As their names suggest, data TSVs are responsible for conveying the actual raw data, whereas control TSVs carry control information and flow control data.

The *VLinks* are designed to provide:

1. **Serialization and Deserialization:** The width of the incoming and outgoing data stream is adapted to the number of available physical TSVs.
2. **Buffering:** A configurable number of data words can be buffered to allow statistical multiplexing and back pressuring
3. **Clock Domain Crossing:** Synchronization between TSV and interconnect clock domain

A VLink is configurable regarding the following aspects (amongst other):

1. **Link width:** Number of protocol side data bits (m)
2. **TSV array width:** Number of physical side data bits (number of data TSVs in the array) (n_d)
3. **Clock Synchronization:** The relation of the read and write clocks (synchronous, mesochronous, asynchronous)
4. **QoS level:** The Quality of Service (QoS) level (guaranteed service or best effort)
5. **Buffer size:** The depth of the buffering queue on both input and output side

The termination building blocks are flavored with respect to two dimensions, namely buffering and synchronization. Buffering is required when best effort links are used and back pressuring is applied. For guaranteed service links a simple register is sufficient. However, such a register has to provide a handshaking interface when a clock domain is crossed. The synchronization type determines the synchronization logic to be used. The asynchronous block, which allows the read and write clock

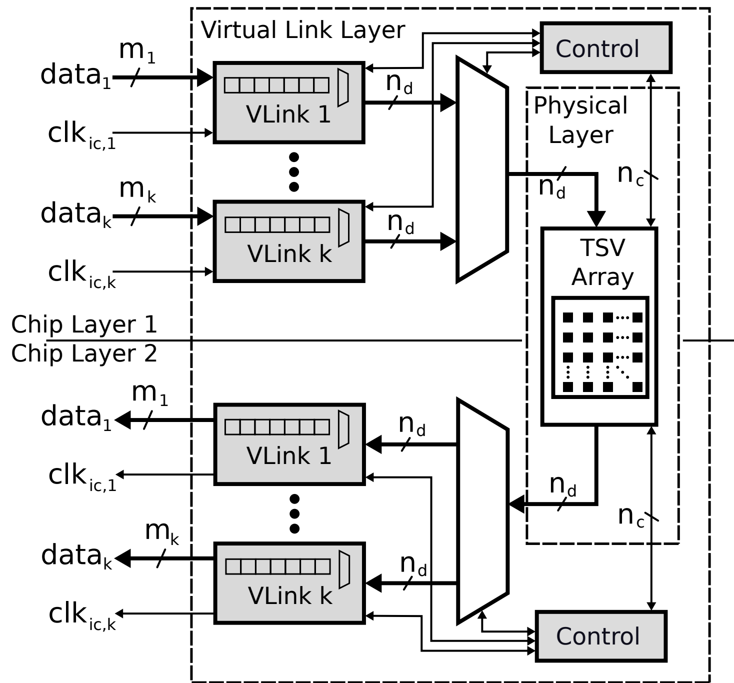


Fig. 3.21.: Multiple virtual links transported with the TSV-Hub

Tab. 3.2.: VLink Termination Methods

		Level of synchronization			
		Asynchronous	**Mesochrounus**	**Synchronous**	**Ratiochronous**
QoS Level	**Guaranteed Service**	Register with Handshaking	Register Handshaking	Register	Register
	Best Effort	Dual-Clock FIFO	FIFO	FIFO	FIFO

to be fully independent, offers the highest flexibility but induces also the highest logic complexity of all synchronization types. The mesochronous blocks can be used if the TSV clock and the interconnect clock share the same clock source, hence show a mesochronous relationship. The mesochronous synchronizers show a smaller logic complexity and also exhibit a smaller delay. The design of the mesochronous synchronizers is inspired by [172]. Obviously, in cases where the TSV-Hub is fully synchronous with the data or destination source clock, no synchronization logic is required at all and even handshake signaling is can be omitted. However, the synchronous design requires a fully synchronous clock distribution network, which is unrealistic for 3D-ICs.

In cases where *VLinks* are continued over more than one layer boundary, directly adjacent TSV-Hubs can be operated synchronously and no synchronization logic is required. This scenario is depicted in Fig. 3.22.

Tab. 3.2 shows the assignment of synchronizers for QoS-level and level of synchronization.

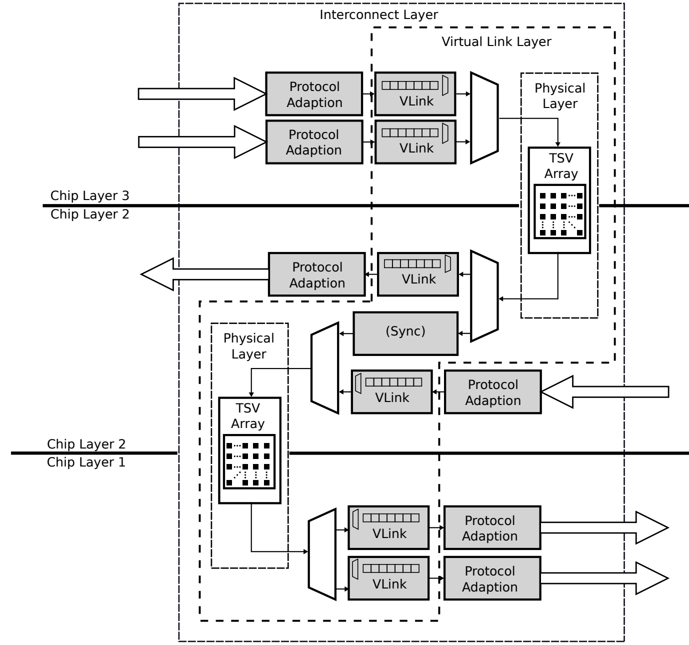


Fig. 3.22.: TSV-hubs in series

3.2.2.2.1 Serialization

An analytical study for serializing a single link inter-layer connection can be found in [105]. We pick-up the naming conventions and extend it for a multi-link scenario were links are first serialized and then multiplexed in the time domain to be transported over a single TSV array.

As depicted in Fig. 3.21, individual links with the data width m are first serialized, such that its word size matches the number of data TSVs n_d. Therefore, the relation of both numbers gives the *Serialization Rate* $S_i = m_i/n_d$. After serialization, the links are transported with a TDMA scheme over the TSV array at a clock rate of f_{tsv}. This TSV clock is usually higher than the clock rates of the individual interconnects $f_{ic,i}$. The ratio of both gives the link speedup $L_i = f_{TSV}/f_{ic,i}$

When multiple links are transported with different data sizes (m_i) and different interconnect frequencies ($f_{ic,i}$), we have to ensure that the TSV array provides the same capacity as the aggregated bandwidth of all links:

$$\sum_{i=0}^{N-1} f_{ic,i} \cdot m_i \leq f_{tsv} \cdot n_d \Leftrightarrow \sum_{i=0}^{N-1} \frac{S_i}{L_i} \leq 1 \tag{3.10}$$

Most of the parameters in (3.10) are fixed. The (maximum) TSV clock is given by the TSV process used, and link clocks and data sizes are given by the existing 2D design

or requirements from a more abstract view on the communication requirements. However, we are flexible regarding the number of TSVs in the array (n_d). Our objective is to keep this number as low as possible for the sake of cost and area savings. Therefore, we resolve (3.10) for n_d and get:

$$n_d \geq \sum_{i=0}^{N-1} \frac{f_{ic,i}}{f_{tsv}} \cdot m_i \Leftrightarrow n_d \geq \sum_{i=0}^{N-1} \frac{m_i}{L_i} \tag{3.11}$$

In (3.10) and (3.11) we assume that the individual links are fully utilized. However, in many cases, links can be idle for longer periods. A good example for such a scenario are the address channels of the AXI protocol. In such cases, it can be meaningful to oversubscribe the TSV array and apply statistical multiplexing such that (3.10) and (3.11) are not fulfilled anymore. However, the used TDMA scheduler has to be capable of handling the idle time frames and assign the "empty" time slots to other links.

Two designs have been implemented and investigated for the serialization logic:

1. A **static serializer** where the input word which is m-bit wide and is divided into $\lceil m/n_d \rceil$ portions which then are serially transmitted. For non-integer serialization rates ($m \mod n_d \neq 0$), the last of the $\lceil m/n_d \rceil$ serialized words per round is partly empty.
2. A **dynamic serializer** which always uses up the available space in all of the serialized portions. For non-integer serialization rates ($m \mod n_d \neq 0$) two consecutive words of the link data stream have to be combined at their word boundaries to form a word of the serialized stream. The logic complexity is higher compared to the static version, but it also comes with the advantage that the serialization rate can be adapted during runtime in the bounds of $(n_d - k_{max}) < n < n_d$. This type of serializer is used when a resilient design is created and fault tolerance is required.

3.2.2.2.2 Scheduling and Multiplexing

The multiplexing of the serialized data streams is carried out in a TDMA fashion. This is possible due to the clock speed-up of the TSV array compared to the intra-layer interconnects. For scheduling the individual streams, a combination of a classic TDMA and a dynamic Time Division Multiple Access (dTDMA) [177] scheduler is applied.

Fig. 3.23 shows the basic schematic of the implemented hybrid scheduler. The lower part forms the TDMA scheduler implementing a classic round-robin scheme. A token (in the form of a single bit) is moving through a shift register in a circular pattern. Each flip-flop embodies one time slot that cannot be skipped. Therefore,

each flip-flop allocates a constant share of the available TSV bandwidth. An output of one of the TDMA flip-flops can therefore be directly assigned to a VLink without the need of further flow control. This configures the link for guaranteed and constant bandwidth QoS-level. A time slot of the TDMA part can, however, also serve as an input to the dTDMA scheduler section to be dynamically distributed among the VLinks. This configures the link for best effort QoS-level.

The dTDMA scheduler is depicted in the upper part of Fig. 3.23. It works in a similar way as the classic TDMA scheduler but allows flip-flop stages to be skipped if there is no data to be transmitted for a certain VLink.

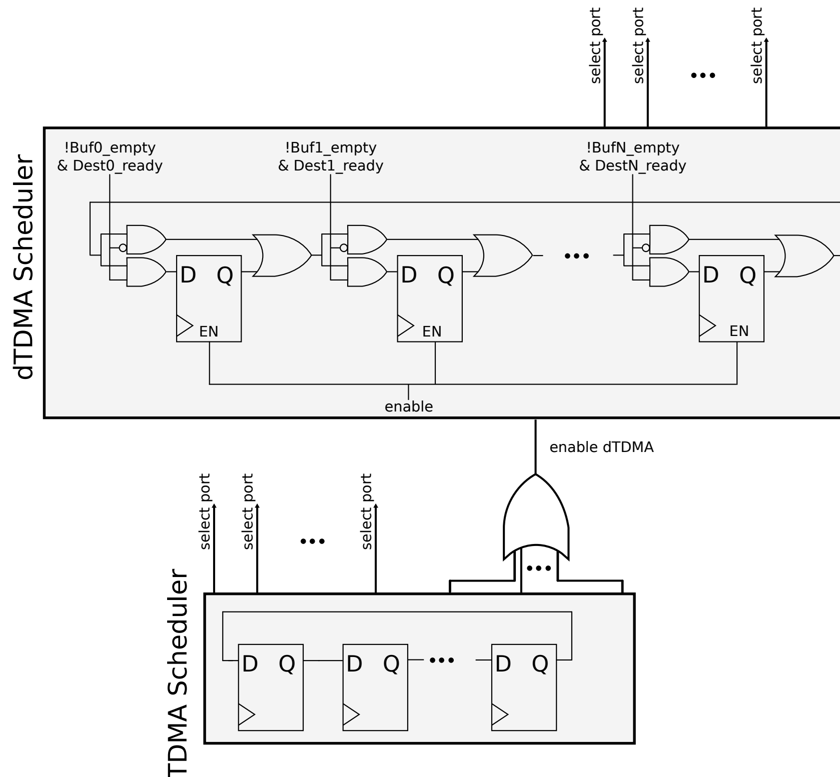


Fig. 3.23.: Hybrid TDMA and dTDMA Scheduler

Note that VLinks served by the dTDMA part of the scheduler also provide a guaranteed bandwidth: the algorithm cannot lead to starvation of links since each flip-flop is guaranteed to get hold of the token within a fixed number of slots as long as there is data available in the respective link. However, this guaranteed bandwidth is usually lower than the link's peak bandwidth since dynamic scheduling is usually only used in cases where statistical multiplexing is applied.

Depending on the configuration vector, it can also occur that only a TDMA scheduler is instantiated (if only constant bandwidth links are used) or that only the dTDMA scheduler is instantiated (when all of the offered TSV capacity can be distributed dynamically to the participating links). If both types are involved, the hybrid scheduler with both a TDMA and a dTDMA section is instantiated.

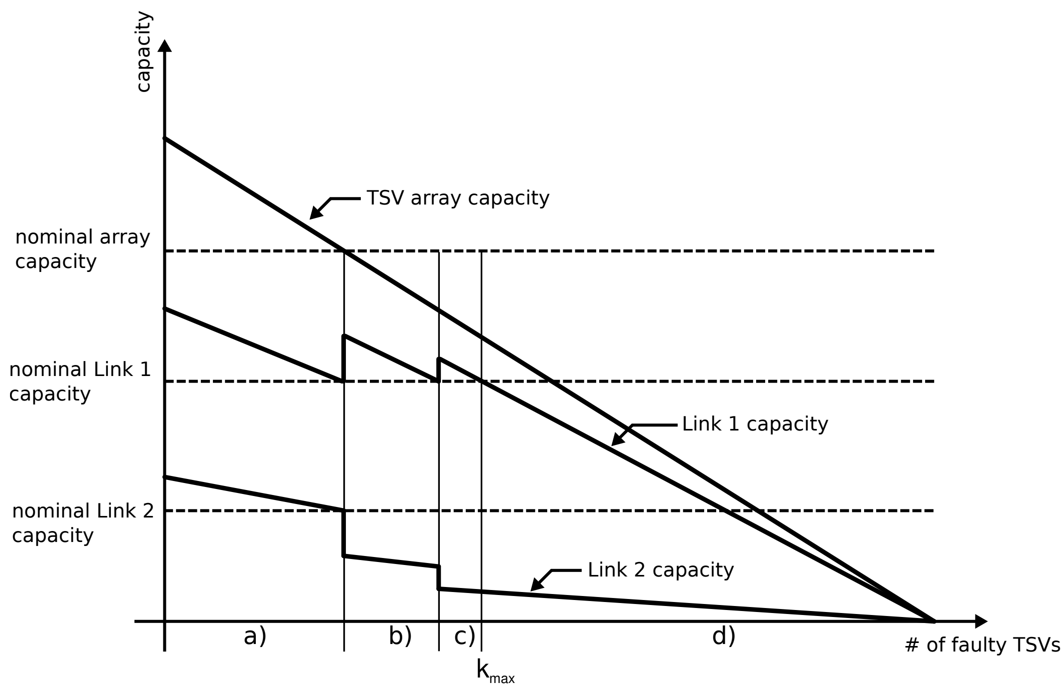


Fig. 3.24.: Link performance as a function of the number of faulty TSVs

Fig. 3.24 gives a qualitative illustration of the described concept for a two-link scenario. Link 1 in the figure is assumed to be configured for guaranteed service QoS-level. This could be for instance a critical point-to-point link in a SoC. Link 2, on the other hand, is more flexible (e. g. a NoC link) and is therefore configured for best effort QoS-level. The solid lines show the actual capacity of the full array and both links as a function of the number of defective TSVs. The dashed lines show their nominal capacity, hence the specified requirement on the throughput.

With a growing number of faulty TSVs (section a), the offered bandwidth of all links declines linearly. At some point (section b in Fig. 3.24), the capacity falls below the specified minimum, and the chip normally had to be considered faulty at this point. With a small alteration of the scheduler, we can assign an additional time slot to Link 1 (and take it away from Link 2). Thereby the capacity of the critical link is lifted above the specification again. With even more TSVs beginning to fail, the link capacities again start to decline and again falls below the specification (section c). Another time the time slot assignment is changed in favor of Link 1. This could theoretically go on multiple times, but at some point all time slots will be used up. Moreover, allowing too many faulty TSVs will at some point be no longer economical since it requires complex switch boxes. Therefore, k_{max} has to be set at a meaningful point and forms a trade-off between yield/resilience and increased cost for higher logic complexity.

Numerical example: Fig. 3.25 shows the principle for an example with realistic numbers. We assume to serve a critical link (Link a) with a nominal throughput of $19.2\,\text{Gbit/s}$ ($32\,\text{bit}$ at $300\,\text{MHz}$) and Link b with a target capacity of $38.4\,\text{Gbit/s}$

(64 bit at 300 MHz) configured for best effort service level. The TSV array is operated at a frequency of 1.8 MHz. An array of 32 data TSVs is required to provide enough capacity for both links. The 32 bit are filled with the data of the two links in six time slots per round. Therefore, two slots (one third) are assigned to Link a and four slots (two thirds) to Link b. For a TSV array with a total count of 40 TSVs, the initially provided bandwidth yields 72 Gbit/s as long as no TSVs are defective. Consequently, as long as 32 or more TSVs are operational, both links are provided with a higher capacity than the specified one. When more than 8 TSVs fail, the time slots get re-assigned and both links now receive 3 time slots per round. The critical Link a is still operated above its specification. We can reassign the time slots one more time, once less than 22 TSVs are functional. Finally, if less than 16 TSVs are functional, we consider the system as defective. Theoretically, the re-assignment could be triggered once more, but in this example, we opt for $k_max = 24$ to be a good trade-off in order to keep the complexity of the switch boxes manageable.

With a growing number of defective TSVs the number of effectively available data TSVs is reduced. Therefore, the serializers used must be able to adjust the portion of a link data word in the range of $n - k_{max}$ to n. This is accomplished by using a logarithmic shifter based serializer design. At this point, one could suggest to apply a serializer that already considers defective TSVs and spares positions of defective TSVs. However, such a serializer design would require a complex crossbar layout, resulting in a large area consumption.

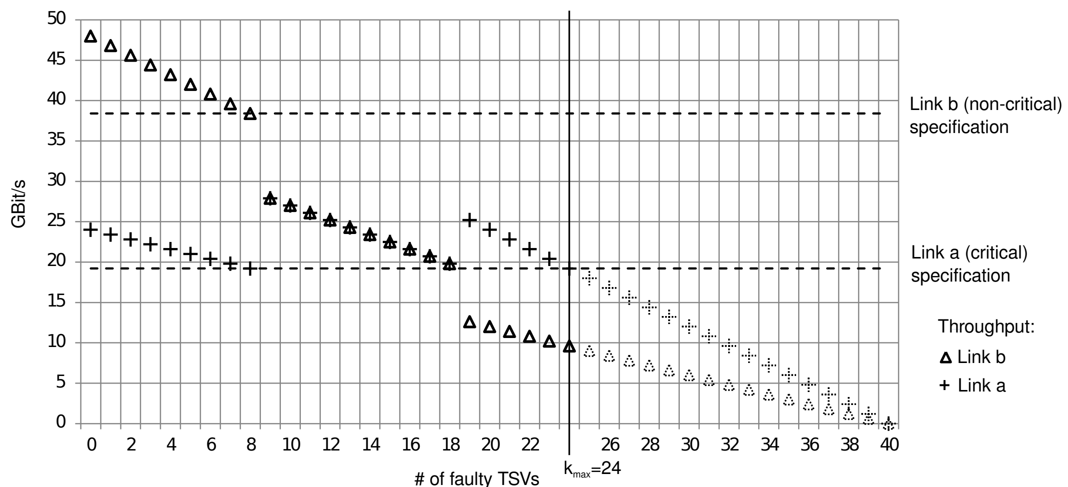


Fig. 3.25.: Link performance as a function of the number of faulty TSVs

3.2.2.3 Interconnect layer

The Interconnect Layer provides the interfacing to actual on-chip interconnection protocols and standards like buses, NoCs, socket proctocols, etc. The layer provides adapters which on one hand, adapt to the interconnection protocol, and on the other use one or multiple *VLinks* from the Virtual Link Layer for transportation. The complexity of such adapters varies significantly depending on the requirements

of the protocols. Protocols with stringent inter-cycle dependencies require more complex adapters with state machines on both ends to emulate signals. On the other hand, protocols that operate based on a "Fire-and-Forget" approach, can have very simple adapters, which mainly are reduced to perform an adaption of handshake signals.

The amount of *VLinks* used by an adapter usually depends on the number of channels the protocol uses internally. For NoC protocols, for instance, a *VLink* is required for each virtual channel. Furthermore, bus protocols, which internally use independent channels, will use multiple *VLinks*. An AXI adapter, for instance, will use a total of five *VLinks*, one link per channel.

3.3 Experimental Results and Discussion

The building blocks described in the previous sections have been implemented on the Register Transfer Level (RTL) and have been synthesized for the $65\,\mathrm{nm}$ and $45\,\mathrm{nm}$ technology node of a standard cell Complementary Metal–Oxide–Semiconductor (CMOS) process. For modeling the TSV process, the parameters have been extracted from [86]. One critical aspect with 3D-ICs is the clock distribution network. The TSV-Hubs require a high-speed clock, generated by a common clock source. However, no H-Tree is created for delivering this clock, and the impact of the, therefore, mesochronous dependencies is handled within clock synchronization stages. The simulations show that, with the chosen physical parameters from [86], a TSV clock of $f_{tsv} = 2\,\mathrm{GHz}$ is feasible with a standard cell process.

3.3.1 Physical Layer

3.3.1.1 Switch Boxes

The switch boxes are composed of a set of $n-1$ multiplexers with n being the number of data TSVs in the array. However, not all multiplexers in the set are equally complex, and their size and composition depends on both n and k_{max}, the maximum number of allowable defective TSVs. In Fig. 3.20 two examples are shown, one (Fig. 3.20b) with $n = k_{max}+1 = 6$, hence with the maximum possible allowed faults (for the case of $n = 6$), and one (Fig. 3.20a) with $n = 6$ and $k_{max} = 2$, hence two possible faulty TSVs. Regarding the multiplexer sizes, it can be generalized that its input count is $k_{max}+1$, unless its position $i = 1..n$ is $i \leq k_{max}$, then its input count is reduced to just i. Therefore, in total, we can calculate the input count as follows:

$$\begin{aligned} I(n, k_{max}) &= \left(\sum_{i=2}^{k_{max}} i\right) + (k_{max}+1)\,(n-k_{max}) \\ &= \frac{{k_{max}}^2 + k_{max}}{2} - 1 + (k_{max}+1)\,(n-k_{max}) \\ &= -\frac{1}{2}{k_{max}}^2 + \left(n - \frac{1}{2}\right) k_{max} - 1 + n \end{aligned} \tag{3.12}$$

Since the size of a Multiplexer is directly proportional to its number of inputs, $I(n, k_{max})$ is a measure for its size complexity $Size\ \mathcal{O}$.

Fig. 3.26 shows the *size complexity* curve with respect to k_{max} for different array sizes. Fig. 3.27 shows the actual consumed switch box silicon area as a result of a logic synthesis for a $45\,\mathrm{nm}$ and $65\,\mathrm{nm}$ standard cell process. The measured curve resembles the expected theoretical complexity curve of (3.12), which is also shown in the figure. We see the strongest deviation from the theoretical curve at at $k_{max} = 16$

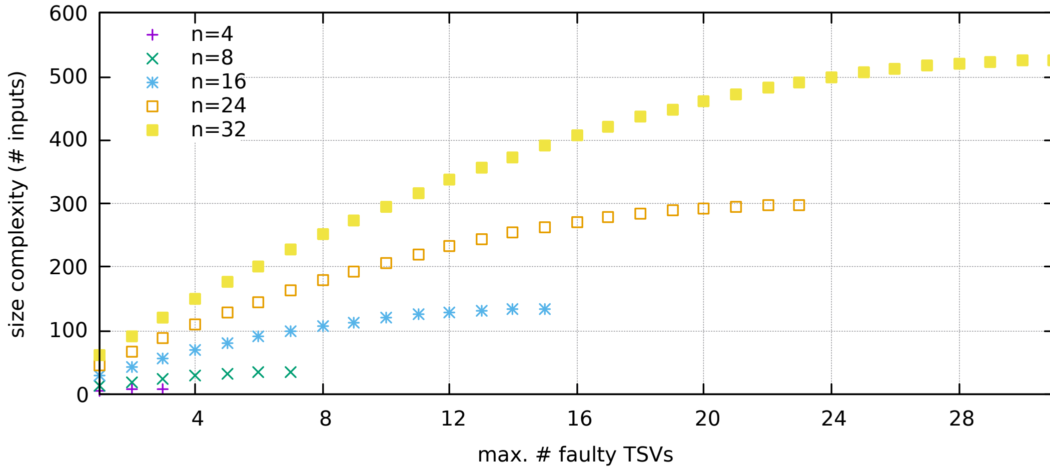


Fig. 3.26.: Switch box size complexity with respect to maximum number of faulty TSVs

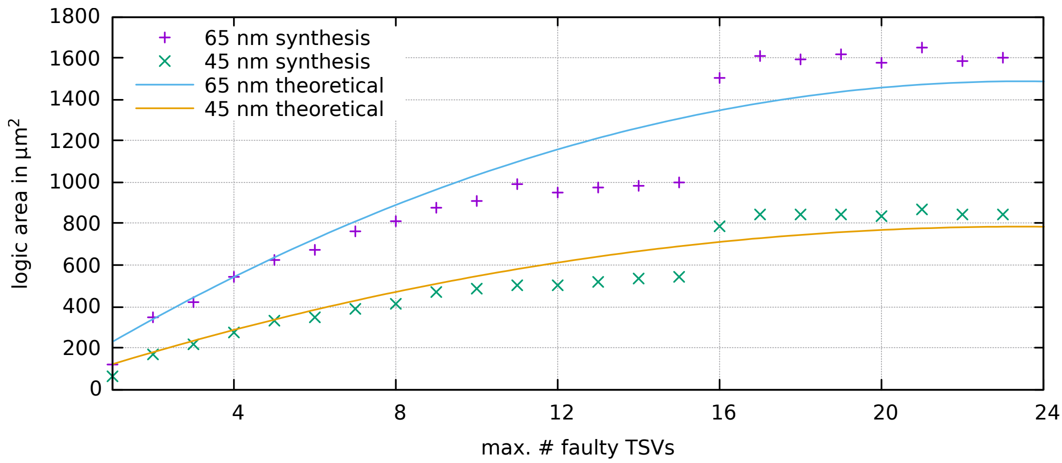


Fig. 3.27.: Switch box area as a function of k_{max} for an array with 24 TSVs

which accounts for standard cells with increased fan out being instantiated due to longer logic paths.

