# DISSERTATION

submitted to the

Combined Faculty for the Natural Sciences and Mathematics

of

Heidelberg University, Germany

for the degree of

Doctor of Natural Sciences

Put forward by
Stefan Kosnac, M.Sc.

born in
Eberbach, Germany

Heidelberg, 2021

# Analysis of On-Chip Inductors and Arithmetic Circuits in the Context of High Performance Computing

Advisor: Prof. Dr.-Ing. Ulrich Brüning

Date of oral examination: .....

# Abstract

The increase in computing performance of integrated circuits over the last decades through shrinking transistor and interconnect sizes by several orders of magnitude has enabled many technological advancements. Today the design of high performance computing (HPC) systems is mainly limited by two major factors: I/O-bandwidth and power consumption. This work takes an in-depth view on one aspect of each of those limitations. Firstly, the I/O-bandwidth is addressed with an analysis of on-chip inductors for high-speed SerDes designs. Secondly, the design of arithmetic circuits for addition, multiplication, and floating-point computation is analyzed.

While the on-chip structures have scaled down extremely, the pin count of packages could not be increased accordingly, which is known as pin limitation. Thus, I/O cells are required to increase the bandwidth per pin to serialize the parallel data for transmission. Such SerDes have to cope with signal degradation caused by channels for multi-gigahertz signals by employing dedicated circuits. This includes an adequate signal termination, which is however compromised by the parasitic capacitance of the electrostatic discharge protection. This work analyzes how the capacitance can be compensated with on-chip inductors to reduce reflections back into the channel.

Unlike other devices, on-chip inductors are rarely provided by process design kits (PDKs) so layout and modeling has to be done by the designer. This work presents an evaluation of different methods for parameterized cells for inductors. The background on analytical inductance calculations is researched and different approximations of the mean distance formula found in literature are compared. A new expression for the internal inductance of straight wires is derived. The influence of oxide and substrate, skin effect, metal fill, and process corners is analyzed. Inductor simulation with field solvers and associated problems are presented. This work closes the gap between simulated S-parameters and lumped models used in circuit design with a Markov-Chain Monte-Carlo fitting technique. Using these results, it is shown how a lumped model can be expressed in terms of layout geometry, similar to what is commonly available for PDK devices. The result is accurate enough to be used for circuit optimization.

The processors used in data-centers, HPC systems, and many other devices are almost exclusively proprietary designs from established companies with instruction set architectures (ISAs) associated with high licensing costs. In 2010, the open-standard RISC-V ISA was released by the University of California, Berkeley, and has obtained great momentum in the last few years. It is said to start a "new era of computer architecture" as many companies and universities start to develop – often small – custom RISC-V cores. It is crucial to include very fast arithmetic units into such cores for RISC-V to enter the HPC environment.

Therefore, this work analyzes an IEEE 754 floating-point unit for fused multiply-add operations developed at the Computer Architecture Group at Heidelberg University. Design, verification, and a power-performance-area (PPA) analysis are presented. Comparable designs available in the open-source space tend to prefer high-level descriptions of multiplication and addition. This work investigates if there are potential gains in PPA that can be achieved through gate-level implementations of arithmetic circuits. However, the results strongly suggest that modern back-end tools are perfectly capable to outperform custom structures and design effort should be directed towards other aspects.

# Zusammenfassung

Die Rechenleistung integrierter Schaltungen konnte in den letzten Jahrzehnten aufgrund immer kleinerer Strukturen extrem gesteigert werden und ermöglichte einen signifikanten Fortschritt in vielen Bereichen der Technologie. Der Entwurf von Hochleistungsrechnern ist heutzutage durch zwei wesentliche Beschränkungen bestimmt: I/O-Bandbreite und Leistungsaufnahme. Diese Arbeit schaut im Detail auf jeweils einen Aspekt, der diese beiden Beschränkungen betrifft. Im Bereich der Bandbreite behandelt sie eine Analyse des Entwurfs integrierter Spulen für SerDes-Technologie. Im zweiten Teil beschäftigt sie sich mit arithmetischen Schaltungen für Multiplikation, Addition und Gleitkommaoperationen.

Die Anzahl an Pins, die eine integrierte Schaltung mit der Umgebung verbinden, konnte nicht im gleichen Maße gesteigert werden wie die Rechenleistung ("pin limitation"). Daher muss die Datenrate pro Pin gesteigert werden, indem parallele Daten für die Übertragung serialisiert werden. Dies ist die Aufgabe von SerDes-Schaltungen, die Daten im Bereich von mehreren Gigabit pro Sekunde übertragen. Dies erfordert unter anderem eine Impedanzanpassung, die verhindert, dass Teile des Signals in den Übertragungskanal reflektiert werden. Allerdings wird die Qualität dieser Schaltung durch die Kapazität von Schutzdioden gegen elektrostatische Entladungen beeinträchtigt. Diese Arbeit beschäftigt sich mit möglichen Schaltungen, um diese Kapazität durch Spulen zu kompensieren.

Im Vergleich zu anderen Bauteilen sind Spulen nur selten in Process Design Kits (PDKs) zu finden, sodass deren Layout und Modellierung vom Designer selbst übernommen werden müssen. Daher stellt diese Arbeit verschiedene Methoden der Layout-Erzeugung für Spulen vor. Zusätzlich werden die Möglichkeiten analytischer Methoden zur Berechnung der Induktivität erforscht und ein neuer Ausdruck für die interne Induktivität von geraden Drähten wird hergeleitet. Die Auswirkungen von Oxid, Substrat, Skin Effekt, Metal Fill und Prozesstoleranzen, sowie die Verwendung von Feldsimulatoren werden untersucht. Mit Hilfe eines Markov-Chain Monte-Carlo Fitting-Algorithmus kann die Design-Lücke zwischen Ersatzschaltbild und simulierten S-Parametern geschlossen werden. Mit diesen Erkenntnissen wird am Beispiel einer T-coil ein Modell erzeugt, welches – ähnlich wie PDK-Modelle – zum Optimieren von Schaltungen genutzt werden kann.

Mikroprozessoren in Rechenzentren und Endnutzergeräten enthalten fast ausschließlich proprietäre Technologie großer Firmen, u.a. Instruktionssätze (ISAs), die hohen Lizenzgebühren unterliegen. An der University of California, Berkeley, wurde daher 2010 die offene RISC-V ISA veröffentlicht, die großen Zulauf in den letzten Jahren erhalten hat. Viele Firmen und vor allem Universitäten entwickeln nun – oftmals kleinere – eigene RISC-V-Prozessoren. Damit RISC-V auch im Bereich der Hochleistungsrechner eingesetzt werden kann, werden schnelle arithmetische Schaltungen gebraucht.

In dieser Arbeit wird eine IEEE 754 Gleitkommaeinheit für "fused multiply-add"-Operationen bzgl. Leistung, Taktrate und Flächenbedarf analysiert, die am Lehrstuhl für Rechnerarchitektur der Universität Heidelberg entwickelt wurde. Vergleichbare Einheiten im "open-source"-Bereich nutzen abstrakte Beschreibungen für Multiplikation und Addition. In dieser Arbeit untersucht daher, ob auf Gatter-Ebene entworfene Schaltungen bzgl. der oben genannten Metriken einen Vorteil bieten. Es stellt sich allerdings heraus, dass heutige Synthesewerkzeuge in der Regel bessere Ergebnisse erzielen und der Aufwand vorzugsweise in andere Bereiche investiert werden sollte.

# Acknowledgments

A huge project like this thesis is never done without the support of others, whom I would like to thank here. First and foremost my wife Vera Wolthoff, who encouraged me to keep going, especially at times where the work seemed overwhelming. Her support was invaluable during the final months of writing this thesis. I am thankful to have her in my life.

I would like to express my gratitude to my advisor Prof. Ulrich Brüning, who provided me with the opportunity to write this thesis at the Computer Architecture Group. He supported me with valuable advice throughout the course of this work and my entire studies.

Special thanks go to Markus Müller and Maximilian Thürmer from Extoll GmbH for their advice and useful discussions, as well as for arranging the opportunity to include my inductor designs into actual silicon. Markus Müller also provided some valuable support during phases when the tools just did not seem to ever simulate my designs.

I am grateful to my colleague and fellow PhD student Tobias Markus for the great time and discussions we had throughout the years.

Finally, I would like to thank my parents for their support over almost a decade of studying. Their continued encouragement motivated me to complete this thesis.

# Contents

Chapter 1

# Introduction

Substantial advances in semiconductor fabrication have accelerated technological innovation in the last decades according to Moore's Law. Device sizes have shrunk several orders of magnitude and while early computers had separate components for various tasks, i.e. external graphic processing units or I/O controllers, it is now possible to integrate nearly all functionality onto one die, or at least package. As a consequence, these designs are now called System-on-Chips (SOCs). High integration comes with reduced energy consumption for the same performance since interconnects between and the size of the components is reduced. This allows to integrate more and more functionality and to also raise the performance, which in turn leads to an overall increase in power consumption of the SOC. However, this trend is heavily constrained by two major factors: pin count and power. This thesis will address one specific aspect concerning each of these two limitations.

With increased functionality, more pins are needed to supply the SOC with data, but pin size has not scaled down enough to keep up with the increasing transistor count. Fig. 1.1 shows the pin count of sockets, mainly for x86 processors from Intel and AMD, starting with the 16-pin Intel 4004 from 1971. The TR4/sTRX4 sockets for AMD Ryzen Threadripper and the SP3 socket for AMD EPYC processors have the highest pin count with 4094. This is an increase of around 256 times within the last 50 years, during which the transistor count increased from the Intel 4004 with 2250 transistors to around 40 billion transistors of an EPYC Rome processor. Thus, the pin increase is faced with roughly an 18 million times increase in transistors. Therefore, the number of package and die pins is a serious limitation for large SOCs and conversely the data rate per pin has to be increased. Wide parallel data paths are still used on-chip at comparatively moderate frequencies in the high megahertz and low gigahertz regimes. To communicate

with other chips, this parallel data has to be serialized and increased in frequency in the same ratio to match the bandwidth. This functionality is provided by a *serializer*, and *deserializer* at the receiving end, with the pair usually called *SerDes*. Given the analog nature of signals on cables or Printed Circuit Board (PCB) tracks, generally referred to as *channels*, a SerDes is also the interface from the digital domain of computing to the analog domain of signal transmission. As an interface to the outside, the SerDes needs to be protected against Electrostatic Discharge (ESD), which may occur during assembly or handling of the chip. This is actually done with all sensitive pins, however, especially at high data-rates, the ESD protection degrades the signal quality at the SerDes in- and output. It also causes reflections back into the channel, which harm signal integrity and lead to reduced link performance or even failure. To achieve both ESD robustness and good signal quality, on-chip inductors can be applied and their design for this application is one aspect addressed by this thesis.

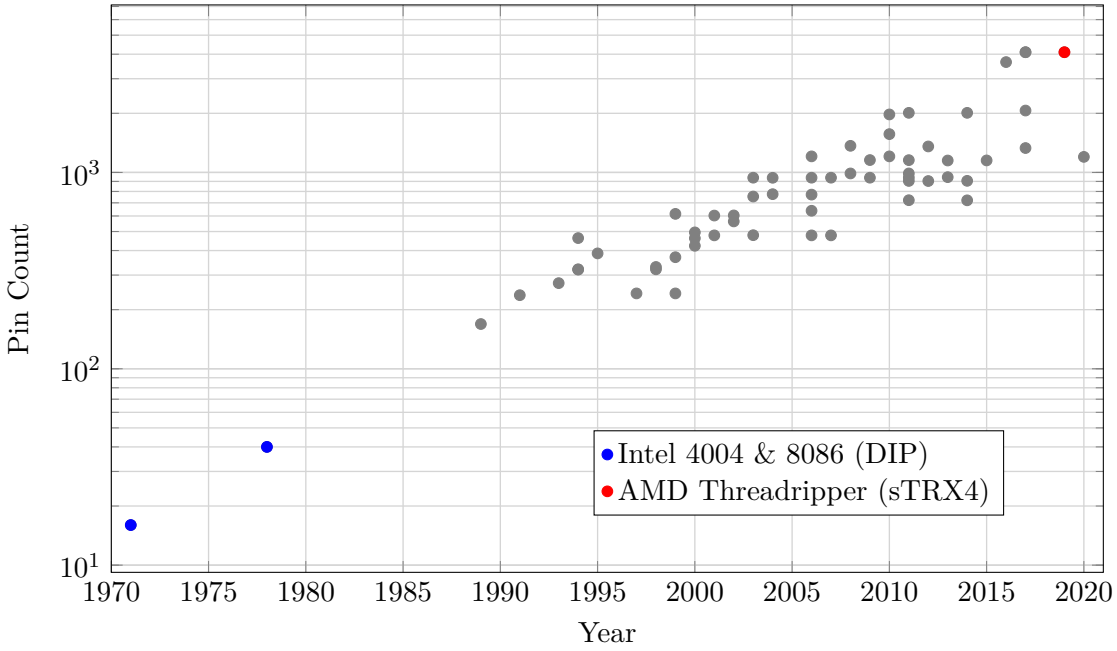



**Fig. 1.1:** Socket pin count of x86 desktop and server CPUs, starting with the Intel 4004 in a dual in-line package (DIP) with 16 pins, and up to 4094 pins of a current AMD Threadripper sTRX4 socket. Data taken from [1].

The second major limitation is the electrical power consumed by chips, the so called *power wall*, which is directly related to their operating temperature. Hence, the thermal integrity of the chip limits the maximum power it can draw in order to allow the corresponding cooling technique to dissipate the generated heat. As power is proportional to the product of clock frequency and the square of voltage, this also places a performance constraint on digital logic. Historically, processor speed has been improved by increasing

the clock frequency. If the clock frequency scaling of the 1990s would have continued, processors would have run at more than 10 GHz by 2010. Obviously, this did not come true and at the end of 2020, the fastest clocked commodity desktop processor – the Intel Core i9-10900K – offers a *single-core* "turbo" clock frequency of 5.3 GHz [2]. Although significantly higher clock rates have been achieved through "over-clocking", this is only possible via expensive cooling solutions and increased operating voltages. The latter enables higher speeds due to faster signal slew-rates, which effectively causes power to depend more than linear on frequency while computing performance stays linear. Hence, increasing the voltage above the nominal value results in a decreasing "performance per watt" ratio. As a consequence, CPU design has shifted towards *multi-core* microprocessors, and clock frequencies have remained more or less constant, except for minor increases. Modern multi-core processors are capable of frequency and voltage scaling to lower their power consumption while idle or increase performance when demanding software is executed. A general observation is that single-core clock frequencies are often higher than multi-core clock frequencies, which is primarily due to power limitation when all cores are used. Assuming software can be parallelized, adding more cores increases performance and power consumption approximately linearly. As many workloads are embarrassingly parallel, for example Monte-Carlo codes or hosting a large number of clients on a single server, this has allowed to continue the scaling of CPU performance with transistor count. The largest core count for x86 CPUs is offered today by AMD's Threadripper and EPYC processors with 64 cores and 128 threads via Simultaneous Multithreading (SMT). However, somewhat countering the trend towards higher integration, AMD uses a *chiplet* approach to maintain the yield for CPUs this large. Instead of a single die, up to nine dice are used within an EPYC Rome package [3]. The dice are connected within the package via AMD's Infinity Fabric on-package, a SerDes-based communication scheme for short ranges of 10 to 20 mm [3]. Besides the power wall, which caps the maximum performance, energy costs and battery life are also two major drivers towards more energy-efficient hardware.

In addition to the technical limitations for high performance chips, economical costs are also to be considered. Top of the line processors are nowadays almost exclusively manufactured by large companies – for example Intel and AMD for x86 processors in the desktop and server market. Whereas mobile devices are dominated by ARM-based chips, which is a Reduced Instruction Set Computer (RISC) architecture that has to be licensed from ARM Ltd., either in form of IP cores or an architectural license, if detailed customization is intended. The high licensing costs associated with the aforementioned Instruction Set Architectures (ISAs) in conjunction with the high engineering and manufacturing costs has kept most smaller companies or universities from attempting custom processor development. To change this, the RISC-V ISA was created at the

University of California, Berkeley (UC Berkeley) [4]. It has gained great momentum in the last years as the ISA has a high quality and zero licensing costs. Additionally, a tremendous ecosystem in terms of software support has developed in the last years, including compiler and operating system support. First hardware implementations have also been manufactured by various companies and research groups. An example are Western Digital Corp.'s SweRV cores, which are used to replace their current state-machine logic and flash controllers, and are also employed for other embedded tasks [5]. The source code has actually been made open-source by Western Digital on GitHub [6], further strengthening the popular belief that RISC-V will become the "Linux of hardware". To develop and maintain the RISC-V ISA itself, the RISC-V foundation was created. The availability of this free, well supported ISA has started a new "era" of computer architecture. In the context of this new ISA and the trend towards energy-efficient high performance computing, this thesis presents an analysis of addition and multiplication in terms of Power-Performance-Area (PPA). Furthermore, a Fused Multiply-Add (FMA) floating-point unit developed by Kaiser at the Computer Architecture Group (CAG) at Heidelberg University [7] is analyzed regarding PPA.

## 1.1 Motivation

As highlighted before, this work presents two topics addressing the current main bottlenecks of High Performance Computing (HPC) systems, pin and power limitation. The first part of this work is motivated by the necessity to incorporate ESD protection into high-speed SerDes designs to avoid the failure of expensive and critical components due to ESD damage. The ESD protection devices are usually diodes capable of deflecting high currents for a short amount of time. However, they possess a parasitic capacitance, which will impede the signal termination at the I/O pin. Most standards, e.g. Peripheral Component Interconnect Express (PCIe), allow only a certain amount of signal reflection. Thus, the parasitic capacitance has to be compensated to be standard compliant and provide ESD protection. This is usually achieved with on-chip inductors [8]. However, the challenge is not only to understand the circuit level of such termination networks but also how to map their inductors to a corresponding layout. While several designs of different termination networks are discussed in literature, e.g. [9] and [10], the procedure of how these results were achieved is not shown. In particular, the inductor design process is not discussed, it is only benchmarked to show that it is feasible in the given context. Hence, it seems the design of custom on-chip inductors does heavily rely on the experience of the designer.

Somewhat simplified, on-chip inductors are wires routed with the intention to leverage their parasitic inductance. Besides the design rules, there are few constraints, which

creates many degrees of freedom for the designer. This freedom is usually mitigated for other circuits with relatively fast feedback through simulation and parameterized cells for layout design. However, for on-chip inductors both are not available. So designs in literature have to be heavily experience-based with a certain amount of "trial and error". In any case, the topic is poorly covered in literature so far. Most work concerning the layout and properties of on-chip inductors is relatively old, with the oldest one known to the author dating back to 1972 [11]. Thus, the motivation for this work is to analyze the physics and tools needed to conduct a more "straight forward" inductor design, replacing the usual procedure of iterating through many designs.

The previously introduced RISC-V ISA has sparked a large number of developments in processor design throughout industry and universities. The CAG at Heidelberg University also planned to organize a project in this context, together with several partners from universities and companies [12]. Part of this proposal was to develop a scalable energy-efficient processor architecture based on RISC-V. The proposal was eventually rejected, but development of a RISC-V pipeline and related components continued within the framework of several bachelor and master theses at the CAG, e.g. [13], [14]. Every processor development eventually needs an Arithmetic Logic Unit (ALU), where the actual processing is performed, and which is therefore a special point for potential optimizations with regard to energy-efficiency and performance. Floating-point performance in particular is crucial for most scientific applications. Thus, an FMA unit was developed and verified during a master's project by Kaiser [7] at the CAG. Here, this FMA unit was synthesized and a power analysis was conducted in a 22nm technology. The results of design, verification, and synthesis were published in [15]. The multiplier within this unit turned out to be a limiting factor. Therefore, this work analyzes several multiplier and adder structures in terms of PPA to investigate if there are potential gains by implementing them with structured code on the gate-level. Most open-source designs only use built-in operators to describe these operations and rely on the synthesis tools to efficiently implement them.

## 1.2 Contributions

This work contributes to the aforementioned topics in the following ways:

#### ESD Compensation and On-Chip Inductors

- An analysis of circuits using two-terminal inductors and T-coils to compensate the parasitic ESD device capacitance for SerDes designs in advanced nodes is conducted. Special attention is paid to the capacitance of the pads that connect the SerDes to

the package, and the series resistance of the T-coil.

- On-chip inductor design is heavily based on the experience of the designer and usually an iterative process. Custom on-chip inductors are often necessary because foundry-provided parameterized cells (PCells) are rarely available and not flexible enough to cover the required properties. So this work evaluated several methods to create PCells for on-chip inductors.
- Analytic approaches for on-chip inductor sizing have largely vanished from literature in the last one to two decades. The emphasis of most publications is on the schematic level or on results measured or simulated for a specific design. However, this does not aid the design of new on-chip inductors. A lot can still be learned from researching analytic methods for inductance calculation developed in the last 100 to 150 years. This work provides a comprehensive overview on the basics of inductance calculation and derives a new expression for the internal inductance of round wires, which has not been published before to the knowledge of the author. Furthermore, the background on the mean distance method for inductance calculation of wires with an arbitrary cross-section is explained.
- The design and optimization of inductors is usually done with simplified schematics called lumped models. However, the actual layout does comprise several additional effects that can only be partly represented with such models. This work discusses the influence of oxide and substrate, skin effect, metal fill, and process corners.
- Accurate electromagnetic simulation of on-chip structures like inductors has become a necessity at data-rates supported by state-of-the-art serial links. Simple layout extraction does not suffice anymore. The difficulties and experiences with some tools are reported in this work. Furthermore, it presents an automated characterization flow for on-chip inductors based on existing Electronic Design Automation (EDA) tools, which significantly speeds up the design space exploration. To close the loop to the schematic design phase, the extraction of lumped model parameters from simulated S-parameters has been implemented with a Markov-Chain Monte-Carlo (MCMC) code. Some early results were presented at CDNLive 2018 [16].
- Unlike other devices used in integrated circuit designs, custom inductors can hardly be optimized in the schematic design phase. Devices like transistors can be sized and simulated with a foundry provided model that translates the geometry into electrical characteristics. This is not possible for custom inductors. Therefore, a methodology to create an analytic inductor model has been derived with the help of the automated characterization flow, the fitting technique and formulas for analytic inductance calculation. This methodology facilitates a forward design flow of on-chip inductors, instead of the usual “trial and error” approach. The resulting

model may be used for "inductor synthesis", i.e. mapping a lumped model inductor schematic to a corresponding layout that is optimized with regard to the created model. This presents an important step towards closing the design gap of on-chip inductors compared to other devices.

- A brief evaluation of ESD compensation test structures applying T-coils in a 22nm technology, including measurement results, is presented. Furthermore, some lessons learned in regard to the termination layout are summarized.

#### Arithmetic Circuits

- A discussion of a SystemVerilog FMA unit design and its verification, as well as a PPA analysis in a 22nm technology, are presented. The results have been published in vol. 6 no. 2 of *Supercomputing Frontiers and Innovations* in 2019 [15].
- Open-source RISC-V core designs do not incorporate specialized circuits for addition and multiplication but instead rely on synthesis tools to efficiently map Hardware Description Language (HDL) operators like "+" and "∗" to hardware. This work analyzes potential benefits of structural code for adders and multipliers in modern technologies and with modern EDA tools. Therefore, the theory behind hardware adders and multipliers is revisited and the resulting circuits have been designed at the gate-level and compared in terms of PPA.

## 1.3 Outline

This thesis takes an in-depth view at two aspects of high-speed integrated circuit design. Chapters 2 and 3 are dedicated to the first aspect, on-chip inductors for high-speed SerDes designs. The second aspect is the analysis of arithmetic circuits and is covered in chapters 4 and 5.

Chapter 2 introduces SerDes technology and places the ESD protection into this context. It then discusses different circuits using inductors to compensate the parasitic capacitance of the ESD devices. In chapter 3, the focus switches to the design of on-chip inductors. It starts with the layout process and presents multiple ways to automate it with PCells. This is followed by a derivation of the inductance of a straight wire and the introduction of the mean distance method for inductance calculations. The third section of this chapter discusses different ways to model inductors and the influence of oxide and substrate, skin effect, metal fill, and process corners. It should then be obvious that no exact analytic formulas exist, so the next section focuses on experiences with field solvers and layout extraction tools. It also describes the automated characterization flow developed in this work. In the fifth section, the MCMC fitting technique is presented, which fits lumped

circuit models to simulated S-parameters. Finally, all previously discussed aspects of on-chip inductors are combined to derive a model that maps geometry to lumped models parameters. The chapter is then concluded with a brief discussion of Time-Domain Reflectometer (TDR) measurement results of test structures in a 22nm node, and some lessons learned.

Chapter 4 starts with a short introduction to floating-point number formats, namely IEEE 754 and Posits. It then presents design, verification, and a PPA analysis of the RISC-V-conform IEEE 754 FMA floating-point unit developed at the CAG. Chapter 5 goes deeper into the details of arithmetic circuits and presents different architectures for adders and multipliers. They are compared in terms of PPA to generic "`+`" and "`*`" operators, which can be used in SystemVerilog.

The thesis is concluded in chapter 6 with a summary of the results, the insights gained from them, and an outlook towards future work.

Chapter 2

# High-Speed Serial Communication

With the ever increasing demand for computing performance and data storage, I/O technology is required to further increase bandwidth. However, as a consequence of pin limitation, this leads to increasing data-rates per pin, requiring circuit designers to deal with a variety of additional electrical effects, which become significant at higher signal frequencies. Another challenge on top of this is the ESD protection, which is accompanied by a parasitic capacitance that degrades high frequency signals. To put the ESD device compensation into context, section 2.1 provides an overview on some aspects of SerDes design. Furthermore, it briefly presents the architecture of the SerDes this work is based on, which is described in more detail by Müller [17] and Thürmer [18]. Section 2.2 continues with background on ESD protection circuits, while section 2.3 deals with on-chip inductors as a method to compensate the negative side-effects on signal quality introduced by the ESD protection.

## 2.1 SerDes Architecture

In addition to the aforementioned pin limitation, there are other reasons not to use parallel data transmission for off-chip communication in most applications. The routing of large interfaces needs more engineering time as well as more costly PCBs with a large number of layers. Furthermore, signal skew between the lanes of a parallel interface has to be carefully controlled [19]. An example is the routing of Dynamic Random Access Memory (DRAM) chips or even Dual Inline Memory Modules (DIMMs). Hence, scaling I/O bandwidth by using more lanes is very expensive and therefore not a feasible method. Instead, the bandwidth per pin is usually increased from generation to generation. PCIe is a perfect example for this development as the number of lanes a link can comprise

has been constant over the last iterations, but the data-rate per lane has significantly increased. This increase is much faster than the quality improvement of the transmission medium, e.g. of PCB traces, connectors and cables, called *channel*. As a consequence of the detrimental channel properties, the signal quality is heavily impacted in the domain of multiple Gbps links. To push data-rates further, additional circuits have been employed on both transmitter and receiver to mitigate these effects.

This section is split into three parts, firstly the properties of channels are discussed, secondly signal termination is explained in the context of serial links, and lastly an overview of state-of-the-art SerDes components is presented.

### 2.1.1 Channels

In conjunction with the data-rate, the channel is the most defining constraint for a SerDes design as it determines the quality of the transmitted signal to a large extend. The entire connection between transmitter output pad and receiver input pad is usually referred to as the channel. Channels typically exceed on-chip wire dimensions and are physically large compared to the wavelength of signals and are therefore modeled as transmission lines. However, they still vary in length significantly from tens of millimeters to tens of meters. The distortions a channel introduces into the signal are best visualized by the Single-Bit Response (SBR). An example is shown in Fig. 2.1.