Fig. 3.28 gives another view on the switch box complexity. It shows the data points of a logic synthesis for two technology nodes with respect to the TSV count. However, the maximum number of allowable faulty TSVs is proportionally increased with the TSV count, such that $k_{max} = 0.5n$. If we substitute k_{max} in (3.12) we get:

$$I(n) = -\frac{1}{2}{k_{max}}^2 + \left(n - \frac{1}{2}\right) k_{max} - 1 + n \tag{3.13}$$

From (3.13) we can derive the circuit complexity with respect to n:

$$Size\ \mathcal{O}\left(I(n)\right) = \mathcal{O}\left(n^2\right) \tag{3.14}$$

This is confirmed by the observations of Fig. 3.28 which resembles a quadratic relationship.

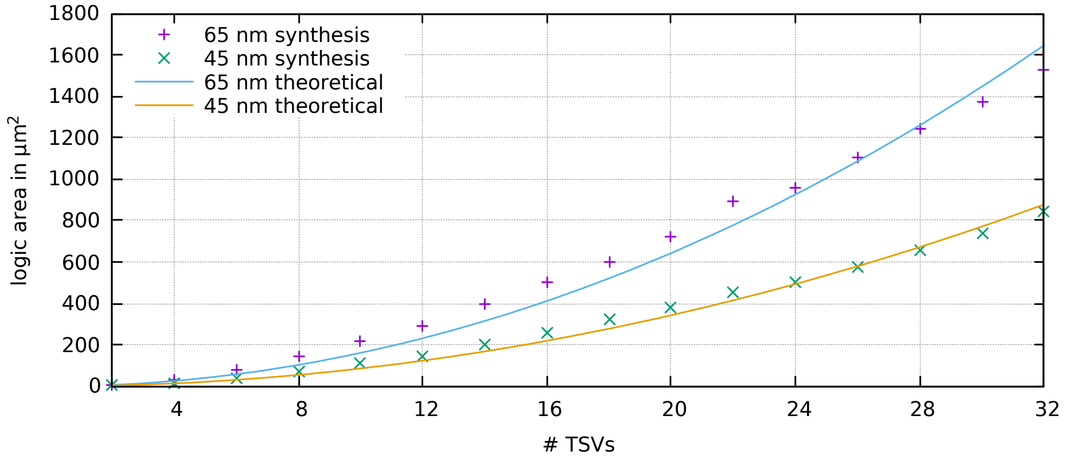


Fig. 3.28.: Switch box area with $k_{max} = 0.5n$

In Fig. 2.11 the theoretical overall system yield was shown as a function of the TSV count. With switch boxes in place, the yield is improved but nevertheless still a probabilistic figure which now also depends on the number of allowable faulty TSVs k_{max}. The appropriate stochastic process to model the yield at this point is a Bernoulli process, and the number of faulty TSVs is a binomially distributed random variable (K in the following).

The probability of having exactly k faulty TSVs within n total TSVs is therefore given by:

$$P(K = k) = \binom{n}{k} (1 - p_{tsv})^k {p_{tsv}}^{n-k} \tag{3.15}$$

where p_{tsv} is the yield of a single TSV, hence the probability of a single TSV to be functional.

With a given k_{max} the full array is still functional with the following probability:

$$P(K \leq k_{max}) = \sum_{k=0}^{k_{max}} \binom{n}{k} (1 - p_{tsv})^k {p_{tsv}}^{n-k} \tag{3.16}$$

Fig. 3.29 shows the improved yield when switch boxes are in place with respect to TSV count for different number of allowable faulty TSVs and a single TSV yield of $p_{tsv} = 0.9$. The solid line represents the yield function when no switch box is in place, hence no faulty TSV is allowed.

Note that although the term “yield” is used to quantify the probability that a TSV is functional in this work, the presented approach and calculations are also applicable for run-time failures. Run-time failures are characterized by the Mean Time To Failure (MTTF) and a characteristic survival function. The probability that a TSV is still functional at time t is then denoted as $S(t)$. Therefore, when assuming that

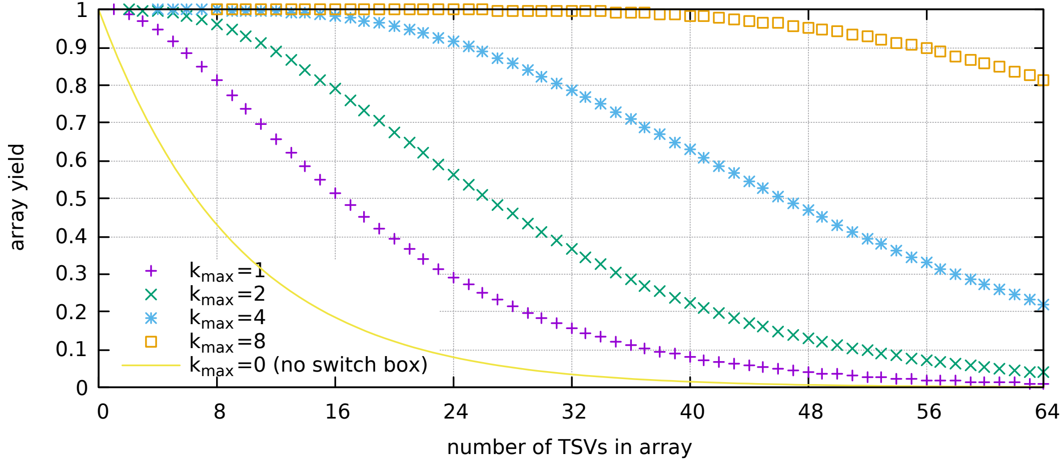


Fig. 3.29.: Yield for switch box TSVs with respect to TSV count ($p_{tsv} = 0.9$)

the MTTF of all TSVs in an array is identical, we can replace the yield p_{tsv} value by $S(t)$.

At this point, the question arises, if spending additional logic for the switch boxes can be justified economically. Regarding production faults, this depends on the unit costs of the complete IC. Or in other words: Does adding the resilient logic improve the overall yield to such an extent that the saved ICs compensate for the additional costs of the switch boxes? This question can hardly be answered in this context since the numbers are very specific for the individual case. Regarding run-time faults the TSV yield value is not meaningful, instead the MTTF is needed.

However, we want to analyze the amount of yield improvement that can be reached for a specific investment of additional chip area. So far, in Fig. 3.29 and Fig. 3.26, either the total number of TSVs n or the number of maximal faulty TSVs k_{max} was set to constant. Setting n to constant and increasing k_{max} clearly increases the yield but it also reduces the provided capacity of the TSV array. This might be tolerable for uncritical links, but for a fair comparison to other solutions we need to ensure that the nominal bandwidth is held constant when measures are taken to improve the yield. Thus, for every allowed faulty TSV an additional TSV has to be added in the array. Below we set $n = n_{nom} + k_{max}$, where n_{nom} is number of TSVs that is needed to retain the nominal bandwidth. Regarding the switch box complexity, we have seen in Fig. 3.26 that complexity increases with higher number of allowable faulty TSVs. We take the same approach here and set $n = n_{nom} + k_{max}$. We call these quantities *effective yield* P_{eff} and *effective complexity* I_{eff} hereafter.

Based on (3.16) the effective yield P_{eff} with respect to k_{max} reads to:

$$P_{eff}\left(K \leq k_{max}\right) = \sum_{k=0}^{k_{max}} \binom{n_{nom} + k_{max}}{k} (1 - p_{tsv})^k {p_{tsv}}^{n_{nom}+k_{max}-k} \tag{3.17}$$

Based on (3.12) the effective switch box complexity with respect to k_{max} reads to:

$$\begin{aligned} I_{eff}(k_{max}, n_{nom}) &= I(n_{nom} + k_{max}, k_{max}) \\ &= \frac{1}{2}{k_{max}}^2 + \left(\frac{1}{2} + n_{nom}\right) k_{max} - 1 + n_{nom} \end{aligned} \tag{3.18}$$

Note that the complexity measure is actually the input count (fan-in) which is (for the case of multiplexers) directly proportional to the required area since multiplexers are AC* complex ($Size\ \mathcal{O}(n)$). To get the actual area penalty a multiplier is needed that can be gained from synthesis experiments. Also the additional TSVs contribute to the area penalty.

The full area penalty for increasing the fault tolerance level is therefore:

$$A(k_{max}, n_{nom}) = a_{fi} \cdot I_{eff}(k_{max}, n_{nom}) + k_{max} \cdot a_{tsv} \tag{3.19}$$

with the footprint for a single TSV a_{tsv} and the switch box area per multiplexer input a_{fi}.

The area penalty and the effective yield are both depicted with respect to the number of allowable faulty TSVs in Fig. 3.30 for an example with a single TSV yield of $p_{tsv} = 0.8$ and an array with a nominal TSV count of 24. The TSV pitch is set to $d_{tsv} = 10\,\mu\text{m}$ and logic area multiplier is gained from synthesis for a $45\,\text{nm}$ standard cell process.

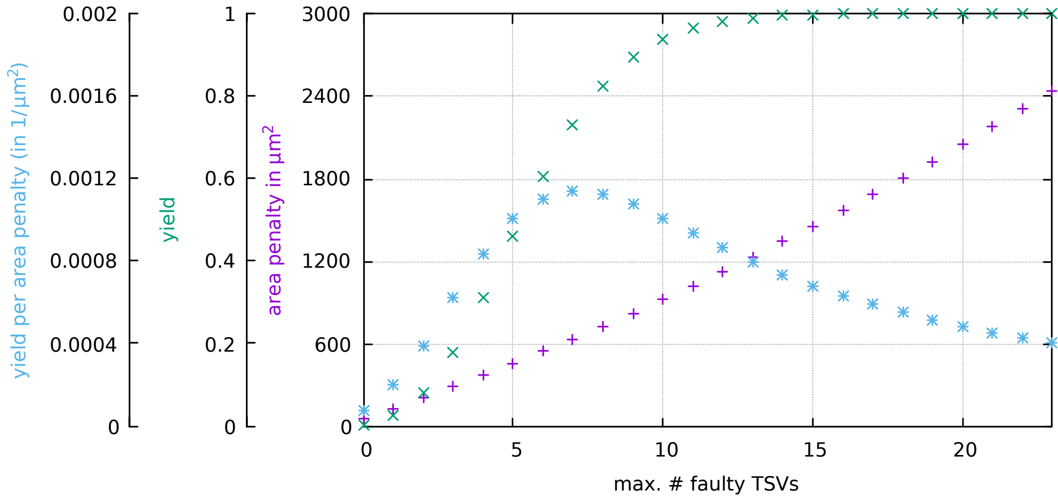


Fig. 3.30.: Effective yield with respect to allowable faulty TSVs

The more faulty TSVs allowed, the higher the yield, but also the complexity, hence area penalty increases. An interesting question at this point is how much yield improvement is achieved with a specific investment of additional area. This can be answered by relating the yield to the area penalty which is also shown in Fig. 3.30. It shows that for the given example this ratio is maximized with $k_{max,opt} = 7$.

*A Circuit Complexity class with $Size\ \mathcal{O}(n)$ and $Depth\ \mathcal{O}(\log n)$

Generalized, we can obtain $k_{max,opt}$ as follows:

$$k_{max,opt} = \arg\max_{k_{max}} \frac{P_{eff}(K \leq k_{max})}{A(k_{max}, n_{nom})} \quad (3.20)$$

Finally, the switch box based approach is compared to other solutions for improving fault tolerance. A baseline reference forms the simple solution of just adding additional TSVs next to existing ones, such that two or more TSVs transport the signal of one wire. Fig.3.31 shows the principle of what is called *reinforced TSVs* in this work.

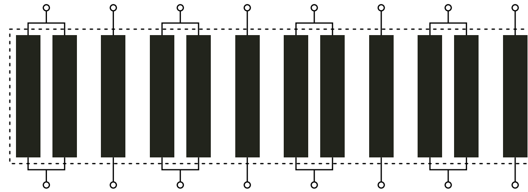


Fig. 3.31.: TSV array with 4 *reinforced TSVs* among eight total vertical signals

The additional n_{spare} TSVs are assumed to be equally distributed, i. e. as long as $n_{spare} \leq n_{nom}$ no bundle contains more than two TSVs. Theoretically also more than two TSVs per bundle would be possible, but with a tremendous area overhead.

With the probability p_{tsv} of a single TSV to be functional, the probability that a bundle of two TSVs is still functional is increased to: $p_{bundle} = 1 - (1 - p_{tsv})^2$.

Therefore, for the yield of the full array we get:

$$p_{reinforced} = p_{tsv}{}^{n_{nom}-n_{spare}} \left(1 - (1 - p_{tsv})^2\right)^{n_{spare}} \quad (3.21)$$

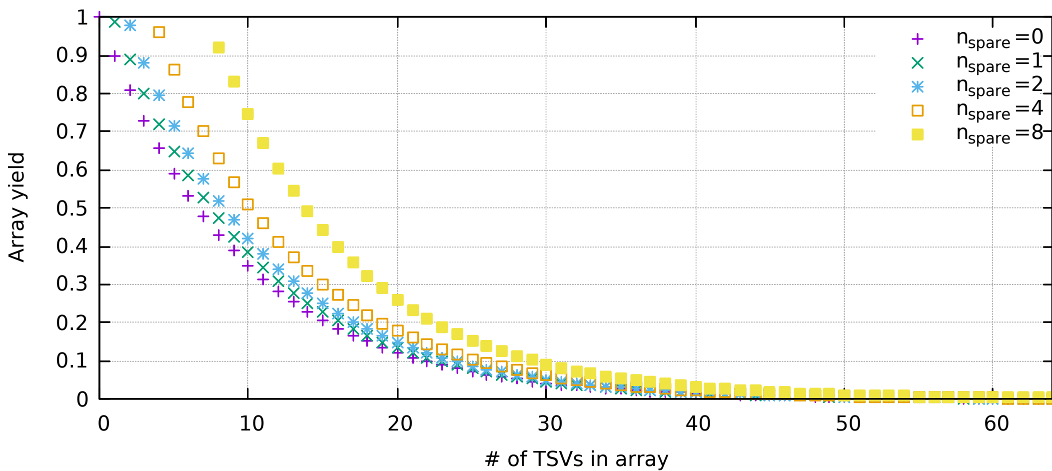


Fig. 3.32.: Yield for TSV arrays with reinforced TSVs ($p_{tsv} = 0.9$)

In Fig. 3.32 this array yield is shown for different numbers of reinforced TSVs. When comparing this figure with Fig. 3.29, it becomes clear that the array yield drops

significantly faster with increasing TSV count. However, for a fair comparison, we have to consider that for the switch box based approach we need to add additional TSVs to maintain the nominal bandwidth, and we need to consider that switch box complexity also increases due to the increased TSV count.

Therefore, the real question is: How much yield improvement do we get for additional area investment while maintaining nominal bandwidth for both the switch box based approach and reinforced TSVs?

To answer this, we consult (3.18) and (3.19) again, which return the effective area penalty with respect to the number of tolerable faulty TSVs while maintaining nominal bandwidth. We need to solve these equations for k_{max} in order to get the possible allowable faulty TSVs for a specific area investment. The result of this transformation can then be inserted in (3.17) to get the effective array yield. The full transformation can be found in Appendix A.1.

For the reference case with reinforced TSVs the calculation is straight-forward and is given by (3.21) with setting $n_{spare} = \lfloor \frac{a}{a_{tsv}} \rfloor$ with the area investment a.

Fig. 3.33 shows the results for $p_{tsv} = 0.9$ and $p_{tsv} = 0.8$ for both the switch box based and reinforced TSV solution. The figure shows that yield gain per area investment is much higher for the switch box solution. Note also the stair shaped curve progression due to integer numbered TSV count finally leading to discrete yield values.

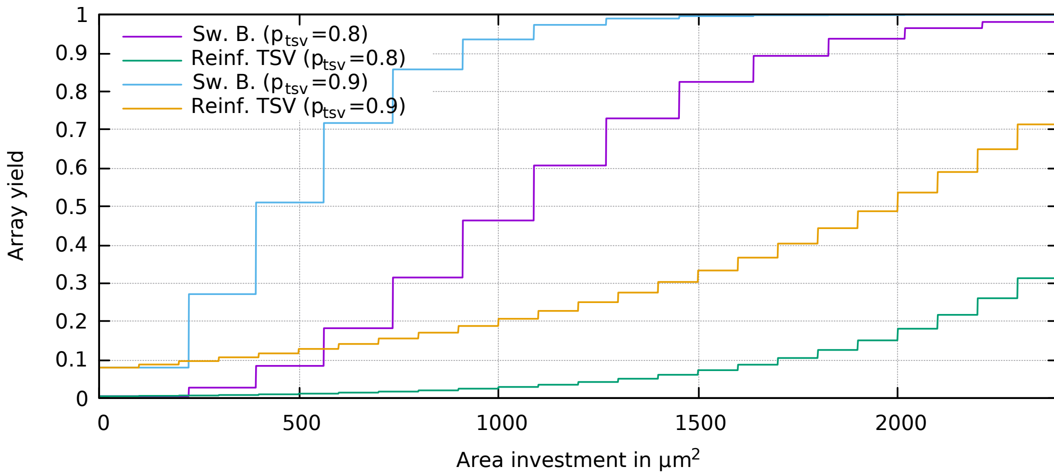


Fig. 3.33.: Yield per area investment for switch box based resilience and reinforced TSVs

3.3.2 Virtual Link Layer

3.3.2.1 Serialization

The serialization step is carried out by a shifter design based on the concept of a logarithmic shifter [178]. Such a shifter comes with a size complexity of $Size\,\mathcal{O}(n\log_2 n)$ and a depth complexity of $Depth\,\mathcal{O}(\log_2 n)$. In contrast to the conventional logarithmic shifter, the modified version used here is designed in an asymmetric fashion, i. e. the output can be narrower as the input. This removes some of the complexity of the shifter. With its $\log_2 n$ depth complexity, the length of the critical path is manageable even for wider link sizes. Each doubling of the data width will add one stage. Our simulations show that such a design is capable of being operated at up to $3\,\mathrm{GHz}$ for the used $45\,\mathrm{nm}$ and $65\,\mathrm{nm}$ technology.

A low-weight alternative to the logarithmic shifter is a standard multiplexer. It is significantly lower in complexity, but cannot shift arbitrarily. Therefore, if the width of the unserialized data stream is not an integer multiple of the serialized one, bandwidth will be wasted.

Fig. 3.34 shows the area consumption for both variants for the serialization of a 64-bit wide link with respect to the width of the serialized data word.

Fig. 3.34.: Area consumption of serializer implementations (64-bit link)

3.3.3 Protocol Layer

3.3.3.1 AXI Case Study

For analyzing and demonstrating the TSV-Hub concept, a set-up for the continuation of two AXI links was implemented and synthesized. Fig. 3.35 gives a high-level view on this system, with two fully independent AXI links. With the TSV-Hub design, it is also possible to continue different protocols by using the specific adapters, but for

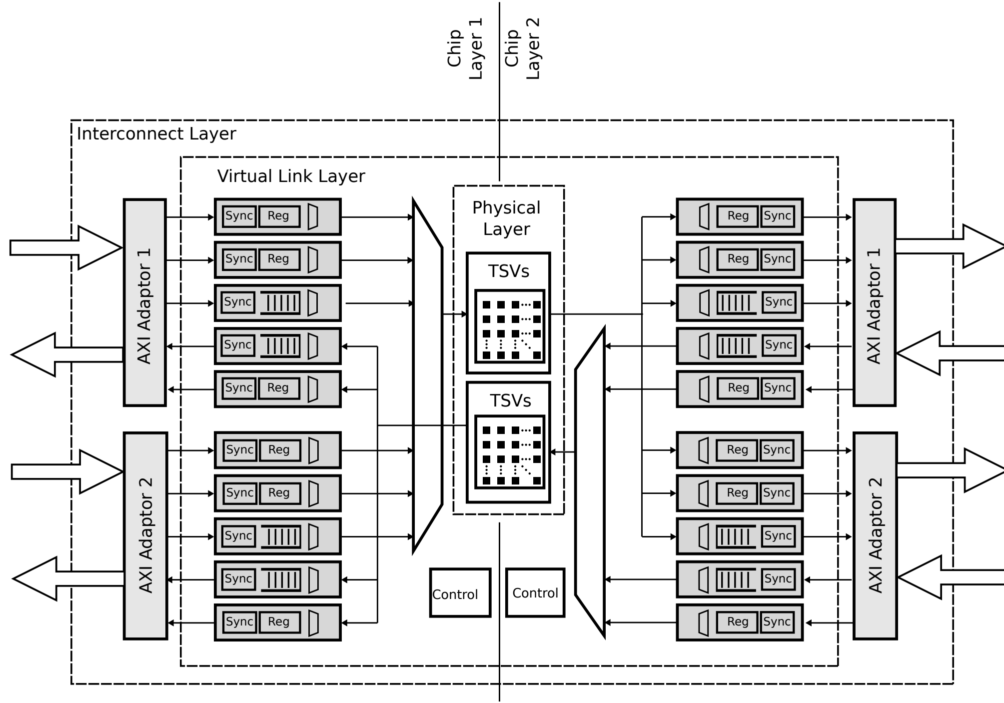


Fig. 3.35.: Continuation of two AXI links

the sake of clarity and to reduce the number of parameters this setup was chosen as a case study with the involvement of a single protocol.

The protocol adapters used for the AXI system make use of five VLinks, one link per AXI channel. This guarantees an independent flow on each channel. The word sizes of the individual channels are given in Tab. 3.1 and range from 9 to 55 bit for the 32-bit version and from 9 to 74 bit for the 64-bit version.

With burst transfers AXI's address channel idles most of the time. Only when a new transfer is initiated, an address is conveyed.

Therefore, the net data rate required for the complete bus depends on the burst size or, statistically seen, on the distribution of burst sizes. The highest utilization will be observed for single word random access transactions, and the lowest utilization for subsequent transfers with the maximum possible burst size.

Formally, the downstream (master to slave / write) bandwidth requirement can be written as:

$$B_{axi,down} = f_{ic}\left(W_{wdata} + \sum_{i=1}^{N_w} \frac{W_{waddr}}{i} r_{w,i} + \sum_{i=1}^{N_r} \frac{W_{raddr}}{i} r_{r,i}\right) \tag{3.22}$$

Tab. 3.3.: Area consumption of the mesochronous FIFOs in μm^2

Channel	65 nm	45 nm
Read Data	3473	1778
Write Data	3570	1828

and the bandwidth for the upstream (slave-to-master / read) direction reads as:

$$B_{axi,up} = f_{ic} \left(W_{rdata} + \sum_{i=1}^{N_r} \frac{W_{wresp}}{i} \cdot r_{w,i} \right) \tag{3.23}$$

For evaluating the TSV-Hub's performance, simulations were carried out for different compositions (with respect to the number of AXI links and the word sizes of the data channels). The simulations were performed on RTL level. Additionally, to verify the timing for the targeted clock frequencies, standard cell synthesis was completed for two different libraries (65 nm and 45 nm). In the synthesized net list, the impact of TSVs was added by assigning the estimated capacitive load and resistance to the driver's outputs. Both front-end and back-end synthesis were performed.

A test bench creating AXI traffic at different burst sizes was used to stimulate the design. The results of these simulations were compared with the throughput of a directly interconnected AXI system (without a TSV-Hub). The results are shown in Fig. 3.37 for the downstream/write direction. In this figure, the throughput achieved with a shifter based serializer (a serializer allowing non-integer serialization rates) is represented by dashed lines. The solid lines show the result for the multiplexer based serializer.

The throughput in Fig. 3.37 is given with respect to the number of data TSVs. As expected, the throughput starts to drop once the required capacity is not provided anymore. With a multiplexer based serializer, performance can be sacrificed for the sake of a lower area consumption. In this case, Pareto optimal design points can be found. For instance, for the case of a 64-bit wide link, 98% of the throughput can still be achieved with using 55 TSVs and 97% of the throughput with 37 TSVs (compared to 77 TSVs for 100% throughput).

Both upstream and downstream performance are combined to an average value which is given in Fig. 3.36 for a 32-bit burst scenario (two 32-bit wide links) with respect to upstream and downstream TSV count. Here, for example, the design point at 20 upstream and 21 downstream TSVs results in 95% of the unserialized throughput (compared to 40 upstream and 42 downstream TSVs for 100%).

Tab. 3.5 gives the area consumption of the TSV-Hub for the different scenarios. It lists both the area consumed by TSVs and the area used for the serialization, multiplexing

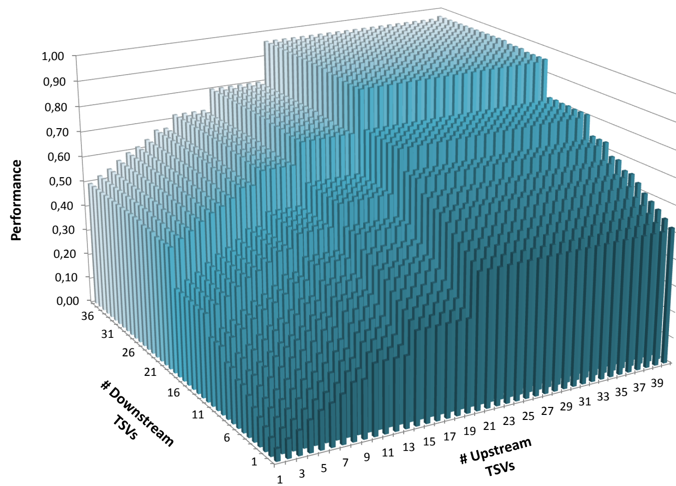


Fig. 3.36.: Performance with respect to TSV count for upstream and downstream

Tab. 3.4.: Area consumption of the mesochronous synchronizers in μm^2

Channel	65 nm	45 nm
Read Address	1528	852
Write Address	1711	964
Write Response	490	250

and control logic. The last column in the table gives a *reference area*. This is the area consumed by TSVs when no serialization/multiplexing is in place, hence no logic overhead is induced but a higher number of TSVs is required. Tab. 3.3 lists the area consumption for the mesochronous synchronization FIFOs and Tab. 3.4 the ones for the mesochronous handshaking registers.

Continuing only two 32-bit AXI links without using any serialization or multiplexing would require 390 TSVs, resulting in an area consumption of $0.156\,\text{mm}^2$ when assuming a $20\,\mu\text{m}$ TSV pitch. This TSV induced area consumption can be reduced to an area of $0.023\,\text{mm}^2$ by using the TSV-Hub's serialization and multiplexing capabilities. This obviously adds a logic overhead that must be considered as well. The overhead for the example above is $0.041\,\text{mm}^2$ and thus a resulting total area for the vertical link of $0.064\,\text{mm}^2$. This is 41% of the reference area. By using a standard cell process with a smaller feature size, the effect is enhanced, since the logic overhead is reduced (in terms of area). For the same example as above but with a $45\,\text{nm}$ technology, the total link area is reduced to 29% of the reference area. On the other hand, with smaller TSV pitches the area savings are reduced as the impact of the TSV occupied area gets smaller. The break-even can be found at a TSV pitch of roughly $7\,\mu\text{m}$ for the $65\,\text{nm}$ process node and $5\,\mu\text{m}$ for the $45\,\text{nm}$ process. Tab. 3.6 shows the area savings for different configurations.

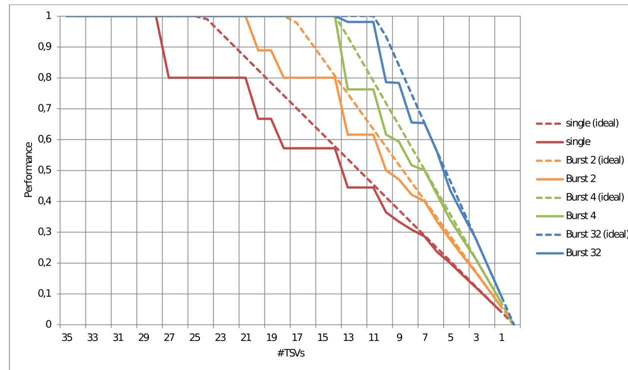
Tab. 3.5.: Area consumption of full TSV-Hub (in μm^2)

	Logic Area		TSV area		TSV (20 μm pitch) + Logic		TSV (10 μm pitch) + Logic		Reference area
	65 nm	45 nm	pitch 20 μm	pitch 10 μm	65 nm	45 nm	65 nm	45 nm	
1x 32 bit	21876	11548	14000	3500	35876	25548	25376	15048	78800
2x 32 bit	40662	21489	23200	5800	63862	44689	46462	27289	156000
1x 64 bit	34369	18050	18800	4700	53169	36850	39069	22750	103600
2x 64 bit	64383	34002	36000	9000	100383	70002	73383	43002	207200

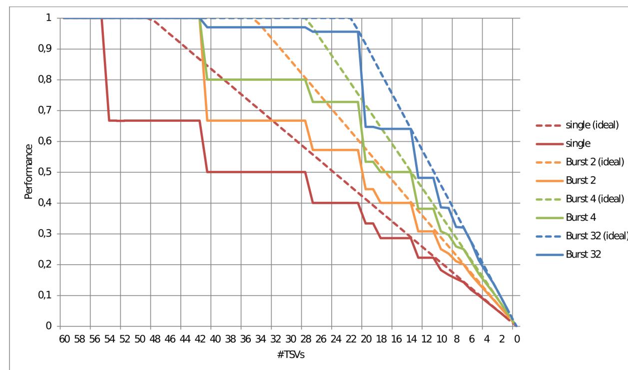
Tab. 3.6.: Percentage of original TSV induced area

	20 μm TSV pitch		10 μm TSV pitch	
	65 nm	45 nm	65 nm	45 nm
1x 32 bit	45 %	32 %	128 %	76 %
2x 32 bit	41 %	29 %	119 %	70 %
1x 64 bit	51 %	36 %	152 %	88 %
2x 64 bit	48 %	34 %	142 %	83 %

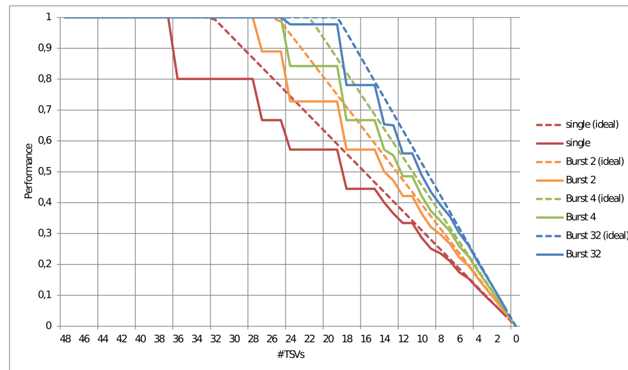
In addition to area savings, the improved yield that comes with a reduced TSV count has to be considered. Even if the total area of the vertical link cannot be reduced because the two effects cancel out each other, it might be beneficial to use serialization and multiplexing.



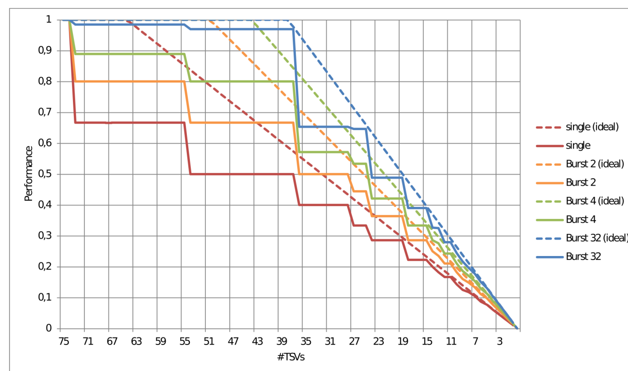
(a) 32-bit single link



(b) 32-bit double link



(c) 64-bit single link



(d) 64-bit double link

Fig. 3.37.: Performance degradation of TSV continued AXI links with respect to burst size

TSV Property Aware 3D-Network-on-Chip Synthesis

4

Designing a Network-on-Chip (NoC) for regular architectures like Chip-Multi-Processors (CMPs) is fairly straight-forward. As the final application and its communication patterns are usually unknown during the design process of such a general-purpose system, the network must equally be as generic as possible. Therefore, it will usually be as regular as the arrangement of the *PEs* (e. g. a mesh structure), and designing it is feasible by manual methods.

Most modern Multi-Processor System-on-Chips (MPSoCs), however, are highly irregular and, in most cases, assembled from multiple general purpose processing cores and many dedicated modules like hardware accelerators and interfacing modules. The scalability problem that modern SoCs pose to classical bus-based interconnect methodologies is evident with dedicated modules, as well.

The crucial difference is, however, that much more knowledge on the communication patterns is available already at design-time. This information can be formally specified in a suitable data structure, e. g. an annotated graph. Fig. 4.1 shows such a graph (without annotations) for a three-layer system. It depicts communication flows between the individual PEs.

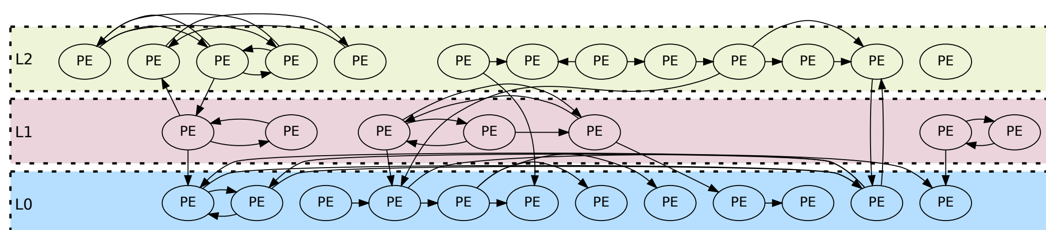


Fig. 4.1.: Example of a communication graph for a three-layer system. The processing elements are already assigned to the chip layers of the 3D-IC

Starting from such a graph, a set of constraints, process parameters and design objectives, a fully customized application specific NoC can be created. However, the design space is huge, and manually creating an optimized on-chip network for an MPSoC is not realistic.

All challenges described above are already apparent in a pure 2D design flow but are intensified for most 3D-ICs. The only challenge that is possibly relaxed in a 3D-scenario is the wire-length problem, since a 3D-IC with the same module count

as its two-dimensional counterpart shows an average reduction of global wire length by a factor of $\sqrt{n}$ (with n being the number of chip layers).

However, this assumption only holds if TSVs are simply considered as vertical wires and are inserted to continue conventional global wires to other chip layers without fully leveraging TSV performance or pursuing an economical TSV use. Considering the area cost, reliability and physical parameters of TSVs, this would be a naive approach and lead to costly systems with low reliability and low yield.

Therefore, the main objective of the synthesis flow proposed in this Chapter is to incorporate the impact of TSV properties early in the NoC synthesis process.

The main differentiators compared to existing NoC synthesis solutions are:

- Incorporate cost factors for Inter-Layer-Links
- Implement TSV-Hubs in order to use TSVs with the following objectives:
  - Leverage the full TSV capacity
  - Use TSVs sparingly
  - Provide fault tolerance for inter-layer-links

The presented 3D-NoC synthesis solution builds up on a flow for 2D-NoC-Synthesis, developed at the TUM Chair of Electronic Design Automation (EDA) by Todorov et al. [33]–[35]. In collaboration with the authors of the approach, it has been extended in this work with respect to the points listed above. The original 2D synthesis flow is depicted in Fig. 4.2.

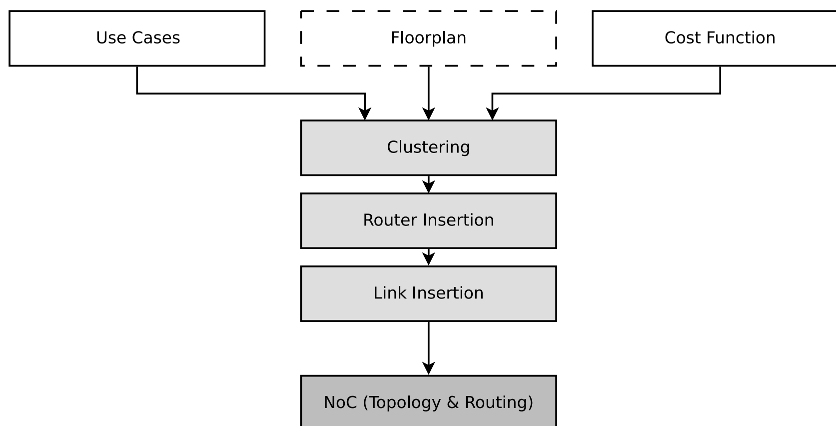


Fig. 4.2.: Synthesis flow of Todorov et al. [34], [35]. Clustering is performed based on multiple graphs (use cases) and cost factors. In [35] also a floor plan input is considered. Links are inserted by means of a Greedy Algorithm to generate the final topology and routing instructions.

4.1 NoC Synthesis process

4.1.1 Problem formulation

In theory, a 2D-Synthesis flow like the one outlined in Fig. 4.2 could also be used to create application specific networks for 3D-Systems with very little modification and without changing the underlying cost model or adding further steps. However, the result would be biased towards high performance at exorbitant costs since TSV costs are not captured sufficiently. A typical outcome is illustrated in Fig. 4.3. The algorithm would maximize the inter-layer-link count since such links are, from the perspective of the tool, very cheap, basically free. Hence, the tool would place cores communicating intensively in close proximity and form clusters spread over multiple layers. Even when TSV length is considered adequately, the length of the Inter Layer Links (ILLs) is negligible compared to BEOL wiring. The resulting system would in fact be of high performance but it would also be very costly and most likely would not be realizable due to the large TSV induced area overhead.

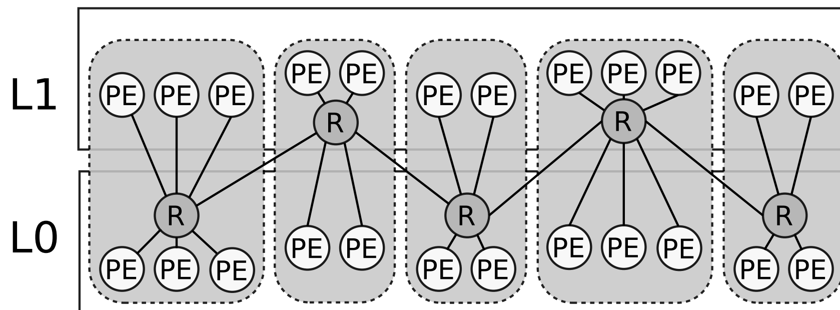


Fig. 4.3.: Example of a typical outcome when using a 2D-Synthesis flow for 3D-Design. Inter-Layer links are maximized since they are physically short. Hence, TSV count is massively increased.

A specialty of the algorithm this work is based on, is that it considers multiple variants of a CCG as input, the so-called *use cases* $\mathcal{U}$. This allows defining requirements for different application scenarios which are temporally mutual exclusive but have all to be satisfied by the generated on-chip network. Thus, each of the possible scenarios forms a use case $U_x \in \mathcal{U}$ [34].

Having an algorithm that supports multiple uses cases is of particular importance for the presented synthesis flow, since it allows efficient vertical link bundling (as described in section 4.1.5).

We pick up the notation from [34] to formalize the CCG and use cases: Each use case represents a directed graph $U_x = (P_x, \Phi_x)$, where the nodes P_x represent active PEs in the use case U_x and the edges Φ_x represent the communication requirements (between two nodes). A flow $\varphi_{x,i} \in \Phi_x$ in turn is comprised of a bandwidth requirement $\beta_{x,i}$ and a latency constraint $\alpha_{x,i}$.

The first step in the synthesis process is the clustering phase. Clustering is based on the CCGs to one part. Cores communicating heavily with each other are more likely to be connected to a common router. The other part is formed by the floorplan input in terms of spatial distances. For the 3D scenario, the vertical distance has to be added. However, this distance must be treated separately from the horizontal one. The number of chip layer crossings between two PEs is essential here, not the actual metric distance. Using the actual metric distance $d_{i,j}$ between two cores P_i and P_j as $d_{i,j} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2 + (z_i - z_j)^2}$ would massively underrate the role of the vertical extent since this is very low (tens to few hundreds of $\mu\mathrm{m}$) compared to the lateral distances (up to few mm). What actually creates costs though, are TSVs inserted at layer crossings.

Once the network synthesis process is finished (the parts that are already defined for the 2D flow), the 3D flow needs further steps for realizing inter-die communication (TSV insertion). A simple but costly and area consuming approach would be to insert a TSV for every wire in each link at each layer crossing. A significant reduction of TSV count can be achieved with a clock speed-up at the TSV array of each link and by applying link serialization. Further reducing TSV count is possible by bundling links and inserting a TSV-Hub for each link bundle and applying multiplexing.

However, to form such bundle, another clustering problem opens up since the algorithm must decide which links to bundle and how many TSV-Hubs to insert at a layer crossing. This can range from a single TSV-Hub (transporting all links) to a dedicated hub for every crossing link. Furthermore, the positions of the TSV-Hubs have to be determined.

4.1.2 Overview

The synthesis flow is outlined in Fig. 4.4. It contains the steps from the 2D flow of [35] and adds the steps of *Vertical link mapping* and *TSV-Hub insertion*.

The initial, intermediate and final stages of the flow are shown in Fig. 4.5 by means of a simplified example (showing only a single lateral dimension).

The process starts with the clustering phase on the basis of CCGs and a 3D-floorplan (Fig. 4.5a). The system is partitioned into smaller sections consisting of only a few PEs each. The clusters can be limited to a single chip layer or span multiple layers (Fig. 4.5b). In the next step (*Router insertion*) a NoC router is instantiated for each cluster and all PEs of a cluster are connected to this router (Fig. 4.5c).

In the *Link insertion* step routers are interconnected, thus forming the topology of the NoC (Fig. 4.5d). Links traversing layer chip boundaries are *virtual* for now since they require TSVs to be inserted. Intra-layer-links, however, already represent physical links. The virtual vertical links are clustered based on temporal correlation

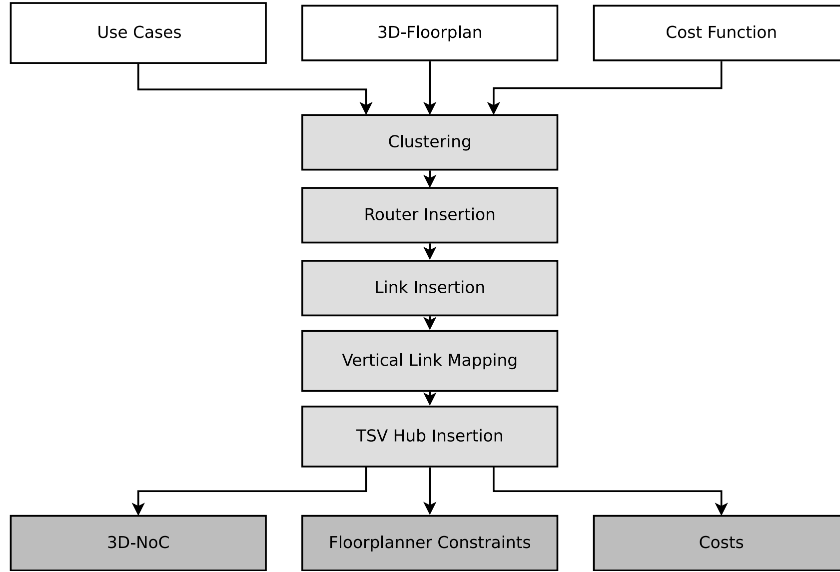


Fig. 4.4.: 3D-NoC synthesis flow

of the carried flows and spatial distances of the layer crossing points in the *Vertical link mapping* phase (Fig. 4.5e). Finally TSV-Hubs are inserted for the vertical link bundles (Fig. 4.5f).

4.1.3 Clustering and Router Insertion

Clustering is the step of creating meaningful partitions of the communication graph and inserting a router for each partition. To this end, we need to define a metric for the optimization process and thereby set a design objective.

The authors of [35] have chosen *spectral clustering* as partitioning algorithm. In contrast to the traditional Min-Cut approach, Spectral Clustering creates clusters with the objective to not only minimize the inter-cluster affinity, but also to maximize the intra-cluster affinity, hence it tries to find the normalized cut [179] of the input.

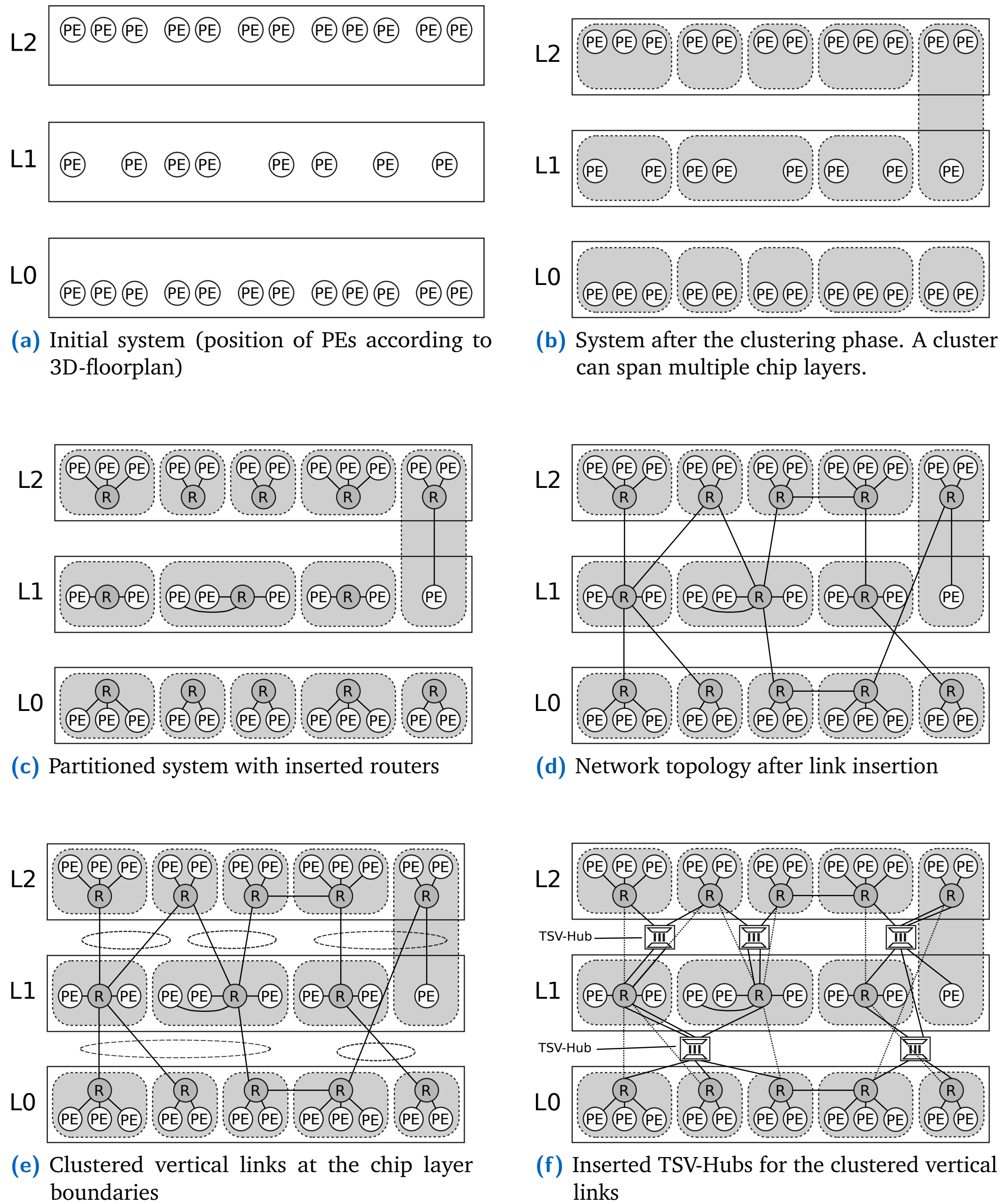
For two clusters k_a and k_b, the normalized cut is given as follows [34]:

$$\overline{\mathrm{cut}}(k_a, k_b) = \frac{\mathrm{cut}(k_a, k_b)}{\mathrm{vol}(k_a)} + \frac{\mathrm{cut}(k_a, k_b)}{\mathrm{vol}(k_b)} \tag{4.1}$$

with $\mathrm{vol}(k)$ being the sum of all weights of all internal edges of a cluster k and $\mathrm{cut}(k_a, k_b)$ being the sum of all weights of the edges between two clusters k_a and k_b.

The optimal clustering C for the full system is then given by [34]:

$$C = \arg\min_{\hat{C}} \sum_{k_a \in \hat{C}} \sum_{\substack{k_b \in \hat{C} \\ k_b \neq k_a}} \frac{\mathrm{cut}(k_a, k_b)}{\mathrm{vol}(k_a)} \tag{4.2}$$



(a) Initial system (position of PEs according to 3D-floorplan)

(b) System after the clustering phase. A cluster can span multiple chip layers.

(c) Partitioned system with inserted routers

(d) Network topology after link insertion

(e) Clustered vertical links at the chip layer boundaries

(f) Inserted TSV-Hubs for the clustered vertical links

Fig. 4.5.: Intermediate stages of the 3D-NoC synthesis flow

The problem of (4.2) is NP hard [180], hence no closed form calculation is feasible. With spectral clustering, a global optimum of the problem is approached.

However, spectral clustering cannot be directly applied to the CCG since it needs an undirected graph with a single value annotation as input. Moreover, the floorplan information has to be incorporated in this graph.

Therefore, an *affinity matrix* [34] is calculated first. It is created from a weight function including all the annotations of the CCG's edges and the distances from the floorplan.

In [34] the latency constraint and bandwidth is considered, here we add the vertical and lateral distance. This gives:

$$a_{i,j} = c_\beta \beta_{i,j} + \frac{c_\alpha}{\alpha i, j} + \frac{c_h}{h_{i,j}} + \frac{c_v}{v_{i,j}} \tag{4.3}$$

with the summed bandwidth requirement of all flows between PE_i and PE_j $\beta_{i,j}$, the maximum latency constraint between the two PEs $\alpha_{i,j}$ and the horizontal and vertical distances $h_{i,j}$ and $v_{i,j}$ respectively. Note that the horizontal distance is the actual metric value, whereas the vertical distance is the distance measured in chip layers. The weighting factors in (4.3) c_β, c_α, c_v and c_h have to be set by the user to tune the design objective and account for different cost factors.

Spectral clustering is then applied on the basis of the affinity matrix $\mathbf{A}$ with the elements calculated according to (4.3). The algorithm was implemented as described in [34].

Finally, a router is inserted for each cluster.

4.1.4 Generation of NoC-Links

In this step, the Routing Path Allocation (RPA) phase, the links between the individual routers are created, hence the network's topology is formed. The algorithm used for this task is identical with the one from the 2D flow [34]. The only difference is, that the full 3D synthesis is not yet finished after this step since TSV insertion has to follow.

Therefore, the algorithm from [33] is used without modification. The algorithm creates a deadlock free network by avoiding cycles in the channel dependency graph. Since TSV-Hub insertion happens at a lower abstraction layer and does not modify this graph, the deadlock freedom is maintained in the following steps.

4.1.5 Vertical Link Bundling and TSV-Hub-Insertion

For a planar design, the links created in the previous phase represent actual physical connections to be realized by a set of wires. In a 3D system, this is only the case for links that do not cross chip layer boundaries. For layer crossing links, however, both BEOL wiring and TSVs have to be used. Such links are considered *virtual* at this point (virtual Inter Layer Link (vILL) below) and need to be mapped on TSV arrays to construct their actual physical path.

The naive approach would be to insert a single TSV for every wire at every layer crossing and directly connect them in series. However, this would drive up area cost, and reduce both yield and reliability. Moreover, it would not fully leverage TSV capacity.

In Chapter 3.2 the TSV-Hub was introduced as a building block to efficiently operate TSV arrays. At this stage of the proposed synthesis process, TSV arrays are instantiated as another building block, besides routers and physical links.

A crucial feature which makes the TSV-Hub superior to a pure serialization solution is that it also provides link multiplexing and, more important, statistical multiplexing. However, for a meaningful application of statistical multiplexing knowledge (the statistic) needs to be available about which flows are likely to be active at the same time and which are not. This information can be derived from the bandwidth annotations of the different use cases in the CCG.

The TSV-Hub insertion and mapping process is carried out independently for each layer boundary. For systems with more than two layers a binary tree sequence (Fig. 4.6) is used, starting in the middle of the stack (this is covered in more detail in section 4.1.5.2).

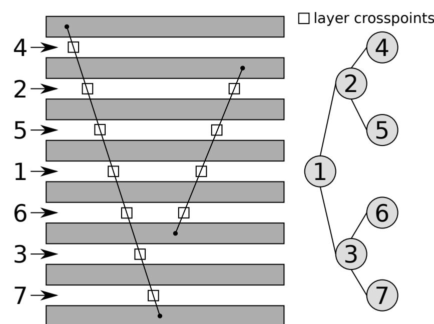


Fig. 4.6.: Example of the order for bundling the links in form of a binary tree, gaps are gradually closed. Layer intersection points are calculated based on a theoretical direct line. (Note that a 3D-IC with that many layers is an academic example and not feasible with current stacking technology.)