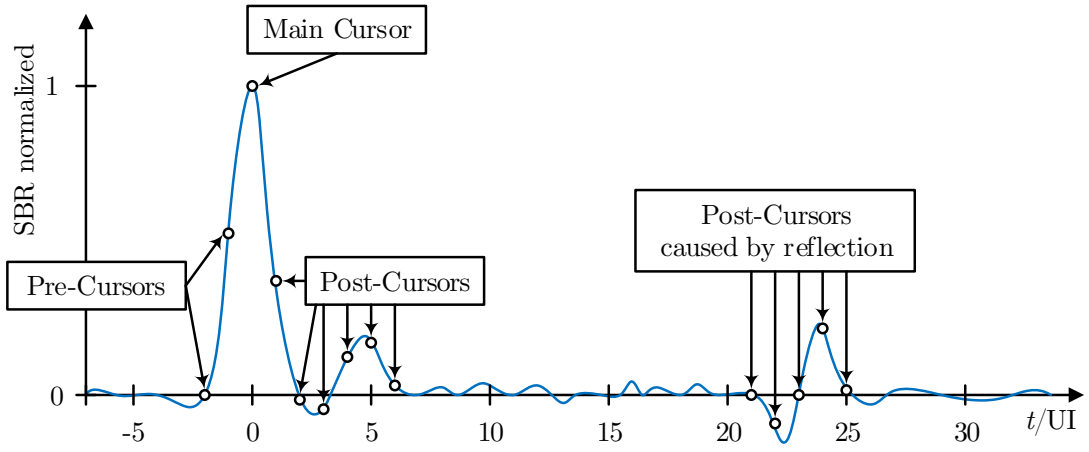



**Fig. 2.1:** Example of a single-bit response. Graphic constructed based on [19].

The SBR of a channel is the time-domain response to a singular rectangular pulse with a width of one Unit Interval (UI), i.e. a single bit. It is sampled by the receiver with the receiver clock, thus the magnitude is only of interest at discrete, UI spaced times marked with a small circle. The receiver clock is ideally aligned to the maximum of the SBR, the *main cursor*. Fig. 2.1 shows that a single bit actually causes non-zero cursors at many more sampling times, meaning symbols are interfering with each other, which is

called Intersymbol Interference (ISI). Depending on their position relative to the main cursor, these cursors are called *pre-* and *post-cursors*. Usually an SBR of a channel has much more post-cursors than pre-cursors [19]. One reason for this is that reflections at impedance discontinuities need some time to bounce back and forth to be seen at the receiver. Such late post-cursors are very difficult to compensate, which makes it crucial to avoid discontinuities within the channel, and at the transmitter and receiver. The former is part of package, track, connector or cable design while the latter is an explicit part of SerDes design.

### 2.1.2 Termination

To avoid reflection at the receiver or transmitter both have to present an input impedance identical to the characteristic impedance of the channel, $Z_0$, which is in many cases a $50\,\Omega$ impedance. This may seem arbitrary but has been chosen for most channels as the 30 dB cutoff frequency is maximized at this channel impedance for a coaxial cable [18]. So to achieve this, a termination resistor $R_T$ is applied, as shown in Fig. 2.2.

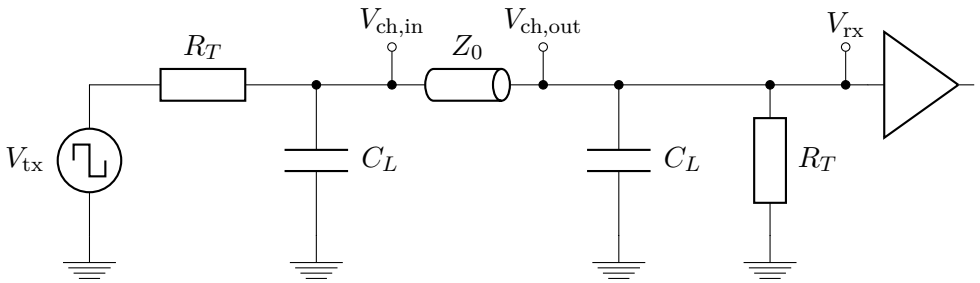



**Fig. 2.2:** Simplified transmission system with TX (left), channel and RX (right). Note that $V_{\text{rx}}$ and $V_{\text{ch,out}}$ are not the same voltage, even if it is drawn here due to simplicity. $V_{\text{ch,out}}$ comes from the channel and has therefore a source impedance of $Z_0$, which acts in series.

However, the ESD diodes add a significant capacitance $C_L$ and thus compromise the termination. In addition to $C_L$, there is also the pad capacitance increasing the capacitive load even further. The transfer functions and termination impedances for Fig. 2.2 are given by Eqs. 2.1 to 2.3. Here $x||y$ means the impedance of $x$ and $y$ in parallel and it is assumed to bind more than other arithmetic operators.

$$H_{\text{tx}} = \frac{V_{\text{ch,in}}}{V_{\text{tx}}} = \frac{Z_0||C_L}{R_T + Z_0||C_L} \tag{2.1}$$

$$H_{\text{rx}} = \frac{V_{\text{rx}}}{V_{\text{ch,out}}} = \frac{R_T||C_L}{Z_0 + R_T||C_L} \tag{2.2}$$

$$Z_{\text{tx}} = Z_{\text{rx}} = R_T||C_L \tag{2.3}$$

An important property of a transfer function is its cut-off frequency, where the magnitude falls below $1/\sqrt{2}$. For $H_{\mathrm{rx}}$ and $H_{\mathrm{tx}}$ the cut-off frequency is given by Eq. 2.4. Since $R_T = Z_0 = 50\,\Omega$ is desired for signal integrity, the expression can also be simplified a bit.

$$\omega_c = \frac{1}{C_L Z_0 || R_T} \overset{R_T \to Z_0}{=} \frac{2}{C_L Z_0} \tag{2.4}$$

Every impedance discontinuity along the channel and at its ends causes a part of the signal to be reflected. The amount of reflection is given by the one-port S-parameter magnitude $|S_{11}|$, where $Z_T$ is the termination impedance.

$$\Gamma = |S_{11}| = \left| \frac{Z_T - Z_0}{Z_T + Z_0} \right| \tag{2.5}$$

Note that SerDes usually operate on differential signals as this has various benefits like reduced crosstalk. Nevertheless, it is valid to consider the termination circuits for most calculations to be single-ended as the coupling between the channels is assumed to be zero. To ensure compatibility between devices, standards like PCIe or Ethernet also specify the quality of the termination a SerDes has to achieve. This is usually done by specifying an upper limit of the reflection as a function of frequency. As an example, Fig. 2.3 shows the masks for 10G and 25G Ethernet, PCIe 4.0 and JESD204C. The reflection for a $C_L$ of 300 fF is included for reference as this is a typical value for an ESD capacitance. It is lower compared to the standards' requirements, however these comprise not solely the level required for the ESD compensation alone but also include the pad and package. Note that the common mode return loss and sometimes even the common to differential conversion have separate constraints that need to be met.

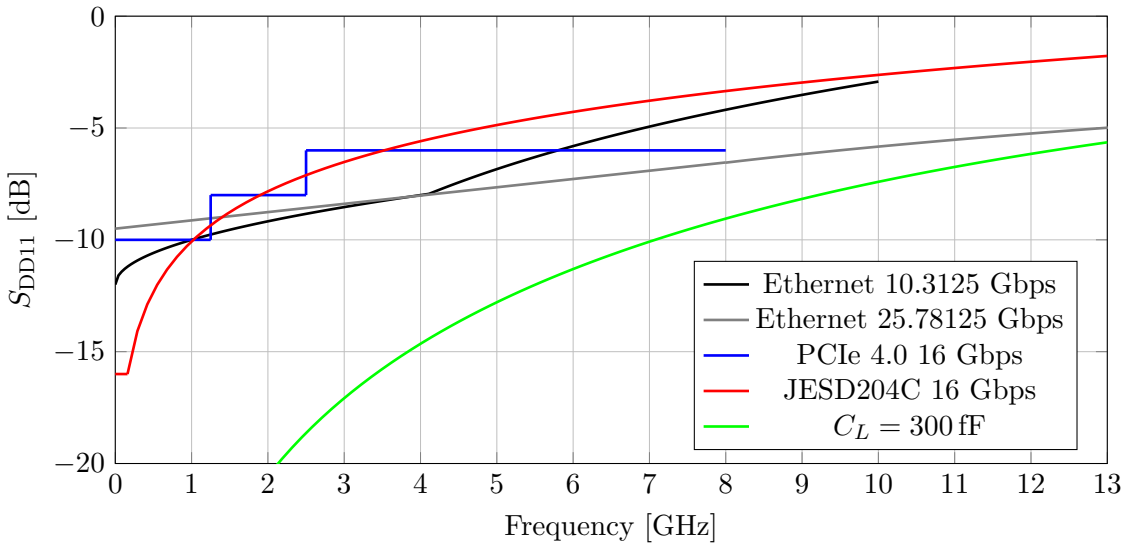



**Fig. 2.3:** Compliance masks for differential return loss $S_{\mathrm{DD11}}$, including package.

### 2.1.3 Components

While the requirements for SerDes are manifold depending on their target application, the encountered problems are similar and common methodologies to deal with them are applied in state of the art designs. A SerDes is composed of a number of different functional blocks that are dedicated to lower the Bit Error Rate (BER) of the link. Fig. 2.4 shows the coarse architecture of the SerDes that provided the context for this work. The upper part and bottom parts show the receiver and the transmitter units, respectively, which together are also called *transceiver*. In addition, a Phase-Locked Loop (PLL) provides the clock signal for both. The overall idea of this architecture is to use equalizers to flatten the system transfer function. The three most common techniques to do this are a Feed-Forward Equalizer (FFE) at the TX and a Continuous-Time Linear Equalizer (CTLE) and a Decision Feedback Equalizer (DFE) at the RX. Furthermore, as the system is self-clocking, a clock-recovery is implemented to recover the clock from the data-stream. Together, these two mechanisms are sufficient to achieve high data-rates for typical channels. This subsection takes a brief look at some of these components.

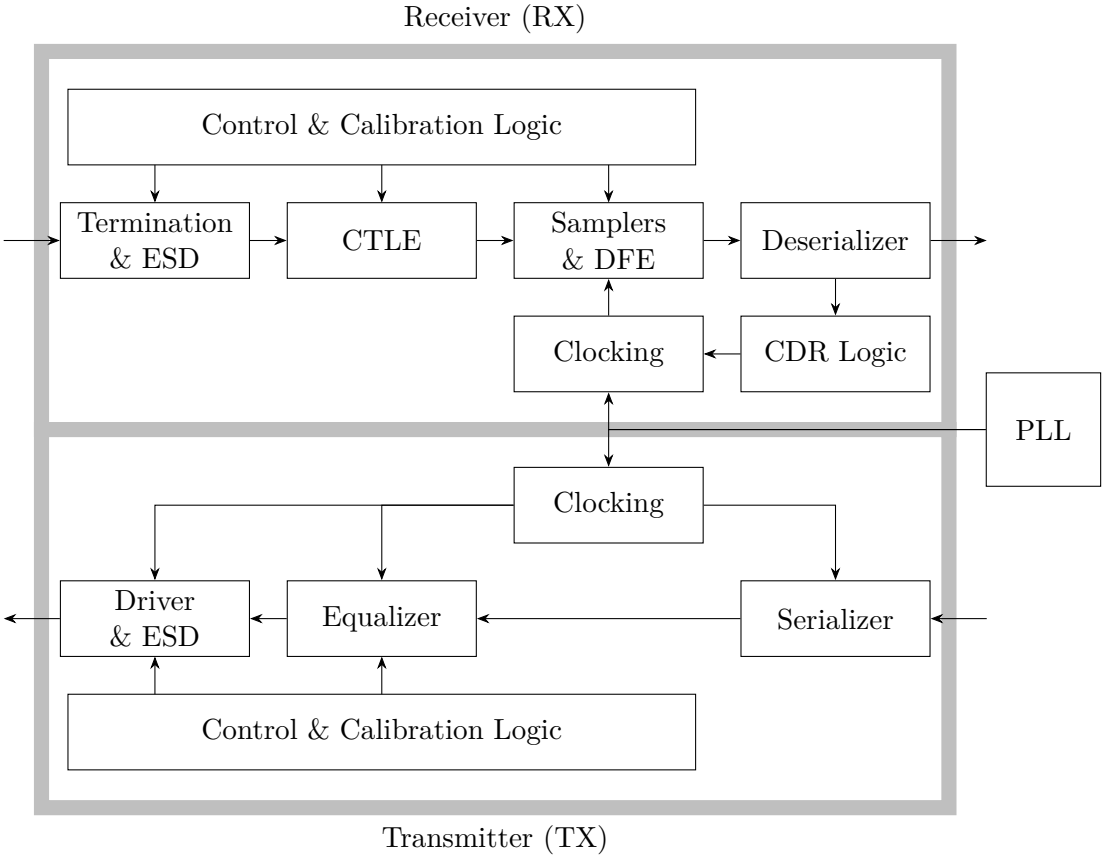



**Fig. 2.4:** Simplified SerDes architecture overview with the serial interface on the left and the parallel interface on the right (based on [17]).

### Feed-Forward Equalizer

The FFE is usually a Finite Impulse Response (FIR) filter, i.e. it combines delayed versions of its input to generate a weighted sum output signal. This is usually done at the TX because the delays can be realized with flip-flops without introducing noise. Depending on the choice of the coefficients, pre- and post-cursors of the channel can be lowered, but it is not feasible to use the FFE to cancel reflection induced cursors. Since the signal swing is limited, the low frequency content is damped to emphasize the high frequency content in the signal. This technique can also be used to modulate a PAM-4 signal, which is a 4-level signaling scheme used to increase the bandwidth that is expected to be part of PCIe 6.0. According to an analysis of Yuan [19], most designs use a maximum of four taps due to diminishing returns. Certain (minimum) configurations can also be demanded by standards, e.g. PCIe 4.0 demands a 3-tap FFE. As digital bits are weighted to produce the output signal, the FFE is usually described in the $z$-domain, as shown in Fig. 2.5. Common topologies for the driver itself are Current Mode Logic (CML) or Stub Series Terminated Logic (SSTL). The implementation of an SSTL driver for this SerDes architecture is described in [20].

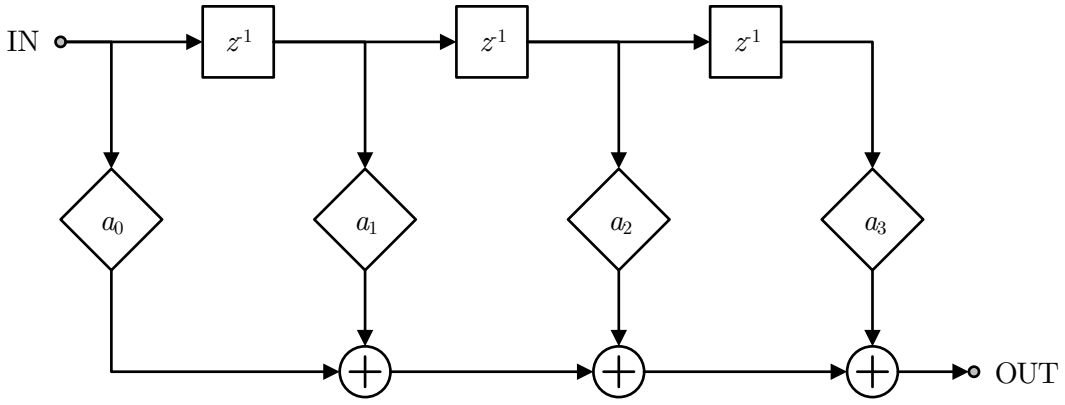



**Fig. 2.5:** Block-diagram of a 4-tap FIR filter.

### Continuous-Time Linear Equalizer

Within the analog front-end of the receiver, the overall low-pass behavior of the channel is equalized with a CTLE in most SerDes designs. An example schematic of an active version is shown in Fig. 2.6. It is basically an amplifier with a split current source to implement a source degeneration using $R_S$ and $C_S$. The transfer function is given in Eq. 2.6 and it can be tuned by adapting the degeneration with an equalization algorithm. The zero is used to cancel the first-order pole of the channel, while the poles are used to

amplify the higher frequencies depending on the data-rate.

$$H_{\mathrm{CTLE}}(s) = \frac{g_m}{C_D} \cdot \frac{s + \frac{1}{R_S C_S}}{\left(s + \frac{1+g_m R_S/2}{R_S C_S}\right) \cdot \left(s + \frac{1}{R_D C_D}\right)} \tag{2.6}$$

A drawback of the CTLE is the noise amplification as it cannot distinguish between signal and noise. Thus, Signal-to-Noise Ratio (SNR) is traded with bandwidth. Therefore, the sizing of the CTLE circuit comprises a lot of trade-offs, which also involve other components of the receiver. More details on CTLE design are found in [18].

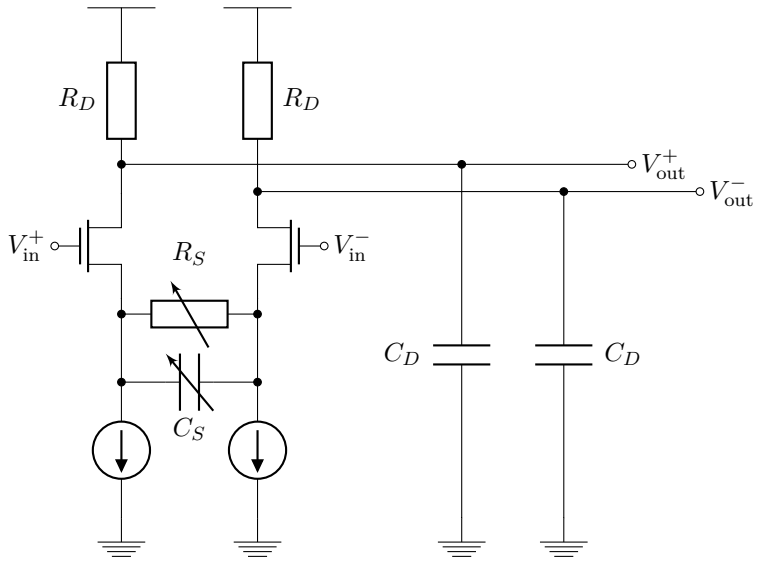



**Fig. 2.6:** Active CTLE with source degeneration.

### Decision Feedback Equalizer

In contrast to the CTLE, the DFE does not amplify noise and is an efficient method to remove post-cursor ISI. It is build as an Infinite Impulse Response (IIR) filter by feeding back the received bits the sampler has already decided on. The received data is weighted according to the post-cursors of the channel that are to be compensated and added to the sampler input to "correct" the sampled voltage, hence increasing the probability for a correct decision. This is of course a recursive approach that assumes the received bits were decided correctly, otherwise the correction will likely increase the error rate. However, typical serial links are operated at error rates at or even below $10^{-12}$, which is enough to get the DFE back if a single error occurs. The weighted summation of the previous bits has to be completed before the current bit is sampled, which imposes a tight timing constraint on the feedback path, especially when considering that a *full-rate* design, as shown in Fig. 2.7, has to run with the line speed. Since this is especially critical

for the first tap as the sampler typically has a longer clock to output delay than a digital flip-flop, so called *loop-unrolled* or *speculative* DFE architectures are built. They use two samplers, where each has the first tap statically applied with a different sign, thus the settling time is removed and a multiplexer then chooses the correct decision based on the previous bit. Additionally, the operating frequency of the DFE can be reduced using half- or even quarter-rate designs. A half-rate DFE splits the data stream into "even" and "odd" bits and uses basically two DFEs – but running at half the clock frequency – to sample the bits in both streams. This is possible by cross-coupling every second feedback path between the two and can also be extended to quarter-rate architectures at the cost of a huge increase in complexity. A detailed analysis of DFE architectures can be found in a previous work of the author [21].

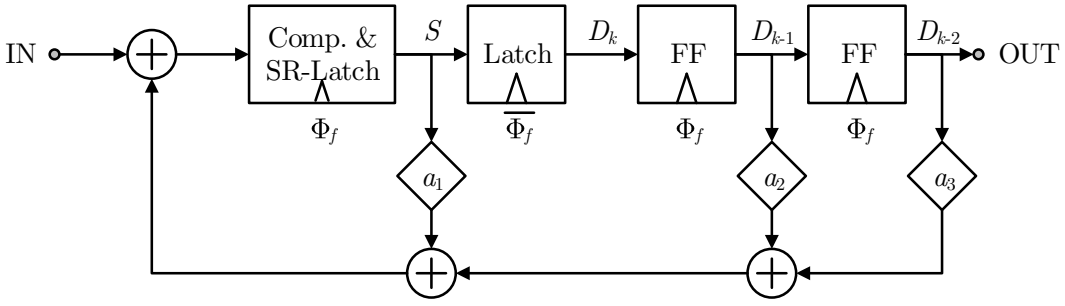



**Fig. 2.7:** 3-tap full-rate direct DFE architecture. $\Phi_f$ is the full-rate clock.

### Clocking

Serial links normally do not transmit a distinct clock signal along with the data, which would also be hard to do for todays ten to hundred Gbps links. The low bit times would require an extremely accurate skew management in the range of pico seconds. Thus, the data-stream has to be self-clocking, i.e. the TX has to assure enough transitions are present in the data stream to allow the RX to use a Clock Data Recovery (CDR) circuit to extract the clock and align the sampling position to the data. A detailed analysis of the CDR used in this SerDes can be found in [17].

### Physical Coding Sublayer

Besides the "raw" analog to digital interface composed of the components described so far, which is called the Physical Media Attachment Layer (PMA), additional measures are needed to provide a functional interface to the Media Access Layer (MAC). One standard for such an interface is the PHY Interface for PCI Express Architectures (PIPE). It defines a set of signals to ensure compatibility between the PHY (contains PMA and PCS) and MAC layer. The PCS performs data encoding, i.e. extending the data payload

by additional bits as well as *scrambling*, to ensure a minimum of transitions and enable clock recovery. A common coding scheme is an 8b/10b code, which adds two additional bits to each byte to ensure a maximum run length of five bits. However, this wastes one fifth of the link bandwidth and was thus replaced in more recent standards by 64b/66b and 128b/130b coding. It is used in PCIe 1.0 to 2.0 and some older Ethernet standards, whereas PCIe 3.0 to 5.0 have switched to a 128b/130b code. In addition to the line coding, data scrambling is often applied, which pseudo-randomizes the data stream so that it contains many transitions. This is usually done through an XOR combining the data with a Pseudorandom Binary Sequence (PRBS) generated by a Linear-Feedback Shift Register (LFSR). The same LFSR is used at the receiver to descramble the data. This does not need synchronization since the receiving LFSR can extract the state of the transmitting LFSR from the data stream (self-synchronizing).

## 2.2 Electrostatic Discharge

The oldest descriptions of electrostatic phenomena by humans date back into ancient Egypt, nearly 5000 years ago, when the Egyptians referred to electric fish as “thunderer of the Nile” [22]. The probably most commonly known electrostatic phenomenon is the separation of charge by rubbing amber on fur, which was observed in ancient Greece. The built-up electrostatic potential can discharge suddenly upon contact with conductive materials. If this happens with pins of fragile electric components, irreversible damage can occur – if they are not properly protected. So called ESD events can occur over the whole lifetime of semiconductor devices. Chip design, for instance, needs to follow antenna rules to avoid relatively large floating shapes connected to a gate, which could build up charge during manufacturing, ultimately leading to a breakdown of the gate insulator material (oxide breakdown). After manufacturing, in packaging and application, especially the pins need to be protected. This is done by handling sensitive devices in ESD controlled environments where charge cannot build up due to grounded machines and personnel. However, ESD events can still happen outside these environments or through careless handling. As a consequence, protection schemes need to be applied at the device level, i.e. on-chip. To verify these protection schemes, a series of ESD testing models has been standardized, which are presented in the next subsection. After this, a short overview of possible ESD failures is presented. The section is concluded with a discussion of ESD protection circuits usually applied in chip design.

### 2.2.1 ESD Testing Models

To verify the ESD robustness of a given protection circuit, may it be by physical testing or in simulation, there are several models specifying different ESD wave forms, which are assumed to cover certain scenarios of ESD. The most common ones are covered in this subsection. These models can be grouped according to the number of pins of the Device Under Test (DUT) involved. The first group involves two pins, so the DUT forms a conductive path between the ESD source and ground. An exhaustive test of all combinations of two pins might take a long time since modern packages can have thousands of pins. To provide an example, the current AMD Socket TR4, used for Ryzen Threadripper processors, is designed for ICs with a 4094-contact Flip-Chip Land Grid Array (FCLGA) package [23]. The second group involves only one DUT pin. Here the DUT and a conductive object are charged to different potentials and, upon contact, a displacement current flows to equalize the potentials.

The requirements for testing ESD robustness are specified in a plethora of standards. It is quite complicated to get an overview as several organizations are involved. However, the Electrostatic Discharge Association (ESDA), an independent trade association in the

United States, provides a comprehensive overview of ESD standards [24]. Historically, the first ESD standards where developed by the United States Military, which include MIL-STD-750, MIL-STD-883 and MIL-STD-1686 [25]. However, the effort of developing separate military standards decreased and the United States Department of Defense assigned the ESDA to transfer MIL-STD-1686 into the commercial standard ANSI/ESD S20.20. As the name of this standard suggests, the ESDA is accredited by the American National Standards Institute (ANSI). On an international level, the International Electrotechnical Commission (IEC), an international standards organization located in Geneva, Switzerland, has released standards under the name IEC 61340. These are also used in Germany as DIN IEC/TR 61340. Furthermore, there is IEC 61000-4-2, which concerns testing ESD immunity. Starting in 2010, the ESDA combined some of their standards with JEDEC Solid State Technology Association (JEDEC) standards. The most recent versions of the combined standards are called ANSI/ESDA/JEDEC JS-001-2017, which covers the Human Body Model (HBM), and ANSI/ESDA/JEDEC JS-002-2018, which covers the Charged Device Model (CDM). These two models are the most important ones for semiconductor devices. There are many more ESD related standards [24], but listing them here is beyond the scope of this document.

### Human Body Model (HBM)

The HBM aims to model the discharge from a charged human body. Note that this is a two pin event, i.e. the charge enters via one device pin and leaves over any one other pin to ground. Charge can build up in the body due to charge separation when walking. Shoes and/or dry air act as an insulator to the floor and environment. Particularly in clean rooms where the air is very pure, special measures need to be taken to assure charge cannot build up in personnel. It is possible to test structures with the HBM at wafer level during the development of new technology nodes [26].

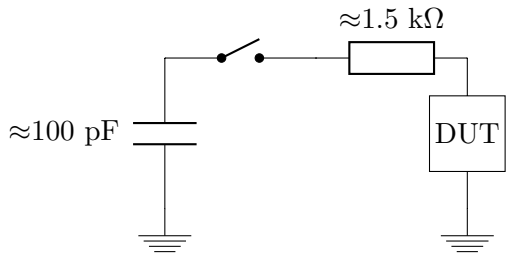



**Fig. 2.8:** Human Body Model discharge circuit according to JS-001-2010.

Fig. 2.8 shows the equivalent circuit used to test HBM ESD robustness. The charge stored on the human body is modeled with a capacitor, usually assumed to be 100 pF large, and the discharge happens through a relatively large resistor. Before testing, the

capacitor gets pre-charged to a defined voltage via a large resistor in the low MΩ range. The capacitor is then switched with a high-voltage relay to the discharge resistor of 1.5 kΩ. Commonly a voltage of ±2 kV is used, there are however several classes ranging from under 250 V to more than 8 kV according to the previously mentioned JS-001-2017 standard (this was taken from the 2010 version of the standard as the 2017 version is not freely available). The current waveform of the ESD event is also specified in JS-001-2010 and shown for 2 kV in Fig 2.9.