To create the bundles of vertical links another round of spectral clustering is performed. This time, links are clustered (not PEs). To this end, another affinity matrix is required. Its affinity values are created based on:

- Bandwidth correlation
- spatial distance of layer crossing points

4.1.5.1 Bandwidth correlation

After the RPA phase, the flows are mapped on physical intra-layer links and vILLs, and a path $\mathcal{P}_{u,f} = \text{path}(f_{u,f})$ is formed for each of the flows defined in the CCG. This path is a sequence of the source node, a series of links and routers and the destination node. Note that a flow is always only part of a single use case u. Of course, a flow with the same properties and the same source and destination node can be added initially in multiple use cases, but they are logically considered as multiple flows.

A link, however, is not assigned to a specific use case. It is the transporting vehicle for a hop of at least one but usually multiple flows and can be present in multiple paths. Therefore, a link has a certain bandwidth share in each use case (which can also be zero if no flow of the respective use case is mapped on the link). Hence, a link l has a bandwidth share of $b_{l,u} > 0$ in a use case u if $\exists i\,(l_i \in \mathcal{P}_{u,i})$.

This is illustrated in Fig. 4.7 for a simple example of a two-layer system with two use cases. In the synthesized system on the right (before TSV-Hub insertion), link l_4 is carrying all four flows, while the other links (l_0,l_1,l_2,l_3) are carrying two flows each. However, all links have a share in each use case since all of them carry links of both use cases.

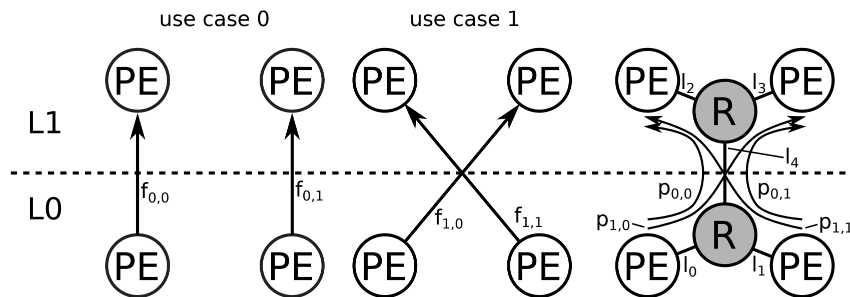


Fig. 4.7.: Simple example showing the CCGs for two use cases of a two-layer system. After clustering and RPA the system on the right has been created. The flows are mapped on a sequence of links each.

To quantify the bandwidth share in the individual use cases, we form a vector $\mathbf{b}_l \in \mathbb{R}^{|\mathcal{U}|}$ for each link l, where the i-th element denotes the bandwidth share in use case u_i and is defined as follows:

$$b_{l,i} = \sum_{\{f:l\in\mathcal{P}_{u,f}\}} \beta_{u,f} \tag{4.4}$$

hence the sum of all bandwidth requirements $\beta_{u,f}$ of all flows $\mathcal{F}_u$ of a use case u which are transported by link l.

Based on the bandwidth vector, it is now possible to calculate a correlation coefficient $r_{i,j}$ for a pair of vILL (l_i, l_j). This correlation coefficient tells if the links have bandwidth shares mainly in the same or in different use cases. It will take values between -1 and 1. The corner cases indicate if the links are active in the same use cases ($r_{i,j} = 1$) or only in different use cases ($r_{i,j} = -1$).

The correlation coefficient of a single link pairing can be calculated as follows:

$$r_{i,j} = \frac{\sum_{k=1}^{|\mathcal{U}|} \left(b_{i,k} - \overline{b_i}\right) \left(b_{j,k} - \overline{b_j}\right)}{\sqrt{\sum_{k=1}^{|\mathcal{U}|} \left(b_{i,k} - \overline{b_i}\right)^2 \sum_{k=1}^{|\mathcal{U}|} \left(b_{j,k} - \overline{b_j}\right)^2}} \tag{4.5}$$

with the number of uses cases $|\mathcal{U}|$, the bandwidth share figures $b_{i,k}$ and $b_{j,k}$ for the bandwidth in Use Case k for the i-th and j-th link respectively, and $\overline{b_i}$ and $\overline{b_j}$ being the arithmetic means of the bandwidth vectors.

A correlation matrix $\mathbf{R} \in \mathbb{R}^{|\mathcal{L}| \times |\mathcal{L}|}$ is defined which contains all the correlation coefficients for all possible link pairings for a specific layer crossing:

$$\mathbf{R} = \frac{1}{|\mathcal{U}| - 1} \left(\mathbf{B} - \bar{\mathbf{b}} \mathbf{1}_{|\mathcal{U}|}^{\mathbf{T}}\right) \left(\mathbf{B} - \bar{\mathbf{b}} \mathbf{1}_{|\mathcal{U}|}^{\mathbf{T}}\right)^{\mathbf{T}} \tag{4.6}$$

where $\mathbf{B} \in \mathbb{R}^{|\mathcal{L}| \times |\mathcal{U}|}$ is the data matrix that contains all of the bandwidth vectors as column components, hence $\mathbf{B} = \left[\mathbf{b}_1 \, \mathbf{b}_2 \ldots \mathbf{b}_{|\mathcal{L}|}\right]$. $\bar{\mathbf{b}}$ is a column vector containing the arithmetic means of all bandwidth vectors, hence $\bar{\mathbf{b}} = \left[\overline{b_1} \, \overline{b_2} \ldots \overline{b_{|\mathcal{L}|}}\right]^T$. $\mathbf{1}_{|\mathcal{U}|}$ is the all-ones vector of size $|\mathcal{U}|$ (number of uses cases).

The term $\mathbf{B} - \bar{\mathbf{b}} 1_{|\mathcal{U}|}^T$ yields the centered data matrices containing the bandwidth values. Therefore, (4.6) can also be written using centering matrices:

$$\mathbf{R} = \frac{1}{|\mathcal{U}| - 1} \mathbf{B} \mathbf{C}_{|\mathcal{U}|} \mathbf{B}^T \mathbf{C}_{|\mathcal{U}|}^T \tag{4.7}$$

where $C_{|\mathcal{U}|}$ is the centering matrix of size $|\mathcal{U}|$.

The cross-correlation coefficient matrix $\mathbf{R}$ will contain values between -1 and 1. A positive value in the i-th row and the j-th column indicates that links i and j are mainly active in a similar share of use cases, and a negative value indicates that they are utilized in different/orthogonal use cases. Those link pairings having negative cross correlation factors are best suitable for link multiplexing.

For spectral clustering, an affinity matrix is required. The bandwidth correlation matrix forms one part of this matrix (the other one is formed by the spatial distance, see next section). In terms of clustering vertical links together to be operated

in a multiplexed fashion, a high affinity value is assigned to links with negative correlation values.

4.1.5.2 Spatial Distance of Link Crossing Points

Besides the temporal traffic correlation of the links, their spatial distance is incorporated in the clustering trade-off. Links that cross the same layer boundary in close proximity should be more likely to be grouped since costs for the wiring overhead is then low. At this point, of course, only the horizontal distance matters since the link clustering is performed layer by layer. A problem is that a link naturally does not have a position assigned to it nor is it contained in the floorplan. Therefore, we use a theoretical layer crossing point for the distance calculations. This point is located at the center point of a direct line from the source to the destination (see Fig. 4.6).

To capture the distances between all links at a layer boundary layer, a link distance matrix $\mathbf{D} \in \mathbb{R}^{|\mathcal{L}|}$ is formed where the element $d_{i,j}$ denotes the metric distance of the layer crossing point of link l_i to the one of link l_j (with $\mathcal{L}$ being the set of all links crossing the currently considered layer boundary). This matrix forms the second component of the link affinity matrix besides the bandwidth correlations.

If $\mathbf{p}_s = \mathrm{Pos_s}(l)$ gives the position vector of the source node of link l and $\mathbf{p}_d = \mathrm{Pos_d}(l)$ yields the position of its destination node, then, linearly calculating the crossing point gives:

$$\mathbf{p}_{c,i} = \frac{b_c - \mathrm{L_s}(i)}{\mathrm{L_d}(i) - \mathrm{L_s}(i)} \left(\mathrm{Pos_s}(i) - \mathrm{Pos_d}(i)\right) + \mathrm{Pos_s}(i) \tag{4.8}$$

with $\mathrm{L_s}(l)$ being the layer of the source node of link l, $\mathrm{L_d}(l)$ being the layer of the destination node and b_c the currently handled layer boundary. In the scale of layers, the layer boundaries are intermediate layers, e. g. the boundary between layer 1 and layer 2 would be denoted as $b = 1.5$.

$$d_{i,j} = ||\mathbf{p}_{c,i} - \mathbf{p}_{c,j}|| \tag{4.9}$$

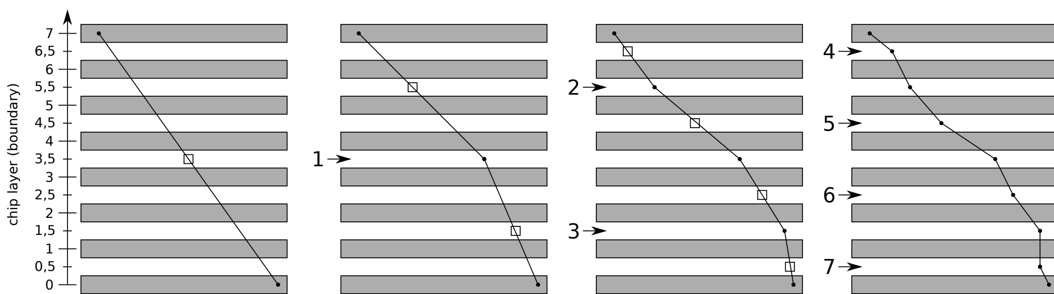


Fig. 4.8.: Successive preliminary TSV-Hub placement (only one horizontal dimension is shown)

If a vILL is traversing multiple layer boundaries, it has to be routed over multiple TSV-Hubs (one at each layer boundary). In a scenario where a dedicated TSV-Hub

(one that is not transporting any other links) is inserted for the link at each boundary, all the hubs are (preliminarily) placed on a straight line from the source to the destination node. However, in most cases, the hubs are transporting multiple links, and a consensus position has to be found. To this end, the hubs are placed layer by layer in a binary-tree fashion (see Fig. 4.6). In Fig. 4.8 the process is shown for an (academic) example of an eight-layer system and a vILL traversing the full stack. First, the position where the link is crossing the middle boundary ($b = 3.5$) is calculated. Next a TSV-Hub is inserted, however, since this hub is also continuing other links (which are not shown), the hub is not placed at exactly the crossing point. By placing the hub, the vILL is cut in half, forming two new links. Thus, when performing the link clustering for boundary $b = 5.5$ in the second step, we get $\mathrm{L_s}(l) = 3.5$ as new source layer since the link is now terminated at the newly inserted TSV-Hub. The process continues in the same pattern until all gaps are closed.

4.1.5.3 Link Affinity

For the clustering of links at a layer boundary b with spectral clustering, an affinity matrix $\mathbf{A}_b \in \mathbb{R}^{|\mathcal{L}|\times|\mathcal{L}|}$ is created.

An element $a_{i,j}$ of this matrix results from a weighting function based on the traffic correlation and the spatial crossing point distance:

$$a_{i,j} = \frac{c_r}{(r_{i,j} + 1)} + \frac{c_d}{d_{i,j}} \tag{4.10}$$

the coefficients c_r and c_d are parameters again to tune the algorithm.

The link affinity matrix can also be seen as an undirected graph $U_b = (\mathcal{L}_b, \mathcal{W}_b)$, where the nodes $\mathcal{L}_x$ represent the links traversing the layer boundary b and the edges w_x represent a tupel (r, d) of the link's bandwidth correlation coefficient r and spatial distance of their crossing points d.

The link affinity matrix forms the basis for the spectral clustering algorithm to create the link bundles.

4.1.5.4 TSV-Hub Preliminary Placement

The final placement of the TSV-Hubs has to be carried out by a 3D-floorplanner after the network synthesis. A conventional 2D-floorplanner is not sufficient since TSV-Hubs have to be placed on two adjacent chip layers at a common position.

However, the algorithm yields a preliminary position for each TSV-Hub. This position is computed as a "center of mass" based on the link sizes connected to the TSV-Hub. Hence, the hub is shifted into the direction of high capacity links in order to reduce overall wire length.

The preliminary TSV-Hub position is calculated as follows:

$$\mathbf{p}_{tsv} = \frac{1}{B} \sum_{\{i:l_i \in \mathcal{L}_h\}} \beta_i \mathbf{p}_i \tag{4.11}$$

with the set of all links connected to the hub $\mathcal{L}_h$ and the position vector of link i $\mathbf{p}_i$. B represents the sum of all bandwidth requirements of all links connected to the hub, hence $B = \sum_{\{i:l_i \in \mathcal{L}_h\}} \beta_i$.

Fig. 4.9 depicts an example showing the preliminary placement of a TSV-Hub transporting four different links with different bandwidth requirements, hence implemented with different link sizes. The position of the hub is shifted towards the 64-bit-wide link.

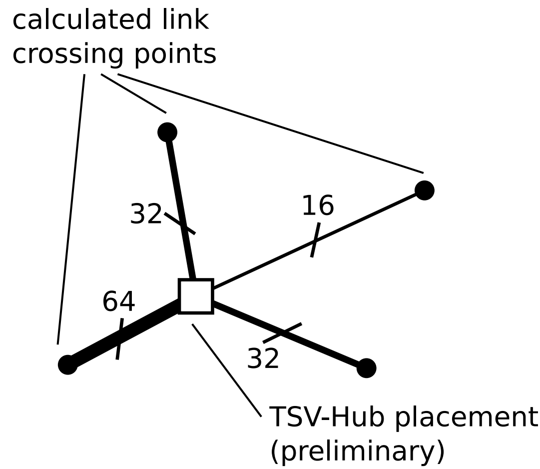


Fig. 4.9.: The TSV-Hub position is computed as "center of mass", based on the link bandwidth and the theoretical layer crossing points of the individual length

4.1.5.5 Determining Optimal Number of Vertical Link Bundles

Determining the number of clusters is not part of the spectral clustering algorithm. In the spectral clustering run for partitioning the 2D network, the method of the maximum eigengap is used which provides a natural measure for the number of clusters [180].

Another way is a brute-force solution: Here we sweep from the smallest possible cluster count (a single cluster) to the maximum one (as many clusters as elements), perform spectral clustering for this size and afterwards calculate the cost of this solution. This approach is not feasible for the network partitioning for two reasons: Firstly, the number of PEs is usually large compared to the inter-layer-links at a single layer (hence a large sweep space) and secondly, to calculate the actual network cost, the subsequent RPA phase had to be carried out for each configuration. The RPA step, however, is time consuming.

In contrast, for the inter-layer link clustering, we expect a manageable number of links to be clustered. Furthermore, the cost assessment of the created partitioning is of low complexity. It comprises two categories:

- Horizontal interconnect overhead
- TSV-Hub complexity

The lower the number of clusters, the higher the horizontal interconnect overhead. This is because the TSV-Hub has to be placed at a consensus point (computed as described in section 4.1.5.4) when more than one link is transported over the array, hence the conventional wiring is longer for each link since it must be routed over the array. If a TSV-Hub is only servicing a single link, this hub can be optimally placed at the link's crossing point, and this hub does not induce an overhead at all. When the maximum number of clusters is used, this is the case for every hub, so there is then no horizontal wiring overhead at all.

The TSV-Hub complexity is also a function of the cluster count. With more clusters more TSV-Hubs are needed. However, for higher cluster counts, the complexity of an individual hub is expected to be smaller compared to one of a partitioning with fewer clusters. This is because the hub of the low cluster count variant has to continue more links and therefore needs more complex switching logic.

The model for computing the TSV costs is based on the findings of Chapter 3.3 and includes:

- Fixed costs for control logic
- Variable costs for control logic
- Variable costs for TSVs
- Variable costs for multiplexers
- Variable costs for buffering
- Variable costs for synchronization
- Variable costs for switch boxes (in case a fault tolerance design is targeted)

4.1.5.6 TSV-Hub Configuration

For each link cluster, a TSV-Hub has to be instantiated and is configured with the following parameters:

- Number of ports
- Link size per port
- Number of TSVs
- Preliminary position

The number of ports is already given by the link count in the cluster and is straightforward.

For scaling the capacity of the TSV-Hub and thus the number of TSVs, we search for the use case placing the highest load on the hub. To this end, all bandwidth vectors of a cluster are summed up to form an aggregated bandwidth vector containing the overall share of cluster within the individual use cases:

$$\mathbf{b}_C = \sum \{\mathbf{b}_i : l_i \in \mathcal{C}\} \tag{4.12}$$

where $\mathcal{C}$ denotes the set of vILL contained in the cluster.

From this vector, the element with the highest value gives the bandwidth requirement for the use case with the highest utilization, and therefore the capacity the hub has to provide:

$$b_{tsv} = \max_{b_i \in \mathbf{b}_C} b_i \tag{4.13}$$

The preliminary position $\mathbf{p}_{tsv}$ is calculated as described in section 4.1.5.4. Finally, the TSVs have to be placed by using a 3D-floorplanner together with other IP-cores.

Besides the individual parameters, which are different for each hub, global TVS-Hub parameters are defined system wide:

- The intra-layer interconnect clock frequency (NoC clock) f_{ic}
- The targeted TSV clock frequency f_{tsv}
- The type of synchronization
- The maximum number of allowable faulty TSVs in a hub k_{max}. This is expressed as a per TSV rate k_{max}/n_d since the TSV number is variable.

4.1.5.7 Algorithm Summary

The full algorithm for the link clustering is given in Algorithm 2. It is formed of a two-level hierarchy: The procedure *ClusterLinks* (line 2 to 11) determines the order in which clustering for the individual layer boundaries is completed. It divides the stack in the middle and starts with the middle boundary (or the layer above in case of an equal number of boundaries). This is continued for the remaining sections, such that the sequence is carried out in form of a binary tree until all layer boundaries have been clustered.

The *ClusterLinks* procedure calls the *ClusterBoundary* procedure (line 12 to 39) for each layer, forming the second level in the hierarchy. Here, first the column vectors $\mathbf{b}_{l,u}$ of the bandwidth matrix $\mathbf{B}_l \in \mathbb{R}^{|\mathcal{L}_\lfloor| \times |\mathcal{U}|}$ are computed (line 13 to 21), where $\mathcal{L}_\lfloor$ denotes the set of inter-layer-links at this layer boundary and $\mathcal{U}$ denotes the set of use cases.

In the next step (line 22 to 28), based on the bandwidth matrix, the bandwidth cross correlation matrix $\mathbf{R} \in \mathbb{R}^{|\mathcal{L}_{\lfloor}| \times |\mathcal{L}_{\lfloor}|}$ is computed. In the same loop, the distances between all link crossing points are stored in the link distance matrix $\mathbf{D} \in \mathbb{R}^{|\mathcal{L}_{\lfloor}| \times |\mathcal{L}_{\lfloor}|}$.

Next, both bandwidth correlation matrix and distance matrix are subject to a weighting function (line 30) which outputs the affinity matrix $\mathbf{A} \in \mathbb{R}^{|\mathcal{L}_{\lfloor}| \times |\mathcal{L}_{\lfloor}|}$ to be used in the spectral clustering algorithm. Here, this step also includes determining the optimal cluster count as described in section 4.1.5.5. The clustering step (line 31) yields the partition matrix $\mathbf{C} \in \mathbb{R}^{|\mathcal{L}_{\lfloor}| \times |\mathcal{B}_{\lfloor}|}$, where $\mathcal{B}_{\lfloor}$ denotes the set of clusters. In $\mathbf{C}$ $c_{i,j} = 1$ implies that link l_i is part of cluster C_j and $c_{i,j} = 0$ implies that it is not.

Multiplying the transposed version of the partition matrix with the bandwidth matrix (line 32) gives the cluster-bandwidth matrix $\mathbf{M} \in \mathbb{R}^{|\mathcal{B}_{\lfloor}| \times |\mathcal{U}|}$ which denotes the bandwidth requirement of a full link bundle for a specific use case. This information is used in the next step for dimensioning the TSV-Hubs.

In the final step (line 33 to 38) a TSV-Hub is configured for each bundle, hence for each row in M.

Algorithm 2 Clustering of Vertical Links

```
1:  //Iterate over all layer boundaries
2:  procedure ClusterLinks(l_m in, l_m ax)
3:      b ← ⌈(l_max − l_min) /2 + l_min − 0.5⌉ + 0.5
4:      ClusterBoundary(b)
5:      if l_max − b > 1 then
6:          ClusterLinks(b + 0.5,l_max)
7:      end if
8:      if b − l_min > 1 then
9:          ClusterLinks(l_min,b − 0.5)
10:     end if
11: end procedure

12: procedure ClusterBoundary(b,𝓛_b)
13:     //Compute bandwidth matrix
14:     for all f ∈ 𝓕_u do
15:         𝓟_{u,f} ← path(f)
16:         for all l ∈ 𝓛_b do
17:             if l ∈ 𝓟_{u,f} then
18:                 b_{l,u} ← b_{l,u} + β_{u,f}
19:             end if
20:         end for
21:     end for

22:     //Compute traffic correlation and link distances
23:     for all l_i ∈ 𝓛_b do
24:         for all l_j ∈ 𝓛_b do
25:             R(i, j) ← corr(b_i, b_j)
26:             D(i, j) ← dist(l_i, l_j)
27:         end for
28:     end for

29:     //Perform weighting and clustering
30:     A ← weight(R, D)
31:     C ← spectralClustering(A)
32:     M ← C^T × B

33:     //instantiate and configure TSV arrays
34:     for all k ∈ rows(M) do
35:         tsv_{k,bw} ← max(k)
36:         tsv_{k,pos} ← centerOfMass(k)
37:         CreateTsvArray(tsv_{k,bw},tsv_{k,pos})
38:     end for
39: end procedure
```

4.2 Synthesis Output

The 3D-NoC synthesis software generates the following types of output:

- An **XML description** of the generated system, in order to finally place the IP-Cores, routers and TSV-Hubs. In addition, it serves for evaluation of the generic
- A **SystemC [181] simulation model** of the constructed NoC with traffic generators and sinks. This generated model is cycle accurate.
- A **synthesizable RTL** model of the constructed NoC

The XML description is a structural description of the generated system, extending the original input XML file. In addition to the input file, it now contains NoC-Routers and TSV-Hubs and the network links between the PEs, routers and hubs. The original channel items are annotated with path data, defining the mapping of each channel on a sequence of links. This information serves as basis for the construction of the routing tables for the NoC-routers. Furthermore, the XML description serves as input to a final floorplanning stage.

The SystemC model provides a cycle accurate timing simulation of the generated 3D-NoC. For each channel, traffic is injected at its source node. This traffic is modeled with an Interrupted Poisson Process (IPP) traffic generator. Such a traffic generator is based on a Markov chain of two states, one *generating* state and one *not-generating* state (Fig. 4.10).

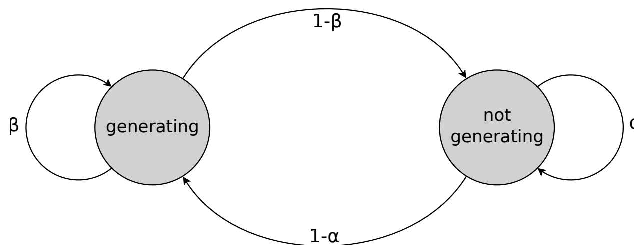


Fig. 4.10.: Markov modeled IPP

In the *generating* state a flit is emitted in each cycle and in the *not-generating* state it is not. For each state a probability (α or β) is defined which gives the probability that the state is not left in the next cycle. With the two probabilities both the emitted traffic rate and the average burst/packet length can be set. A full derivation of the probabilities can be found in Appendix A.3.

Like the SystemC based simulation model, the RTL model also forms a structural description of the synthesized network. Here, a model described with the synthesizable subset of the Hardware Description Languages (HDLs) *VHDL* (for TSV-Hubs) and *Verilog* (Routers) is generated. The routers are adapted from the *LISNOC* project [36]. Note that this model does only contain the on-chip network, not the full system

including the IP-Cores. The PEs are only contained in form of traffic source and sink dummies.

4.3 Experimental Results

The synthesis flow and the software tool implementing the flow were validated with multiple examples. In this section, we first present a simple example to study the complexity of the inserted TSV-Hubs with respect to the extent of vertical link bundling. Next, more complex randomly generated systems are simulated to evaluate the impact of the inserted TSV-Hubs on the NoC timing. Finally, we outline a case study for the synthesis of a heterogeneous mobile SoC. The input data in terms of communication requirements and IP-core count where obtained from a mobile SoC vendor. Five different use cases were defined by the vendor for this system (e. g. Video playback, LTE communication, etc.).

4.3.1 Complexity and Costs of Vertical Link Bundling

Bundling vertical links via TSV-Hubs increases the complexity of the system and adds additional circuitry. Consequently, while costs are saved on one side by reducing TSV count, other costs are increased.

To study the impact of vertical link bundling, we consider a basic example of a two-layer system with 16 independent flows. For this, 32 PEs are required, forming 16 independent pairs of two PEs each, connected by a single flow. The parameters of the initial clustering process and the RPA phase, as well as the distances between the PEs, have been selected such that each PE pair forms a cluster and a router is inserted for each pair. Consequently, 16 virtual vertical links are crossing the layer boundary. This is depicted in Fig. 4.11.