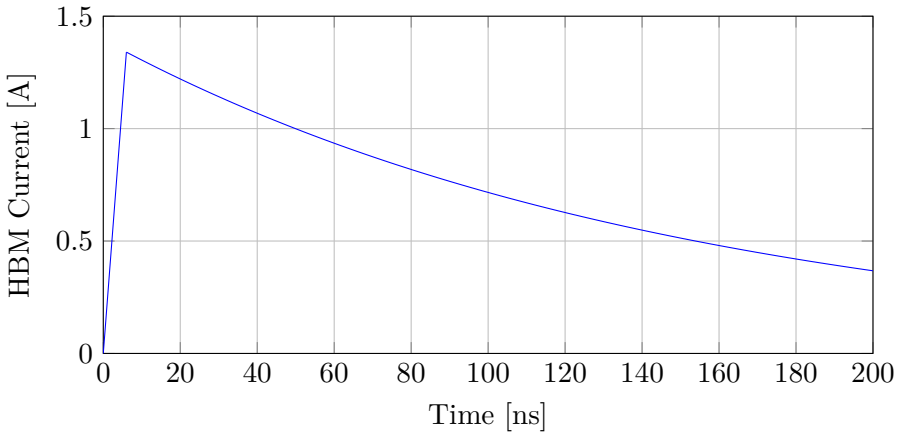



**Fig. 2.9:** Simplified HBM current waveform according to JS-001-2010 for a 2 kV discharge. The rise time usually lies between 2 ns and 10 ns and is probably caused by the finite switching speed of the relay – but is not present in the equivalent circuit. Peak current (1.2 A to 1.48 A) and decay time (130 ns to 170 ns) are defined by the resistor, capacitor, and pre-charge voltage.

### Charged Device Model (CDM)

This model considers a charged device instead of a charged external entity, and is to be classified as a single pin event. Charge is built up during handling of the device, e.g. when it "*slides along the plastic rails of the production line and discharges upon touchdown on the PCB*" [25]. The CDM "*is considered to be the [...] best representation of what can occur in automated handling equipment used in manufacturing and the assembly of integrated circuits (ICs) today. It is well known that the largest cause by far of ESD damage to an IC during device handling [...] is from charged device events*" [27]. Since the JS-002 standard is not freely available, the following information is from its predecessor ESDA standard ANSI/ESD S5.3.1-2009. For testing purposes, a controlled charge is required to be generated on the device. There are two methods described: The first is the direct charging method where the DUT pins are charged via a 100 MΩ resistor – either all simultaneously or only a $V_{SS}$ pin with ohmic connection to the substrate. The second method, the Field-Induced Charged Device Model (FICDM), uses an electric field plate

to charge the device. Both methods discharge after charging via one pin and repeat the procedure until all pins are tested.

The module capacitance is assumed to be in the range of 1 pF to 30 pF. Note that the S5.3.1-2009 standard does only specify waveforms for "verification modules" of 4 pF and 30 pF as the waveform depends on the DUT. The equivalent circuit and waveform shown in Fig. 2.10 and 2.11 are therefore only typical examples. The CDM event involves high but also highly damped currents. The important distinction to the HBM is the significantly shorter duration of around 2 ns. Similar to the HBM, several voltage classes are defined ranging from 125 V to 2 kV.

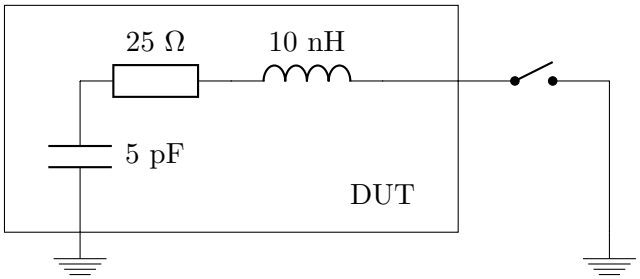



**Fig. 2.10:** Typical Charged Device Model equivalent circuit [25].

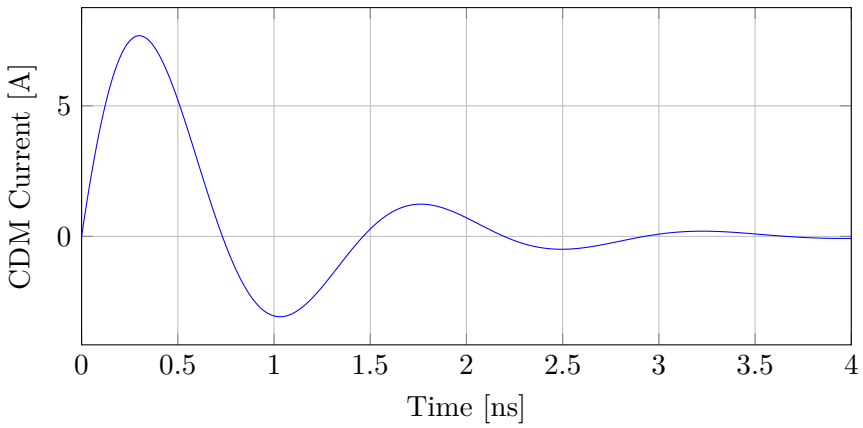



**Fig. 2.11:** Simplified CDM current waveform according to the circuit in Fig. 2.10, similar to S5.3.1-2009, for a 500 V discharge.

### Machine Model (MM)

This model simulates an arc discharge [28] of a charged machine component across two pins of the DUT. However, its relevance has decreased and it is, e.g., not required in the automotive industry anymore [25]. This fact is also underlined by the non-existing merged standard from the ESDA and JEDEC, which does exist for the HBM and the CDM – but it is still showing up sometimes and is therefore worth mentioning here.

The ESDA standard for the Machine Model (MM) is ANSI/ESD S5.2-2009. It gives an example circuit for testing as shown in Fig. 2.12. Compared to the HBM, the resistance is very low and the size of the capacitor is roughly doubled.

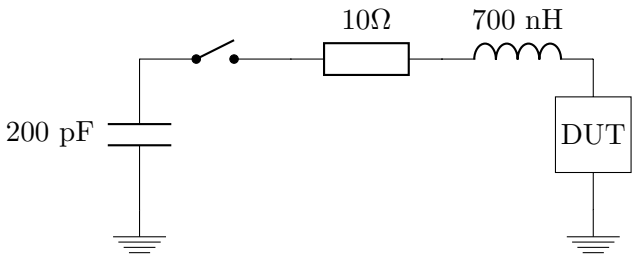



**Fig. 2.12:** Machine Model equivalent circuit according to S5.2-2009.

The current waveform, shown in Fig. 2.13, is a weakly damped oscillation with higher peak currents compared to the HBM. Again, there are several voltage classes defined by S5.2-2009, reaching from 25 V to 400 V.

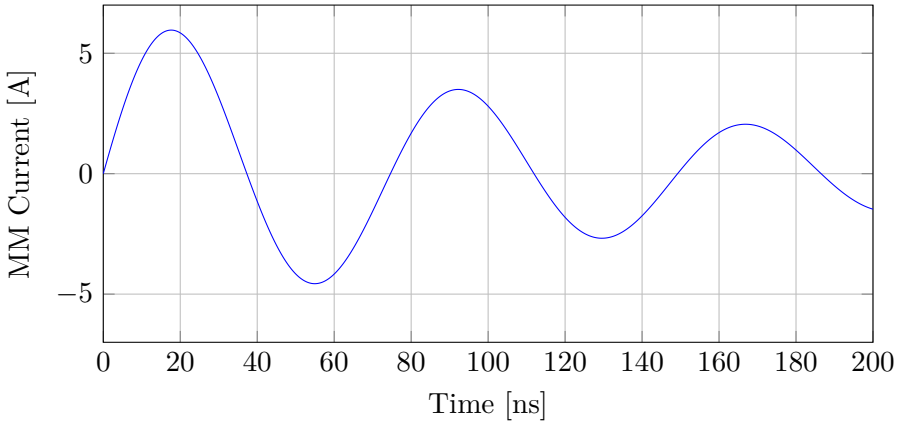



**Fig. 2.13:** Simplified MM current waveform according to S5.2-2009 for a 400 V discharge with the specified peak current within 7.0 A ± 10%. The period of the first pulse is specified to be between 66 ns and 90 ns.

### Transmission Line Pulse (TLP)

The Transmission Line Pulse (TLP) is not a required ESD test, but it is used during technology development to obtain the current voltage characteristics of the semiconductor devices during ESD events [26]. Standards on TLP testing are ANSI/ESD STM5.5.1-2014 and IEC 62615:2010. First commercial TLP testers became available in the 1990s, while the idea dates back to the time after World War II. Using short pulses in the form of square waves allows to measure the device properties without thermal destruction. The rise time is usually below 10 ns for a 100 ns pulse and the total energy is chosen to be identical to the HBM event [28]. To test shorter events like the CDM, the Very Fast

Transmission Line Pulse (VFTLP) has been developed, allowing for pulses with rise times in the low hundred pico second and pulse widths in the low nano second range. A standard for VFTLP is ANSI/ESD SP5.5.2-2007.
A TLP tester uses a voltage source to pre-charge a transmission line, which is then allowed to discharge into the DUT. The pulse width is directly determined by the length and propagation speed of the transmission line.

#### Other ESD Models

There are a more ESD models, listed for example in [28], but they are either not considered or not required in the technologies in focus of this work. So they are only briefly named and described here.

- *Charged Cable Model* – Discharge from a cable onto a chip or system. The cable is modeled by a relatively large capacitor in the order of 1 nF which is discharged through a resistor. Compared to Radio-Frequency (RF) signals the event is much slower and its duration is determined by the cable length.
- *Charged Cassette Model* – Models the discharge of a charged storage or game cartridge during mounting into the respective socket as it may appear in consumer electronics. Seems to have been developed in the era of popular handheld game consoles. The current waveform is similar to the MM, although the capacitance is much lower at around 10 pF.

#### Discharge Path

It should be noted that the current waveforms shown in this subsection are assuming that the discharge happens through a short. This is usually valid as working ESD protection circuits should ideally act as a short circuit to deflect the ESD current. However, predefined ESD modules for circuit simulators sometimes model the event with a fixed current source, which is either due to the above assumption of a short or some other reason.

### 2.2.2 ESD Failures

Another aspect of ESD are the failures inflicted to unprotected circuits, or by ESD events too strong for the protection devices. These failures can occur to almost any device on a chip if it is affected by the ESD event. The simulation of the ESD protection is usually limited to applying an ESD source module in a circuit simulator and observing voltages at different nets, which are assumed to break if certain voltages are exceeded. An example for this may be the gate of the input amplifier of a receiver circuit. This provides a baseline, but actual ESD failures can be much more complex and can occur at

unexpected places. Voldman [29] provides a comprehensive discussion of possible failures and their consequences. He also observed that an ESD design methodology or EDA tool is still missing, especially in comparison to advances made in other areas of chip-design. Nevertheless, it should be noted that a significant part of ESD design and testing is, and only can be reasonably done, by semiconductor companies – not chip design teams.

The first failure condition is a high temperature along the discharge path. Melting materials can change crystal structure or cause changes in doping concentration as well as displace material. Liquid metal may flow into the dielectric [29], which changes the resistance of interconnects. Even without melting, thermal strain caused by different expansion coefficients might cause mechanical stress between metal and dielectrics. The behavior of material under electrical heating can be described by various electro-thermal models [29], [30]. The second failure mechanism is exceeded breakdown voltages, which mostly concerns insulators. The most delicate of these is the gate oxide, which gets thinner with each node shrink to increase the control of the gate over the channel. However, this also decreases the voltage necessary to reach the critical electrical field strength for a breakdown. The same mechanism applies to any two neighboring metal lines, both vertically and horizontally, as well as vias. Therefore, metal capacitors formed from densely packed metal stripes, separated by dielectric, are vulnerable. The same is true for inductors built from interconnect wires, which are one main aspect of this work. Fig. 2.14 shows how the "under-pass" of an inductor presents a critical point for ESD failure. For the T-coil network discussed in the next section, Voldman identifies the part connecting pad and ESD devices to be the point of potential failure. However, as stated before, the simulation of possible breakdowns within interconnects is not supported in any feasible manner and is not focused on in this work.

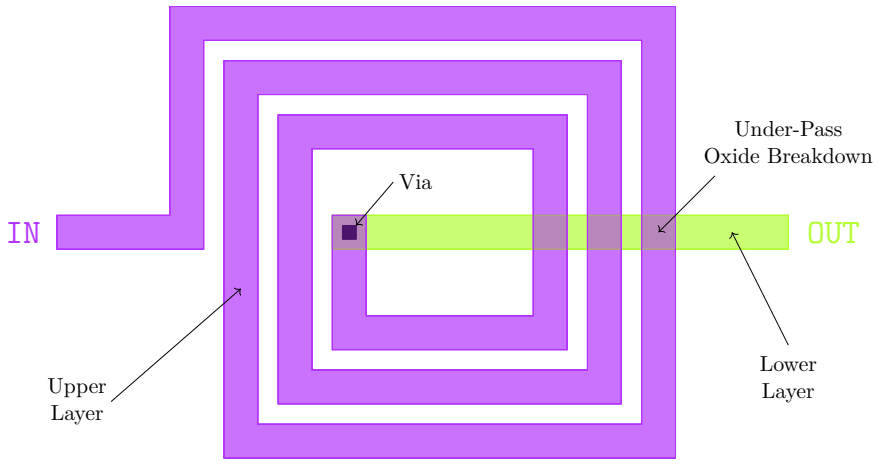



**Fig. 2.14:** Possible ESD failure of an on-chip inductor.

And thirdly, besides a complete breakdown of a device, there is also the possibility that it is only degraded. A MOSFET device may have its threshold voltage altered, e.g. through charge injection. Another failure mechanism is a so-called *soft breakdown* where the gate charge leaks into the dielectric. Such a soft breakdown usually leads to a *hard breakdown* in a relatively short time [29]. Also passive components may suffer from degradation as a change in impedance may modify properties like the reflection coefficient at RF pins.

Historically, ESD robustness of semiconductor chips has been at its peak in the late 1990s. It has increased until 1997 through improvements in design and process technology. However, when the performance of designs could no longer be increased as easily as "promised" by Moore's Law, lowering the ESD robustness was accepted. As data-rates continue to rise, this trend will likely shrink ESD protection in the future.

### 2.2.3 ESD Protection Circuits

So far, possible ESD events and failures have been presented so to conclude this section, a common ESD protection circuit is discussed. The circuit shown in Fig. 2.15 uses diodes and power clamps as protection devices. It includes all three different kinds of pins, power, ground, and I/O. There are other protection strategies using NFETs or thyristors, but they are not considered in this work as the latter, for instance, may have a trigger speed too low for fast CDM events.

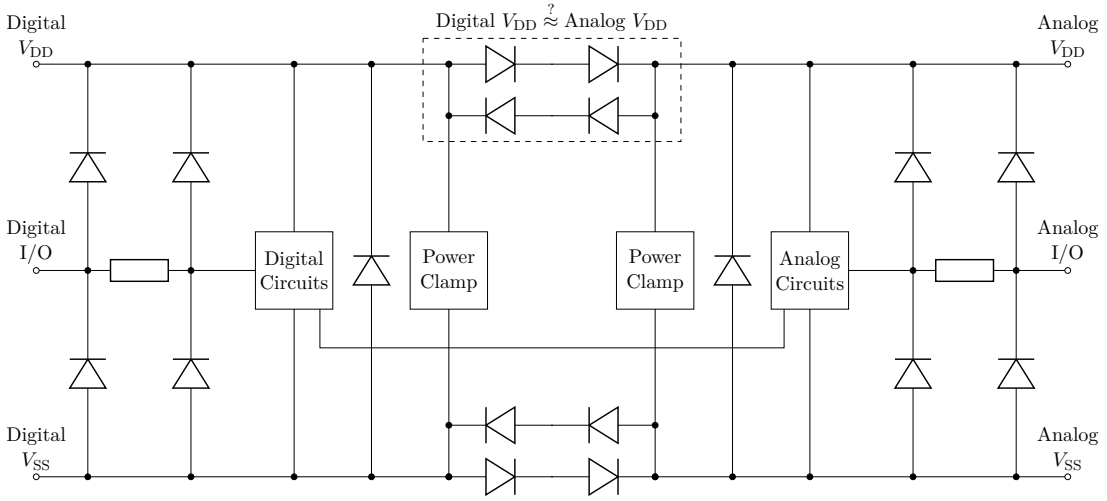



**Fig. 2.15:** ESD protection circuit for designs with two power domains, which can deflect any discharge polarity between any two pins.

The fundamental idea of ESD protection circuits is to deflect high currents through dedicated devices to protect the fragile internal circuits. So it would be ideal if the

protection device could act like a short circuit during an ESD event, while behaving like an open circuit during normal operation. This leads to diodes as basic elements for this job as they can conduct high currents or isolate two nodes. During normal operation, $V_{\mathrm{DD}}$ usually is larger than or equal to the I/O voltage, which in turn is larger than or equal to $V_{\mathrm{SS}}$. Thus, diodes can only be applied in one direction between those nets, otherwise they would prevent the protected circuit from functioning. Due to this however, it is also not possible for the diodes to deflect ESD events with the same voltage relation. Hence, an additional circuit is needed to also protect against this second half of events, which is called *power clamp*. A frequency-triggered power clamp as it is commonly applied [31] is shown in Fig. 2.16.

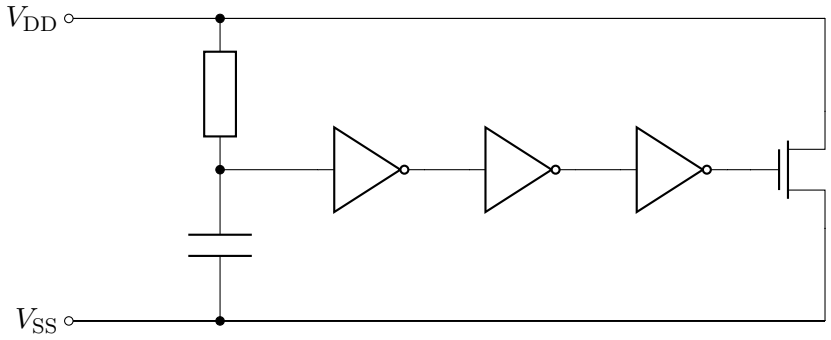



**Fig. 2.16:** RC- or frequency-triggered power clamp.

The power clamp is switched off at operating conditions, but in case of an ESD event, the RC branch triggers the inverter chain and turns on the discharge transistor, called *clamp device*. To trigger the inverter chain, the time-constant of the RC branch has to be greater than the HBM (and MM) time constant so that the spike on $V_{\mathrm{DD}}$ is not transferred to the input of the first inverter and it registers a "zero" due to the risen $V_{\mathrm{DD}}$. Furthermore, the RC time constant also determines the on-duration of the clamp device, which has to be long enough to deflect most of the discharge current. However, it cannot be chosen too large to avoid accidentally triggering the power clamp during ramp-up of $V_{\mathrm{DD}}$ at power-on. In more recent designs, the three inverters have been reduced to a single one for improved trigger speed [31]. Conversely, the inverter chain had the advantage of a smaller first stage that had less load on the RC branch. An alternative to the frequency-triggered is the voltage-triggered power clamp. It triggers when $V_{\mathrm{DD}}$ is above a certain threshold. In this design, the clamp device is held off via a resistor to $V_{\mathrm{SS}}$ and activated via a string of series-connected diodes connected to $V_{\mathrm{DD}}$. A pre-built power clamp is usually available in Process Design Kits (PDKs).

### Protection across Power Domains

The circuit in Fig. 2.15 is split into two power domains, which is a commonly encountered scenario, e.g. digital and analog domains are separated to avoid noise transfer from the digital to the analog circuits via the power and ground rails. The coupling of the two power domains via a pair of anti-parallel diodes is necessary to conduct discharge currents between pins belonging to two different voltage domains. However, this can only work if the voltages are similar enough to keep the diodes from conducting during normal operation. With more diodes in series per string, higher differences can be tolerated. Furthermore, as power domains are usually kept separate to lower noise, a trade-off in terms of ESD protection and coupling between the power domains has to be found. The device connecting the power rails can be omitted, but the ground device is usually required.

### HBM Protection

In advanced nodes, typically only HBM and CDM events are considered. The HBM event involves a discharge through any two pins, even pins associated with different voltage domains. So the protection scheme needs to provide a save current path for a large number of combinations of two pins and on top of that, for both negative and positive ESD voltages. Moreover, this protection circuit is expected to work while the chip is not powered, which is important as ESD events tend to cover scenarios during device handling. Looking only at a single voltage domain, there are six different discharge scenarios possible (three pin combinations times two voltage polarities). Their corresponding discharge path is listed in Tab. 2.1. Fig. 2.15 also shows that two stages of diodes are applied at the I/O pin. The ones closest to the I/O-pin are called *primary*, and the ones closest to the internal circuitry, *secondary* diodes.

| First Pin | Second Pin | Discharge Path |
|---|---|---|
| $V_{\mathrm{DD}}$ | I/O | power clamp and lower primary diode |
| $V_{\mathrm{DD}}$ | $V_{\mathrm{SS}}$ | power clamp |
| I/O | $V_{\mathrm{DD}}$ | upper primary diode |
| I/O | $V_{\mathrm{SS}}$ | upper primary diode and power clamp |
| $V_{\mathrm{SS}}$ | $V_{\mathrm{DD}}$ | diode parallel to power clamp |
| $V_{\mathrm{SS}}$ | I/O | lower primary diode |

**Tab. 2.1:** Possible ESD scenarios in a single power domain and their discharge path. This assumes a positive ESD voltage at the first pin. Swapping the pins results in the corresponding negative voltage discharge path.

The table shows that the power clamp, although it is placed between the power and ground rails, is also used for half of the ESD events involving the I/O pin. Thus, the

resistance of this whole path is important for ESD robustness. Regarding events between pins of different power domains, they are deflected similarly but are conducted across the inter-domain devices.

### CDM Protection

In contrast to the HBM event, the CDM emulates the discharge of a charged substrate through any single pin. In case this is an I/O pin, the primary diodes are insufficient to avoid damage at input gates. As the CDM event involves higher currents than the HBM event at around 5 A to 30 A, an active primary diode with an on-resistance of around 1 Ω still results in too high voltages across the receiver gate. Therefore, the secondary diodes, which are much smaller (roughly ten times), and an ESD resistor are used to implement a voltage divider that significantly lowers the gate-source voltage to avoid oxide breakdown. This additional protection is usually only required for gates, i.e. typically at input pins, and omitted for outputs.

### Parasitic Diode Capacitance

During normal operation, the diodes are in reverse direction and their significant property influencing RF signals is their reverse capacitance. This capacitance is caused by the depletion region around the p-n junction in reverse operation. Its thickness depends on the magnitude of the reverse voltage and as a consequence the capacitance does so, too. A common relation for this is given by Eq. 2.7.

$$C_j(V) = C_j(0) \cdot \left(1 - \frac{V}{V_{bi}}\right)^{-n_j} \tag{2.7}$$

The built-in or diffusion voltage $V_{bi}$ depends on the doping strengths of the p and n area. $n_j$ depends on the doping gradient from one contact to the other. An abrupt junction yields $n_j = 0.5$, while a constant gradient yields $n_j \approx 0.3$. The voltage dependent capacitance is to be considered when the compensation circuit is designed, which is done by simulating the diodes at the correct DC bias. However, it will not be possible to design voltage dependent inductors compensating the transient, signal dependent change in capacitance.

## 2.3 ESD Device Compensation

The previous section has motivated the importance of ESD protection, as well as its negative impact on RF designs. This section provides a discussion of on-chip inductor-based capacitance compensation. There are other means for compensation like PCB based inductors [32], however they are not considered here as the SerDes should be a stand-alone Intellectual Property (IP) block. Furthermore, the separation between ESD devices and a PCB inductor is too large in the multi-GHz domain. Another method uses high-speed amplifiers to mirror the input voltage to an intermediate node of the diodes to cancel their capacitance [33] (called *bootstrapping*). As these amplifiers need power, are also making use of an inductor, and present an additional capacitive load, this method is not applied here. Instead, the focus lies on using only passive inductors. For these, a variety of different circuits are proposed in literature for a multitude of applications. They can be classified as shown in Fig. 2.17.

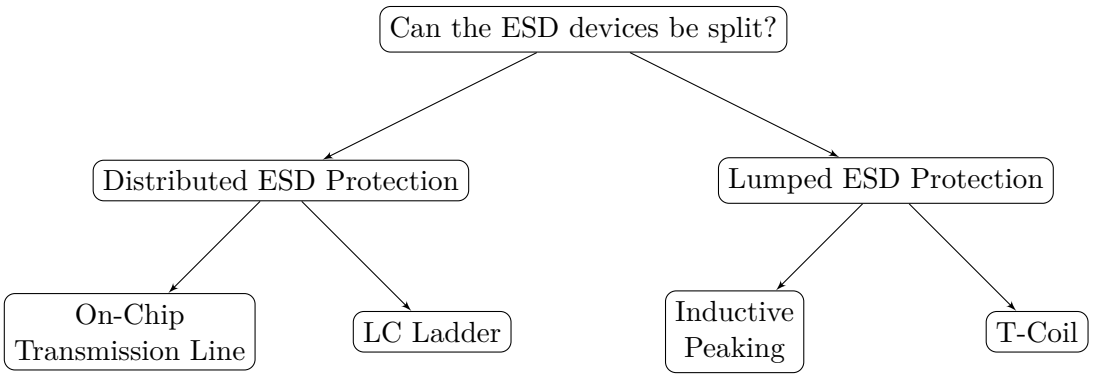



**Fig. 2.17:** Attempt to classify on-chip inductor-based circuits for ESD compensation.

This section will begin with the distributed circuits, which are then followed by the lumped circuits. As the T-coil allows to ideally compensate the diode capacitance, neglecting all kinds of secondary effects, it is the circuit of choice. The section is therefore concluded with an analysis of two additional parasitic effects on the T-coil design – the series resistance of the inductor windings and the capacitance of the pad. Where necessary, this section already makes some arguments that are related to the physical layout of the circuits. For more details on these refer to chapter 3. The previous section has shown that the receiver CDM network includes a set of secondary diodes and an ESD resistor. These are attributed to the input stage of the receiver and, along with its input capacitance, neglected. As such, the calculations done in this section can be transferred unchanged to the transmitter output. A co-design of ESD compensation, and receiver front-end and transmitter driver is beyond the scope of this work but will likely be beneficial in future designs, though also much more complicated.

### 2.3.1 Distributed ESD Protection

The idea behind a distributed compensation scheme is to continue the channel on-chip and incorporate the diode capacitance in this transmission line [34]. Such a scheme makes use of the possibility to split ESD devices in smaller parts, which is not always possible. The termination resistor is then free of the parallel capacitor and can easily terminate the channel. This structure is shown in Fig. 2.18.