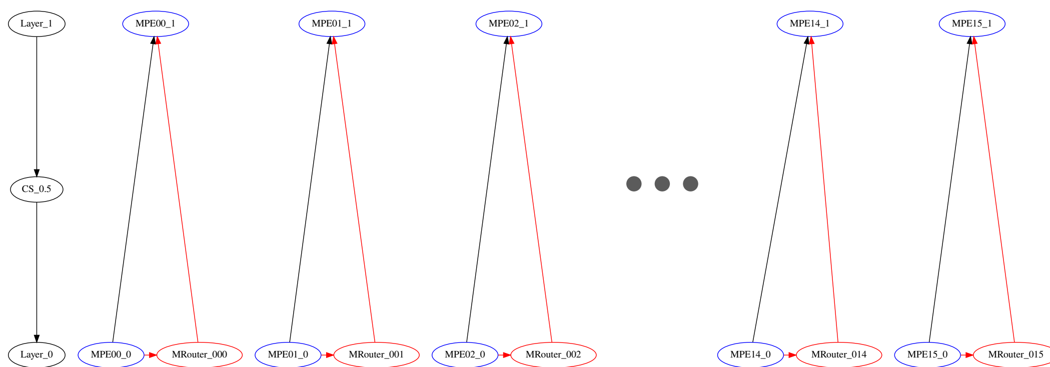
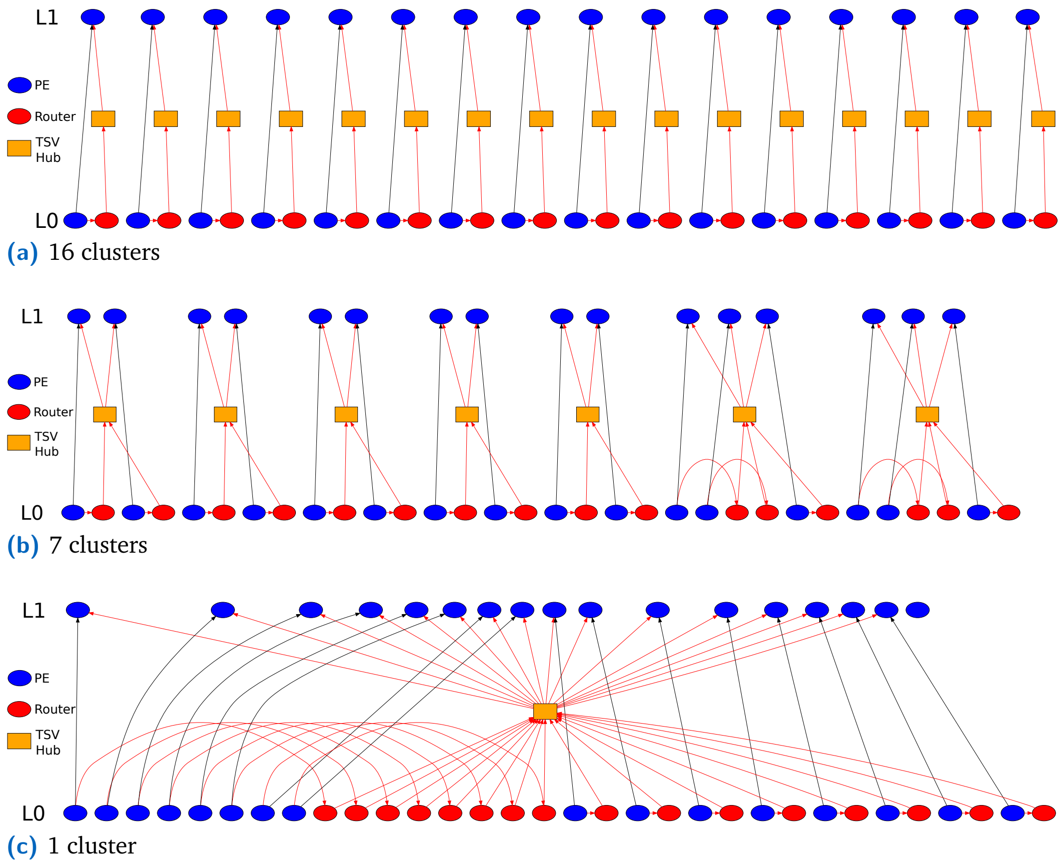


Fig. 4.11.: Basic system with 32 PEs forming 16 independent pairs. After the RPA phase 16 virtual links crossing the layer boundary are present. This system serves as input for the vertical link clustering phase. Black lines represent flows, red lines represent network links.

For the vertical continuation of the layer crossing links, TSV-Hubs are inserted by the synthesis tool. The number of links per hub and therefore the hub count is determined by the algorithm such that overall costs are minimized. Fig. 4.12 shows

the final output for a cluster count of 16 (Fig. 4.12a), 7 (Fig. 4.12b) and 1 (Fig. 4.12c) clusters.

Using a low cluster count results in larger link bundles and allows to perform statistical multiplexing, however wiring overhead is added.



(a) 16 clusters

(b) 7 clusters

(c) 1 cluster

Fig. 4.12.: Synthesis output for different cluster counts. The black lines represent flows, the red lines represent network links.

For this experiment, the algorithm has been modified such that a sweep over all possible hub counts (1 to 16) is performed and the costs are tracked. The results are given in Fig. 4.13. The overall costs for each case can be split into three portions:

- The **hub logic costs** account for the serialization, multiplexing and control logic of all TSV-Hubs inserted by the synthesis tool. The underlying cost model reflects the complexity analysis as described in Chapter 3.2. In Fig. 4.13 a slight increase of the logic costs towards higher cluster counts is observable. This can be explained as follows: The logic costs can be further broken down into a fixed (per hub) part and one that accounts for the multiplexing and serialization logic. The fixed part is mainly induced by the scheduler and control logic. The multiplexer and serialization logic, however, is independent from the cluster count (at least for the single use case situation).
- The **TSV costs** are induced by the area consumption of the TSV array, hence the actual diameter and Keep-out-Zones. In the considered scenario, this cost

is equal for all cluster counts, since the total bandwidth that has to be provided by the TSV-Hubs is always the same. The TSV costs become dependent on the cluster count and size once multiple uses cases are considered (see below).

- Bundling multiple vertical links for the continuation by a common TSV-Hub requires rerouting of conventional wires, and therefore, **2D interconnect costs** are induced. In Fig. 4.13 the highest 2D interconnect costs are observed for the single cluster scenario. Towards higher cluster counts, these costs drop significantly. Finally, for the case of a single hub for each vertical link (16 clusters), no 2D interconnect overhead is present anymore, since the hubs can be placed at the optimal position.

The individual costs are subject to weighting factors which can be used for tuning the synthesis process. In Fig. 4.13 the costs are normalized to the single cluster, single use case scenario.

With the used weighting factors the synthesis flow would select six clusters, since this is the solution with the lowest overall cost. Note, however, that this can be different with different weighting factors.

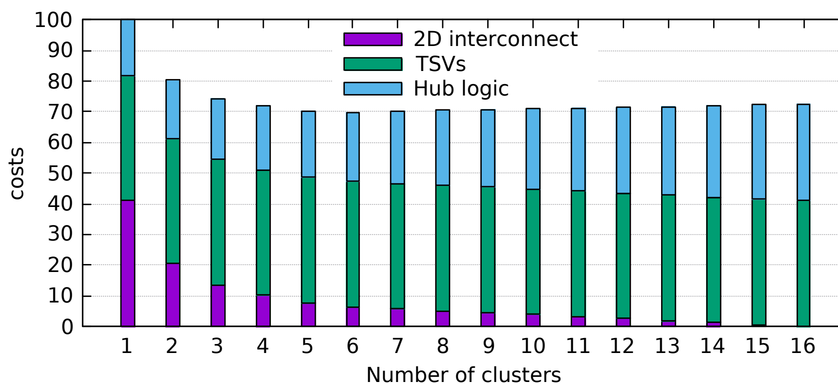


Fig. 4.13.: Weighted costs with respect to cluster count for a single use case system with 32 PEs as depicted in Fig. 4.11

When considering multiple use cases, the TSV costs can be reduced. This is shown in Fig. 4.14 and Fig. 4.15. The results are normalized with the same factor as in Fig. 4.13. With multiple use cases, not all flows are active at the same time and, depending on the mapping of the flows on the network link, the utilization varies with the currently active use case. This information can be used for optimizing the TSV-Hub configuration to perform statistical multiplexing (see Chapter 4.1.5).

In Fig. 4.14 and Fig. 4.15 all subgraphs show the costs for a specific vertical link cluster count for the system described above. Hence, each subgraph represents a single bar of Fig. 4.13, but the results are given for different use case counts each from 1 to 16 use cases. In this simple example, each flow is mapped onto a sequence of only two network links and each network link has only a single flow mapped onto it. For the scenario of a single use case (first bar in each graph), all flows are

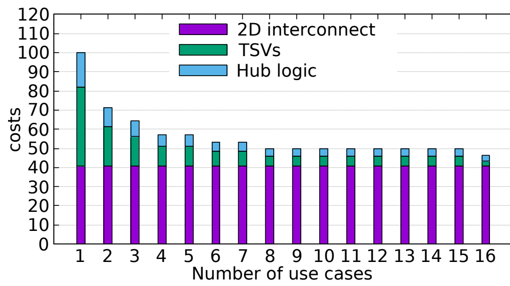
assigned to the same single use case. For the scenario of 16 use cases (last bar in each graph), each use case is only used for a single flow. For the cases in between, the use cases are assigned to the flows in round-robin fashion.

In Fig. 4.14a only a single hub is present. For the single use case scenario, this hub has to provide the total sum of the bandwidth of all links. With more use cases and the knowledge that some links are now active mutual exclusively, the TSV bandwidth can be reduced.

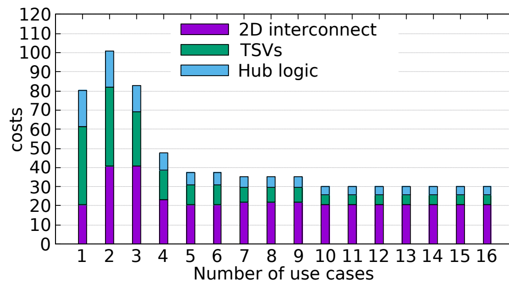
Using 16 different clusters forms the other corner case as depicted in Fig. 4.15b. Here TSV costs cannot be reduced, since each flow is using its dedicated TSV-Hub, and no statistical multiplexing can be performed. For all other cases, TSV costs are reduced with higher numbers of use cases.

The interconnect overhead varies with the use case count for most situations. This is a result of the individual link cluster composition. The algorithm assigns a strong affinity to links which carry flows of different use cases to increase the gain of the statistical multiplexing. Consequently, also links which have a larger spatial distance of their layer crossing points are sometimes clustered if this is beneficial for the TSV costs. The spatial difference and traffic correlation form antagonists, and both contribute to the affinity matrix used for the clustering (see Chapter 4.1.5).

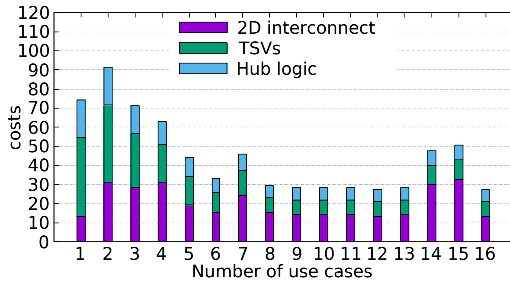
Note that the number of uses case is neither a parameter to the synthesis flow, nor can it be chosen by the algorithm itself. The use cases are a part of the input model describing the systems communication behavior.



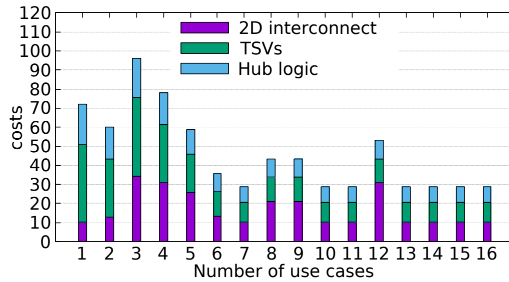
(a) 1 cluster



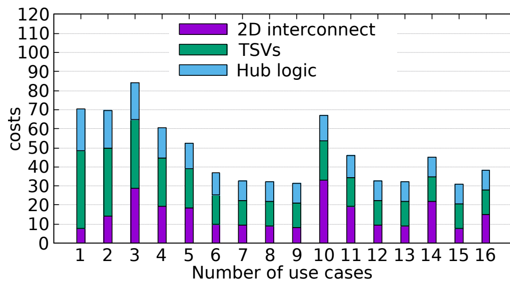
(b) 2 clusters



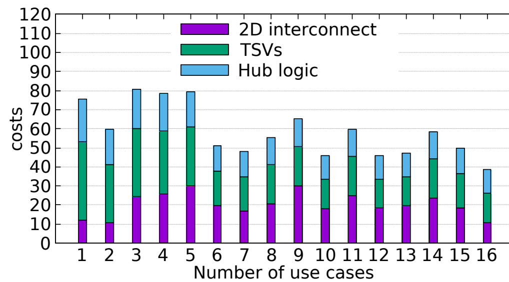
(c) 3 clusters



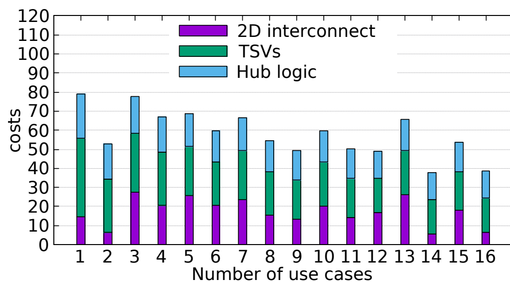
(d) 4 clusters



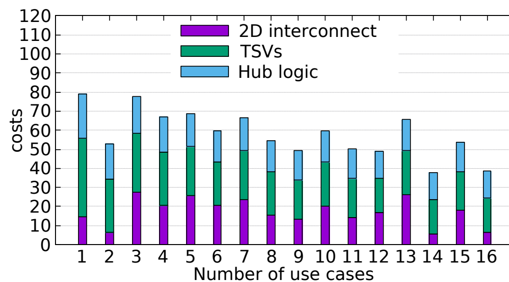
(e) 5 clusters



(f) 6 clusters

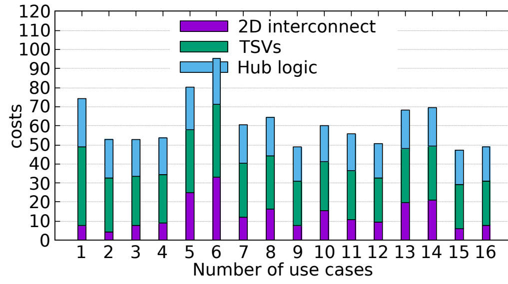


(g) 7 clusters

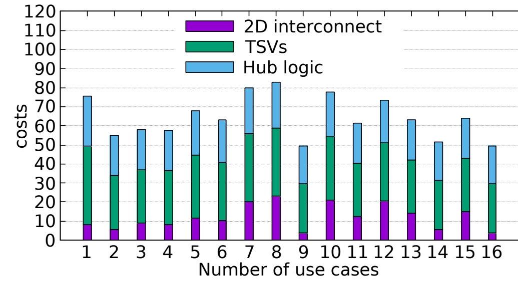


(h) 8 clusters

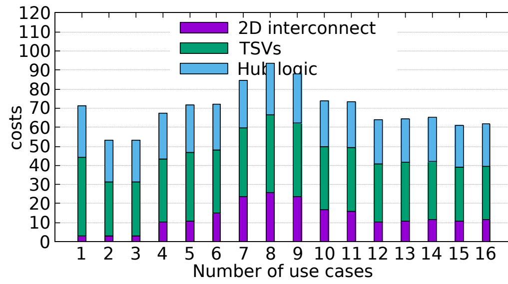
Fig. 4.14.: Weighted costs with respect to number of use cases for cluster counts from 1 to 8 clusters



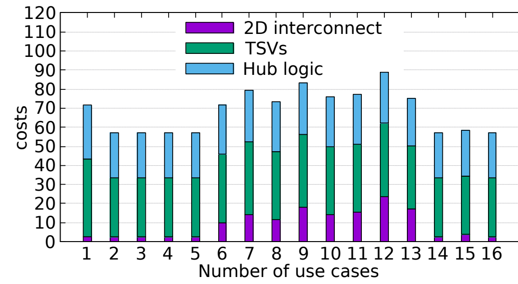
(a) 9 clusters



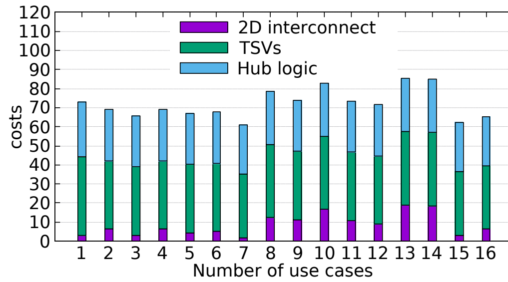
(b) 10 clusters



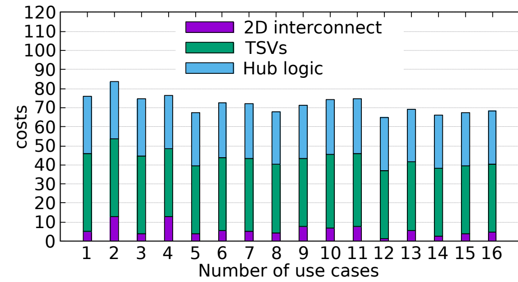
(c) 11 clusters



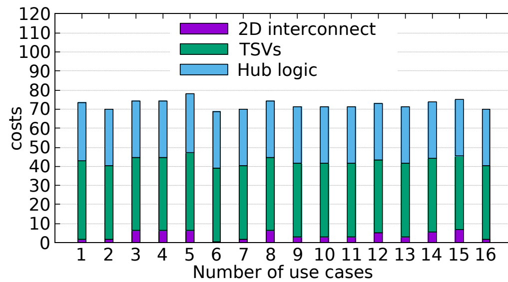
(d) 12 clusters



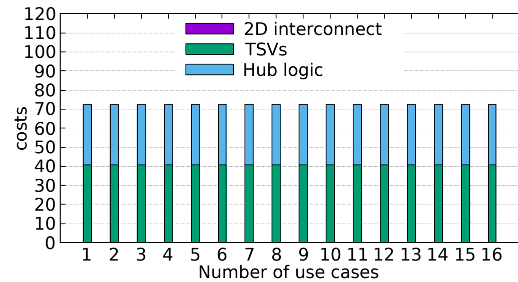
(e) 13 clusters



(f) 14 clusters



(g) 15 clusters



(h) 16 clusters

Fig. 4.15.: Weighted costs with respect to number of use cases for cluster counts from 9 to 16 clusters

4.3.2 Latency Analysis

Adding TSV-Hubs to the on-chip network clearly impacts the packet latency since additional buffer stages and queues are added to the network channels. This impact is studied in this section. To this end, the flit latency of the 3D network is compared to a baseline 2D network. This 2D network always reflects the same network topology and communication patterns as the 3D network but vertical links are considered as 2D links. Note, however, that the baseline 2D system is an academic construct. Distributing all cores on a single layer would induce much more horizontal wiring. The 2D baseline system can therefore also be seen as a 3D system but without TSV-Hubs being used, hence a dedicated TSV for each wire of each layer-crossing link, which is, however, also not feasible in most cases because of the high TSV count.

When increasing the generation rate, the network is driven into congestion for both the 3D and the 2D version. The expectation regarding the 3D-network is that the network does not become congested significantly earlier, and that we only see a slight increase of the delay when the network is in the non-congested state. When the layer count is increased, we expect an increase of the latency since more layer boundaries have to be crossed in average.

Furthermore, we expect to not encounter any deadlocks since the TSV-Hub insertion does not alter the network topology, and this topology is by construction deadlock free.

A major challenge at this point is to find a suitable reference system on which latency analysis can be performed. Using regular mesh NoC topologies is not practical as they do not use links with different capacities and application specific router configurations or use cases. Moreover, the clustering approach is not meaningful with regular systems.

Analyzing real world examples is also problematic since many different variants are needed to produce representative results.

Another option is to use randomly generated input models. This allows to carry out any amount of simulations and syntheses and finally average over the results and check confidence intervals. To this end, a model generator was developed. It creates a set of input models with the following parameters:

- Number of cores
- Number of layers
- Average burst size
- Average load
- Minimal and maximal bandwidth of channels
- Chip side length

- Number of use cases
- Minimal and maximal number channels per core

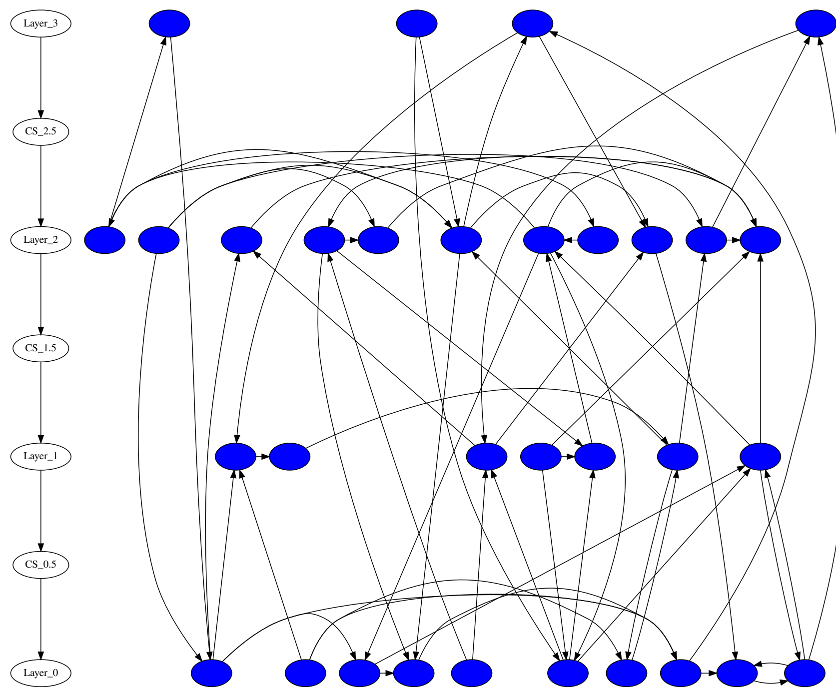
The generation process is then performed as follows:

1. The configured number of PEs are randomly distributed on the chip within the bounds give in the parameter set (number of layers and side length).
2. For each PE, a random value is generated defining the number of channels originating from the core (within the bounds set in the parameters). For each channel, a randomly selected PE acts as destination.
3. Each channel is configured with the average burst size and a bandwidth parameter that is randomly selected between the bounds of the configuration parameters
4. Each channel is randomly assigned to one of the available use cases

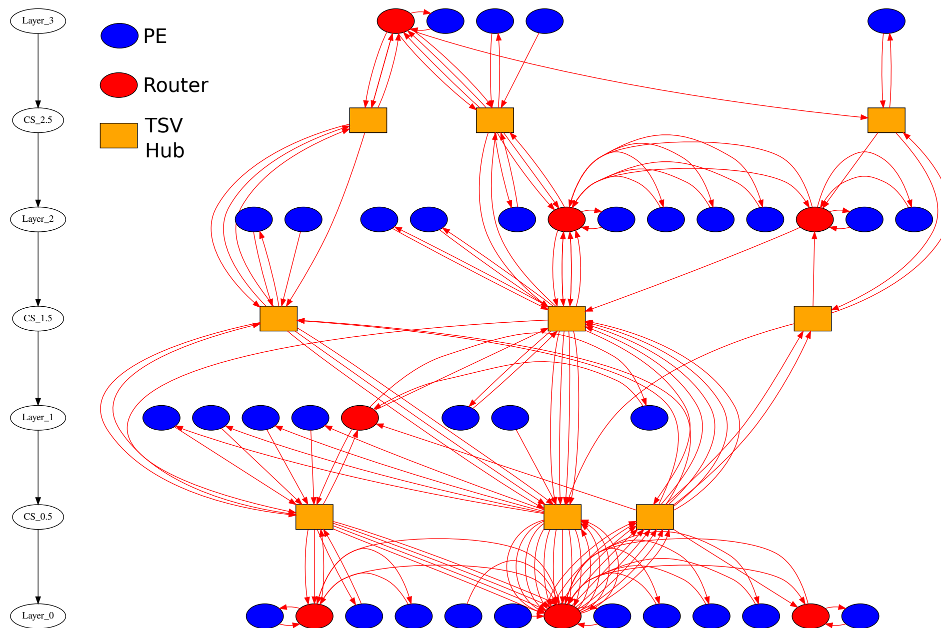
Fig. 4.16 exemplary shows a possible auto-generated 4-layer system with 32 Cores and between 1 and 4 channels originating from each PE. Fig. 4.16a shows the system's CCG, and Fig. 4.16b depicts the synthesis result for this system in form of the 3D network graph (including routers and TSV-Hubs).

For all scenarios discussed below, a batch generation of 24 different random systems was carried out. Next, a 3D-NoC synthesis was performed for every system. In this step also a SystemC simulation model was created for each system. Then, a batch simulation of each of the generated systems was performed with a different flit generation rate per run. In each run, the generation rate of each channel source was increased by 1.25% (from 1.25% to 98.75%). For each single run, all flit latencies were measured for a simulation time of 100 000 cycles (of the 400 MHz interconnect clock), and an average was formed. Finally, for each generation rate step, an average over all 24 simulated systems was formed. Here, we also captured the variance to show the confidence intervals in the graphs.

Fig. 4.17 shows the result of a system with only 4 cores distributed on 4 (Fig. 4.17a) and 10 (Fig. 4.17b) layers, respectively. In this scenario, infinite buffer was assumed. The graphs show that in the non-congested range the average latency is increased by a few cycles. This is a natural consequence of the additional register stages in the TSV-Hubs. For the 10 layer system (note that this is a theoretical example), this increase is higher since more layer boundaries have to be crossed. Both systems, the 3D-NoC and the 2D baseline NoC are congesting with a comparable pace when generation rate is increased. We see a fairly high variance on the latency measures of the 3D system. This is explainable with the low number of cores. Among the 24 systems there will be some where most cores are located in close vertical proximity and not many layer boundaries have to be crossed. On the other hand, there will be some systems with maximized vertical distance. We expect this variance to be lowered for systems with a higher PE count, due to the *law of large numbers*.

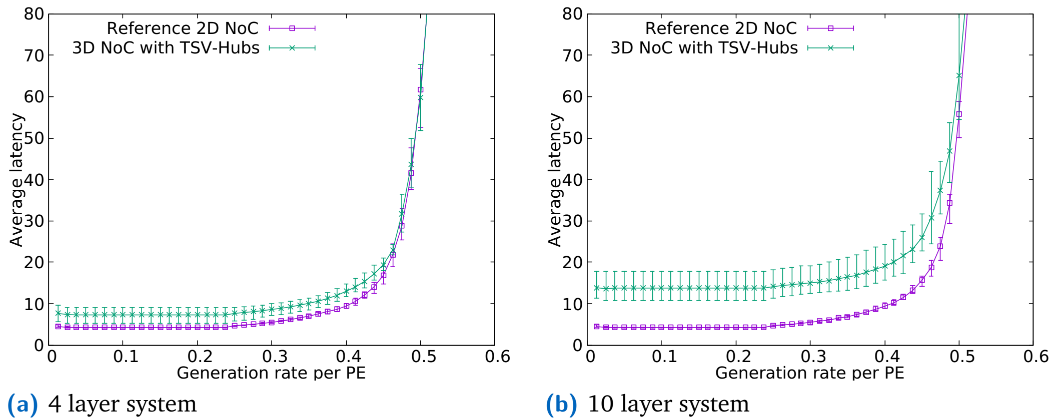


(a) Core Communication Graph



(b) Core Communication Graph

Fig. 4.16.: Auto-generated communication graph with 4 layers, 30 cores, between 1 and 4 channels originating at each core. All use cases are shown.