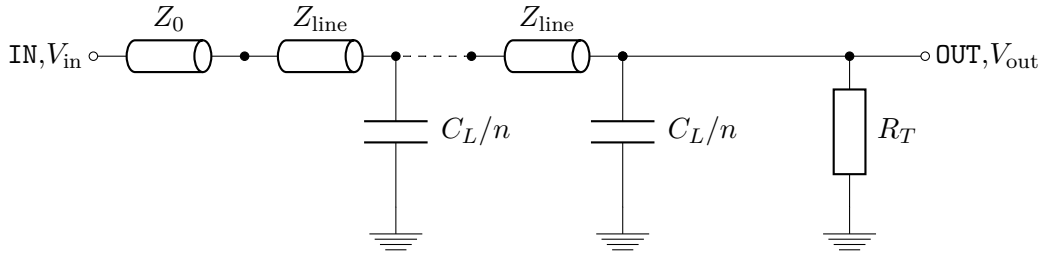



**Fig. 2.18:** Distributed compensation scheme with parasitic ESD device capacitance split into $n$ parts.

The on-chip transmission line impedance, $Z_{\text{line}} = \sqrt{L_{\text{line}}/C_{\text{line}}}$, is chosen to satisfy the condition in Eq. 2.8.

$$Z_0 \stackrel{!}{=} \sqrt{\frac{L_{\text{line}}}{C_{\text{line}} + C_L/n}} \tag{2.8}$$

This distributed structure has several drawbacks. Chaining ESD devices along a transmission line will result in different resistances from the pad to each of them. This may cause imbalanced ESD currents flowing through each device, with the highest current stress on the one closest to the pad. Consequently, this device may break and limit ESD tolerance [8]. Another important aspect is the size necessary to build a transmission line on chip. There are basically three different regimes of electrical design, separated by the relative size of the system to the wavelength of the signals [35]. If the geometric size of the system is much smaller than the signal wavelength ($l \ll \lambda$), problems are solved with circuit theory where voltages are assumed to be constant over the extent of a net. For similar sizes ($l \approx \lambda$), in the regime of microwave engineering, the variation of voltage due to phase differences over the length of a conductor has to be taken into account. And for very high frequencies ($l \gg \lambda$), problems can be treated with the methods of geometrical optics [35]. Considering the range of SerDes bit-rates from 10 Gbps to 32 Gbps, which provided the context for this work, frequencies up to 16 GHz – and at most 48 GHz as the second term in the Fourier series of a periodic rectangular function – can be expected. For typical medium length channels the 48 GHz will never actually cross the channel twice to make reflection at this frequency of any concern. Thus, overestimated by quite a bit, the shortest wavelength to be considered is $\lambda = c/48\,\text{GHz} \approx 6.25\,\text{mm}$. This is more

than ten times the length of the entire SerDes block ($\approx 600\,\mu m$). Hence, it is unrealistic to work with transmission line properties for capacitance compensation. The circuit will actually behave like an LC ladder network.

Another aspect of this is that a simple estimation of the wire length required to provide the necessary $L_{\text{line}}$ also results in lengths above 1 mm. In section 3.2, an approximate formula for the inductance of a straight wire with rectangular cross-section will be derived. Assuming $C_{\text{line}}$ to be zero, $C_L$ to be 300 fF, and width plus thickness of the wire to be $10\,\mu m$, Eq. 2.8 can be transformed into Eq. 2.9 relating the number of segments $n$ with the length of a segment $l$.

$$n \cdot l \cdot \left( \ln\left( \frac{l}{5\,\mu m} \right) + 0.5 \right) = 3750\,\mu m \tag{2.9}$$

Choosing $n = 10$ results in a segment length of $l = 106\,\mu m$. This is of course only approximate but highlights space as the main drawback of this approach. The large space required is also pointed out by Kim et al. [36].

### Two-Stage LC Ladder

Instead of a transmission line, the $Z_{\text{line}}$ can be simply replaced by an inductor to build an actual LC ladder network. A typical inductor for this application takes around $(40\,\mu m)^2$, hence it is not possible to use more than two inductors per pin (see Fig. 2.22). The schematic is shown in Fig. 2.19. While it is technically a lumped circuit, it is still considered a "distributed ESD protection" in this work because the ESD devices are split.

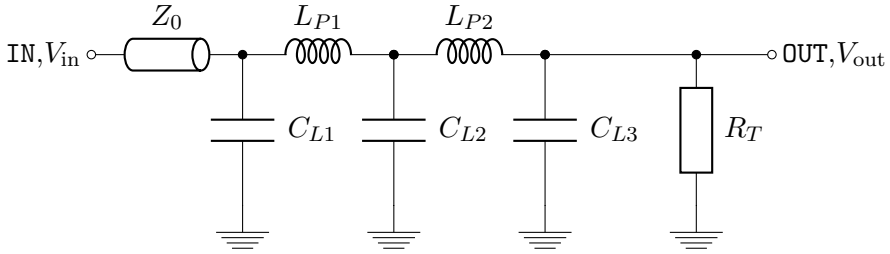



**Fig. 2.19:** Two-stage LC ladder network.

The performance of this circuit, as well as of the following circuits in this section, is optimized by minimizing the reflection around $s = 0$, where $s = j\omega$ is the complex angular frequency. This is done by using the free parameters to cancel the same number of terms in a series expansion of $\Gamma$ at $s = 0$. Transfer functions are analyzed after the optimal parameters are inserted and can therefore only be optimized by choosing the location of the output voltage or with parameters not needed to optimize the reflection. One might argue that a complex metric to trade reflection versus bandwidth and group delay may

be a better way to achieve lower bit error rates. However, this requires knowledge about the exact channel, which is often not possible, e.g. PCIe cards have to operate in a large number of different mainboard designs. Hence, it is hardly possible to assess how much reflection is too much. Nevertheless, the priority of reflection over the transfer function in this context is not pointed out in literature very well to the author's knowledge and is therefore highlighted here.

This two-stage peaking circuit has four free parameters, $L_{P1}$, $L_{P2}$, and two of the capacitors, as their overall sum is fixed to $C_L$. The optimal configuration is found for $C_{L1} = C_{L3}$ and $C_{L2}/C_L = (\sqrt{5} - 1)/2 \approx 0.618$. Note that this is the golden ratio. The inductors are sized to $L_{P1} = L_{P2} = C_L Z_0^2/2$ in this configuration. Fig. 2.20 compares the reflection to the uncompensated case, as well as another (suboptimal) configuration that simply splits the diodes into two halves. The corresponding transfer function magnitudes are shown in Fig. 2.21. The location of the output in this circuit is optimal in terms of flat magnitude and group delay, which are two properties usually desired to reduce signal distortion.

The suboptimal variant using $C_{L1} = 0$, $C_{L2} = C_{L3}$ and $L_{P1} = L_{P2}$ was implemented in this work for a SerDes design in a 28nm technology. A micrograph of this is shown in Fig. 2.22. The diodes consisted of twenty elements and were split into two packs of ten. Despite being far from the optimal solution for this topology, this layout was chosen due to difficulties with tools, the methodology, and a lack of experience in the first months of this work. The inductors were sized based on analytic formulas found in literature. However, this was possibly quite inaccurate (more on this in chapter 3). It was originally planned to already use a T-coil design, but this attempt was discarded due to a missing methodology to simulate and extract the electrical properties of the T-coil layout.

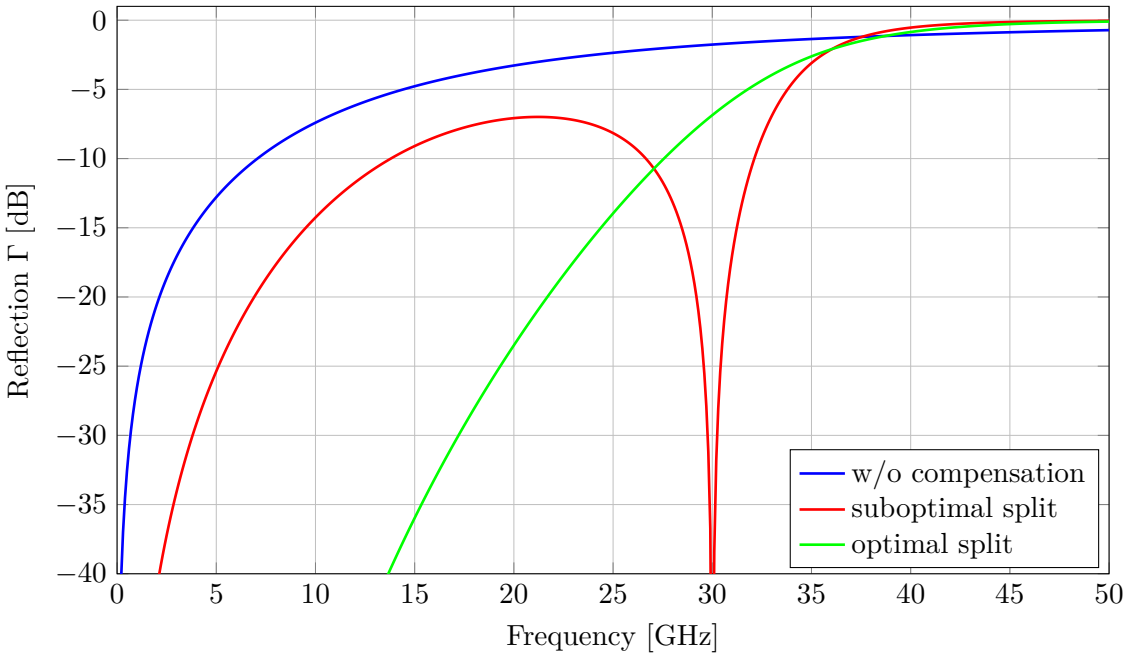



**Fig. 2.20:** Reflection of the two-stage ladder network for $C_L = 300\,\mathrm{fF}$.

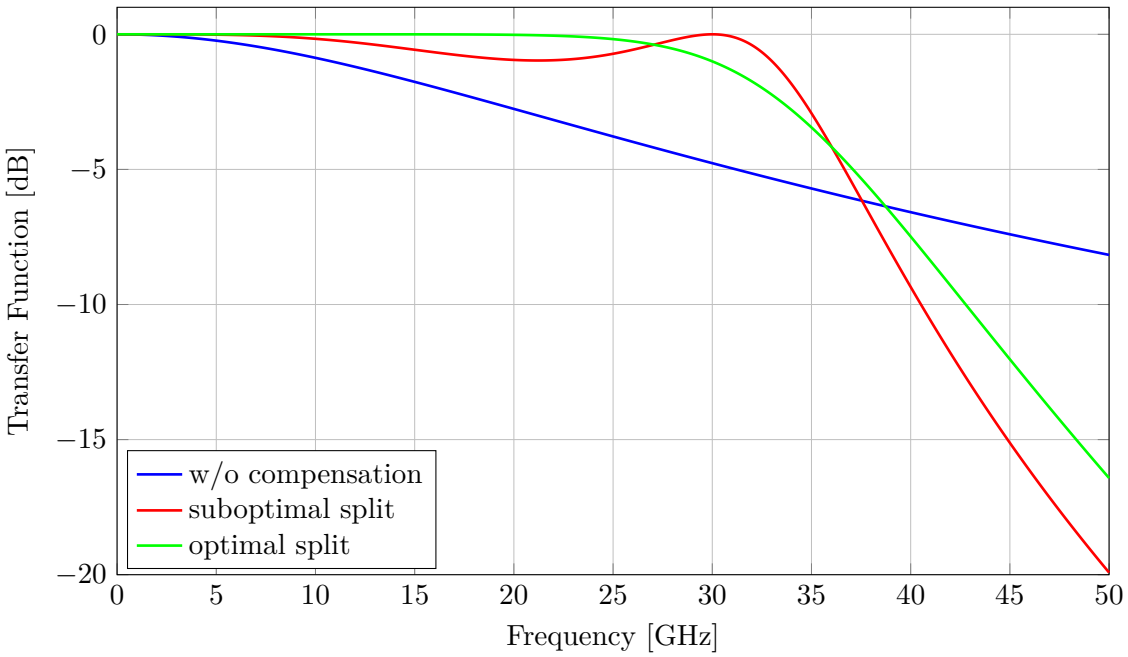



**Fig. 2.21:** Transfer function of the two-stage ladder network for $C_L = 300\,\mathrm{fF}$.

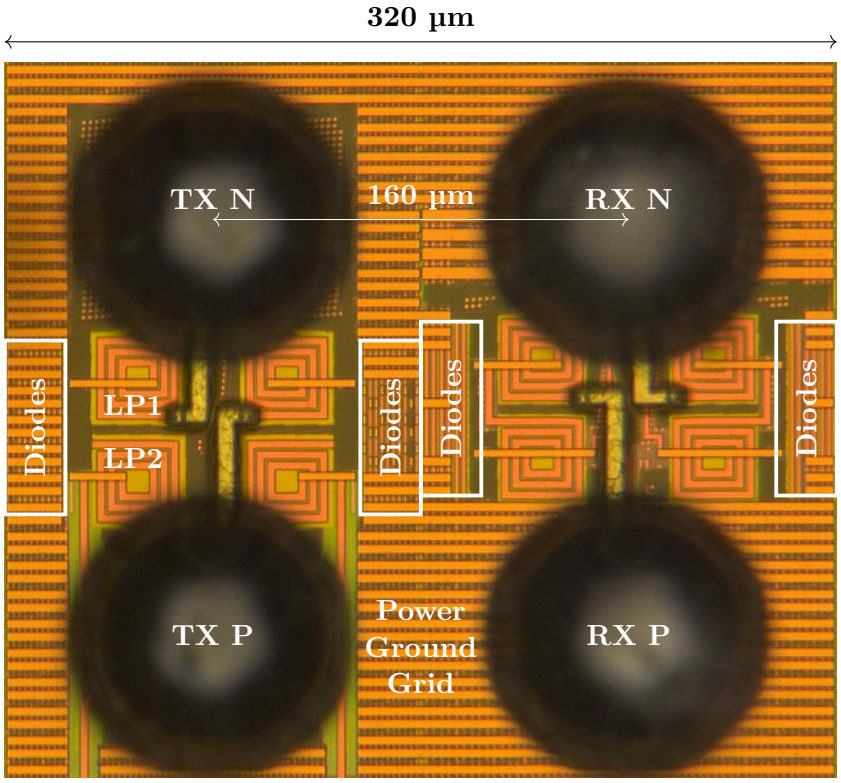



**Fig. 2.22:** A two-stage ladder network applied to a 28nm SerDes design. The black circles are the bumps which are connected to $L_{P1}$ via the thick, grainy, yellow, 'L'-shaped traces.

### 2.3.2 Lumped ESD Protection – The T-Coil

As distributed ESD devices can impede the protection, it is worthwhile to take a closer look on lumped ESD protection schemes. The simplest is inductive peaking using a single inductor in series with $R_T$. Unfortunately, the improvement in reflection is very poor [8]. This can be changed by inserting more peaking inductors into the circuit, e.g. one in every branch as shown in Fig. 2.23. The three inductors are covering basically any relevant version of this circuit in terms of input impedance. Note that a parallel inductor to ground is not possible since this would cause complete reflection at low frequencies. Before considering any transfer functions, the reflection has to be optimized. It is given in Eq. 2.10. From the DC point, $R_T = Z_0$ is obvious and already inserted to shorten the expression.

$$\Gamma = \frac{(L_z + L_r - C_L Z_0^2)s + (L_z - L_r)C_L Z_0 s^2 + (L_z L_r + L_z L_c + L_r L_c)C_L s^3}{2Z_0 + (L_z + L_r + C_L Z_0^2)s + (L_z + L_r + 2L_c)C_L Z_0 s^2 + (L_z L_r + L_z L_c + L_r L_c)C_L s^3} \tag{2.10}$$

Fortunately, the numerator polynomial of the reflection has three coefficients that vanish for $L_z = L_r = C_L Z_0^2/2$ and $L_c = -C_L Z_0^2/4$. The immediate observation is the negative $L_c$, which cannot be realized with a simple inductor. However, negative inductors can be

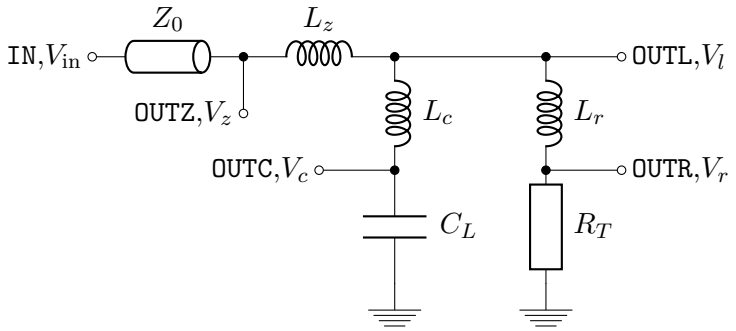



**Fig. 2.23:** Possible inductor positions for an inductive peaking network.

created through coupling of two or more positive inductors. This line of thought may have led to the development of the so-called *T-coil*. A T-coil is a 3-terminal inductor that is created from a 2-terminal inductor by splitting it in two halves with an additional third terminal called *center tap*. In principle, this center tap can be placed anywhere on the wire between the other two terminals. The inductance seen as a 2-terminal inductor is $L_{\text{tot}}$, which is consequently split into two parts by the center tap. Then the self-inductances of both parts are called $L_a$ and $L_b$. However, the splitting demands to take the mutual inductance $L_m = k\sqrt{L_a L_b}$ between both parts into account, where $k \in [0, 1]$ is the coupling factor. The total inductance is now given by $L_{tot} = L_a + L_b + 2L_m$. This coupling also existed in the two-stage ladder network but could be neglected there as lateral coupling is much weaker than vertical coupling. Hence, to achieve a significant coupling, the inductor halves of the T-coil are stacked above each other on different layers. This is also a significant advantage of the T-coil compared to the two-stage ladder as it uses only the space of a single inductor.

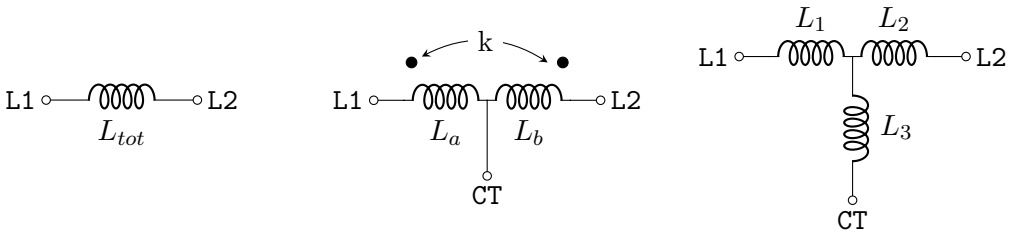



**Fig. 2.24:** Center tap splits coil (left) in two coupled parts (center), which are represented by a three inductor equivalent circuit (right).

A relation between $(L_a, L_b, L_m)$ and $(L_1, L_2, L_3)$ can be derived as the equivalent circuit has to show the same behavior when the center tap is floating as the 2-terminal inductor. Therefore, $L_a + L_b + 2L_m = L_{tot} = L_1 + L_2$ has to be true. Similar conditions are found for a floating L1- or L2-terminal: $L_1 + L_3 = L_a$ and $L_2 + L_3 = L_b$. Solving for the equivalent circuit parameters arrives at the results in Eqs. 2.11 to 2.13. It is worth

to point out that some publications tend to prefer working with the equivalent circuit, i.e. $(L_1, L_2, L_3)$, while others favor the coupled inductors, i.e. $(L_a, L_b, L_m)$. The former is preferred in this thesis as it has the advantage of removing the coupling factor from the expressions derived for $L_1$ and $L_2$.

$$L_1 = L_a + L_m = L_a + k\sqrt{L_a L_b} \tag{2.11}$$

$$L_2 = L_b + L_m = L_b + k\sqrt{L_a L_b} \tag{2.12}$$

$$L_3 = -L_m = -k\sqrt{L_a L_b} \tag{2.13}$$

Fig. 2.25 shows the lumped ESD protection network with a T-coil. Compared to the peaking circuit, an additional capacitor $C_B$ is added. The vertical stacking of the inductor halves causes a significant *bridging capacitance* between them, which is included as a capacitor from `L1` and `L2`. The most important aspect is that it does not prohibit perfect impedance matching but can be used to tune certain transfer functions. Since the internal node of the inductors is not existing anymore, `OUTL` cannot be used any longer to connect input or output of the internal circuitry.

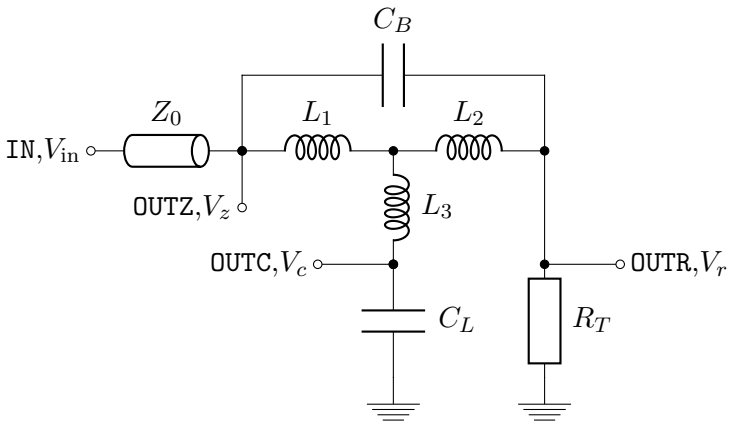



**Fig. 2.25:** Termination network using a T-coil.

The reflection can be easily calculated through a Π-to-T conversion of the Π-network formed by $L_1$, $L_2$, and $C_B$, as shown in [37] and Fig. 2.26. This allows to easily calculate the termination impedance and therefore the reflection. Eq. 2.14 only shows the numerator since the denominator is not needed to set Γ to zero.

$$\Gamma = \frac{(L_1 + L_2 - C_L Z_0^2) \cdot s + C_L Z_0 (L_1 - L_2) \cdot s^2 + C_L (L_1 L_2 + (L_3 - C_B Z_0^2)(L_1 + L_2)) \cdot s^3}{\ldots} \tag{2.14}$$

Every coefficient has to be zero, which yields three conditions. The one for $s^2$ demands $L_1 = L_2$. From this, the coefficient of $s$ sets $L_1 = L_2 = C_L Z_0^2/2$. Putting this into the coefficient for $s^3$ results in $L_3 = (C_B - C_L/4)\, Z_0^2$. Thus, the condition of a matching network sets all inductances of the network to defined values, but $C_B$ remains as a degree of freedom.

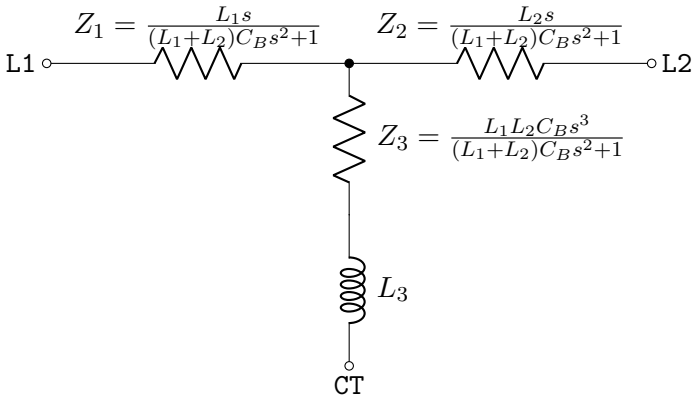



**Fig. 2.26:** T-coil converted to star topology.

### Transfer Functions in the Matched T-Coil Network

There are three possibilities to connect the input/output of the internal circuitry to the T-coil network, `OUTZ`, `OUTR` and `OUTC`. This consideration can be divided in two sub-parts: One for applying the network at the input of a receiver and another one for the output of a transmitter. For calculating the transfer function, it is important to consider the channel as it occurs as a source impedance of the input voltage at the receiver and as a load impedance for transmitter circuits. Since some papers apply the T-coil for bandwidth improvement of amplifiers, their load impedance might be different and therefore different results are obtained for these circuits. At DC, each transfer function has a value of 1/2 because of the voltage divider of $Z_0$ and $R_T$. This offset of $-6\,\mathrm{dB}$ does not alter the shape of the transfer functions, hence they are normalized to $H(0) = 1$. Another assumption made is that internal circuitry does not present any load to the T-Coil network, and thus their properties do not influence the reflection and transfer functions. This is of course a simplification. Calculating the transfer functions under the above conditions for the receiver yields Eqs. 2.15 to 2.17. This is a perfect transfer function at `OUTZ` and a second order all- and low-pass at `OUTR` and `OUTC`, respectively.

$$H_z(s) = 1 \tag{2.15}$$

$$H_r(s) = \frac{C_B C_L R_T^2 s^2 - C_L R_T s/2 + 1}{C_B C_L R_T^2 s^2 + C_L R_T s/2 + 1} \tag{2.16}$$

$$H_c(s) = \frac{1}{C_B C_L R_T^2 s^2 + C_L R_T s/2 + 1} \tag{2.17}$$

Before discussing the above functions, it is insightful to point out why they are identical for transmitter circuits. A CML transmitter acts like an ideal current source under the above simplifications. As such, it does not load the node it is connected to and does not change the termination impedance. The resulting transfer impedance $V_z/I_{in}$

can be transformed into a transfer function by using the transfer conductance $g_m$ of the transistor: $I_{in} = g_m V_{in}$. Normalizing so that $H(0) = 1$ then cancels the $g_m$ and therefore also results in the above transfer functions. In case of an SSTL driver, the only possible configuration is the all-pass as it has to be in series with $R_T$. At this point, it seems obvious where the RX and TX should be connected. However, due to the series resistance of a real inductor, the associated voltage drop can be too much for the internal circuitry when directly attached to the pad. For the other two transfer functions, $C_B$ can be chosen for specific properties. The low-pass has a maximally flat amplitude response with $C_B = C_L/8$ (MFA, Butterworth) and a maximally flat group delay with $C_B = C_L/12$ (MFED, Bessel). The all-pass achieves maximally flat group delay as well, also at $C_B = C_L/12$, which cancels the second order term in the group delay series expansion (Eq. 2.18). A flat group delay is usually preferred for signal transmission.

$$\tau_{\mathrm{g,all}}(\omega) = C_L Z_0 + \left(3 C_B C_L^2 Z_0^3 - \frac{C_L^3 Z_0^3}{4}\right) \omega^2 + \mathcal{O}(\omega^4) \tag{2.18}$$

The discussed transfer functions are plotted in Fig. 2.27 and 2.28 along with the uncompensated diodes. The reflection is not plotted as it is trivial for this circuit.

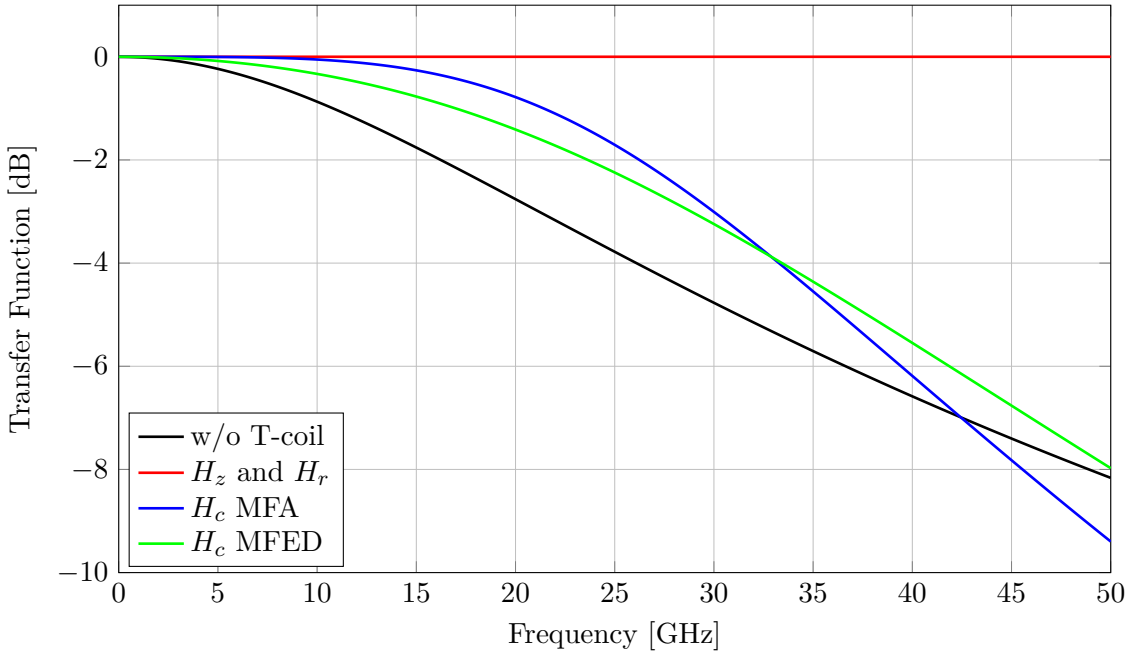



**Fig. 2.27:** Magnitude of T-coil network transfer functions for $C_L = 300\,\mathrm{fF}$.

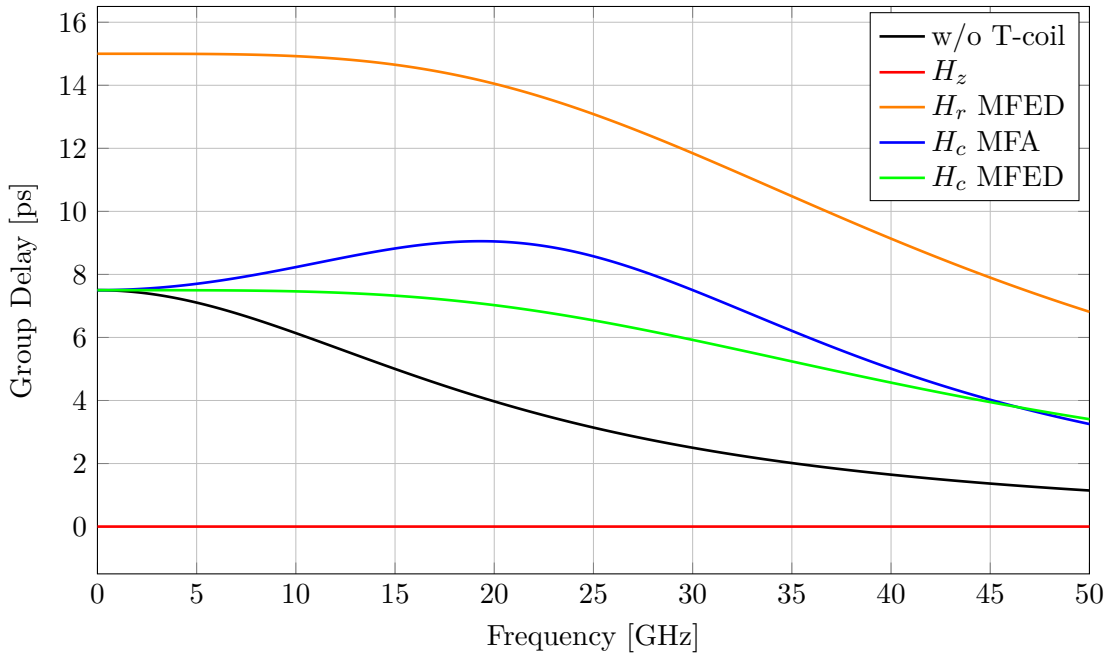



**Fig. 2.28:** Group delay of T-coil network transfer functions for $C_L = 300\,\text{fF}$.

To get a possibly more expressive parameter for the pole tuning than $C_B$, a variety of mappings to other parameters is introduced by various papers. This does not produce much insight into the T-coil circuit but can avoid confusion when comparing different sources in literature.

### Coupling Factor

The inductor coupling factor $k$ is commonly used to replace $C_B$. This is done via Eq. 2.13 and by inserting the inductor values as calculated for perfect matching. The resulting quadratic equation yields only one valid result, for which $L_3$ has to be negative.

$$L_3 = -k\sqrt{(L_1 + L_3)(L_2 + L_3)} \Rightarrow C_B = \frac{C_L}{4}\frac{1-k}{1+k} \tag{2.19}$$

In turn, also $L_3$ can be expressed in terms of $k$.

$$L_3 = -\frac{C_L Z_0^2}{2}\frac{k}{1+k} \tag{2.20}$$

### Damping Factor

The low-pass transfer function $H_c$ is sometimes compared to the canonical form of a second-order function with a cut-off frequency $\omega_0$ and a damping factor $\zeta$. Paramesh et al. [38] use this to map $C_B$ to $\zeta$. It is also common to encounter $L_3$, $k$, and $\omega_0$ in terms

of $\zeta$ as shown in Eqs. 2.21 to 2.24.

$$\frac{1}{C_B C_L Z_0^2 s^2 + C_L Z_0 s/2 + 1} \stackrel{!}{=} \frac{1}{s^2/\omega_0^2 + 2\zeta s/\omega_0 + 1} \Rightarrow C_B = \frac{C_L}{16\zeta^2} \tag{2.21}$$

$$L_3 = -\frac{C_L Z_0^2}{4}\left(1 - \frac{1}{4\zeta^2}\right) \tag{2.22}$$

$$k = \frac{4\zeta^2 - 1}{4\zeta^2 + 1} \tag{2.23}$$

$$\omega_0 = \frac{4\zeta}{C_L Z_0} \tag{2.24}$$

This is done to achieve an expression for the bandwidth. It is found that the $-3\,\mathrm{dB}$ frequency of this circuit is given by Eq. 2.25. Note that [38] is missing the outer square root in the below expression.

$$\omega_{\text{-3dB}} = \frac{4\zeta}{C_L Z_0}\sqrt{1 - 2\zeta^2 + \sqrt{(1-2\zeta^2)^2 + 1}} \tag{2.25}$$

To quantify the improvement achieved with the T-coil, the bandwidth extension factor (BWXF) is defined in [38]. However, the circuit considered in [38] is an amplifier stage with only $C_L$ as load. In case this amplifier would drive a channel, $Z_0$ would have to be considered in parallel to $C_L$, effectively reducing the BWXF by a factor of 2.

$$\mathrm{BWXF}_{\text{amplifier}} = \frac{\omega_{\text{-3dB}}}{1/(C_L Z_0)} \stackrel{\zeta=1/\sqrt{2}}{=} 2\sqrt{2} \tag{2.26}$$

### Pole Angle

The poles of the canonical second order system are located at $s_{1/2} = -\omega_0(\zeta \mp \sqrt{\zeta^2 - 1})$ in the complex plane. Galal et al. [8] use this to express $\zeta$ as a function of the pole angle. Hence, the known pole angles for MFA ($\theta = 135°$) and MFED ($\theta = 150°$) can be inserted directly to obtain the corresponding transfer function.

$$\tan(\theta) = \frac{\mathrm{Im}\left\{s_{1/2}\right\}}{\mathrm{Re}\left\{s_{1/2}\right\}} = \frac{\mp\sqrt{\zeta^2 - 1}/j}{\zeta} \Rightarrow \zeta = \frac{1}{\sqrt{1 + (\tan(\theta))^2}} \tag{2.27}$$

$$C_B = \frac{C_L}{16}\left(1 + (\tan(\theta))^2\right) \quad \stackrel{150°}{=} \frac{C_L}{12} \quad \stackrel{135°}{=} \frac{C_L}{8} \tag{2.28}$$

$$L_3 = -\frac{C_L Z_0^2}{16}\left(3 - (\tan(\theta))^2\right) \quad \stackrel{150°}{=} -\frac{C_L Z_0^2}{6} \quad \stackrel{135°}{=} -\frac{C_L Z_0^2}{8} \tag{2.29}$$

$$k = \frac{3 - (\tan(\theta))^2}{5 + (\tan(\theta))^2} \quad \stackrel{150°}{=} \frac{1}{2} \quad \stackrel{135°}{=} \frac{1}{3} \tag{2.30}$$

### 2.3.3 Parasitic Series Resistance

The previous subsection analyzed the most basic design equations for the ideal symmetric T-coil. A real T-coil will suffer from parasitics, with the most dominant parasitic intrinsic to the T-coil being its finite series resistance. Fig. 2.29 shows the equivalent circuit for the resistive T-coil.

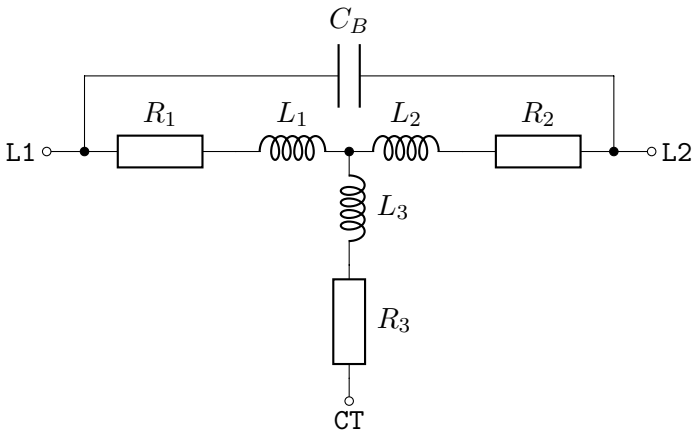



**Fig. 2.29:** T-coil with parasitic resistance and bridging capacitance.

Adding three additional resistors renders exact solutions as shown before very complicated. It is possible to derive some expressions, but they are very involved and do not provide much insight. $R_1$ and $R_2$ are, depending on the metal thickness, below $3\,\Omega$. $R_3$ is much smaller and below $100\,\mathrm{m}\Omega$ as it is only a contact wire to access the mid point of the T-coil. Since the winding resistance changes with the inductance, it is difficult to optimize it concurrently. Therefore, the resistance is usually tolerated. Nevertheless, it is interesting to see how much this degrades the impedance matching for a circuit designed with the ideal T-coil. For this, $R_1 = R_2 = 10R_3 = R_w$ is assumed. The termination resistor is tuned for each curve in Fig. 2.30 to $R_T = Z_0 - R_1 - R_2$. This is also done within the SerDes adaptation via a tuneable $R_T$, which is also used to compensate process variation of the termination resistor. The bridging capacitance is set to $C_B = C_L/12$ as it influences the termination impedance when the T-coil becomes resitive. It can be concluded that the resistance of a typical T-coil, as used in this work, is still low enough to be neglected during inductor sizing.

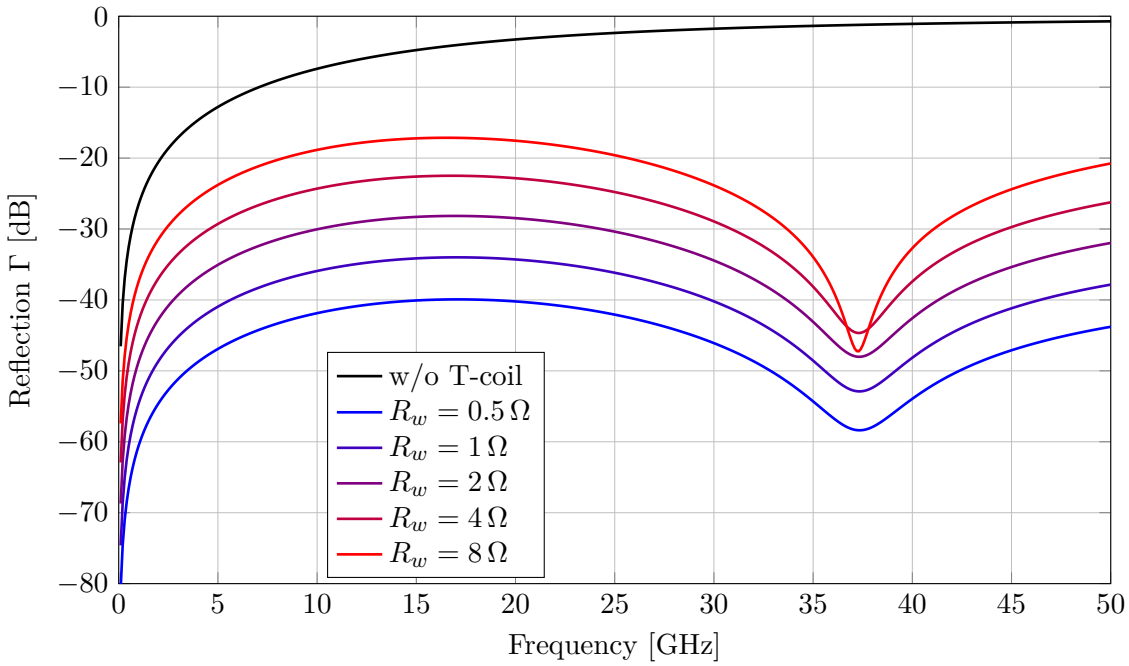



**Fig. 2.30:** Reflection of resistive T-coil for $C_L = 300\,\text{fF}$.

### 2.3.4 The Pad Capacitance

Up to this point only the ESD diode capacitance $C_L$ was considered, however the pad itself is actually a large plate capacitor whose size heavily depends on the structures beneath it. As such, it is highly interesting to analyze the potential influence it might have on the termination quality of the circuit. Fig. 2.31 again shows the T-coil network but with an added $C_P$ to represent the pad. Unlike $C_L$, $C_P$ can neither be split nor moved behind an inductor.

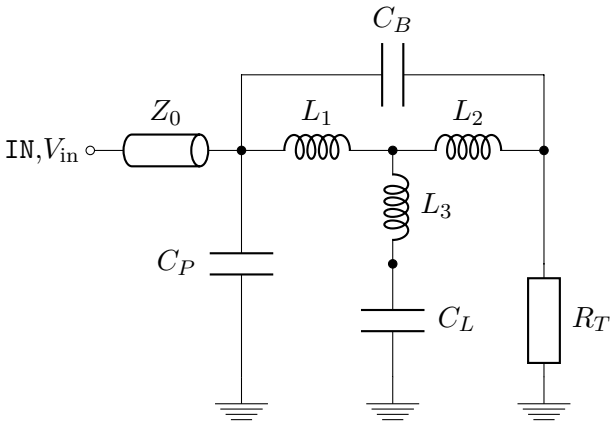



**Fig. 2.31:** Termination circuit using a T-coil, including the pad capacitance $C_P$.

Therefore, the reflection cannot be canceled with a T-coil anymore, but the second to fourth terms in the series expansion can be set to zero by using an asymmetric T-coil with the parameters in Eqs. 2.31 to 2.34. $C_B$ cannot be used to cancel the fifth term and is chosen to minimize the group delay for the all-pass transfer function. The below equations show that $L_1$ and $L_2$ are generally larger but also skewed depending on the ratio of $C_P/C_L$. The asymmetry of the resulting T-coils rises very fast with higher $C_P$. For $C_P = C_L/3$, they are already a factor of two apart. The equations do not result in a valid T-coil for arbitrary $C_P$, which limits the maximum pad capacitance to be tolerated by an asymmetric T-coil. Although the limit is set by $C_B$ to $C_P \approx 0.315C_L$, it can be set to zero for higher pad capacitances. Then the limit is imposed by $k \geq 0$ and increased to $(\sqrt{2}-1)C_L$. Although these limits seem to be reasonable, a T-coil using stacked inductors with zero bridge capacitance cannot be built. The same goes for zero coupling. At these limits, the circuit gradually transforms into the LC ladder network ($C_{L1} = C_P$, $C_{L2} = C_L$, $C_{L3} = 0$), requiring two independent inductors.

$$L_1 = \left(1 + \frac{C_P}{C_L}\right) \cdot \frac{(C_L + C_P)Z_0^2}{2} \qquad L_1 > 0 \Rightarrow C_P \geq 0 \tag{2.31}$$

$$L_2 = \left(1 - \frac{C_P}{C_L}\right) \cdot \frac{(C_L + C_P)Z_0^2}{2} \qquad L_2 > 0 \Rightarrow C_P \leq C_L \tag{2.32}$$

$$L_3 = -\left(1 - 3\frac{C_P^2}{C_L^2}\right) \cdot \frac{C_L Z_0^2}{6} \qquad 0 \leq k \leq 1 \Rightarrow C_P \leq C_L/\sqrt{3} \tag{2.33}$$

$$C_B = \frac{C_L^4 - 6C_L^2C_P^2 - 12C_LC_P^3 - 3C_P^4}{12C_L(C_L + C_P)^2} \qquad C_B > 0 \Rightarrow C_P \lessapprox 0.315C_L \tag{2.34}$$

Since symmetric T-coils are easier to build than asymmetric ones, it is worthwhile to consider the optimal parameters for this configuration. It will perform worse due to a missing degree of freedom but still has the layout advantage of single T-coil instead of two inductors. The third term in the series expansion of the reflection cannot be canceled due to the missing degree of freedom, but the second and fourth terms vanish for the inductances in Eqs. 2.35 to 2.37. $C_B$ is chosen to optimize the group delay of the all-pass transfer function again. This time, $C_P$ is limited to $C_L/3$ by $k \leq 1$.

$$L_1 = L_2 = \frac{(C_L + C_P)Z_0^2}{2} \qquad L_1, L_2 > 0 \Rightarrow C_P \geq 0 \tag{2.35}$$

$$L_3 = -\left(1 + 3\frac{C_P}{C_L}\right) \cdot \frac{C_L Z_0^2}{6} \qquad 0 \leq k \leq 1 \Rightarrow C_P \leq C_L/3 \tag{2.36}$$

$$C_B = \frac{C_L^3 + 6C_L^2C_P + 21C_LC_P^2 + 12C_P^3}{12(C_L + C_P)^2} \qquad C_B > 0 \Rightarrow C_P \geq 0 \tag{2.37}$$

The T-coil has the main drawback that the pad capacitance cannot be incorporated

into the ESD capacitance, which is possible with the ladder network. Furthermore, the asymmetric T-coil transforms into the ladder network when $C_P \geq (\sqrt{2}-1)C_L$. Due to this, the two-stage ladder network is revisited again, and $C_P$ is now considered as a part of $C_{L1}$. It turns out that the size of $C_P$ defines four different regimes where the circuit topology changes as some components become zero.

| $C_P$ range | Limit | Topology |
|---|---|---|
| $(C_P/C_L) \in [0, \sqrt{5}-2)$ | $C_{L1} \geq C_P$ | optimal split |
| $(C_P/C_L) \in [\sqrt{5}-2, \sqrt{2}-1)$ | $C_{L2} \leq C_L$ | suboptimal split ($C_{L1}$ is too large) |
| $(C_P/C_L) \in [\sqrt{2}-1, 1)$ | $L_{P2} \leq 0$ | no split ($C_{L1} = C_P$, $C_{L2} = C_L$, $C_{L3} = 0$) |
| $(C_P/C_L) \in [1, \infty]$ | - | inductive peaking ($L_{P2} = 0$) |

**Tab. 2.2:** The size of the pad capacitance defines four different regimes for the two-stage ladder network.

The larger $C_P$ becomes, the more the circuit transforms into simple inductive peaking. In the first regime, an optimal split is still possible, and $C_{L1}$ contains $C_P$ plus a small portion of $C_L$. At approximately $0.24C_L$, $C_{L1}$ consists only of the pad capacitance and cannot get smaller, thus resulting in an suboptimal split. Raising $C_P$ further moves all parts of $C_L$ from $C_{L3}$ to $C_{L2}$ causing it to become zero at $C_P = (\sqrt{2}-1)C_L$. Note how this coincides with the asymmetric T-coil at this pad capacitance. The next regime causes $L_{P2}$ to decrease, which becomes zero at $C_P = C_L$, as does $L_2$ in the asymmetric T-Coil. For pads with a greater capacitance than the ESD devices, only a single inductor peaking circuit is left.

Fig. 2.32 shows that the ladder network has very poor impedance matching when $C_P$ exceeds the value that still allows for an optimal split. The asymmetric T-coil performs better than the symmetric one, which is expected. The transfer functions are also heavily impacted by the pad capacitance. It should therefore be reduced as much as possible.

To assess the actual value of $C_P$ that is to be expected in a typical design and potential layout optimizations, several variations of the pad layout have been simulated with the EMX field solver. The eight different layouts are shown in Fig. 2.33. Blue is the redistribution (uppermost) layer, while purple is directly below and green below that. The octagonal shape is the pad, and the vertical lines on the redistribution layer are a common way to run power and ground. They are connected to the horizontal stripes in an alternating pattern. From these lines, all circuits in this area are connected to power and ground. The pattern is called *power grid*. The structure was fitted to a $\Delta$-topology of capacitors between pad, power, and ground. The $C_P$ given below the subfigures is calculated as the sum of the capacitance from pad to power and pad to ground. For reference, also the capacitance of the power grid is given.

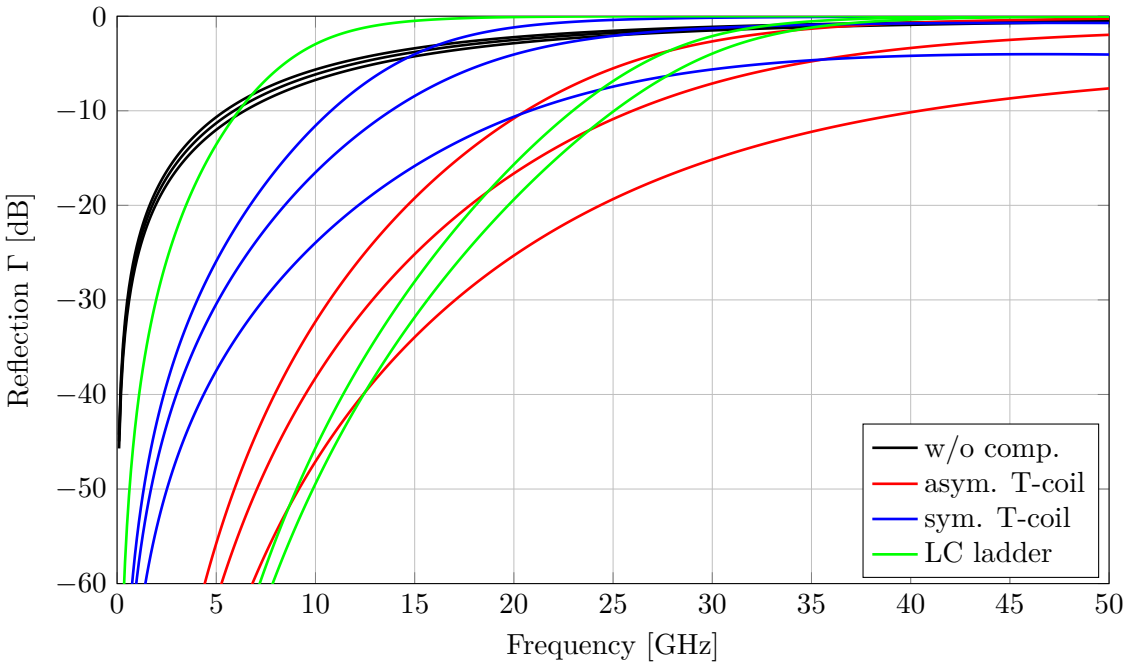
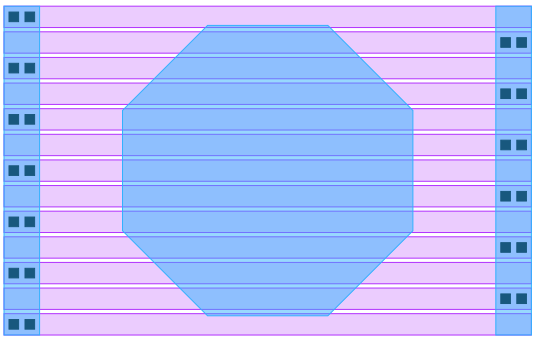
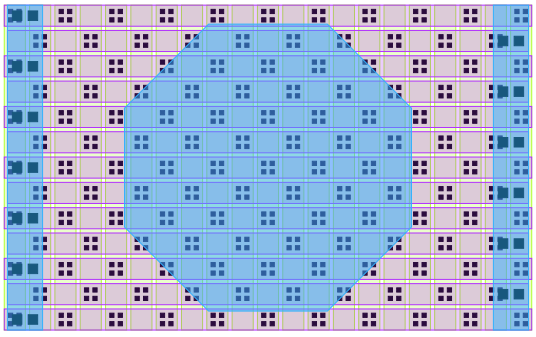



**Fig. 2.32:** Reflection of the asymmetric and symmetric T-coil networks, compared to the two-stage ladder and uncompensated network. Each is plotted for a pad capacitance of 10%, 20%, and 30% of $C_L = 300\,\text{fF}$.

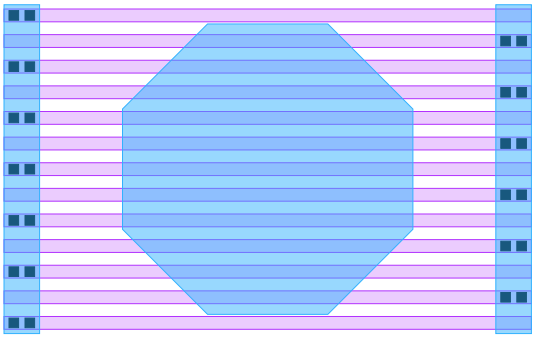
The layouts in the first and third row are using a width-to-spacing ratio of 5:1, and the ones in the second and fourth row a 1:1 ratio. Additionally, different variants are presented in the four quadrants. The upper left quadrant only uses power grid on the layer below the pad, while the upper right uses power grid on both layers. On the lower left, the layer below the pad is left empty, while on the lower right, metal fill is inserted. The simulation shows that the capacitance varies significantly between all designs. So just for matching of the differential signal paths, it is already worth it to carefully layout the structures below the pad. Fortunately, this effort can be limited to the next two layers below the pad as the layouts on the left clearly show an immense decrease in capacitance by skipping only a single layer. Furthermore, the upper half of the layouts shows that the grid below the pad greatly shields the pad from structures below that grid. Finally, the lower half of layouts shows that metal fill has a significant impact on the pad capacitance. Hence, it has to be kept at a minimum. The analyzed circuits have shown that the pad capacitance should be at most around $0.24C_L$, which is $C_P = 72\,\text{fF}$ for the typical diodes used in this work. Therefore, only minimum metal fill can be allowed on the layer directly below the pad and the power grid below should be kept as thin as possible. This shows that there is already extreme effort required just to achieve a relatively reasonable pad compensation.