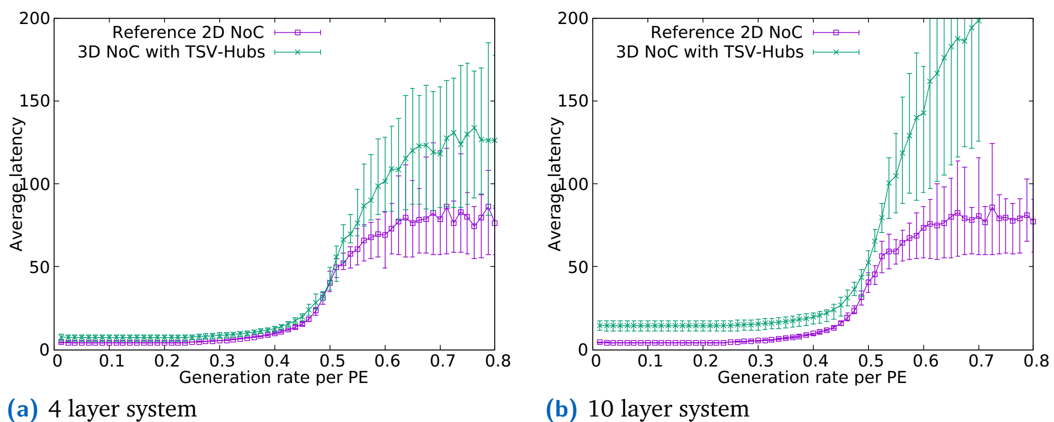


(a) 4 layer system **(b)** 10 layer system

Fig. 4.17.: Average flit latency in a 3D system with 4 cores and infinite buffer space compared to a 2D reference NoC

Fig. 4.18 shows again the system of 4 cores distributed on (Fig. 4.18a) and 10 (Fig. 4.18b) but this time with a limited buffer space of 30 words per link. As expected, we see that average flit latency is cut-off at a threshold when the generation rate is increase beyond the point where the system gets congested and buffers are completely filled most of the time. Consequently, in this region, not all newly generated packets can be inserted in the network at the traffic generators anymore and have to be discarded at the cores network interface. We also see that this threshold is higher for the 3D-system. This is a result of a higher amount of overall buffer space in the 3D system since here not only the routers offer buffering but also the TSV-Hubs.

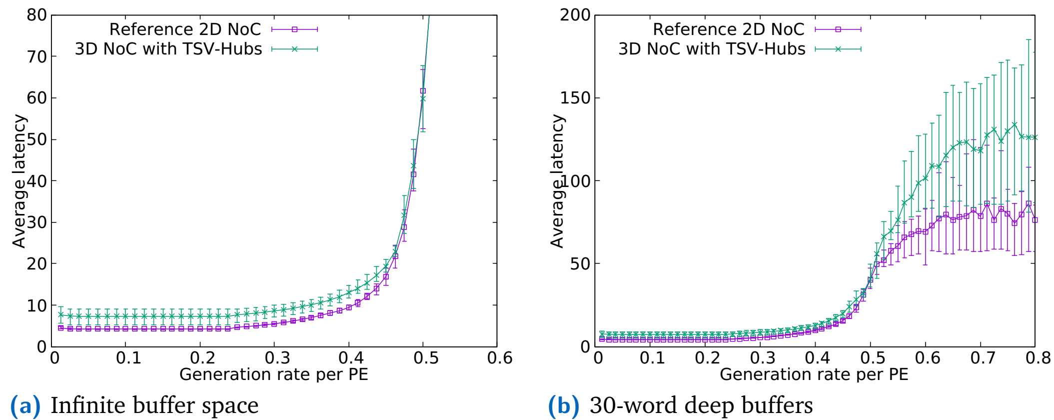
We see that the average latency varies widely over the generated systems in the congested region. This is due to the fact that the latency threshold depends on the amount of buffer space, which again depends on the number of routers, TSV-Hubs and their link count and is different for each of the generated systems.



(a) 4 layer system **(b)** 10 layer system

Fig. 4.18.: Average flit latency in a 3D system with 4 cores and with 30 words deep queues in routers and hubs compared to a 2D reference NoC with same configuration

Finally, Fig. 4.19 shows the results for a system with 64 cores being distributed on 4 layers. The infinite buffer space scenario is shown in Fig. 4.19a, and the results for a system with 30-word long queues per link are shown in Fig. 4.19b. As expected, the higher number of cores leads to smaller confidence intervals since the likelihood that many cores are concentrated on a single layer is much smaller now. The results confirm that inserting TSV-Hubs does neither lead to premature congestion nor to deadlocks. The inevitable latency increase of few clock cycles is traded-off for a reduced TSV-count and saved costs.



(a) Infinite buffer space

(b) 30-word deep buffers

Fig. 4.19.: Average flit latency in a 3D system with 64 cores distributed on 4 layers compared to a 2D reference NoC with same configuration

4.3.3 Case Study

In this section the 3D-NoC synthesis for a real world application of a heterogeneous mobile SoC is presented. The data on communication requirements used for five different use cases were obtained were obtained from a mobile SoC vendor. Originally, the system was not designed as a 3D-SoC. Therefore, a 3D-floorplanning tool [182] from project partners from the Institute of Microelectronic Systems of the University of Hannover was used for an initial placement of the PEs. The floorplanning was carried out for a 3D-system consisting of three chip layers.

Other relevant parameters for the network synthesis are given in Tab. 4.1. A NoC clock (intra-layer clock) of $500\,\mathrm{MHz}$ was chosen. For the TSV process a clock of $1.5\,\mathrm{GHz}$ was assumed and a TSV pitch of $20\,\mu\mathrm{m}$. This allows a serialization rate of $S = \frac{clk_{tsv}}{tsv_{noc}} = 3$ when pure serialization is used or up to three equally sized links to be transported over a similar sized TSV-Hub when pure link multiplexing is applied.

Tab. 4.1.: Case study key parameters

# of PEs	25
# of flows	128
# of use cases	5
# of silicon layers	3
NoC clock	$500\,\mathrm{MHz}$
TSV clock	$1.5\,\mathrm{GHz}$
TSV pitch	$20\,\mu\mathrm{m}$
TSV-Hub process node	$45\,\mathrm{nm}$ standard cell

The initial CCG is depicted in Fig. 4.20a. Note that only one use case is depicted, although five different uses cases are considered in the synthesis.

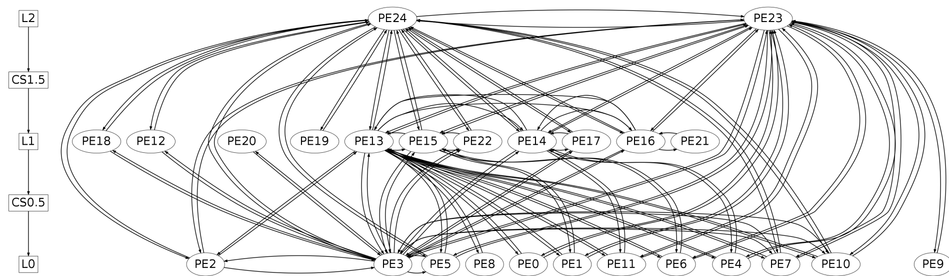
Tab. 4.2.: Case study results

# of inserted routers	5
# of inserted NoC links	70
# of layer crossing links	26
# of TSV-Hubs (serialization only)	26
# of TSV-Hubs (link clustering + serialization)	6
TSV-Hub sizes (# TSVs each)	105,54,43,68,255,151
# of TSV (no serialization or multiplexing)	˜ 2900
# of TSVs (serialization only)	981
# of TSVs (link clustering + serialization)	676
# TSVs saved (multiplexing gain)	31%
NoC synthesis tool run-time	˜ 30 min

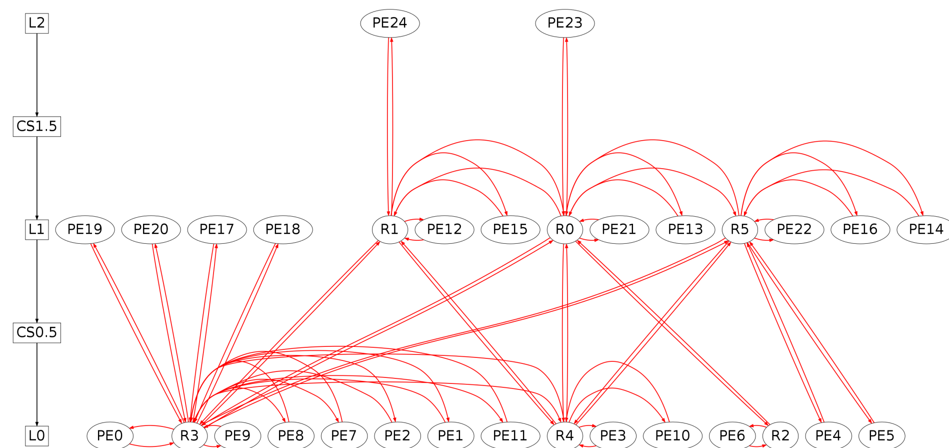
The synthesis results are listed in Tab. 4.2. Network synthesis inserts a total of five NoC routers and interconnects the routers and the PEs by a total of 70 NoC links. Out of these links, 26 are traversing chip layer boundaries. Hence, if only

serialization was applied, 26 smaller TSV-Hubs would be inserted, one per vertical link. The network synthesis result is shown in Fig. 4.20b.

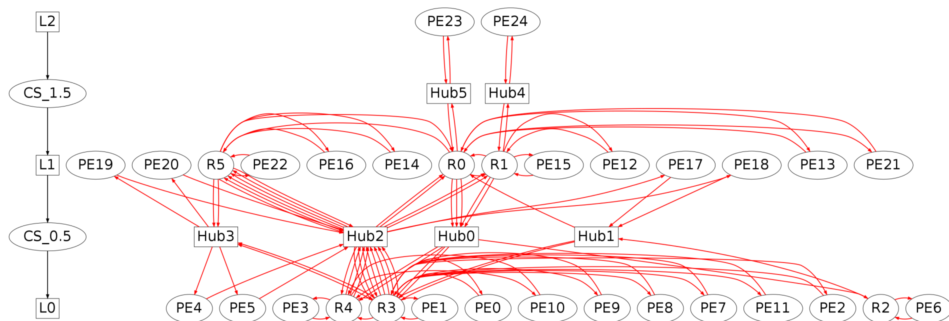
When applying neither link multiplexing nor link serialization, a total of 2900 TSVs is required to fulfil the bandwidth requirements of the vertical links. In this case, the TSVs are operated at the same clock speed as the intra-layer links, hence $500\,\mathrm{MHz}$. With applying serialization, the TSV count is reduced to 981 TSVs, meaning a reduction of roughly 66%. In this case the TSVs are operated at a clock speed of $1.5\,\mathrm{GHz}$. The network graph including TSV-Hubs is depicted in Fig. 4.20c.



(a) CCG



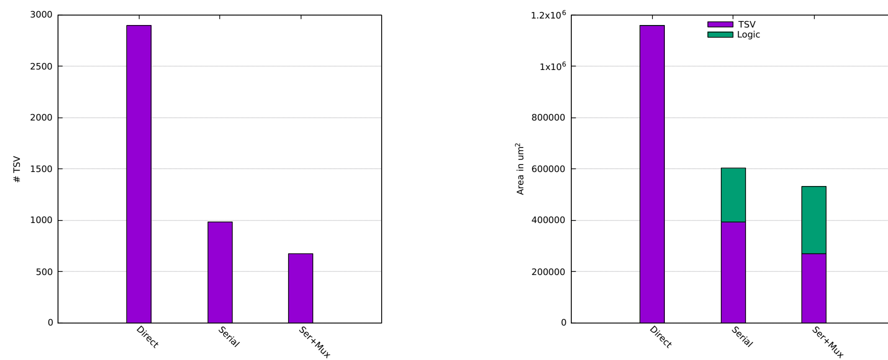
(b) Synthesized network; vertical links are still virtual and have to be connected via TSV-Hubs



(c) Placed TSV-Hubs for continuing vertical links

Fig. 4.20.: CCG, synthesized on-chip network and 3D-continued network of a 3D mobile SoC

With additional link multiplexing, the TSV clock is not further increased, but the temporal correlation of the link's data traffic is exploited. Links which are not operating in the same use case are combined and continued over a common TSV-Hub. This results in only six TSV-Hubs to be added. However, those TSV-Hubs are larger in size compared to the hubs that are inserted in case of a pure serialization solution since they are transporting multiple links. Although, as statistical multiplexing is performed now, the hub's capacity is smaller than the sum of the bandwidth requirements of all the vertical links the hub is transporting. The actual sizes (in terms of TSV count) of the six inserted hubs are given in Tab. 4.2. Regarding the number of TSVs, the TSV count is further reduced to just 676 TSVs, hence by roughly 31% (*multiplexing gain*). Note, however, that statistical multiplexing can only be applied when more than one use case is defined and therefore knowledge on the temporal correlation of the traffic flows is available.



(a) TSV count for direct connection, serialization and both serialization and link multiplexing

(b) System after the clustering phase. A cluster can span multiple chip layers.

Fig. 4.21.: TSV count and area comparison

Both serialization and multiplexing induce an area overhead since multiplexers are required as well as control logic for scheduling and arbitration. With a TSV pitch of $20\,\mu\text{m}$ a total area of $270\,400\,\mu\text{m}^2$ is required for TSVs (compared to roughly $1.16\,\text{mm}^2$ for the baseline reference with no serialization and multiplexing). Logic synthesis has been carried out for the six TSV-Hubs with the configuration as in Tab. 4.2 for a $45\,\text{nm}$ standard cell library. The resulting logic area of all hubs combined sums up to $525\,000\,\mu\text{m}$. Together with the TSV induced area this gives a total required area of $795\,000\,\mu\text{m}$ for realizing the vertical interconnects.

TSV count and total area are compared in Fig. 4.21.

Conclusion and Outlook 5

Three-dimensional integrated circuits have emerged as a promising solution for meeting future requirements for integrated systems regarding performance, power consumption, and diversity. By stacking multiple active dies, they achieve a higher integration density in general, providing more silicon area without further increasing footprint, and they allow the combination of dies, manufactured with different processes (heterogeneous integration). While advanced packaging technologies like Systems in Package (SiPs) also offer these advantages, 3D-ICs additionally contain high-performance die-to-die interconnects, usually in the form of Through Silicon Vias (TSVs) and allow for systems with high internal communication throughput to be distributed to multiple chip layers.

For designers, migrating from 2D-designs to stacked designs presents a challenge. They have to cope with issues that become apparent when continuing on-chip communication links to other chip layers. This includes issues like TSV area consumption, TSV reliability and TSV throughput optimizations. In 3D-ICs, on-chip interconnect structures or Networks-on-Chips (NoCs) have to be built such that they spread over multiple active layers. This work presented solutions to facilitate this migrating while keeping economic aspects in mind.

In the following sections, a summary of the scientific contribution is given. Finally, an outlook concludes this thesis.

5.1 Summary of Contributions

The problem of high area consumption of TSV arrays was addressed by using multiplexing and serialization techniques on inter-layer links. The original link capacity is maintained by applying a clock speed-up to the TSV arrays. This is possible due to the physical properties of TSVs (small capacitive load, low electrical resistance). Pure serialization is fully transparent (apart from required register stages to be inserted), and each link is continued over its own TSV array which has a smaller bit-width than the intra-layer part of the link. Additionally, link multiplexing and statistical oversubscription of TSV arrays, allow to leverage idle phases of individual links.

Methods for the efficient vertical continuation of conventional bus protocols were researched. It was shown that TSV count can be reduced by distributing the bus matrix to multiple chip layers. The influence of serialization and multiplexing on

bus timing was investigated. It was shown that modern channel based protocols like AMBA AXI can largely benefit from link multiplexing. It was researched how inserting register stages for serialization and deserialization into the address or data paths of bus-protocols affects performance. It was concluded that classical bus-protocol with strong inter-cycle dependencies suffer from performance degradation with register stages being inserted. More modern, channel based protocols do not suffer from this degradation and only receive a latency increase.

A solution for making TSV arrays robust against TSV defects was presented. Compared to other resilient design approaches, the number of possible faulty TSVs is configurable by an input parameter set at synthesis time. This allows the design to be adapted for different TSV processes with different yield figures. Currently, as TSV processes are still developed and optimized at the chip foundries, there is no reliable data regarding the final yield or fault rate of TSVs. However, it is predicted that yield and reliability will be low compared to conventional wiring. Once the actual expected fault rate of a TSV process is known, this fault tolerance concept can be tweaked with a single parameter to an appropriate value, providing just enough redundancy without spending more logic overhead than needed.

The multiplexing, serialization and fault-tolerance mechanisms were implemented in a configurable IP-Core, the so-called *TSV-Hub*. Internally, this module makes use of generic virtual links which are transported over a virtualized shared TSV array. The channels are configurable regarding their bit-width and quality-of-service level. A dynamic TDMA scheduler is used for arbitration and for assigning the TSV array to the individual links on a time slot basis. It has been shown that two independent AXI links can be continued over a TSV array of only 58 TSVs (compared to 390 TSVs without using the TSV-Hub). Taking the logic overhead induced by the TSV-Hub into account, this results in an overall area reduction to 29% of the original area in case of a $45\,\mathrm{nm}$ process node and a TSV pitch of $20\,\mu\mathrm{m}$.

The TSV-Hub can be used as a building block in a NoCs fabric to bundle multiple vertical links and continue them efficiently over a shared TSV array. A 3D NoCs synthesis algorithm was proposed which synthesizes an application specific NoC based on formally defined communication requirements. It automatically combines inter-layer-links and dimensions and inserts TSV-Hubs at appropriate locations.

The synthesis algorithm is based on a state-of-the-art 2D synthesis algorithm which has been extended at relevant points. It considers the vertical distance at an early stage and later bundles links with respect to their spatial distance and expected temporal correlation of carried traffic. In a case study, the on-chip network for a mobile SoC was synthesized, and it was shown in the results that TSV count can be reduced by roughly 70% with link multiplexing and even further with additional serialization of the individual links (2900 to 981 and 2900 to 676 TSVs respectively).

5.2 Future Work

This work is, to the best of our knowledge, the first one presenting a concept for incorporating link multiplexing at TSV arrays and including this in an automated design process. Naturally there are possible enhancements and further aspects that can be studied.

In general, since TSV manufacturing processes are still under research at chip foundries, most process parameters are unknown. The mechanisms and tools proposed in this work are flexible with respect to such parameters (e. g. TSV yield, TSV manufacturing cost, maximal layers, TSV diameter, size of keep out areas, TSV capacity and resistance). At a later point in time, when more concrete numbers are known for such parameters they can be used to calibrate the method and the impact of the still variable parameters can be studied.

Furthermore, the following aspects are worth being studied in the future:

In the context of the TSV-Hub design:

- **Incorporate packet switching in TSV-Hubs:** As already hinted in Chapter 3.1.1 a TSV-Hub performing link multiplexing already contains the data paths for a full crossbar, since at deserialization the data words can be assigned to another than the original link. This requires only enhancements in the control logic. This, however, cannot replace a full NoC router since it cannot switch packets between the inter-layer ports. However it can most likely be used to reduce complexity of some of the routers by outsourcing some switching functionality to the hubs for the cost of only a small logic overhead in the TSV-Hub's control logic.
- **Impact of different process nodes for the chip layers:** The TSV-Hub, as outlined in this work, assumes that both, receiving and sending side are realized in the same integration technology. However, one major benefit of 3D-ICs is that they allow to stacking multiple heterogeneous dies. Therefore the impact of having multiple traditional integration processes is worth studying. This includes the impact of different intra-layer clocks on different layers.
- **Irregular placement of TSVs:** The TSV arrays that are used in the TSV-Hub design are regular lattice structures. However, other topologies are also possible and even a fully irregular placement and mixture with conventional logic could be beneficial.

In the context of the 3D-NoC synthesis process:

- **Incorporate switching:** As already touched in the paragraph above, packet switching could be partly included in the TSV-Hubs. This feature could then

also be used in the synthesis flow for placing NoC routers of reduced complexity and outsourcing the switching partly to the hubs.

- **Timing model:** The granularity of timing model (latency constraints) is on the level of hops. This can be improved to a more accurate timing model.
- **Floorplanning loop:** The design used for the case study was initially placed by a 3D floorplanning tool (without communication infrastructure). The floorplanning tool has to be used again after the network synthesis since IP-Cores, networking elements and TSV-Hubs will most likely overlap. This can be extended to a loop where the network synthesis is carried out again for the newly placed elements. Thus, this further extends design space, and a converging solution for this problem has to be found.

5.3 Outlook

First commercial products based on 3D-integration and TSV technology are available today in the form of stacked DRAM chips and high-end FPGAs. The new 3D-memory standards Hybrid Memory Cube (HMC) [63] and High Bandwidth Memory (HBM) [65] outperform the conventional DDR memory standards by far and offer a maximum bandwidth of more than $200\,\mathrm{GB/s}$. However, end-products containing such modules are situated at the absolute high-end range of the spectrum and cost thousands of dollars per unit. This is expected to change in the near future. The third version of HBM is scheduled for 2019/2020. This revised version of the standard provides even more memory bandwidth and reaches the terabytes per second range [183]. Most importantly, though, it will be less costly to produce and rather targets a consumer product range [183]. Therefore, we observe an evolution as with most new technology: a solution at first deployed in high-end professional equipment moving into affordable consumer products.

This evolution will most likely repeat itself in the area of 3D integrated SoCs. As of today, there are no 3D-SoCs available on the market, but the first ones expected are high-end products, like server processors or professional GPUs. However, with SoCs there are also other aspects when it comes to cost than pure manufacturing. SoCs are much more diverse than regular structures like DRAMs or FPGAs. The design costs play an important role, and the costs for IP-cores can be decisive. 3D-SoCs can only be competitive if not every contained module has to be redesigned from scratch or has to be significantly modified because the interface has become incompatible with the on-chip interconnect backbone. Moreover, the design process for the interconnect infrastructure must be straight forward and ideally automated to a large part. Therefore, when 3D-ICs finally will be custom-made as SoC designs, solutions like the ones presented in this work, facilitating migration from the 2D realm and providing an automated flow for interconnect synthesis, will become crucial.

An unknown factor is the progress of TSV miniaturization. So far, the shrinking of TSV diameters and pitch has not advanced with the pace as it was predicted when first prototypes where built. In 2009, the International Technology Roadmap for Semiconductors (ITRS) for the first time included a *3D-TSV roadmap* in its interconnect report [184]. It predicted TSV diameters of $2\,\mu m$ to $4\,\mu m$ for global interconnects and $0.8\,\mu m$ to $1.5\,\mu m$ for intermediate interconnects. For the actual area consumption the TSV pitch is relevant, which is also estimated in the report. Values from $4\,\mu m$ to $8\,\mu m$ for global interconnects and $1.6\,\mu m$ to $3\,\mu m$ for intermediate interconnects are listed for the same time frame from 2013-2015. In later versions of the report [49], [185], we find more or less the same values, however shifted to later years. In the 2015 edition [185], the estimates are given for the time frame from 2019 to 2022 and are even slightly increased.

This is mainly due to the many challenges IC manufacturers still face like thinned wafer handling, alignment issues, TSV defects, heat dissipation, etc. However, in the meantime, the shrinking process according to Moore's Law still continues against many predictions. The experiments in this thesis were carried out for the $65\,nm$ and $45\,nm$ process node. This year (2017), first ICs produced at the $10\,nm$ nodes are shipped [186], and $7\,nm$ is on the horizon [187]. The logic overhead for serialization, multiplexing and rerouting logic for resilient design to cope with TSV defects will be much smaller for more advanced process node. Therefore, area savings will be even more significant compared to the experiments shown here.

Appendix A

A.1 TSV-Array Yield per Area Investment

The effective yield of a TSV array with respect to number of allowable faulty TSVs is given by:

$$P_{eff}\left(K \leq k_{max}\right) = \sum_{k=0}^{k_{max}} \binom{n_{nom} + k_{max}}{k} (1 - p_{tsv})^k {p_{tsv}}^{n_{nom}+k_{max}-k} \tag{A.1}$$

The effective Switch Box complexity with respect to k_{max} reads to:

$$\begin{aligned} I_{eff}(k_{max}, n_{nom}) &= I(n_{nom} + k_{max}, k_{max}) \\ &= \frac{1}{2}{k_{max}}^2 + \left(\frac{1}{2} + n_{nom}\right) k_{max} - 1 + n_{nom} \end{aligned} \tag{A.2}$$

And the overall area increase for improving resilience is:

$$\begin{aligned} A(k_{max}, n_{nom}) = a_{fi} \left(\frac{1}{2}{k_{max}}^2 + \left(\frac{1}{2} + n_{nom}\right) k_{max} - 1 \right. \\ \left. + n_{nom}\left(k_{max}, n_{nom}\right)\right) + k_{max} \cdot a_{tsv} \end{aligned} \tag{A.3}$$

When solved for k_{max} this yields:

$$k_{max}(a) = \pm \frac{1}{a_{fi}} \left(-a_{fi}n - \frac{a_{fi}}{2} - a_{tsv} + \frac{1}{2}\sqrt{8aa_{fi} + c}\right) \tag{A.4}$$

with the constant c :

$$c = 4{a_{fi}}^2 n^2 - 4{a_{fi}}^2 n + 9{a_{fi}}^2 + 8a_{fi}a_{tsv}n + 4a_{fi}a_{tsv} + 4{a_{tsv}}^2$$

The negative version of (A.4) can be omitted.

Since only integer values of k_max are possible, the floor function is used and the integral part of k_max is inserted in (A.1), to get the yield of the TSV array with respect to the area investment $p_{array}(a)$:

$$p_{array}(a) = \sum_{k=0}^{\lfloor k_{max}(a) \rfloor} \binom{n_{nom} + \lfloor k_{max}(a) \rfloor}{k} (1 - p_{tsv})^k {p_{tsv}}^{n_{nom} + \lfloor k_{max}(a) \rfloor - k} \tag{A.5}$$

A.2 TSV Sidewall Depletion Area

Any MOS structure operates in the depletion region when a voltage between the *Flatband Voltage* V_{fb} and the *Threshold Voltage* V_{th} is applied. The actual values of V_{fb} and these voltages strongly depends on the doping level of the involved silicon substrate. TSVs punch through the bulk silicon which has of all the doped parts in an IC usually the lowest doping level in the magnitude of $N_a = 1 \times 10^{-16}\,\mathrm{cm}^{-2}$.

The Threshold Voltage for doping levels in this order of magnitude is negative for both p- and n-doped silicon. Therefore, if we assume that only positive voltages are applied to a TSV, we will operate the TSV always in the inversion area. Operating a MOS structure in the inversion region, means that the depletion area width d_{dep} has reached its maximum extent $d_{dep,max}$ and also the depletion capacitance C_{dep} is at its maximum value $C_{dep,max}$. For DC applications the overall capacitance is further increased due to the formation of the inversion zone. For AC applications ($f_{clk} > 100\,\mathrm{MHz}$) however, such an inversion zone is never formed.

Since the overall capacitance of a MOS-Structure is formed of the oxide capacitance C_{ox} and the depletion capacitance C_dep in series and C_ox being independent from the voltage level, the overall capacitance is then also independent from the voltage level and can be denoted as:

$$C_{mos,min} = \frac{C_{ox} C_{dep,max}}{C_{ox} + C_{dep,max}} \tag{A.6}$$

Fig. A.1 from [85] shows the relation of the TSV capacity and applied voltage. Due to the negative V_{th} the minimal capacitance is already reached below $0\,\mathrm{V}$ and the overall capacitance C_{tsv} is constant for all positive voltages.

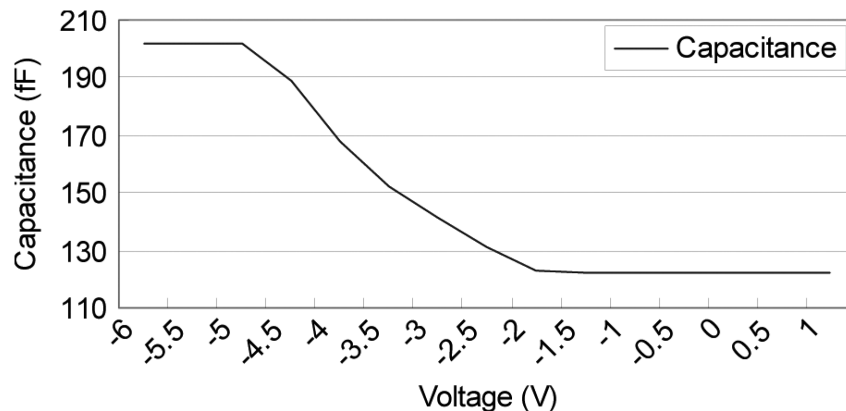


Fig. A.1.: TSV capacitance with respect to applied voltage [85]

For planar MOS structures the maximal depletion zone width $d_{dep,max}$ can be calculated as:

$$d_{dep,max} = \sqrt{\frac{4\varepsilon_{si}kT}{q^2N_a}\ln\left(\frac{N_a}{n_i}\right)} \tag{A.7}$$

However (A.7) is not valid for the cylindrical shape of TSVs, this is neglected in [73], in [75] it is considered but no full solution for calculating $d_{dep,max}$ is given. In [79] it shown, that the accurate depletion width for a cylinder approaches the one from (A.7) for larger TSV diameters. For smaller diameters, however ($d_{tsv} < 10\,\mathrm{nm}$) there is a significant deviation which introduces an error of up to 10% for realistic numbers of TSV diameters.

Therfore for calculating the accurate $d_{dep,max}$ max the following non-linear equation from [79] (originally from [188]) must be solved.

$$\begin{aligned}\frac{4\varepsilon_{si}kT}{q^2N_a}\ln\left(\frac{N_a}{n_i}\right) &= -0.5{d_{dep}}^2 - d_{dep}\left(0.5d_{tsv} + d_{ox}\right)\\ &+ \left(0.5d_{tsv} + d_{ox} + d_{dep,max}\right)^2\ln\left(\frac{0.5d_{tsv} + d_{ox} + d_{dep,max}}{0.5d_{tsv} + d_{ox}}\right)\end{aligned} \tag{A.8}$$

Equation (A.8) must now be numerical solved for $d_{dep,max}$ to obtain the maximum depletion width.

A.3 Markov Modeled on-off Traffic Generation

To generate random network traffic based on the parameters of average bandwidth (b_{avg}), peak bandwidth (b_{peak}) and average burst length (l_{avg}) a Markov modeled on-off generator can be used. Such a traffic generator is based on a Markov-chain with two states, one *generating* state and one *non-generating* state (see Fig. A.2).

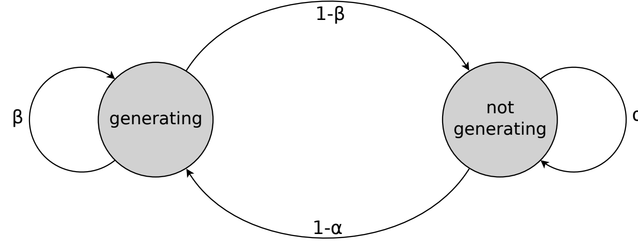


Fig. A.2.: Markov modeled IPP

In the *generating* state a flit is emitted in each cycle and in the *not-generating* state it is not. For each state, a parameter (α or β) is defined setting the probability that the state is not left in the next cycle. With the two probabilities, both the emitted traffic rate and the average burst/packet length can be set.

Let $P(on)$ be the probability that the generator is in the *generating* state. Then

$$P(on) = \frac{b_{avg}}{b_{peak}} \tag{A.9}$$

and

$$P(off) = \frac{b_{peak} - b_{avg}}{b_{peak}} = 1 - \frac{b_{avg}}{b_{peak}} \tag{A.10}$$

From the state diagram it can be derived that:

$$P(on) = P(off) \cdot (1 - \alpha) + P(on) \cdot \beta \tag{A.11}$$

$$P(off) = P(on) \cdot (1 - \beta) + P(off) \cdot \alpha \tag{A.12}$$

$$1 = P(on) + P(off) \tag{A.13}$$

By substituting $P(off)$ in (A.11) by $1 - P(on)$ from (A.13) and $P(on)$ in (A.12) by $1 - P(off)$ we get:

$$P(on) = \frac{1 - \alpha}{2 - \alpha - \beta} \tag{A.14}$$

and

$$P(off) = \frac{1 - \beta}{2 - \alpha - \beta} \tag{A.15}$$

Dividing (A.14) by (A.15) gives:

$$\frac{P(off)}{P(on)} = \frac{1 - \beta}{1 - \alpha} \tag{A.16}$$

together with (A.9) and (A.10) we get

$$\frac{P(off)}{P(on)} = \frac{1 - \beta}{1 - \alpha} = \frac{1 - \frac{b_{avg}}{b_{peak}}}{\frac{b_{avg}}{b_{peak}}} = \frac{b_{peak}}{b_{avg}} - 1 \tag{A.17}$$

This gives the relation of α and β only. To compute the actual values the average burst length has to be considered. A burst is generated as long as the generator operates in the *generating* state. This process is a sequence of Bernoulli experiments (one per cycle). Thus, the burst length is geometrically distributed. The probability of a "success" (leaving the *generating* state) is given by $1 - \beta$. Hence the expected value of the burst length can be written as:

$$\mathrm{E}(BS) = \frac{1}{1 - \beta} \tag{A.18}$$

with BS being the random variable of the burst length.

With the average burst length $l_{avg} = \mathrm{E}(BS)$ (in cycles), β can be calculated as follows:

$$\beta = 1 - \frac{1}{l_{avg}} \tag{A.19}$$

Resolving (A.17) for α and using β from (A.19) finally gives

$$\alpha = 1 - \frac{b_{avg}}{b_{peak} \cdot l_{avg}} \tag{A.20}$$

List of Personal Publications

[1] F. Miller, T. Wild, and A. Herkersdorf, "Tsv-virtualization for multi-protocol-interconnect in 3d-ics", in *15th EUROMICRO Conference on Digital System Design (DSD)*, 2012.

[2] F. Miller, T. Wild, and A. Herkersdorf, "Virtualized and fault-tolerant inter-layer-links for 3d-ics", *Microprocessors and Microsystems*, vol. 37, no. 8, Part A, pp. 823–835, 2013.

[3] F. Miller, T. Wild, and A. Herkersdorf, "Networks-on-chips für 3d-ics", en, in *7. ITG/GI/GMM-Fachtagung*, vol. ITG-Fachbericht, Band 244, Dresden, Germany, Sep. 2013.

[4] F. Miller, V. Todorov, T. Wild, D. Mueller-Gritschneder, A. Herkersdorf, and U. Schlichtmann, "A tsv-property-aware synthesis method for application-specific 3d-nocs", in *Design, Automation Test in Europe Conference Exhibition (DATE), 2013 - Friday Workshop on 3D Integration*, 2014.

[5] K. Hylla, M. Metzdorf, A. Gruenewald, K. Hahn, A. Heinig, U. Knoechel, S. Wolf, F. Miller, T. Wild, A. Quiring, *et al.*, "Needs–nanoelektronik-entwurf fuer 3d-systeme", *GMM-Fachbericht-Zuverlässigkeit und Entwurf*, 2012.

[6] A. Heinig, M. Dietrich, A. Herkersdorf, F. Miller, T. Wild, K. Hahn, A. Gruenewald, R. Brueck, S. Kroehnert, and J. Reisinger, "System integration - the bridge between more than moore and more moore", in *Design, Automation Test in Europe Conference Exhibition (DATE), 2014*, 2014.

[7] E. Koser, F. Miller, and W. Stechele, "Matching detection and correction schemes for soft error handling in sequential logic", in *2015 Euromicro Conference on Digital System Design*, Aug. 2015, pp. 706–713.

Bibliography

[8] L. Durant, O. Giroux, M. Harris, and N. Stam. (2017). Inside volta: The world's most advanced data center gpu, NVIDIA Corporation, [Online]. Available: `https://devblogs.nvidia.com/parallelforall/inside-volta/`.

[9] S. Wallentowitz. (2017). Moore and more, [Online]. Available: `https://github.com/wallento/mooreandmore`.

[10] WikiChip. (2017). Wikichip. accessed: 2017-07-24, WikiChip, [Online]. Available: `https://en.wikichip.org`.

[11] G. E. Moore, "Cramming more components onto integrated circuits, electronics, volume 38, number 8, april 19, 1965", *Also available online from ftp://download. intel. com/research/silicon/moorespaper. pdf*, 1965.

[12] G. E. Moore, "Progress in digital integrated electronics", in *Electron Devices Meeting, 1975 International*, vol. 21, IEEE, 1975, pp. 11–13.

[13] D. Brock and G. Moore, *Understanding Moore's Law: Four Decades of Innovation*. Chemical Heritage Foundation, 2006.

[14] M. Kada, "Research and development history of three-dimensional integration technology", in *Three-Dimensional Integration of Semiconductors: Processing, Materials, and Applications*, K. Kondo, M. Kada, and K. Takahashi, Eds. Cham: Springer International Publishing, 2015, pp. 1–23.

[15] C. A. Mack, "Seeing double", *IEEE Spectrum*, vol. 45, no. 11, pp. 46–51, Nov. 2008.

[16] D. James, "Intel ivy bridge unveiled - the first commercial tri-gate, high-k, metal-gate cpu", in *Proceedings of the IEEE 2012 Custom Integrated Circuits Conference*, Sep. 2012, pp. 1–4.

[17] W. Arden, M. Brillouët, P. Cogez, M. Graef, B. Huizing, and R. Mahnkopf, "Morethan-moore white paper", *Version*, vol. 2, p. 14, 2010.

[18] W. Dally and B. Towles, “Route packets, not wires: On-chip interconnection networks”, in *Design Automation Conference, 2001. Proceedings*, 2001, pp. 684–689.

[19] L. Benini and G. De Micheli, “Networks on chips: A new soc paradigm”, *Computer*, vol. 35, no. 1, pp. 70–78, Jan. 2002.

[20] G. D. Micheli and L. Benini, “Networks on chips: 15 years later”, *Computer*, vol. 50, no. 5, pp. 10–11, May 2017.

[21] K. Schuler. (Apr. 2013). Arteris flexnoc network-on-chip technology designed into majority of mobile socs. accessed: 2017-08-31, Arteris, Inc., [Online]. Available: `http://www.arteris.com/press-releases/arteris_mobility_pr_4_june_2013`.

[22] S. Murali, C. Seiculescu, L. Benini, and G. De Micheli, “Synthesis of networks on chips for 3d systems on chips”, in *Design Automation Conference, 2009. ASP-DAC 2009. Asia and South Pacific*, Jan. 2009, pp. 242–247.

[23] *Amba axi and ace protocol specification*, Arm Ltd., 2011.

[24] D. Wentzlaff, P. Griffin, H. Hoffmann, L. Bao, B. Edwards, C. Ramey, M. Mattina, C.-C. Miao, J. Brown, and A. Agarwal, “On-chip interconnection architecture of the tile processor”, *Micro, IEEE*, vol. 27, no. 5, pp. 15–31, Sep. 2007.

[25] JEDEC, *Wide i/o 2 (wideio2)*, JEDEC SOLID STATE TECHNOLOGY ASSOCIATION.

[26] P. Vivet, D. Dutoit, Y. Thonnart, and F. Clermidy, “3d noc using through silicon via: An asynchronous implementation”, in *VLSI and System-on-Chip (VLSI-SoC), 2011 IEEE/IFIP 19th International Conference on*, Oct. 2011, pp. 232–237.

[27] S. Pasricha, “Exploring serial vertical interconnects for 3d ics”, in *Design Automation Conference, 2009. DAC ’09. 46th ACM/IEEE*, Jul. 2009, pp. 581–586.

[28] F. Sun, A. Cevrero, P. Athanasopoulos, and Y. Leblebici, “Design and feasibility of multi-gb/s quasi-serial vertical interconnects based on tsvs for 3d ics”, in *VLSI System on Chip Conference (VLSI-SoC), 2010 18th IEEE/IFIP*, Sep. 2010, pp. 149–154.

[29] I. Loi, S. Mitra, T. H. Lee, S. Fujita, and L. Benini, “A low-overhead fault tolerance scheme for tsv-based 3d network on chip links”, in *Proceedings of the 2008 IEEE/ACM International Conference on Computer-Aided Design*, ser. ICCAD ’08, San Jose, California: IEEE Press, 2008, pp. 598–602.

[30] U. Kang, H. J. Chung, S. Heo, D. H. Park, H. Lee, J. H. Kim, S. H. Ahn, S. H. Cha, J. Ahn, D. Kwon, J. W. Lee, H. S. Joo, W. S. Kim, D. H. Jang, N. S. Kim, J. H. Choi, T. G. Chung, J. H. Yoo, J. S. Choi, C. Kim, and Y. H. Jun, "8 gb 3-d ddr3 dram using through-silicon-via technology", *IEEE Journal of Solid-State Circuits*, vol. 45, no. 1, pp. 111–119, Jan. 2010.

[31] V. Pasca, L. Anghel, C. Rusu, and M. Benabdenbi, "Configurable serial fault-tolerant link for communication in 3d integrated systems", in *On-Line Testing Symposium (IOLTS), 2010 IEEE 16th International*, Jul. 2010, pp. 115–120.

[32] V. Pasca, L. Anghel, and M. Benabdenbi, "Fault tolerant communication in 3d integrated systems", in *2010 International Conference on Dependable Systems and Networks Workshops (DSN-W)*, Jun. 2010, pp. 131–135.

[33] A. Verma, P. S. Multani, D. Mueller-Gritschneder, V. Todorov, and U. Schlichtmann, "A greedy approach for latency-bounded deadlock-free routing path allocation for application-specific nocs", in *Networks on Chip (NoCS), 2013 Seventh IEEE/ACM International Symposium on*, 2013, pp. 1–7.

[34] V. Todorov, D. Mueller-Gritschneder, H. Reinig, and U. Schlichtmann, "A spectral clustering approach to application-specific network-on-chip synthesis", in *Design, Automation Test in Europe Conference Exhibition (DATE), 2013*, 2013, pp. 1783–1788.

[35] V. Todorov, D. Mueller-Gritschneder, H. Reinig, and U. Schlichtmann, "Deterministic synthesis of hybrid application-specific network-on-chip topologies", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 33, no. 10, pp. 1503–1516, Oct. 2014.

[36] Technical University of Munich. (). Lisnoc, Technical University of Munich, Chair for Integrated Systems, [Online]. Available: `https://lis.ei.tum.de/projects/lisnoc`.

[37] M. G. Smith and S. Emanuel, "Methods of making thru-connections in semiconductor wafers", Patent US 3343256 A, Sep. 1967.

[38] W. Shockley, "Semiconductive wafer and method of making the same", Patent US 3044909 A, Jul. 1962.

[39] (2016). 3dic and 2.5d tsv interconnect for advanced packaging: 2016 business update - report sample. Accessed: 2017-06-11, Yole Développement, [Online]. Available: `https://www.i-micronews.com/advanced-packaging-report/product/p3dic-and-2-5d-tsv-interconnect-for-advanced-packaging-2016-business-update.html`.

[40] M. Santarini, "Xilinx introduces first heterogeneous 3d fpga: Virtex-7 h580t", *Xcell journal*, vol. Issue 80, pp. 8–13, 2012.

[41] K. Fischer, M. Agostinelli, C. Allen, D. Bahr, M. Bost, P. Charvat, V. Chikarmane, Q. Fu, C. Ganpule, M. Haran, M. Heckscher, H. Hiramatsu, E. Hwang, P. Jain, I. Jin, R. Kasim, S. Kosaraju, K. S. Lee, H. Liu, R. McFadden, S. Nigam, R. Patel, C. Pelto, P. Plekhanov, M. Prince, C. Puls, S. Rajamani, D. Rao, P. Reese, A. Rosenbaum, S. Sivakumar, B. Song, M. Uncuer, S. Williams, M. Yang, P. Yashar, and S. Natarajan, "Low-k interconnect stack with multi-layer air gap and tri-metal-insulator-metal capacitors for 14nm high volume manufacturing", in *2015 IEEE International Interconnect Technology Conference and 2015 IEEE Materials for Advanced Metallization Conference (IITC/MAM)*, May 2015, pp. 5–8.

[42] B. K. Kaushik, V. R. Kumar, M. K. Majumder, and A. Alam, *Through Silicon Vias: Materials, Models, Design, and Performance*. Crc Press, 2016.

[43] J. Ouyang, J. Xie, M. Poremba, and Y. Xie, "Evaluation of using inductive/capacitive-coupling vertical interconnects in 3d network-on-chip", in *Proceedings of the International Conference on Computer-Aided Design*, ser. ICCAD '10, San Jose, California: IEEE Press, 2010, pp. 477–482.

[44] M. S. Parekh, P. A. Thadesar, and M. S. Bakir, "Electrical, optical and fluidic through-silicon vias for silicon interposer applications", in *2011 IEEE 61st Electronic Components and Technology Conference (ECTC)*, May 2011, pp. 1992–1998.

[45] J. Reisinger, "3-d integration: A reality?", in *DATE'10 Friday Workshop on 3D Integration - Applications, Technology, Architecture, Design, Automation, and Test - Electronic Workshop Digest*, 2010, pp. 275–288.

[46] Intel Corporation. (May 2011). Intel reinvents transistors using new 3-d structure. accessed: 2017-07-01, [Online]. Available: `https://newsroom.intel.com/news-releases/intel-reinvents-transistors-using-new-3-d-structure/`.

[47] V. F. Pavlidis, "Interconnect-based design methodologies for three-dimensional integrated circuits", PhD thesis, University of Rochester, 2008.

[48] Z. Xu and J. Q. Lu, "Through-silicon-via fabrication technologies, passives extraction, and electrical modeling for 3-d integration/packaging", *IEEE Transactions on Semiconductor Manufacturing*, vol. 26, no. 1, pp. 23–34, Feb. 2013.

[49] ITRS, *International technology roadmap for semiconductors 2013 edition interconnect*, INTERNATIONAL TECHNOLOGY ROADMAP FOR SEMICONDUCTORS, 2013.

[50] C.-W. Wu, "What we have learned from soc is what is driving 3d integration", in *DATE Friday Workshop on 3D Integration - Electronic Workshop Digest*, Conference on Design, Automation and Test in Europe, DATE, 2010, pp. 5–51.

[51] Y. Kim, S.-K. Kang, and S. E. Kim, "Study of thinned si wafer warpage in 3d stacked wafers", *Microelectronics Reliability*, vol. 50, no. 12, pp. 1988–1993, 2010.

[52] H. Takahashi, S. Wang, S. Kameyama, Y. Higami, H. Yotsuyanagi, M. Hashizume, S.-K. Lu, and Z. Roth, "Trends in 3d integrated circuit (3d-ic) testing technology", in *Three-Dimensional Integration of Semiconductors: Processing, Materials, and Applications*, K. Kondo, M. Kada, and K. Takahashi, Eds. Cham: Springer International Publishing, 2015, pp. 235–268.

[53] B. Banijamali, S. Ramalingam, K. Nagarajan, and R. Chaware, "Advanced reliability study of tsv interposers and interconnects for the 28nm technology fpga", in *2011 IEEE 61st Electronic Components and Technology Conference (ECTC)*, May 2011, pp. 285–290.

[54] R. Chaware, K. Nagarajan, and S. Ramalingam, "Assembly and reliability challenges in 3d integration of 28nm fpga die on a large high density 65nm passive interposer", in *2012 IEEE 62nd Electronic Components and Technology Conference*, May 2012, pp. 279–283.

[55] K. Saban, "Xilinx stacked silicon interconnect technology delivers breakthrough fpga capacity, bandwidth, and power efficiency", *Xilinx, White Paper*, vol. 1, wP380, 2011.

[56] S. Lakka, "Xilinx ssi technology concept to silicon development overview", in *2012 IEEE Hot Chips 24 Symposium (HCS)*, Aug. 2012, pp. 1–22.

[57] L. Madden, "Heterogeneous 3-d stacking, can we have the best of both (technology) worlds?", in *Proceedings of the 2013 ACM International Symposium on International Symposium on Physical Design*, ser. ISPD '13, Stateline, Nevada, USA: ACM, 2013, pp. 1–2.

[58] JEDeC, "Jesd79-4 (ddr4 sdram)", *Joint Electron Device Engineering Council*, 2012.

[59] L. Mearian. (Apr. 2014). Hynix reveals world's first 128gb ddr4 memory module. accessed: 2017-07-10, [Online]. Available: `http://www.computerworld.com/s/article/9247500`.

[60] Samsung. (Nov. 2015). Samsung starts mass producing industry's first 128-gigabyte ddr4 modules for enterprise servers. accessed: 2017-07-09.

[61] Hybrid Memory Cube Consortium, "Hybrid memory cube specification 1.0", 2013.

[62] Hybrid Memory Cube Consortium, *Hybrid memory cube specification rev. 2.0*, 2014.

[63] Hybrid Memory Cube Consortium, *Hybrid memory cube specification rev. 2.1*, 2015.

[64] J. Jeddeloh and B. Keeth, “Hybrid memory cube new dram architecture increases density and performance”, in *2012 Symposium on VLSI Technology (VLSIT)*, Jun. 2012, pp. 87–88.

[65] JEDEC, “High bandwidth memory (hbm) dram”, *Standard JESD235A*, 2015.

[66] B. Moyer. (Jan. 2017). Hbm vs. hmc comparing cubes. accessed: 2017-06-09, EE Journal, [Online]. Available: `http://www.eejournal.com/article/20170102-hbm-hmc/`.

[67] J. Macri, “Amd’s next generation gpu and high bandwidth memory architecture: Fury”, in *2015 IEEE Hot Chips 27 Symposium (HCS)*, Aug. 2015, pp. 1–26.

[68] M. Walton. (Apr. 2016). Nvidia unveils first pascal graphics card, the monstrous tesla p100. accessed: 2017-07-10, ars technica, [Online]. Available: `https://arstechnica.com/gadgets/2016/04/nvidia-tesla-p100-pascal-details/`.

[69] JEDEC, *Wide i/o single data rate*, JEDEC, Dec. 2011.

[70] S. M. Alam, R. E. Jones, S. Rauf, and R. Chatterjee, “Inter-strata connection characteristics and signal transmission in three-dimensional (3d) integration technology”, in *8th International Symposium on Quality Electronic Design (ISQED’07)*, Mar. 2007, pp. 580–585.

[71] Y. Liang and Y. Li, “Closed-form expressions for the resistance and the inductance of different profiles of through-silicon vias”, *IEEE Electron Device Letters*, vol. 32, no. 3, pp. 393–395, Mar. 2011.

[72] J. Kim, J. S. Pak, J. Cho, E. Song, J. Cho, H. Kim, T. Song, J. Lee, H. Lee, K. Park, S. Yang, M. S. Suh, K. Y. Byun, and J. Kim, “High-frequency scalable electrical model and analysis of a through silicon via (tsv)”, *IEEE Transactions on Components, Packaging and Manufacturing Technology*, vol. 1, no. 2, pp. 181–195, Feb. 2011.

[73] I. Savidis and E. G. Friedman, “Closed-form expressions of 3-d via resistance, inductance, and capacitance”, *IEEE Transactions on Electron Devices*, vol. 56, no. 9, pp. 1873–1881, Sep. 2009.

[74] I. Savidis, S. M. Alam, A. Jain, S. Pozder, R. E. Jones, and R. Chatterjee, “Electrical modeling and characterization of through-silicon vias (tsvs) for 3-d integrated circuits”, *Microelectronics Journal*, vol. 41, no. 1, pp. 9–16, 2010.

[75] G. Katti, M. Stucchi, K. De Meyer, and W. Dehaene, “Electrical modeling and characterization of through silicon via for three-dimensional ics”, *Electron Devices, IEEE Transactions on*, vol. 57, no. 1, pp. 256–262, Jan. 2010.

[76] E. Rosa and U. S. N. B. of Standards, *The self and mutual inductances of linear conductors*, ser. Bulletin of the Bureau of Standards. U.S. Dept. of Commerce and Labor, Bureau of Standards, 1908.

[77] J. Kim, J. S. Pak, J. Cho, J. Lee, H. Lee, K. Park, and J. Kim, “Modeling and analysis of differential signal through silicon via (tsv) in 3d ic”, in *2010 IEEE CPMT Symposium Japan*, Aug. 2010, pp. 1–4.

[78] I. Ndip, K. Zoschke, K. Löbbicke, M. J. Wolf, S. Guttowski, H. Reichl, K. D. Lang, and H. Henke, “Analytical, numerical-, and measurement based methods for extracting the electrical parameters of through silicon vias (tsvs)”, *IEEE Transactions on Components, Packaging and Manufacturing Technology*, vol. 4, no. 3, pp. 504–515, Mar. 2014.