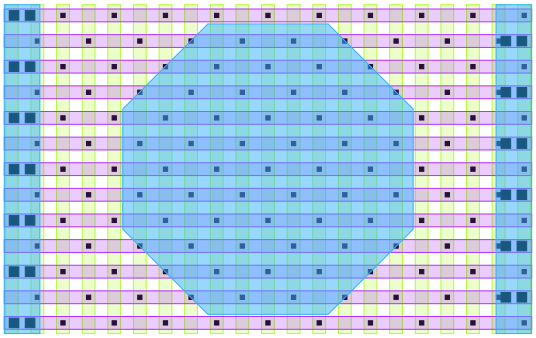
**(a)** $C_P = 351\,\text{fF}$ and $C_{\text{pow2gnd}} = 312\,\text{fF}$



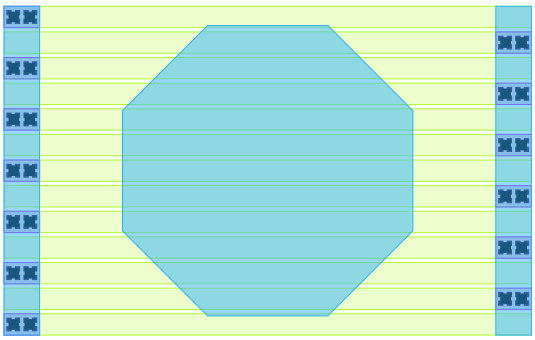
**(b)** $C_P = 355\,\text{fF}$ and $C_{\text{pow2gnd}} = 695\,\text{fF}$



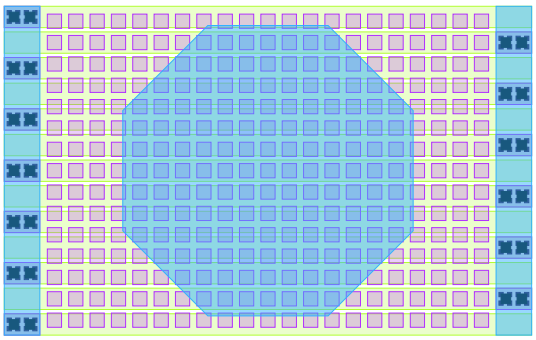
**(c)** $C_P = 256\,\text{fF}$ and $C_{\text{pow2gnd}} = 139\,\text{fF}$



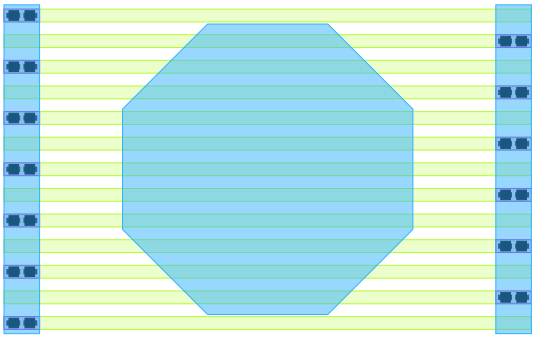
**(d)** $C_P = 257\,\text{fF}$ and $C_{\text{pow2gnd}} = 318\,\text{fF}$



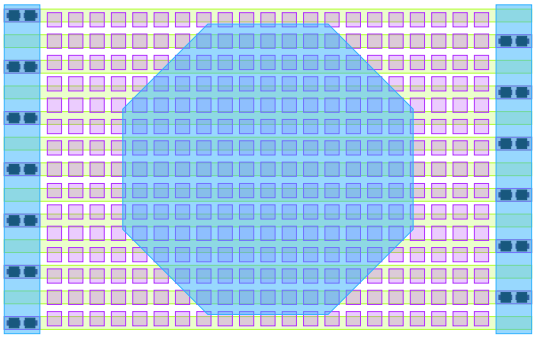
**(e)** $C_P = 54\,\text{fF}$ and $C_{\text{pow2gnd}} = 320\,\text{fF}$



**(f)** $C_P = 93\,\text{fF}$ and $C_{\text{pow2gnd}} = 321\,\text{fF}$



**(g)** $C_P = 52\,\text{fF}$ and $C_{\text{pow2gnd}} = 143\,\text{fF}$



**(h)** $C_P = 87\,\text{fF}$ and $C_{\text{pow2gnd}} = 143\,\text{fF}$

**Fig. 2.33:** Eight variants of possible layouts below the octagonal pad shape.

Chapter 3

# On-Chip Inductors

The previous chapter has elaborated on the importance of inductors in chip design, especially for compensating capacitive input loads. In this chapter, the design process for on-chip inductors is discussed in detail. The first section looks at different methods to generate on-chip inductor layouts as these are very seldom provided by PDKs. Besides the layout, a big question to answer is how its electrical properties correspond to the schematic. The fundamental task is to close the design gap between electrical (schematic) and geometric (layout) parameters. To do this, section 3.2 starts with the basics of inductance calculation by considering the cylindrical wire and shows that even this simple case is more complicated than expected. The mean distance approximation is explained to calculate the self- and mutual inductance for arbitrary wire cross-sections. Then, section 3.3 shows how on-chip inductors are modeled in the schematic, as well as the influence of oxide, substrate, skin effect, metal fill, and process corners. The next section discusses layout extraction and field solvers, and presents a scripted flow that uses the layout generation and various EDA tools to transform geometric parameters into S-parameters. In section 3.5, an MCMC fitting technique is used to map simulated S-parameters to schematic-based models. Using the aforementioned results, the creation of a model for T-coils in this particular process, which can be used for design optimization, is described. The chapter is concluded with an analysis of time-domain measurements from silicon test structures, and a brief overview on the challenges faced to integrate T-coils and ESD devices into SerDes layouts.

## 3.1 Inductor Layout

This section discusses possible inductor layouts and how to generate them automatically. The latter is essential to explore the design space in an efficient manner. The first part shows the various limitations and constraints imposed by advanced nodes and planar technologies in general, and takes a look at feasible inductor geometries and their properties. The following subsections discuss different approaches to *parameterized cells (PCells)*. These cells are the foundation for layout generation and are available in different flavors, namely Synopsys PyCells, Cadence SKILL PCells, and XCells. Cadence Virtuoso, the analog design environment used in this work, is SKILL based, so SKILL PCells are the native form of parameterized cells and are often provided in PDKs for all basic devices. The term *SKILL PCell* is used in this work to differentiate from the general term *PCell*. Since SKILL is a proprietary language, there has been an initiative to develop PyCells, which are based on the Python language and aim to reduce the coding effort. Another approach for a more graphical design of PCells is available from Cadence, the PCell Designer. A tool that allows to graphically design PCells and generate the necessary SKILL code. Due to shortcomings of all these approaches, the XCells were developed at the CAG at Heidelberg University using a Python to SKILL translation layer and a constraint-based approach [39].

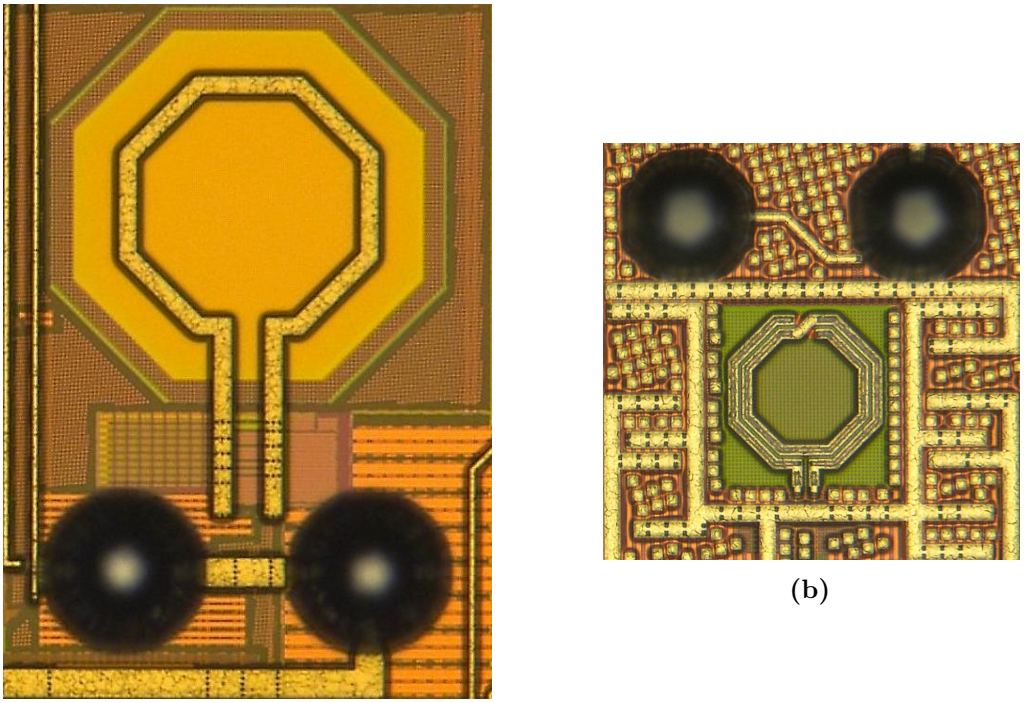
### 3.1.1 Inductor Types

On-chip inductors are essentially metal traces arranged in a way that their geometry causes a significant (parasitic) inductance. By being a parasitic property of a metal trace, the inductor is by default not present in the schematic design phase but has to be included for accurate simulation results. This raises a methodology problem, which is addressed later in this chapter.

Section 2.3 has identified T-coils as a method to compensate the capacitive load caused by ESD diodes. Therefore, this section is limited to discussion of T-coil layouts. The location of the third pin, the center tap, adds a degree of freedom, but the layout of simple two-terminal inductors is easily derived by just omitting this third pin. As a consequence, the layouts and properties of two-terminal inductors are basically identical and most results can be transferred to other on-chip inductors.

Before spending any effort on creating parameterized cells for inductors, it is worth mentioning that the 22nm PDK considered in this work did provide an inductor capable of forming a T-Coil. However, this inductor is intended to be used for oscillator circuits in PLLs. Thus, it is required to provide a high quality factor to reduce the damping of the oscillation. To achieve this, a low resistance is desirable, which comes with thick metal traces provided by the redistribution layer, i.e. the thick top metal layer used for

pads. Furthermore, eddy currents are reduced by special fill patterns to avoid large and especially contiguous metal shapes. To protect the inductor from its surroundings, special guard structures are applied. Photographs of such inductors can be seen in Fig. 3.1. It can be observed that octagonal layouts are chosen for this use case, as a more circular design reduces the length of the traces and therefore decreases the resistance, while the inductance stays roughly the same. More than eight edges are prohibited by most modern technologies as angles other than 45° and 90° are not allowed by the design rules. In fact, 45° angles are almost exclusively limited to the redistribution layer. An exception are again the foundry-sponsored PCells, which can use more relaxed design rules as their environment is carefully controlled and the designs are verified by the foundry.



**(a)** **(b)**

**Fig. 3.1:** Micrographs of inductors used for LC-oscillators in a 28nm (a) and a 22nm (b) PLL design. The inductor on the left has an outer diameter of 169.2 µm, while the one on the right has 100 µm. The displayed size ratio of both inductors is identical to the actual size ratio. The PLLs were designed at the CAG and Extoll GmbH though not in the context of this work.

The special design rules and the verified performance do make the PDK inductor an attractive solution for oscillators, but the foundry PCell does not allow inner diameters below 70 µm. This prohibits placing the inductor in between pads because they have a center to center spacing of 150 µm and a diameter of 81.6 µm. So there is only a spacing of 68.4 µm, which renders this foundry PCell non-feasible for the T-coil design targeted

in this work. The pads, or rather the solder balls above the pads, are the black circles in Fig. 3.1. The SerDes makes use of differential signaling so two T-coils need to be placed between two vertically arranged pads. Placing them below the pads is introducing a huge coupling capacitance and significantly degrades the electrical properties of the inductors, as it presents a large contiguous metal plate that allows eddy currents to flow. This arrangement was analyzed in the early stages of this work with the goal to save area but is of course very naive in retrospect. Eddy currents are mitigated by the PDK inductors by using special minimum-sized fill patterns. Due to these constraints, it becomes necessary to design custom PCells.

The design of inductors in planar technologies is practically compelled to use the lateral dimensions as vertical inductors suffer from different layer thicknesses and the use of vias. With the requirement to use only 90° bends, the square spiral is the most trivial inductor. By placing a third pin, the center tap (`CT`), anywhere on the spiral, a T-coil is created. It is shown in Fig. 3.2 and called *tapped T-coil* in this thesis, analogous to [40]. Note that the red and blue parts indicate the two halves of the T-coil, not different layers. This T-coil has the advantage to require only a single layer so the design does not depend much on the metal stack. However, it is rather difficult to use this layout to build a symmetric T-coil.

- non-symmetric
- arbitrary tapping ratios
- single layer
- high self-inductance
- low terminal-to-terminal capacitance
- medium mutual coupling

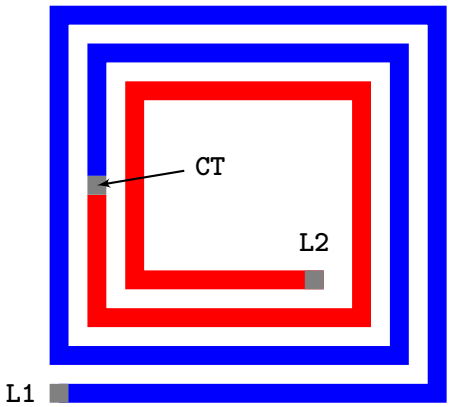
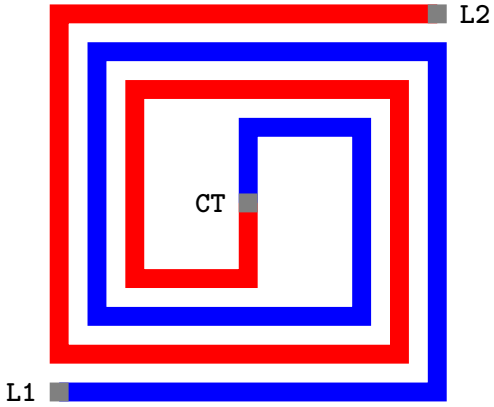



**Fig. 3.2:** Tapped T-coil.

Mohan [40] also shows two more variants of this in a discussion of transformers. A transformer is identical to two inductors in close proximity and is converted into a T-coil by shortening one terminal of each, the first and second inductor. The variants shown by Mohan are the *interleaved* and *stacked* transformer, and their T-coil versions are shown in Fig. 3.3 and 3.4. In contrast to the tapped T-coil, the interleaved T-coil is indeed symmetric, but the opposing current directions in neighboring segments lower the inductance significantly. As a result, it provides too low inductance per area to fit

between pads.




- symmetric
- single layer
- low self-inductance
- low terminal-to-terminal capacitance
- low mutual coupling

**Fig. 3.3:** Interleaved T-coil.

- symmetry depends on layers
- double layer
- high self-inductance
- high mutual coupling
- medium terminal-to-terminal capacitance

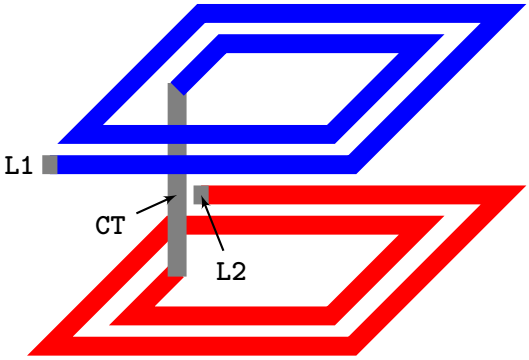



**Fig. 3.4:** Stacked T-coil.

Of these variants, the stacked T-coil fits best for the targeted use case. The need for two relatively thick and similar layers is actually not a hindrance in the considered technologies. Both provide a pair of thicker metal layers below the redistribution layer. And using two layers with current flowing in the same direction results in a high self-inductance per area. The vertical separation also leads to increased mutual-inductance compared to the lateral coupling of the tapped T-coil. Last but not least, it is symmetric due to identical layer types, which is perfect for impedance matching.

All T-coil variants share the problem that an additional metal trace is needed to contact the `CT` pin and route it to the cell boundary. The layer of this trace has to be different from the layers used for the windings. This adds non-ideal properties compared to the equivalent circuit model but cannot be avoided. By making this trace part of the T-coil cell, it is included in the EM-simulation and does not cause layout-dependent changes when the T-coil is inserted in its environment. Lateral coupling to structures outside the cell boundary is smaller than coupling to structures above and below the

cell. So to avoid that other structures are placed below or above the inductor on unused layers, a special foundry-defined fill pattern can and should be included as simulation results show in section 3.3. This fill pattern consists of very small square shapes to assure the inductor is manufactured correctly, but it also reduces eddy currents compared to regular fill. The implications of these fill structures on layout extraction are discussed later in section 3.4.

### 3.1.2 Layout Generation

The different kinds of possible layouts for T-coils can all be easily drawn in the layout entry tool. However, coming from the schematic design phase, there is little indication on what to draw. Thus, many iterations will likely be needed to get the "correct" layout with the desired electrical properties. Each iteration entails altering the layout by hand. Consequently, a lot of repetitive work is saved if the redraw process is automated. Furthermore, as soon as fill or guard structures are introduced, automation is inevitable. Several methods were evaluated and applied over the course of this work. They will be presented in the following subsections in chronological order. This subsection aims to give a brief background on why all these methods were evaluated.

Starting with the first attempt to design a T-coil for ESD capacitance compensation, it was obvious that a method to generate different T-coil layouts will be needed. Without any experience, it was guaranteed that more than one inductor had to be designed before the "right" one was found. The method of choice was to write a SKILL function, accompanied by a set of helper functions, which took the geometric parameters and drew a T-coil layout in the currently active layout entry window. Beforehand, the SKILL functions had to be loaded into Virtuoso. As mentioned before, this first attempt aimed to place the inductor below the pad. However, EM-simulation revealed relatively late that the designed T-Coil was too large and that placing it below the pad was not a good idea. Furthermore, no method to convert the S-parameters produced by the simulator to schematic parameters existed at this point. So due to time constraints, a simpler square spiral two-terminal inductor, like the tapped T-Coil, was finally implemented as SKILL PCell and used in the submitted design. The layout for these inductors was determined via formulas found in literature [41]. This was the 28nm design which is shown in Fig. 2.22. Coming from this, the goal was still to deploy a T-coil in the next design (22nm), albeit smaller and between the pads. A T-coil PCell had to be created but not necessarily with SKILL. The intention was to find an easier and more portable method to design PCells, now that a technology change was ahead. Potentially, even other parts of the SerDes layout could be automated in this way. This led to the idea to try the Python-based PyCells, which are described in the next subsection.

### 3.1.3 PyCells

PyCells were developed by the company Ciranova, which was aquired by Synopsys in 2012. The concept behind the PyCells is to enable the description of layout in Python, while providing existing EDA tool support by writing PCell information to OpenAccess files. OpenAccess is an EDA database format designed for interoperability between tools of different vendors. It is "open" in the sense that the code can be seen and modified when paying the membership fee [42]. The Silicon Integration Initiative (Si2) manages these memberships. It was created to enable design tool interoperability and within it the OpenAccess Coalition formed, a coalition of many major semiconductor companies. OpenAccess provides an API to the database format, and in contrast to other EDA standardization attempts, there is also a C++ reference implementation available, which was donated by Cadence Design Systems in 2002 [43]. The format can store, among many other things, schematic and layout information like ports, shapes, nets, and parasitics. Files use the .oa extension. So to make PyCells compatible with other EDA tools, Ciranova developed a Python wrapper for the reference implementation of the OpenAccess API [44]. This allows to access and modify design files in Python code. An IDE, called PyCellStudio, with debugging capabilities is also provided. The designer can now benefit from object-oriented programming to lower the amount of repetitive code and of course other features Python provides. The PyCells are expected to greatly reduce the coding required for a specific task compared to the Cadence-specific SKILL PCells. Furthermore, with the use of layer-mapping files and technology-independent functions of the PyCell API, a high degree of reusability of existing code in case of a technology change can be achieved.

#### PyCell Code

The research conducted on PyCells was one of the first things done for this thesis. In the meantime, PyCells are still being developed by Synopsys, and also the 22nm technology considered here was recently announced to receive PyCell support, which was not available a few years ago. So some compatibility issues might have vanished by the time this thesis is completed, and also some PyCell API details may have changed. Nonetheless, this subsection takes a look at the PyCell code structure with the example of a simple square spiral inductor, which was created in the framework of this work and is shown in Lst. 3.1. The following information on the PyCell API is taken from the "Python API Reference Manual" [45]. The implemented PyCell named `InductorSpiralSquare` inherits from a class called `DloGen`. "Dlo" on its own stands for "Dynamic Layout Object" and wraps the oaDesign class of the OpenAccess API. Consequently, `DloGen` is the class to inherit from when writing a layout generator. This is an abstract class containing virtual methods, from which the presented code implements three. The first is `defineParamSpecs` and

it allows to specify the parameters, their default values, and a description. It is also possible to include different types of constraints. Via the `RangeConstraint`, the outer diameter is limited to a reasonable range. Turn width and spacing are not constrained here. The number of turns is intended to allow half and quarter turns, which is checked by a `StepConstraint`. If a diameter or number of turns that violates these constraints is entered by the PyCell user, one of three actions can be executed: `REJECT`, `ACCEPT`, and `USE_DEFAULT`. This is specified via the `action` argument and is by default set to `REJECT` the value. Another useful constraint is the `ChoiceConstraint`. It allows to provide a list of choices for the parameter, which are two layer names as strings in this case. The second method, `setupParams`, transfers the PyCell parameters to internal variables. Additional computations with the parameters can be done here. In this example, the parameters are simply assigned and the layer string is converted to a `Layer` object. The third function is `genLayout` and generates the layout, as the name suggests. Here, a square spiral and two pins at the start and end points should be drawn. In line 26, a new (layer,purpose)-pair is created for the pins as the default purpose stored in `self.layer` is “drawing”. The next two lines place the text “`L1`” and draw the pin shape of the first pin. Then, the loop draws the spiral by iterating over the number of sides, i.e. quarter turns `n`. When the last rectangle is placed, the second pin is created at the corresponding position.

```
from cni.dlo import *
from cni.constants import *

class InductorSpiralSquare(DloGen):

  @classmethod
  def defineParamSpecs(cls, specs): # define parameters and default values
    specs('diameter', 35.0, 'outer diameter', RangeConstraint(low=15.0, high=150.0))
    specs('width', 3.4, 'turn width')
    specs('spacing', 1.15, 'turn spacing')
    specs('turns', 2.25, 'number of turns', StepConstraint(step=0.25, action=REJECT))
    specs('layer', 'metal1', 'inductor layer', ChoiceConstraint(['metal1','metal2']))

  def setupParams(self, params): # process parameter values entered by user
    self.diameter = params['diameter']
    self.width = params['width']
    self.spacing = params['spacing']
    self.turns = params['turns']
    self.layer = Layer(params['layer'])

  def genLayout(self): # generate layout
    d = self.diameter
    w = self.width
    s = self.spacing
    n = int(4.0*self.turns)
    layer_pin = Layer((self.layer).getLayerName(), 'pin')
    Text(layer_pin, "L1", Point(w/2.0, w/2.0), 1.0, Location.UPPER_RIGHT)
    Rect(layer_pin, Box(0,0,w,w))
    for i in range(0, n):
      x0, x, x3 = max((i//4-1)*(w+s),0), (i//4)*(w+s), (i//4+1)*(w+s)
      if i%4 == 0: # bottom
        Rect(self.layer, Box(x0, x, d-x, w+x))
        if i==n-1:
          Text(layer_pin, "L2", Point(d-x-w/2.0, w+x-w/2.0), 1.0, Location.UPPER_RIGHT)
          Rect(layer_pin, Box(d-x-w, x, d-x, w+x))
      if i%4 == 1: # right
        Rect(self.layer, Box(d-w-x, x, d-x, d-x))
        if i==n-1:
          Text(layer_pin, "L2", Point(d-x-w/2.0, d-x-w/2.0), 1.0, Location.UPPER_RIGHT)
          Rect(layer_pin, Box(d-x-w, d-x-w, d-x, d-x))
      if i%4 == 2: # top
        Rect(self.layer, Box(x, d-w-x, d-x, d-x))
        if i==n-1:
          Text(layer_pin, "L2", Point(x+w/2.0, d-x-w/2.0), 1.0, Location.UPPER_RIGHT)
          Rect(layer_pin, Box(x, d-w-x, x+w, d-x))
      if i%4 == 3: # left
        Rect(self.layer, Box(x, x3, w+x, d-x))
        if i==n-1:
          Text(layer_pin, "L2", Point(x+w/2.0, x3+w/2.0), 1.0, Location.UPPER_RIGHT)
          Rect(layer_pin, Box(x, x3, x+w, x3+w))
```

**Listing 3.1:** PyCell code for a square spiral inductor.

### PyCell Studio

Synopsys provides a tool environment for the PyCells, which includes an integrated development environment (IDE) called PyCell Studio. It is started via the command `cndbg`, and provides a code editor and a layout pane, which can display the PyCell to provide visual feedback. Running the code via the "play" button, executes a tool called `cngenlib`. It generates an OpenAccess database library, which can then be read by Virtuoso to utilize the PyCells. To get the PyCell generated, it has to be registered for generation in the `definePcells()` function as shown in Lst. 3.2.

```
1 from InductorSpiralSquare import *
2
3 def definePcells(lib):
4     lib.definePcell(InductorSpiralSquare, "InductorSpiralSquare")
```

**Listing 3.2:** Registering InductorSquareSpiral as a PCell to generate it via `cngenlib`.

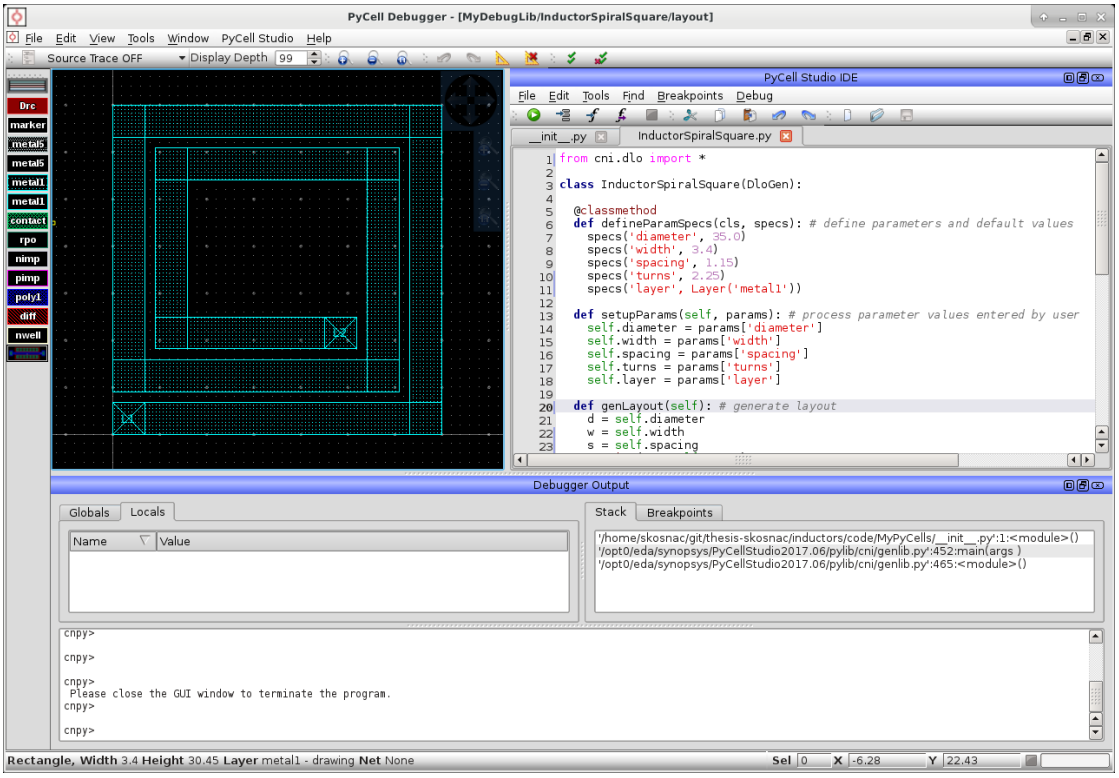



**Fig. 3.5:** The IDE PyCell Studio 2017.06 can be used to develop and debug PyCell code. The `InductorSpiralSquare` from Lst. 3.1 is shown here.