[79] A. E. Engin and S. R. Narasimhan, “Modeling of crosstalk in through silicon vias”, *IEEE Transactions on Electromagnetic Compatibility*, vol. 55, no. 1, pp. 149–158, Feb. 2013.

[80] S. Lim, “3d interconnect extraction”, in *Devices, Circuits, and Systems*, CRC Press, Dec. 2015, pp. 53–79.

[81] D. H. Kim, Y. Kim, J. Cho, B. Bae, J. Park, H. Lee, J. Lim, J. J. Kim, S. Piersanti, F. de Paulis, A. Orlandi, and J. Kim, “Through-silicon via capacitance voltage hysteresis modeling for 2.5-d and 3-d ic”, *IEEE Transactions on Components, Packaging and Manufacturing Technology*, vol. 7, no. 6, pp. 925–935, Jun. 2017.

[82] W. C. Elmore, “The transient response of damped linear networks with particular regard to wideband amplifiers”, *Journal of Applied Physics*, vol. 19, no. 1, pp. 55–63, 1948. eprint: `http://dx.doi.org/10.1063/1.1697872`.

[83] J. Rabaey, A. Chandrakasan, and B. Nikolic, *Digital integrated circuits: A design perspective*, ser. Prentice Hall electronics and VLSI series. Pearson Education, 2003.

[84] C. Hu, *Modern Semiconductor Devices for Integrated Circuits*. Prentice Hall, 2010.

[85] X. Wu, W. Zhao, M. Nakamoto, C. Nimmagadda, D. Lisk, S. Gu, R. Radojcic, M. Nowak, and Y. Xie, “Electrical characterization for intertier connections and timing analysis for 3-d ics”, *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 20, no. 1, pp. 186–191, Jan. 2012.

[86] G. Van der Plas, P. Limaye, I. Loi, A. Mercha, H. Oprins, C. Torregiani, S. Thijs, D. Linten, M. Stucchi, G. Katti, D. Velenis, V. Cherman, B. Vandevelde, V. Simons, I. De Wolf, R. Labie, D. Perry, S. Bronckers, N. Minas, M. Cupac, W. Ruythooren, J. Van Olmen, A. Phommahaxay, M. de Potter de ten Broeck, A. Opdebeeck, M. Rakowski, B. De Wachter, M. Dehan, M. Nelis, R. Agarwal, A. Pullini, F. Angiolini, L. Benini, W. Dehaene, Y. Travaly, E. Beyne, and P. Marchal, "Design issues and considerations for low-cost 3-d tsv ic technology", *Solid-State Circuits, IEEE Journal of*, vol. 46, no. 1, pp. 293–307, Jan. 2011.

[87] M. Aoki, F. Furuta, K. Hozawa, Y. Hanaoka, H. Kikuchi, A. Yanagisawa, T. Mitsuhashi, and K. Takeda, "Fabricating 3d integrated cmos devices by using wafer stacking and via-last tsv technologies", in *2013 IEEE International Electron Devices Meeting*, Dec. 2013, pp. 29.5.1–29.5.4.

[88] R. W. Hamming, "Error detecting and error correcting codes", *Bell System Technical Journal*, vol. 29, no. 2, pp. 147–160, 1950.

[89] S. R. Sridhara and N. R. Shanbhag, "Coding for system-on-chip networks: A unified framework", *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 13, no. 6, pp. 655–667, Jun. 2005.

[90] K. Chakrabarty, S. Deutsch, H. Thapliyal, and F. Ye, "Tsv defects and tsv-induced circuit failures: The third dimension in test and design-for-test", in *2012 IEEE International Reliability Physics Symposium (IRPS)*, Apr. 2012, 5F.1.1–5F.1.12.

[91] A. Eghbal, P. M. Yaghini, N. Bagherzadeh, and M. Khayambashi, "Analytical fault tolerance assessment and metrics for tsv-based 3d network-on-chip", *IEEE Transactions on Computers*, vol. 64, no. 12, pp. 3591–3604, Dec. 2015.

[92] M. Laisne, K. Arabi, and T. Petrov, *Systems and methods utilizing redundancy in semiconductor chip interconnects*, US Patent App. 12/480,754, Mar. 2010.

[93] A.-C. Hsieh, T. Hwang, M.-T. Chang, M.-H. Tsai, C.-M. Tseng, and H.-C. Li, "Tsv redundancy: Architecture and design issues in 3d ic", in *Proceedings of the Conference on Design, Automation and Test in Europe*, ser. DATE '10, Dresden, Germany: European Design and Automation Association, 2010, pp. 166–171.

[94] Y. Zhao, "Investigation into yield and reliability enhancement of tsv-based three-dimensional integration circuits", PhD thesis, University of Southampton, Oct. 2014.

[95] Y. Zhao, S. Khursheed, B. M. Al-Hashimi, and Z. Zhao, "Co-optimization of fault tolerance, wirelength and temperature mitigation in tsv-based 3d ics", in *2016 IFIP/IEEE International Conference on Very Large Scale Integration (VLSI-SoC)*, Sep. 2016, pp. 1–6.

[96] Y. Zhao, S. Khursheed, and B. Al-Hashimi, "Cost-effective tsv grouping for yield improvement of 3d-ics", in *Test Symposium (ATS), 2011 20th Asian*, Nov. 2011, pp. 201–206.

[97] V. Pasca, S. U. Rehman, L. Anghel, and M. Benabdenbi, "Efficient link-level error resilience in 3d nocs", in *2012 IEEE 15th International Symposium on Design and Diagnostics of Electronic Circuits Systems (DDECS)*, Apr. 2012, pp. 127–132.

[98] V. Pasca, L. Anghel, and M. Benabdenbi, "Error resilience exploration in 3d systems", in *IEEE International On-Line Testing Symposium*, Chania: IEEE, vol. 5, 2013.

[99] C. L. Lung, Y. S. Su, H. H. Huang, Y. Shi, and S. C. Chang, "Through-silicon via fault-tolerant clock networks for 3-d ics", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 32, no. 7, pp. 1100–1109, Jul. 2013.

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

[101] S. K. Roy, K. Roy, C. Giri, and H. Rahaman, "Recovery of faulty tsvs in 3d ics", in *Sixteenth International Symposium on Quality Electronic Design*, Mar. 2015, pp. 533–536.

[102] S. K. Roy, S. Chatterjee, C. Giri, and H. Rahaman, "Repairing of faulty tsvs using available number of multiplexers in 3d ics", in *Fifth Asia Symposium on Quality Electronic Design (ASQED 2013)*, Aug. 2013, pp. 155–160.

[103] C. Osewold, W. Büter, and A. García-Ortiz, "A coding-based configurable and asymmetrical redundancy scheme for 3-d interconnects", in *2014 9th International Symposium on Reconfigurable and Communication-Centric Systems-on-Chip (ReCoSoC)*, May 2014, pp. 1–8.

[104] V. Darve, "An asynchronous serial link for a 3d network-on-chip", in *Friday Workshop on 3D Integration - Date Conference 2010*, 2010.

[105] F. Darve, A. Sheibanyrad, P. Vivet, and F. Petrot, "Physical implementation of an asynchronous 3d-noc router using serial vertical links", in *VLSI (ISVLSI), 2011 IEEE Computer Society Annual Symposium on*, Jul. 2011, pp. 25–30.

[106] S. Kimura, T. Hayakawa, T. Horiyama, M. Nakanishi, and K. Watanabe, "An on-chip high speed serial communication method based on independent ring oscillators", in *2003 IEEE International Solid-State Circuits Conference, 2003. Digest of Technical Papers. ISSCC.*, Feb. 2003, 390–391 vol.1.

[107] I.-C. Wey, L.-H. Chang, Y.-G. Chen, S.-H. Chang, and A.-Y. Wu, "A 2gb/s high-speed scalable shift-register based on-chip serial communication design for soc applications", in *2005 IEEE International Symposium on Circuits and Systems*, May 2005, 1074–1077 Vol. 2.

[108] M. Saneei, A. Afzali-Kusha, and M. Pedram, "Two high-performance and low-power serial communication interfaces for on-chip interconnects", *Canadian Journal of Electrical and Computer Engineering*, vol. 34, no. 1/2, pp. 49–56, Winter 2009.

[109] A. J. Martin, "Developments in concurrency and communication", in, C. A. R. Hoare, Ed., Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1990, ch. Programming in VLSI: From Communicating Processes to Delay-insensitive Circuits, pp. 1–64.

[110] M. Belleville, E. Beigne, and A. Valentian, "A single tsv-rail 3d quasi delay insensitive asynchronous signaling", in *IC Design Technology (ICICDT), 2011 IEEE International Conference on*, May 2011, pp. 1–4.

[111] A. Sheibanyrad and F. Petrot, "Asynchronous 3d-nocs making use of serialized vertical links", in *3D Integration for NoC-based SoC Architectures*, ser. Integrated Circuits and Systems, A. Sheibanyrad, F. Petrot, and A. Jantsch, Eds., Springer New York, 2011, pp. 149–165.

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

[113] S. Ogg, E. Valli, B. Al-Hashimi, A. Yakovlev, C. D'Alessandro, and L. Benini, "Serialized asynchronous links for noc", in *Design, Automation and Test in Europe, 2008. DATE '08*, Mar. 2008, pp. 1003–1008.

[114] E. Beigne, F. Clermidy, P. Vivet, A. Clouard, and M. Renaudin, "An asynchronous noc architecture providing low latency service and its multi-level design framework", in *Proceedings of the 11th IEEE International Symposium on Asynchronous Circuits and Systems*, Washington, DC, USA: IEEE Computer Society, 2005, pp. 54–63.

[115] G. Beanato, A. Cevrero, G. D. Michel, and Y. Leblebici, "3d serial tsv link for low-power chip-to-chip communication", in *2014 IEEE International Conference on IC Design Technology*, May 2014, pp. 1–4.

[116] G. Beanato, K. Gharibdoust, A. Cevrero, G. D. Micheli, and Y. Leblebici, "Design and analysis of jitter-aware low-power and high-speed tsv link for 3d ics", *Microelectronics Journal*, vol. 48, pp. 50–59, 2016.

[117] G. Beanato, A. Cevrero, G. D. Micheli, and Y. Leblebici, “Impact of data serialization over tsvs on routing congestion in 3d-stacked multi-core processors”, *Microelectronics Journal*, vol. 51, pp. 38–45, 2016.

[118] T. Robertazzi, *Networks and Grids: Technology and Theory*, ser. Information Technology: Transmission, Processing and Storage. Springer New York, 2007.

[119] A. C. Cangellaris, “The interconnect bottleneck in multi-ghz processors; new opportunities for hybrid electrical/optical solutions”, in *Proceedings. Fifth International Conference on Massively Parallel Processing (Cat. No.98EX182)*, Jun. 1998, pp. 96–103.

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

[121] Y. Zhang, W. Ye, and M. Irwin, “An alternative architecture for on-chip global interconnect: Segmented bus power modeling”, in *Signals, Systems Computers, 1998. Conference Record of the Thirty-Second Asilomar Conference on*, vol. 2, Nov. 1998, 1062–1065 vol.2.

[122] ARM, *Arm amba 5 ahb protocol specification*, ARM Ltd., 2015.

[123] K. Olukotun, O. Olukotun, L. Hammond, and J. Laudon, *Chip Multiprocessor Architecture: Techniques to Improve Throughput and Latency*, ser. Synthesis lectures in computer architecture. Morgan & Claypool Publishers, 2007.

[124] S. Vangal, J. Howard, G. Ruhl, S. Dighe, H. Wilson, J. Tschanz, D. Finan, P. Iyer, A. Singh, T. Jacob, S. Jain, S. Venkataraman, Y. Hoskote, and N. Borkar, “An 80-tile 1.28tflops network-on-chip in 65nm cmos”, in *Solid-State Circuits Conference, 2007. ISSCC 2007. Digest of Technical Papers. IEEE International*, Feb. 2007, pp. 98–589.

[125] S. Vangal, J. Howard, G. Ruhl, S. Dighe, H. Wilson, J. Tschanz, D. Finan, A. Singh, T. Jacob, S. Jain, V. Erraguntla, C. Roberts, Y. Hoskote, N. Borkar, and S. Borkar, “An 80-tile sub-100-w teraflops processor in 65-nm cmos”, 1, vol. 43, Jan. 2008, pp. 29–41.

[126] L. Benini, “Application specific noc design”, in *Proceedings of the Conference on Design, Automation and Test in Europe: Proceedings*, ser. DATE ’06, Munich, Germany: European Design and Automation Association, 2006, pp. 491–495.

[127] V. F. Pavlidis and E. G. Friedman, “Chapter 1 - introduction”, in *Three-dimensional Integrated Circuit Design*, V. F. Pavlidis and E. G. Friedman, Eds., Boston: Morgan Kaufmann, 2009, pp. 1–16.

[128] *Itu-t x.200 data network and open system communications*, Information technology – Open Systems Interconnection – Basic Reference Model: The basic model.

[129] N. E. Jerger, T. Krishna, and L.-S. Peh, "On-chip networks, second edition", *Synthesis Lectures on Computer Architecture*, vol. 12, no. 3, pp. 1–210, 2017. eprint: `https://doi.org/10.2200/S00772ED1V01Y201704CAC040`.

[130] A. Sodani, "Knights landing (knl): 2nd generation intel(r) xeon phi processor", in *2015 IEEE Hot Chips 27 Symposium (HCS)*, Aug. 2015, pp. 1–24.

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

[132] T. Bjerregaard and S. Mahadevan, "A survey of research and practices of network-on-chip", *ACM Comput. Surv.*, vol. 38, no. 1, p. 1, 2006.

[133] V. Pavlidis and E. Friedman, "3-d topologies for networks-on-chip", in *SOC Conference, 2006 IEEE International*, Sep. 2006, pp. 285–288.

[134] B. Feero and P. P. Pande, "Performance evaluation for three-dimensional networks-on-chip", in *ISVLSI '07: Proceedings of the IEEE Computer Society Annual Symposium on VLSI*, Washington, DC, USA: IEEE Computer Society, 2007, pp. 305–310.

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

[136] H. Matsutani, M. Koibuchi, and H. Amano, "Tightly-coupled multi-layer topologies for 3-d nocs", in *Parallel Processing, 2007. ICPP 2007. International Conference on*, Sep. 2007, p. 75.

[137] H. Matsutani, M. Koibuchi, D. Hsu, and H. Amano, "Three-dimensional layout of on-chip tree-based networks", in *Parallel Architectures, Algorithms, and Networks, 2008. I-SPAN 2008. International Symposium on*, May 2008, pp. 281–288.

[138] Y. J. Hwang, J. H. Lee, and T. H. Han, "3d network-on-chip system communication using minimum number of tsvs", in *ICT Convergence (ICTC), 2011 International Conference on*, Sep. 2011, pp. 517–522.

[139] M. Daneshtalab, M. Ebrahimi, P. Liljeberg, J. Plosila, and H. Tenhunen, "Pipeline-based interlayer bus structure for 3d networks-on-chip", in *Computer Architecture and Digital Systems (CADS), 2010 15th CSI International Symposium on*, Sep. 2010, pp. 35–41.

[140] W. J. Dally and C. L. Seitz, "Deadlock-free message routing in multiprocessor interconnection networks", *IEEE Trans. Comput.*, vol. 36, no. 5, pp. 547–553, May 1987.

[141] C. J. Glass and L. M. Ni, "The turn model for adaptive routing", in *Proceedings of the 19th Annual International Symposium on Computer Architecture*, ser. ISCA '92, Queensland, Australia: ACM, 1992, pp. 278–287.

[142] G.-M. Chiu, "The odd-even turn model for adaptive routing", *IEEE Transactions on Parallel and Distributed Systems*, vol. 11, no. 7, pp. 729–738, Jul. 2000.

[143] 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 *2012 IEEE 6th International Symposium on Embedded Multicore SoCs*, Sep. 2012, pp. 167–174.

[144] F. Dubois, A. Sheibanyrad, F. Petrot, and M. Bahmani, "Elevator-first: A deadlock-free distributed routing algorithm for vertically partially connected 3d-nocs", *IEEE Transactions on Computers*, vol. 62, no. 3, pp. 609–615, Mar. 2013.

[145] W. Dally and B. Towles, *Principles and Practices of Interconnection Networks*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2003.

[146] J. Kim, C. Nicopoulos, D. Park, R. Das, Y. Xie, V. Narayanan, M. S. Yousif, and C. R. Das, "A novel dimensionally-decomposed router for on-chip communication in 3d architectures", *SIGARCH Comput. Archit. News*, vol. 35, pp. 138–149, 2007.

[147] A.-M. Rahmani, K. Latif, P. Liljeberg, J. Plosila, and H. Tenhunen, "Research and practices on 3d networks-on-chip architectures", in *NORCHIP, 2010*, Nov. 2010, pp. 1–6.

[148] A.-M. Rahmani, K. Latif, K. Vaddina, P. Liljeberg, J. Plosila, and H. Tenhunen, "Power-efficient inter-layer communication architectures for 3d noc", in *VLSI (ISVLSI), 2011 IEEE Computer Society Annual Symposium on*, Jul. 2011, pp. 355–356.

[149] A.-M. Rahmani, P. Liljeberg, J. Plosila, and H. Tenhunen, "Bbvc-3d-noc: An efficient 3d noc architecture using bidirectional bisynchronous vertical channels", in *VLSI (ISVLSI), 2010 IEEE Computer Society Annual Symposium on*, Jul. 2010, pp. 452–453.

[150] D. Park, S. Eachempati, R. Das, A. Mishra, Y. Xie, N. Vijaykrishnan, and C. Das, "Mira: A multi-layered on-chip interconnect router architecture", in *Computer Architecture, 2008. ISCA '08. 35th International Symposium on*, Jun. 2008, pp. 251–261.

[151] K. Nomura, K. Abe, S. Fujita, and A. DeHon, “Novel design of three-dimensional crossbar for future network on chip based on post-silicon devices”, in *2006 1st International Conference on Nano-Networks and Workshops*, Sep. 2006, pp. 1–5.

[152] L. P. Carloni, P. Pande, and Y. Xie, “Networks-on-chip in emerging interconnect paradigms: Advantages and challenges”, in *Proceedings of the 2009 3rd ACM/IEEE International Symposium on Networks-on-Chip*, ser. NOCS ’09, Washington, DC, USA: IEEE Computer Society, 2009, pp. 93–102.

[153] S. Murali, P. Meloni, F. Angiolini, D. Atienza, S. Carta, L. Benini, G. De Micheli, and L. Raffo, “Designing application-specific networks on chips with floorplan information”, in *Computer-Aided Design, 2006. ICCAD ’06. IEEE/ACM International Conference on*, Nov. 2006, pp. 355–362.

[154] C. Neeb and N. Wehn, “Designing efficient irregular networks for heterogeneous systems-on-chip”, *Journal of Systems Architecture*, vol. 54, no. 3, pp. 384–396, 2008, System and Network on Chip.

[155] C. Seiculescu, S. Murali, L. Benini, and G. De Micheli, “Sunfloor 3d: A tool for networks on chip topology synthesis for 3d systems on chips”, in *Design, Automation Test in Europe Conference Exhibition, 2009. DATE ’09.*, Apr. 2009, pp. 9–14.

[156] R. Marculescu, J. Hu, and U. Ogras, “Key research problems in noc design: A holistic perspective”, in *Hardware/Software Codesign and System Synthesis, 2005. CODES+ISSS ’05. Third IEEE/ACM/IFIP International Conference on*, Sep. 2005, pp. 69–74.

[157] B. Yu, S. Dong, S. Chen, and S. Goto, “Floorplanning and topology generation for application-specific network-on-chip”, in *Design Automation Conference (ASP-DAC), 2010 15th Asia and South Pacific*, Jan. 2010, pp. 535–540.

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

[159] S. Murali and G. De Micheli, “Sunmap: A tool for automatic topology selection and generation for nocs”, in *Proceedings of the 41st Annual Design Automation Conference*, ser. DAC ’04, San Diego, CA, USA: ACM, 2004, pp. 914–919.

[160] J. Hu and R. Marculescu, “Energy- and performance-aware mapping for regular noc architectures”, *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 24, no. 4, pp. 551–562, Apr. 2005.

[161] K. Srinivasan, K. Chatha, and G. Konjevod, "An automated technique for topology and route generation of application specific on-chip interconnection networks", in *Computer-Aided Design, 2005. ICCAD-2005. IEEE/ACM International Conference on*, Nov. 2005, pp. 231–237.

[162] D. Bertozzi, A. Jalabert, S. Murali, R. Tamhankar, S. Stergiou, L. Benini, and G. De Micheli, "Noc synthesis flow for customized domain specific multiprocessor systems-on-chip", *Parallel and Distributed Systems, IEEE Transactions on*, vol. 16, no. 2, pp. 113–129, Feb. 2005.

[163] W. Zhong, S. Chen, F. Ma, T. Yoshimura, and S. Goto, "Floorplanning driven network-on-chip synthesis for 3-d socs", in *Circuits and Systems (ISCAS), 2011 IEEE International Symposium on*, May 2011, pp. 1203–1206.

[164] A. Barzinmehr and S. Tosun, "Energy-aware application-specific topology generation for 3d network-on-chips", in *2017 IEEE 20th International Symposium on Design and Diagnostics of Electronic Circuits Systems (DDECS)*, Apr. 2017, pp. 84–87.

[165] G. Leary and K. Chatha, "A holistic approach to network-on-chip synthesis", in *Hardware/Software Codesign and System Synthesis (CODES+ISSS), 2010 IEEE/ACM/IFIP International Conference on*, Oct. 2010, pp. 213–222.

[166] J. Soumya and S. Chattopadhyay, "Application-specific network-on-chip synthesis with flexible router placement", *Journal of Systems Architecture*, vol. 59, no. 7, pp. 361–371, 2013.

[167] S. Tosun, Y. Ar, and S. Ozdemir, "Application-specific topology generation algorithms for network-on-chip design", *IET Computers Digital Techniques*, vol. 6, no. 5, pp. 318–333, Sep. 2012.

[168] G. N. Khan and A. Tino, "Synthesis of noc interconnects for custom mpsoc architectures", in *Proceedings of the 2012 IEEE/ACM Sixth International Symposium on Networks-on-Chip*, ser. NOCS '12, Washington, DC, USA: IEEE Computer Society, 2012, pp. 75–82.

[169] S. Pasricha and N. Dutt, *On-chip communication architectures : System on chip interconnect*. Amsterdam; Boston: Elsevier / Morgan Kaufmann Publishers, 2008.

[170] Arm, "Multi-layer ahb overview", Tech. Rep. ARM DVI 0045B, 2004.

[171] M. Krstic, E. Grass, F. Gurkaynak, and P. Vivet, "Globally asynchronous, locally synchronous circuits: Overview and outlook", *Design Test of Computers, IEEE*, vol. 24, no. 5, pp. 430–441, Sep. 2007.

[172] I. Loi, F. Angiolini, and L. Benini, “Developing mesochronous synchronizers to enable 3d nocs”, in *Proceedings of the conference on Design, automation and test in Europe*, ser. DATE ’08, Munich, Germany: ACM, 2008, pp. 1414–1419.

[173] P. Teehan, M. Greenstreet, and G. Lemieux, “A survey and taxonomy of gals design styles”, *IEEE Design Test of Computers*, vol. 24, no. 5, pp. 418–428, Sep. 2007.

[174] J. M. Chabloz and A. Hemani, “A flexible communication scheme for rationally-related clock frequencies”, in *2009 IEEE International Conference on Computer Design*, Oct. 2009, pp. 109–116.

[175] L. Jiang, Q. Xu, and B. Eklow, “On effective tsv repair for 3d-stacked ics”, in *Design, Automation Test in Europe Conference Exhibition (DATE), 2012*, Mar. 2012, pp. 793–798.

[176] D. Velenis, E. Marinissen, and E. Beyne, “Cost effectiveness of 3d integration options”, in *3D Systems Integration Conference (3DIC), 2010 IEEE International*, Nov. 2010, pp. 1–6.

[177] T. D. Richardson, C. Nicopoulos, D. Park, V. Narayanan, Y. Xie, C. Das, and V. Degalahal, “A hybrid soc interconnect with dynamic tdma-based transaction-less buses and on-chip networks”, in *Proceedings of the 19th International Conference on VLSI Design held jointly with 5th International Conference on Embedded Systems Design*, Washington, DC, USA: IEEE Computer Society, 2006, pp. 657–664.

[178] M. R. Pillmeier, “Design alternatives for barrel shifters”, 2002.

[179] J. Shi and J. Malik, “Normalized cuts and image segmentation”, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 8, pp. 888–905, Aug. 2000.

[180] U. Von Luxburg, “A tutorial on spectral clustering”, *Statistics and computing*, vol. 17, no. 4, pp. 395–416, 2007.

[181] A. S. Initiative. (). Systemc. accessed: 2017-08-24, Accellera Systems Initiative, [Online]. Available: `http://www.accellera.org/downloads/standards/systemc`.

[182] A. Quiring, M. Olbrich, and E. Barke, “Fast global interconnnect driven 3d floorplanning”, in *2015 IFIP/IEEE International Conference on Very Large Scale Integration (VLSI-SoC)*, Oct. 2015, pp. 313–318.

[183] M. Walton. (Aug. 2016). Hbm3: Cheaper, up to 64gb on-package, and terabytes-per-second bandwidth. accessed: 2017-07-28, [Online]. Available: `https://arstechnica.com/gadgets/2016/08/hbm3-details-price-bandwidth/`.

[184] ITRS, *International technology roadmap for semiconductors 2009 edition interconnect*, INTERNATIONAL TECHNOLOGY ROADMAP FOR SEMICONDUCTORS, 2009.

[185] ITRS, *International technology roadmap for semiconductors 2015 edition interconnect*, INTERNATIONAL TECHNOLOGY ROADMAP FOR SEMICONDUCTORS, 2015.

[186] J. Ribeiro. (Feb. 2017). Samsung's bleeding-edge, 10nm exynos 9 chip arrives before the galaxy s8. accessed: 2017-07-28, [Online]. Available: `http://www.pcworld.com/article/3173376/mobile/samsung-starts-production-of-new-10-nm-exynos-9-series-chip.html`.

[187] Intel Corporation. (Feb. 2017). Intel supports american innovation with 7 billion investment in next-generation semiconductor factory in arizona. accessed: 2017-07-28, [Online]. Available: `https://newsroom.intel.com/news-releases/intel-supports-american-innovation-7-billion-investment-next-generation-semiconductor-factory-arizona/`.

[188] C. Xu, H. Li, R. Suaya, and K. Banerjee, "Compact ac modeling and performance analysis of through-silicon vias in 3-d ics", *IEEE Transactions on Electron Devices*, vol. 57, no. 12, pp. 3405–3417, Dec. 2010.