### Santana Technology File

Besides the PyCell code, it is necessary to provide a technology file to the `cngenlib` command to generate the OpenAccess database library for Virtuoso. This technology file is also needed to display the layers within PyCell Studio and has to be in the *Santana* format, which a reference manual exists for [46]. However, the 22nm PDK only provides the technology file in the Cadence DF II format, which is anticipated in the Santana reference manual, and a conversion tool, called `cntechconv`, is provided. This was tested for the 22nm technology file but yielded errors like "*Illegal quoted expression*". It was found that this error originates from the multi-line `'ref`-expressions in the DF II file. According to the Cadence help system they are used to add a reference ID to constraints, which can then be displayed in the "Annotation Browser" or Design Rule Check (DRC). The conversion error can be solved by commenting out all `'ref` expressions and then the corresponding rules are transferred to the Santana file. A shell script has been created which conducts all these steps and prepares the technology file for the next steps:
`./tech_conv_22nm.sh` *<techfile.tf>*
`cntechconv` *<techfile.tf_modified.tf>* `-o` *<techfile.tf_santana.tech>*

A few issues encountered on the way are listed here:

- There is still a conversion error regarding the substrate layer for the 22nm technology file. The corresponding section has been commented out. This may result in a few missing rules in the converted file.
- The `cngenlib` tool is missing two `minWidth` rules for the layers `CA` and `CB`. The rules were added with the values found in the original technology file. It is not clear why they are not converted.
- The parent directory to the directory with the PyCell code has to be added to the `$PYTHONPATH` environment variable. If this is not done correctly, Virtuoso cannot find the PyCell.
- The `$OA_PLUGIN_PATH` environment variable contains the paths to Virtuoso plugins (.plg). It should be set correctly by sourcing the PyCell Studio script.
- The gcc version of the PyCell Studio and the system running Virtuoso should match to properly use the PyCells.

These issues with the technology file conversion and the lack of support in the 22nm PDK led to the decision to not further use PyCells. Converting technology files can cause all kind of different problems in the long run. Furthermore, building PyCells with transistors requires to code a lot of special cases due to complex design rules in advanced nodes. A lot of error prone work can be saved, when PDK PCells are simply used in custom PCells. It is not feasible to draw transistors, or even code parameterized ones, by hand.

### 3.1.4 SKILL PCells

The previous section has shown that PyCells are a viable method to create PCells, but missing support makes them complicated to apply. Therefore, the focus of this work switched back to write PCells in SKILL to be compatible with Virtuoso and the target technologies. In contrast to the PyCell's `cngenlib` command, SKILL PCell code can be loaded into Virtuoso via its command window using a simple `(load <pcell.il>)` command. The PCell can then be found in the library manager and used in the layout editor like foundry-provided PCells. Except in the case of changes, the code is no longer needed as it is converted into the OpenAccess format and stored in the corresponding Virtuoso library. The basic SKILL function to define a PCell is shown in Lst. 3.3. `pcDefinePCell` is provided with a triple of library, cell, and view name, to determine where to place it in the Virtuoso library manager. Then a sequence of parameters along with their default values can be specified.

```
1 (pcDefinePCell (list (ddGetObj "<library_name>") "<cell_name>" "<view_name>")
2         (
3                 (<param_1_name> <param_1_default_value>)
4                 ...
5                 (<param_n_name> <param_n_default_value>)
6         )
7
8         <pcell_code>
9 )
```

**Listing 3.3:** Function to define a SKILL PCell.

To facilitate the coding process of the different inductor types, several SKILL functions were developed in this work to encapsulate common structures, such as drawing a spiral. Another motivation to encapsulate functionality within functions is to enable support for different technologies. The stack-up change is only the obvious difference; there are also, among others, the purpose of the label layer or the (layer,purpose)-pair for LVS-resistors. The latter is handled as metal layer plus a dedicated purpose in one technology and a single new resistor layer but with different purposes depending on the metal layer in the next technology. Further process dependent functions are needed, e.g. for fill structures and via sizes. Last but not least, there is functionality to support the needs of different tools. Some tools cannot read every (layer,purpose)-pair and assign it the correct properties for simulation. For example, the purpose for fill and differently colored (multi-patterning) metal shapes has to be mapped to the "drawing" purpose to get the shapes recognized as metal and simulated properly. Furthermore, thousands of small fill shapes cannot be simulated with (every) field solver in reasonable time. So as an approximation, the metal shapes on at least the lower layers should provide a way to be scaled up for simulation. Fig. 3.6 shows an example of the layouts generated by the stacked T-coil SKILL PCell but neglects fill shapes etc. The basic geometry of such a

T-Coil is described by four parameters: outer diameter $d$, turn width $w$, turn spacing $s$, and number of turns $n$. Note that $s$ is not the complex angular frequency in this chapter.

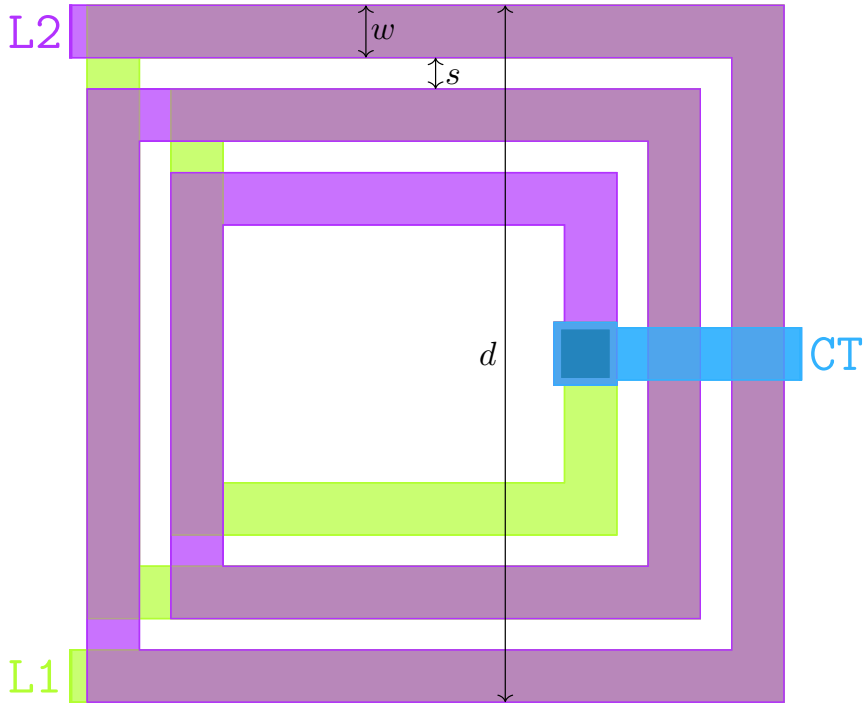



**Fig. 3.6:** Stacked T-coil layout generated with the SKILL PCell code ($n = 2.5$ turns; fill shapes are not shown). The layer order from top to bottom is blue, purple, green, and vias are black.

### PCell Designer

The Cadence PCell Designer is only mentioned here for completeness. It provides a graphical programming interface to create PCells. The GUI can deploy PCells and provides a lot of geometric functions to facilitate the design. For example, "turtle programming" is supported, which follows the idea of a turtle moving with a pen over paper. It can move with either the pen up or down on the paper, i.e. just move, or move and simultaneously draw lines. There are also logic functions operating on 2D shapes, basic control structures like loops, and the possibility to incorporate existing PCells. The PCells are then "programmed" by building hierarchical structures with commands, similar to the tree view known from file browsers on PCs. For more in-depth information and discussion about the PCell Designer, the reader is referred to T. Markus [39]. He implemented a couple of different PCells, which were actually used in a SerDes design, and showed that the use of PCells for common structures found in full-custom design improves quality and speed of the layout procedure. However, the PCell Designer approach was ultimately discarded by Markus in favor of XCells before noteworthy attempts were

made in this work to port the inductor SKILL PCell code. Thus, a PCell Designer implementation of inductors was neglected.

### 3.1.5 XCells

The shortcomings of the SKILL PCell approach led to the investigation of another method for layout generation via parameterizable cells. This new method is based on a socket communication provided by Virtuoso. Via this socket, it is possible to send SKILL commands to, and retrieve answers from, Virtuoso. The key point here is that most functionality provided by Virtuoso is actually accessible via SKILL. Building upon this, an open-source Python tool called *Skillbridge*, available on GitHub [47], was developed at the CAG by T. Markus and N. Buwen. It wraps the socket communication into Python and makes it possible to interact with almost any SKILL functionality Virtuoso provides. The obvious consequence from having all these possibilities at hand in Python, a language far more popular compared to SKILL, is to write code more conveniently to increase the usability of this interface. This work is not open-sourced and part of a collaboration between the CAG and the company EXTOLL GmbH. An in-depth view on this can be found in [39]. Part of this code can be used to develop so-called *XCells*, a novel approach to "parameterized cell"-based layout generation. Lst. 3.3 shows the SKILL code construct used to declare a SKILL PCell. XCells, however, make no use of this and do not register in the Virtuoso Library Manager at all. Instead, they appear in the "Constraint"-mode of the schematic editor as constraints that can be associated with schematic instances. For example, a resistor pair can be constrained to comprise a passive load used in an amplifier circuit, but the sizing of the resistors is done as usual by entering the values into the foundry PCell properties. Moving to the layout stage, the schematic instances are placed somewhere in the layout plane with the Virtuoso "Generate" menu. Switching again to the "Constraint" mode, one finds the previously declared constraints still associated with the placed instances. Via a custom menu created along with the XCell code, layout generation can be triggered for all, or only selected, constraints. This process is parameterized via the constraint parameters, which are natively supported by Virtuoso. Custom parameters required for the specific layout generation task can be added to the constraint in the corresponding XCell Python code. XCells for inductors have not been completed within the framework of this thesis. The corresponding constraint is to be associated with an LVS-resistor.

## 3.2 Inductance

Up to now, this work has presented termination circuit topologies and how to generate inductor layouts. However, the connection between both still remains unclear. Therefore, this section will address the theoretical foundations of what inductance is and how to calculate it for certain geometries. As shown in the previous sections, the motivation to look at on-chip inductors emerged during the SerDes design. Naturally, the research initially focused on literature that addressed ESD device compensation. However, most papers do not even dedicate a short section to elaborating on how to solve the problem without simulation or measuring physical test-structures. Therefore, this work also followed a simulation-based approach for a long time, which is not wrong but lacks a lot of insight, which one tends to compensate with trial and error. This consumes a lot of time, which could be spent on approaching the problem in an analytic manner. A paper by Sunderarajan S. Mohan on *"Simple Accurate Expressions for Planar Spiral Inductances"* [41] finally delivered the correct references, and unfolded a lot of analytical calculations and even some historical perspectives. Therefore, the idea of this section is to provide deep insight into which amount of the inductance calculation can be done analytically. It shows that a significant amount of work was already done by James Clerk Maxwell around 1873 in his book *"A Treatise on Electricity and Magnetism"* [48]. Maxwell even noticed the importance of the Geometric Mean Distance (GMD) of conductor cross-sections, which is presented later in this section. A few decades later, Edward Bennett Rosa, chief-physicist at the U.S. National Bureau of Standards (today NIST), published a comprehensive collection of inductance calculations over the years from 1907 to 1916. In 1999, Mohan combined and approximated these results to derive simple expressions for on-chip inductors in his Ph.D. thesis at Stanford University [40].

### 3.2.1 Self-Inductance

In principle, *self-inductance* $L$, also called simply inductance, is only defined for current loops by the ratio of magnetic flux $\Phi$ through the area enclosed by the loop and the current $I$ flowing in this loop. More complicated topologies, like a coil with $n$ turns, are sometimes considered to enclose the magnetic flux several times, and the enclosed magnetic flux is then called *flux linkage* $\Psi$. This word sometimes occurs in the context of inductance calculation but represents simply the total magnetic flux enclosed and is thus identical to magnetic flux. For completeness though not related to this work, it should be noted that the definition of the term flux linkage is extended in the (controversial) discussion of memristors (see [49]).

$$L := \frac{\Phi}{I} \equiv \frac{\Psi}{I} = \frac{N\Phi_{\text{single-turn}}}{I} \tag{3.1}$$

From Faraday's law of induction, one of the Maxwell equations, the well known relation between voltage and current across an inductor can be derived.

$$V_L = -\oint_{\partial\mathcal{A}} \mathbf{E}\mathrm{d}\mathbf{s} = \int_{\mathcal{A}} \frac{\partial \mathbf{B}}{\partial t}\mathrm{d}\mathbf{A} = \frac{\mathrm{d}\Phi}{\mathrm{d}t} = \frac{\mathrm{d}(LI)}{\mathrm{d}t} = L\frac{\mathrm{d}I}{\mathrm{d}t} \tag{3.2}$$

Note that the first minus sign stems from the usual circuit theory definition where voltage and current direction are equal. Furthermore, it is assumed that the geometry, i.e. $\mathcal{A}$ and $L$, do not change over time. Since on-chip inductors are static topologies, this is a valid assumption. This equation also contains the relation of magnetic flux to magnetic flux density $\mathbf{B}$, which is the actual vector field for magnetism. In general, the flux density is the quantity needed to derive the inductance of an (arbitrary) current distribution. The discussion of the inductance of a straight wire with radius $R$ and length $l$ constitutes the starting point for all following calculations. Its derivation is actually not easily found and a number of misconceptions are spread in the literature, which will be mentioned later at the appropriate steps. It is insightful to start with the magnetic flux density of an infinite straight wire carrying a current $I$ distributed homogeneously over its cross-section. In cylindrical coordinates, the magnetic field is given by Eq. 3.3, where $\hat{\varphi}$ is the tangential unit vector.

$$\mathbf{B}_{w,\infty}(\mathbf{r}) = \frac{\mu_0 I}{4\pi}\hat{\varphi}\begin{cases} 2/r, & r > R \\ 2r/R^2, & r \leq R \end{cases} \tag{3.3}$$

The energy of the magnetic field is computed and set equal to the energy stored in an inductor with inductance $L$ and the identical current $I$.

$$\frac{1}{2}LI^2 \stackrel{!}{=} \frac{1}{2\mu_0}\int_V \mathbf{B}_{w,\infty}(\mathbf{r})\mathbf{B}_{w,\infty}(\mathbf{r})\mathrm{d}V \tag{3.4}$$

$$= \frac{1}{2\mu_0}\int_{-l/2}^{l/2}\int_0^{\infty}\int_0^{2\pi} \mathbf{B}_{w,\infty}(\mathbf{r})\mathbf{B}_{w,\infty}(\mathbf{r})r\mathrm{d}\phi\mathrm{d}r\mathrm{d}z \tag{3.5}$$

$$= \frac{2\pi l}{2\mu_0}\left(\int_0^R \mathbf{B}_{w,\infty}(\mathbf{r})\mathbf{B}_{w,\infty}(\mathbf{r})r\mathrm{d}r + \int_R^{\infty} \mathbf{B}_{w,\infty}(\mathbf{r})\mathbf{B}_{w,\infty}(\mathbf{r})r\mathrm{d}r\right) \tag{3.6}$$

$$= \frac{2\pi l}{2\mu_0}\frac{\mu_0^2 I^2}{4\pi^2}\left(\int_0^R \frac{r^3}{R^4}\mathrm{d}r + \int_R^{\infty}\frac{1}{r}\mathrm{d}r\right) \tag{3.7}$$

$$= \frac{\mu_0 l I^2}{4\pi}\left(\frac{1}{4} + \int_R^{\infty}\frac{1}{r}\mathrm{d}r\right) \to \infty \tag{3.8}$$

$$\Rightarrow L_{w,\mathrm{int}} = \frac{\mu_0 l}{8\pi} \tag{3.9}$$

The inductance $L_{w,\mathrm{int}}$ derived with Eq. 3.4 from the finite part is called *internal inductance* as it is caused by the energy stored in the magnetic field within the conductor volume. However, also the part outside the wire, called *external inductance*, needs to be finite. At first glance, it seems impossible to calculate since the above integral does not converge

even for a wire segment of finite length $l$. In fact, every infinitesimal segment of length $\mathrm{d}z$ of an infinite straight wire carrying a constant current $I$ stores an infinite amount of magnetic energy. This fact leads to a lot of confusion and causes people to assume the inductance of a straight wire cannot be calculated, or they even consider the internal inductance to be the correct result. However, this is not true! The error made in the previous attempt lies in the magnetic field used. While the calculation may have considered only a finite part of the wire in the energy calculation, the magnetic field used was the field of the infinite wire. So to get the correct result, the magnetic field of a finite wire has to be computed. This may seem trivial in retrospect but is hard to figure out when researching the inductance of a straight wire, due to a lot of misconceptions. The main confusion probably arises from the fact that, due to the radial symmetry of the wire and current distribution, it is assumed that the current in a wire segment causes a field only in radial direction, when it actually contributes to the field at every point in space. So the parts that were intended to be neglected by integrating only over the finite wire length actually still contribute to the field in the integration volume.
For time-invariant current density distributions $\mathbf{J}(\mathbf{r})$, the magnetic field can be calculated with the Biot-Savart law.

$$\mathbf{B}(\mathbf{r}) = \frac{\mu_0}{4\pi} \int_V \mathbf{J}(\mathbf{r}') \times \frac{\mathbf{r} - \mathbf{r}'}{|\mathbf{r} - \mathbf{r}'|^3} \mathrm{d}V' \tag{3.10}$$

It can be shown that a uniform current inside a cylinder produces the same magnetic field outside of this cylinder as a current concentrated at the axis of the wire [50]. Thus, an alternative version of the Biot-Savart law for the case of a current $I$ flowing along a curve described by a vector $\mathbf{c}$ can be used. Of course the resulting field is only valid for $r \geq R$.

$$\mathrm{d}\mathbf{B} = \frac{\mu_0 I}{4\pi} \cdot \frac{\mathrm{d}\mathbf{c} \times (\mathbf{r} - \mathbf{c})}{|\mathbf{r} - \mathbf{c}|^3} \tag{3.11}$$

Using the norm of the cross product and replacing the sine function with the ratio of the side opposite to $\theta$ and the hypotenuse (see Fig. 3.7), the following integral for the magnetic flux density is obtained. As expected the result is also a function of $z$.

$$\begin{aligned} \mathbf{B}_{w,\mathrm{ext}}(\mathbf{r}) &= \frac{\mu_0 I}{4\pi} \hat{\varphi} \int_{-l/2}^{l/2} \frac{r \mathrm{d}c}{(r^2 + (c - z)^2)^{3/2}} \\ &= \frac{\mu_0 I}{4\pi} \hat{\varphi} \cdot \left( \frac{l/2 + z}{r\sqrt{r^2 + (l/2 + z)^2}} + \frac{l/2 - z}{r\sqrt{r^2 + (l/2 - z)^2}} \right) \end{aligned} \tag{3.12}$$

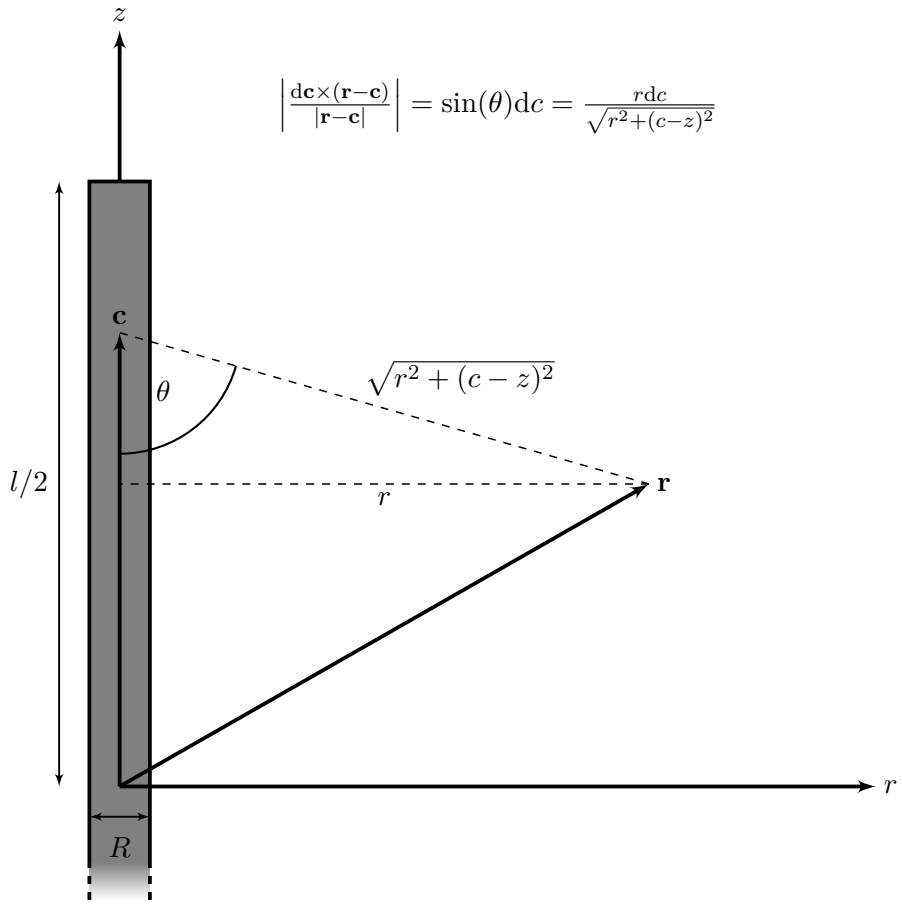



**Fig. 3.7:** Illustration of the magnetic flux density calculation for a finite straight cylindrical wire.

After calculating the external magnetic flux density ($r \geq R$), it is necessary to also consider the internal magnetic field. Rosa assumes it to be the same as for the infinite wire [50]. Although, this assumption may be approximately valid for reasonably long wires at their center ($z = 0$) as will be shown later, it cannot be exact even though Rosa claims to derive an exact formula. The field of the infinite wire can be calculated with Ampère's law.

$$\oint_{\partial\mathcal{A}} \mathbf{B}(\mathbf{r})\mathrm{d}\mathbf{s} = \mu_0 I_{\mathrm{enc}} \tag{3.13}$$

The integral of the magnetic field along a closed loop is proportional to the current flowing through the area enclosed by it. This law cannot be applied in the case of a finite wire because the magnetic field of the return path is not present. The infinite wire however can be thought of as closed at infinity. So this is one reason why Rosa's assumption is invalid. Another one is that the tangential magnetic field – the magnetic field is only tangential here – has to be continuous at the wire surface in the absence of surface currents. However, using the internal field of the infinite wire results in a

discontinuity at $r = R$. So instead of calculating the magnetic field inside the conductor with Ampère's law, this thesis proposes to use the Biot-Savart law again. But this time the current $I$ is scaled by $r^2/R^2$ to only consider the current inside a cylinder with radius $r$. This does not change the integral in Eq. 3.12.

$$\mathbf{B}_{w,\mathrm{int}}(\mathbf{r}) = \frac{\mu_0 I}{4\pi}\hat{\varphi} \cdot \frac{r^2}{R^2} \cdot \left( \frac{l/2 + z}{r\sqrt{r^2 + (l/2 + z)^2}} + \frac{l/2 - z}{r\sqrt{r^2 + (l/2 - z)^2}} \right) \tag{3.14}$$

It is useful to look at the limit of the large term in the brackets for $l$ to infinity, which results exactly in the factor of 2 present in Eq. 3.3. So the case of the infinitely long wire can be derived from the finite one, which is a valuable consistency check. To visualize the magnetic field in $z$ and $r$ direction, Fig. 3.8 and 3.9 show the relative magnitude of the magnetic field over both axes normalized to $R$. The $R$ in the factor on the y-axis originates from the $r$ in front of the square roots in Eq. 3.14 when normalized to $r/R$, while the remaining expression is already dimensionless. Both plots show five curves for different lengths. The figures also visualize how the lines approach the limit of the infinite wire and especially that the radial field is continuous at $r = R$. Although, Rosa's assumption is already a good approximation at $l = 5R$ at the center of the wire ($z = 0$).

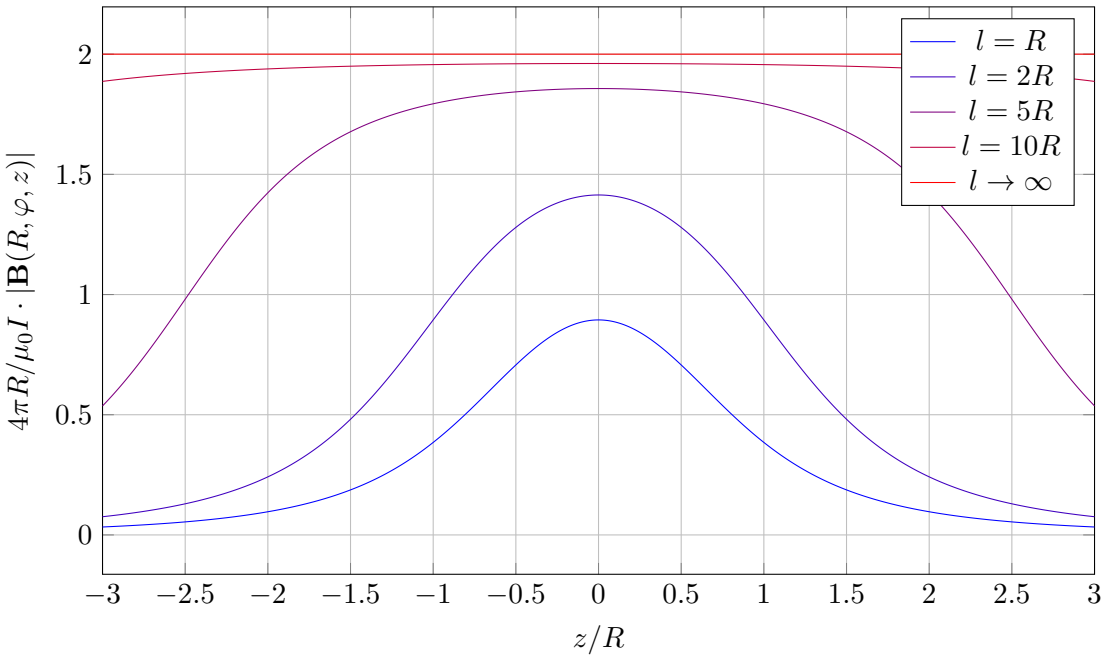



**Fig. 3.8:** The relative magnetic flux density along the z-axis.

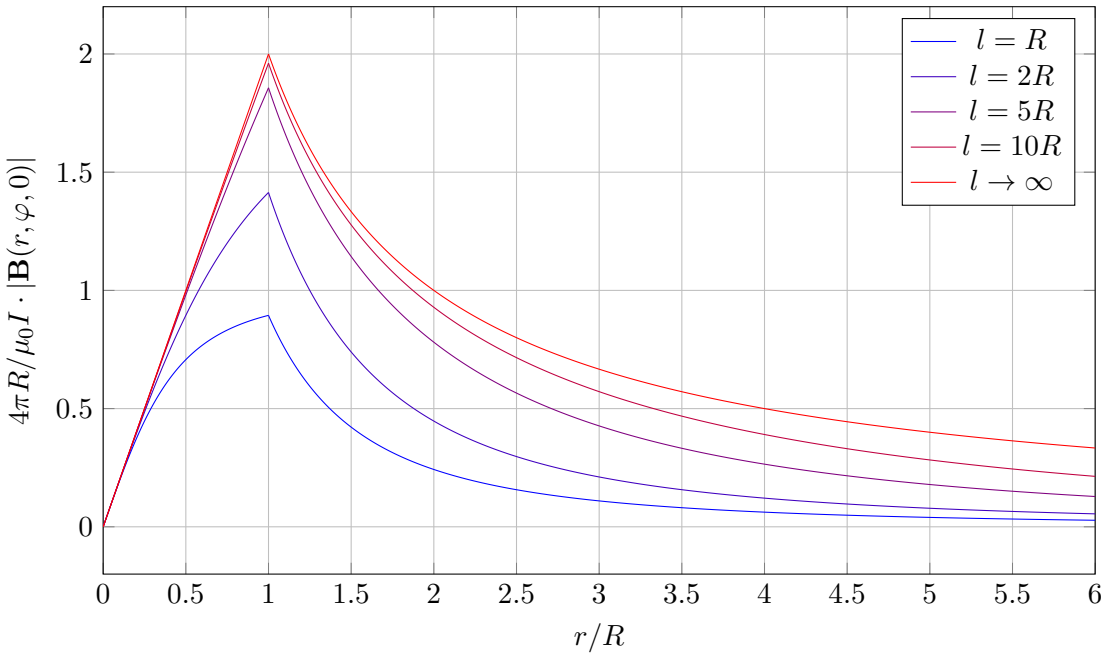



**Fig. 3.9:** The relative magnetic flux density along the r-axis.

The next step is to finally calculate the inductance for the finite wire. A volume integral over the squared field from the energy based approach of Eq. 3.4 is complicated. Hence like Rosa, this thesis continues by calculating the magnetic flux. The conceptual challenge is now the area over which the magnetic flux is integrated as inductance is only defined for closed loops, but a finite straight wire does not enclose a certain area. Rosa saw this problem too and dedicated a short section in [50] to provide an explanation or definition for open circuits: "*The actual self-inductance of any closed circuit of which it [the open circuit] is a part will be the sum of the self-inductances of all the parts, plus the sum of the mutual inductances of each one of the component parts on all the other parts.*" – i.e. it is simply defined as the inductance of a part of a closed circuit. Thus, the flux through the right half-plane due to the external magnetic flux density is to be calculated by integrating over $r$ and $z$.

$$\Phi_{w,\text{ext}} = \int \mathbf{B}_{w,\text{ext}}(\mathbf{r})\mathrm{d}\mathbf{A} \tag{3.15}$$

$$= \frac{\mu_0 I}{4\pi} \int_R^\infty \int_{-\infty}^\infty \left( \frac{l/2 + z}{r\sqrt{r^2 + (l/2 + z)^2}} + \frac{l/2 - z}{r\sqrt{r^2 + (l/2 - z)^2}} \right) \mathrm{d}z\mathrm{d}r \tag{3.16}$$

$$= \frac{\mu_0 I}{4\pi} \int_R^\infty \frac{2l}{r}\mathrm{d}r \to \infty \tag{3.17}$$

However, the integral does still not converge. Rosa found this mistake made by others, too. The actual area for the integration is not the complete right half-plane but only the

area for $-l/2 \leq z \leq l/2$ because only these field lines cross the wire when the current changes. Rosa explains it as follows: "*When the current in the wire decreases, the field everywhere decreases in intensity, and we think of the lines as collapsing upon the wire; that is, moving in from all sides upon the wire. But those lines above BB' and below AA' do not cut the wire, and hence contribute nothing to the self-inductance.*" [50], where "*above BB' and below AA'*" means $|z| \geq l/2$. Due to this, the integration in $z$ is now limited to the range along the wire.

$$\Phi_{w,\text{ext}} = \int \mathbf{B}_{w,\text{ext}}(\mathbf{r})\mathrm{d}\mathbf{A} \tag{3.18}$$

$$= \frac{\mu_0 I}{4\pi} \int_R^\infty \int_{-l/2}^{l/2} \left( \frac{l/2+z}{r\sqrt{r^2+(l/2+z)^2}} + \frac{l/2-z}{r\sqrt{r^2+(l/2-z)^2}} \right) \mathrm{d}z\mathrm{d}r \tag{3.19}$$

$$= \frac{\mu_0 I}{4\pi} \int_R^\infty \frac{2}{r} \left( \sqrt{r^2+l^2} - r \right) \mathrm{d}r \tag{3.20}$$

$$= \frac{\mu_0 I}{4\pi} \left( 2l \ln \frac{\sqrt{l^2+R^2}+l}{R} - 2\sqrt{l^2+R^2} + 2R \right) \tag{3.21}$$

The external inductance is now finite and for the total inductance, Rosa just added the internal inductance from Eq. 3.9. As explained before, this thesis intends to use $\mathbf{B}_{w,\text{int}}(\mathbf{r})$ and calculate the magnetic flux like before. However, in contrast to the external flux, the internal flux density has to be weighted with the wire cross-section it actually crosses, when thinking of a decreasing current. As a check for correctness of this way of thinking, Rosa compared his result (although obtained with the internal field of the infinite wire) to the internal inductance calculation via the field energy.

$$\Phi_{w,\text{int}} = \int \frac{r^2}{R^2} \mathbf{B}_{w,\text{int}}(\mathbf{r})\mathrm{d}\mathbf{A} \tag{3.22}$$

$$= \frac{\mu_0 I}{4\pi} \int_0^R \frac{r^4}{R^4} \int_{-l/2}^{l/2} \left( \frac{l/2+z}{r\sqrt{r^2+(l/2+z)^2}} + \frac{l/2-z}{r\sqrt{r^2+(l/2-z)^2}} \right) \mathrm{d}z\mathrm{d}r \tag{3.23}$$

$$= \frac{\mu_0 I}{4\pi} \int_0^R \frac{2r^3}{R^4} \left( \sqrt{r^2+l^2} - r \right) \mathrm{d}r \tag{3.24}$$

$$= \frac{\mu_0 I}{4\pi} \cdot \frac{2}{15R^4} \left( 2l^5 - 3R^5 + \left(3R^2 - 2l^2\right) \left(l^2+R^2\right)^{\frac{3}{2}} \right) \tag{3.25}$$

Note that the limit $l \to \infty$ does not yield the internal inductance from Eq. 3.9 but $\infty$. This is because the limit has to be applied only for the flux density since the integration borders should not go to infinity.

The combination of both results finally gives the inductance of a straight wire $L_w$.

$$L_w = \frac{\Phi_{w,\mathrm{int}} + \Phi_{w,\mathrm{ext}}}{I} \tag{3.26}$$

$$= \frac{\mu_0}{2\pi}\left(l \ln \frac{\sqrt{l^2+R^2}+l}{R} - \sqrt{l^2+R^2} + R + \frac{2l^5 - 3R^5 + \left(3R^2 - 2l^2\right)\left(l^2+R^2\right)^{\frac{3}{2}}}{15R^4}\right)$$

$$\overset{l \gg R}{\approx} \frac{\mu_0 l}{2\pi}\left(\ln\left(\frac{2l}{R}\right) - \frac{3}{4} + \frac{4R}{5l} - \frac{R^2}{6l^2}\right) \tag{3.27}$$

$$\overset{l \ll R}{\approx} \frac{\mu_0 l}{2\pi}\left(\frac{2l}{3R} - \frac{l^3}{6R^3}\right) \tag{3.28}$$

$$\overset{l \approx R}{\approx} \frac{\mu_0 l}{2\pi}\left(0.589 + 0.488 \cdot \frac{l-R}{R} - 0.097 \cdot \left(\frac{l-R}{R}\right)^2\right) \tag{3.29}$$

The approximations are given for later reference in the next subsections. An example value calculated with this formula is 1.5 µH for a wire with a diameter of 1 mm and a length of 1 m. The approach for the internal field and, as a consequence, the expression for the internal inductance of the finite round wire are not found in literature to the knowledge of the author.

### 3.2.2 Mutual Inductance

Structures more complex than a single straight wire can sometimes be composed of multiple straight wires. In addition to the self-inductance, the interaction between the wires also needs to be considered. A changing current in the first wire will induce a voltage in the second wire and vice versa. This is described by the mutual inductance. Consider two parallel wires of equal length $l$ at a center-to-center distance $d$. Following the line of thought from Rosa [50], the magnetic field lines that cut the second wire, when the current in the first wire decreases, is found by integrating in Eq. 3.19 from $d$ to infinity instead from $R$ to infinity.

$$M_{2w} = \frac{\mu_0 l}{2\pi}\left(\ln \frac{\sqrt{l^2+d^2}+l}{d} - \frac{\sqrt{l^2+d^2}-d}{l}\right) \tag{3.30}$$

This expression is only exactly valid for two filaments, i.e. infinitesimal thin wires. Another, more general method to calculate the mutual inductance between two wire filaments is the Neumann formula shown in Eq. 3.31. It is derived from the vector potential of the magnetic field. Given a number of closed circuit loops, the formula describes the mutual inductance from loop $i$ on loop $j$. Note that the expression is

symmetric in indices, so $L_{ij} = L_{ji}$.

$$L_{ij} = \frac{\mu_0}{4\pi} \oint_{\mathcal{C}_j} \oint_{\mathcal{C}_i} \frac{\mathrm{d}\mathbf{r}_i \mathrm{d}\mathbf{r}_j}{|\mathbf{r}_i - \mathbf{r}_j|} \tag{3.31}$$

For two parallel filaments this can be simplified, since $\mathrm{d}\mathbf{r}_i$ and $\mathrm{d}\mathbf{r}_j$ are parallel. The resulting expression from Eq. 3.32 is identical to Eq. 3.30.

$$M_{2w} = \frac{\mu_0}{4\pi} \int_{-l/2}^{l/2} \int_{-l/2}^{l/2} \frac{1}{\sqrt{d^2 + (z - z')^2}} \mathrm{d}z \mathrm{d}z' \tag{3.32}$$

### 3.2.3 Geometric and Arithmetic Mean Distances

So far, only conductors with a circular cross-section have been discussed. In practice, other cross-sections are also very important, particularly rectangular ones for on-chip inductors. The previous calculations were already quite cumbersome, and it does not seems likely to get handy results for more complicated cross-sections. However, the inductance of conductors with arbitrary cross-sections can be calculated with the Neumann formula (Eq. 3.31). This is based on "*the theorem that the self-inductance of a circuit is equal to the sum of all the mutual inductances of the component parts of the circuit*" [50], i.e. the sum of all the mutual inductances between every pair of current filaments starting at two points on the wire cross-section at a distance $d = |\mathbf{r} - \mathbf{r}'|$ apart. This is done by integrating twice over the area of the cross-section.

$$L_{\text{self}} = \frac{1}{A^2} \int_{\mathcal{A}} \int_{\mathcal{A}'} M_{2w}(d, l) \mathrm{d}A' \mathrm{d}A \tag{3.33}$$

Divided by the square of the cross-sectional area $A$, this is essentially an arithmetic mean of the mutual inductance over the cross-section. The Neumann integral is basically an integral over an inverse distance and will contain a dominant term in the form of $\ln(d)$ [51]. Other terms may appear depending on the shape of the curves $\mathcal{C}_i$ and $\mathcal{C}_j$, e.g. for two straight wires it was the second term in Eq. 3.30. For circular filaments, the mutual inductance is given by elliptic integrals. This is also described in the various references used in this section. For on-chip inductors, mainly straight filaments are relevant. $M_{2w}$ can hardly be integrated analytically. For this reason, an approximation is made by Rosa [50] for small distances compared to the filament length because wires are usually longer than their diameter (Eq. 3.34). As a side note, an analytical solution for the mutual inductance of two rectangular wires is given by [52]. However, the resulting expression is very large even without the integration borders inserted.

$$M_{2w} \overset{l \gg d}{\approx} \frac{\mu_0 l}{2\pi} \left( \ln(2l) - \ln(d) - 1 + d/l - d^2/4l^2 \right) \tag{3.34}$$

The arithmetic mean of these terms is much easier to calculate. The first one would be $\ln(d)$. Note that the arithmetic mean of the logarithms of some values $x_i$ is the logarithm of the geometric mean of these values (Eq. 3.35).

$$\frac{1}{n}\sum_{i=1}^{n}\ln\left(x_i\right) = \ln\left(\sqrt[n]{\prod_{i=1}^{n}x_i}\right) \tag{3.35}$$

This observation leads to the importance and definition of the *Geometric Mean Distance (GMD)* in analytic inductance calculations.

$$\ln\left(\mathrm{GMD}\right) := \frac{1}{A^2}\int_A\int_{A'}\ln\left(d\right)\mathrm{d}A'\mathrm{d}A \tag{3.36}$$

The linear term defines the *Arithmetic Mean Distance (AMD)*, and the square term the *Arithmetic Mean Square Distance (AMSD).*

$$\mathrm{AMD} := \frac{1}{A^2}\int_A\int_{A'}d\mathrm{d}A'\mathrm{d}A \tag{3.37}$$

$$\mathrm{AMSD}^2 := \frac{1}{A^2}\int_A\int_{A'}d^2\mathrm{d}A'\mathrm{d}A \tag{3.38}$$

As a first application of this method, the self-inductance of a straight wire of length $l$ and radius $R$ is calculated again. Maxwell derived the GMD of a circle in [53]. He was presumably the first one to notice the importance of the GMD in inductance calculation. To find the AMD and AMSD of a circular cross-section, one should take a closer look at the expression to be integrated.

$$d = |\mathbf{r}-\mathbf{r}'| = \sqrt{r^2+r'^2-2rr'\cos\left(\varphi-\varphi'\right)} \tag{3.39}$$

While the AMSD removes the square-root in the integral and renders it therefore trivial to calculate, the AMD is not easily calculated. It was derived in an example in [54]. This paper provides "*bounds on the average distance between two points uniformly and independently chosen from a compact convex subset of the s-dimensional Euclidean space*" [54]. So the mean distances for a disc are the following.

$$\mathrm{GMD}_\bullet = \exp(-1/4)\cdot R \approx 0.779R \tag{3.40}$$

$$\mathrm{AMD}_\bullet = \frac{128}{45\pi}\cdot R \approx 0.905R \tag{3.41}$$

$$\mathrm{AMSD}_\bullet = R \tag{3.42}$$

Replacing the terms in Eq. 3.34 with the appropriate mean values then yields an approximate expression for the self-inductance of a wire with a circular cross-section.

$$L_{w,\mathrm{md}} = \frac{\mu_0 l}{2\pi}\left(\ln(2l) - \ln(\mathrm{GMD}_\bullet) - 1 + \mathrm{AMD}_\bullet/l - \mathrm{AMSD}_\bullet^2/4l^2\right) \tag{3.43}$$

$$= \frac{\mu_0 l}{2\pi}\left(\ln(2l) - \ln(R) - \frac{3}{4} + \frac{128}{45\pi}\cdot\frac{R}{l} - \frac{R^2}{4l^2}\right) \tag{3.44}$$

This expression for a wire with circular cross-section was not found in literature by the author, only its components. The exact value for $\mathrm{AMD}_\bullet$ never showed up in the context of inductance calculation. Some even believe "*This integral cannot be solved analytically.*" [55]. Compared to the exact formula in Eq. 3.27, $L_{w,\mathrm{md}}$ has larger magnitudes in the linear and square term.

Another approach taken by Weaver [51], replaces every $d$ in $M_{2w}$ with $\mathrm{GMD}_\bullet$ for an approximate formula. Weaver did not actually explain why but only that he neglects higher order mean distances. In fact, (almost) the same was also done by Greenhouse [56] and criticized by Mohan [40] for not being mathematically well founded. Eqs. 3.45 to 3.47 show the series expansions of the formula proposed by Weaver. The constants are rounded to three digits.

$$L_{w,\mathrm{weaver}} \overset{l\gg R}{\approx} \frac{\mu_0 l}{2\pi}\left(\ln\left(\frac{2l}{R}\right) - \frac{3}{4} + 0.779\cdot\frac{R}{l} - 0.152\cdot\frac{R^2}{l^2}\right) \tag{3.45}$$

$$\overset{l\ll R}{\approx} \frac{\mu_0 l}{2\pi}\left(0.642\cdot\frac{l}{R} - 0.088\cdot\frac{l^3}{R^3}\right) \tag{3.46}$$

$$\overset{l\approx R}{\approx} \frac{\mu_0 l}{2\pi}\left(0.580 + 0.489\cdot\frac{l-R}{R} - 0.094\cdot\left(\frac{l-R}{R}\right)^2\right) \tag{3.47}$$

A comparison of the constants in all three series expansions shows that using the GMD for all $d$ yields an approximate formula that is in very good agreement with the exact expression. One might expect this to be the case for long wires since a comparison of the series expansions of $L_w$ and $M_{2w}$, neglecting the linear and higher terms, results in $d$ being equal to the GMD. For short wires, however, the exact reason why this fits so well remains unclear.

Aebischer et al. discusses the mean distance method and proposes another variant of the mean distance formula [55]. They also use the exact expression for $M_{2w}$, but instead of replacing every $d$ with the GMD, they use the AMD for the linear term and AMSD for every $d^2$. However, they also do a series expansion of the terms, for which they used the AMSD as an approximation, around $d = \mathrm{AMSD}_\bullet$ and compute a correction term to increase the accuracy. Adding both leads to their final formula for which here only the series expansions are shown as they facilitate an easier comparison to other

approximations.

$$L_{w,\text{aebischer}} \overset{l\gg R}{=} L_{w,\text{md}} + \mathcal{O}\left(\frac{R^4}{l^4}\right) \tag{3.48}$$

$$\overset{l\ll R}{\approx} \frac{\mu_0 l}{2\pi}\left(0.061 + 0.642\cdot\frac{l}{R} - 0.101\cdot\frac{l^3}{R^3}\right) \tag{3.49}$$

$$\overset{l\approx R}{\approx} \frac{\mu_0 l}{2\pi}\left(0.634 + 0.475\cdot\frac{l-R}{R} - 0.097\cdot\left(\frac{l-R}{R}\right)^2\right) \tag{3.50}$$

For long wires this results in the same expression as for the initial mean distance formula plus higher order terms. For short wires, a constant terms shows up and for medium lengths the constant term is higher than it should be.

The integration over a conductor cross-section can also be done with a Monte Carlo code. To investigate this, a Monte Carlo Python code has been developed and higher order mean distances have been computed. The results in Tab. 3.1 show that the first three – exactly known – mean distances are computed correctly. They are used for another approximation $L_{w,\text{md,numeric}}$.

| Mean Distance | Integrand | Numerical Value | Absolute Error | Rounded Value |
|---|---|---|---|---|
| GMD | $\ln(d)$ | 0.778799(9656181976) | $8.175\cdot10^{-7}$ | 0.77880 |
| AMD | $d$ | 0.905414(8839886084) | $0.967\cdot10^{-7}$ | 0.90541 |
| AMSD | $d^2$ | 1.000000(7791999586) | $7.792\cdot10^{-7}$ | 1.00000 |
| - | $d^4$ | 1.136221(1315214665) | - | 1.13622 |
| - | $d^6$ | 1.232193(2598736391) | - | 1.23219 |
| - | $d^8$ | 1.304775(4703107082) | - | 1.30478 |
| - | $d^{10}$ | 1.362207(1403827725) | - | 1.36221 |

**Tab. 3.1:** Numerical values from Monte Carlo integration with $10^{10}$ random point pairs in a circle with $R = 1$. The third column shows the error relative to the known exact values and suggests that at least six digits can be trusted. The last column contains the rounded values.

The approximate formulas presented so far are compared to the exact self-inductance in Fig. 3.10 and 3.11. They show the relative error of the corresponding approximate formula $L_\approx$ to the exact formula $L_w$, $\Delta_w = L_\approx/L_w - 1$. The first observation is that for moderately long wires, $l \gtrsim 3R$, the error for all formulas is around 2%, i.e. for all practical cases every formula can be used without relevant error. As expected from the previous series expansions, the formula given by Weaver fits well over the whole range. For $l \to 0$ the error is approximately 3.84%. The formulas approximating the mutual inductance for long wires fall apart around $l \approx R$, as expected. These are $L_{w,\text{md}}$, $L_{w,\text{aebischer}}$ and $L_{w,\text{md,numeric}}$. The result from numeric integration of $M_{2w}$, $L_{w,\text{numeric}}$, however, also diverges from $L_w$ for short wires. This was not expected and cannot be

explained by the author. A numerical problem can be ruled out as several methods lead to the same result: numerical integration with Mathematica, Monte Carlo integration with Python, and Monte Carlo integration with Python using the bigfloat package for increased precision. The internal inductance, which is dominating the self-inductance for "disc-like" wires, proposed by Rosa, $L_{w,\mathrm{int}}$, does also not explain this discrepancy. It fits even worse. Most likely, the Neumann integral is not defined very well if $\mathcal{C}_i$ and $\mathcal{C}_j$ are no proper loops and their separation is large compared to their size. Another reason could be that the Neumann integral is actually not well defined when integrated over the same area twice, which is done for the self-inductance. For large $l$, this might not be a problem as $M_{2w}$ is approximated with a logarithm in this domain, but for small $l$ it is a $1/d$ function.

With this last point remaining unsolved, this discussion of the cylindrical conductor has provided the insight that self- and mutual inductance of wires with arbitrary cross-section can be computed using some kind of mean distance approximation. However, the result is only reliable for at least $l \gtrsim 3R$. This line of thought, despite being much less verbose about the backgrounds, was used by Mohan to derive approximate formulas for on-chip inductors.

The comparison of the previous formulas was done for a round wire because an exact solution is available for it, which the approximations can be compared to. Likely the same relation holds true for different cross-sections, especially wires with a rectangular cross-section as they occur in chip design. There, the lateral dimension is called the width $w$ and the vertical dimension the thickness $t$. Both GMD and AMD of a rectangular cross-section are quite complicated to calculate. The former was done by Maxwell in [57], the latter can be found in [54], and the AMSD is trivial to calculate.

$$\begin{aligned}
\ln(\mathrm{GMD}_{\blacksquare}) &= \ln\left(\sqrt{w^2+t^2}\right) - \frac{w^2}{6t^2}\ln\left(\sqrt{1+\frac{t^2}{w^2}}\right) - \frac{t^2}{6w^2}\ln\left(\sqrt{1+\frac{w^2}{t^2}}\right) && (3.51)\\
&\quad + \frac{2w}{3t}\arctan\left(\frac{t}{w}\right) + \frac{2t}{3w}\arctan\left(\frac{w}{t}\right) - \frac{25}{12}\\
&\approx \ln(w+t) - 1.5 && (3.52)\\
\mathrm{AMD}_{\blacksquare} &= \frac{1}{15}\left(\frac{w^3}{t^2} + \frac{t^3}{w^2} + \sqrt{w^2+t^2}\left(3 - \frac{w^2}{t^2} - \frac{t^2}{w^2}\right)\right) && (3.53)\\
&\quad + \frac{1}{6}\left(\frac{t^2}{w}\ln\left(\frac{w+\sqrt{w^2+t^2}}{t}\right) + \frac{w^2}{t}\ln\left(\frac{t+\sqrt{w^2+t^2}}{w}\right)\right)\\
&\approx \frac{\sqrt{w^2+t^2+0.46wt}}{3} && (3.54)\\
\mathrm{AMSD}_{\blacksquare}^2 &= \frac{1}{6}\left(w^2+t^2\right) && (3.55)
\end{aligned}$$

The approximate formula for $\mathrm{GMD}_{\blacksquare}$ is actually valid for every aspect ratio of the cross-

section. This was reported by Rosa [50] but can also be shown easily by looking at the series expansion for a square or even a one-dimensional cross-section. For AMD$_\blacksquare$, the approximation is given by Mohan and is also very accurate for different aspect ratios [40].

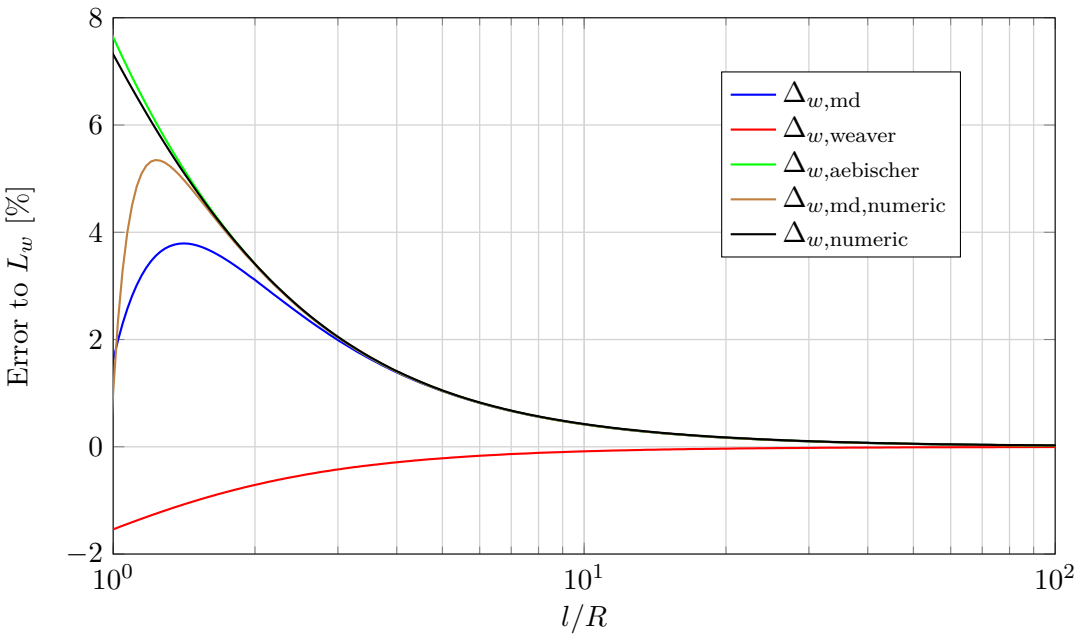



**Fig. 3.10:** Errors of mean distance formulas for the self-inductance of a straight wire with circular cross-section ($l > R$).

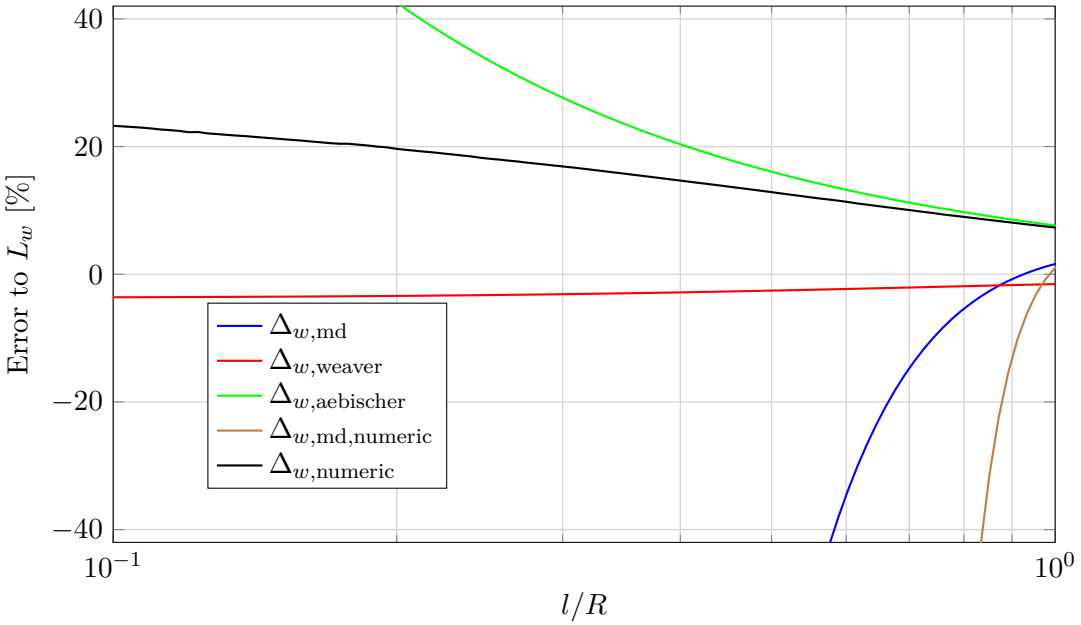



**Fig. 3.11:** Errors of mean distance formulas for the self-inductance of a straight wire with circular cross-section ($l < R$).

## 3.3 Modeling of Inductors

So far, this chapter has presented possible geometries for on-chip inductors and how their layout can be generated, as well as the basics of inductance calculation for straight wires with arbitrary cross-section. This section discusses how inductors are modeled in the circuit design flow. The fundamental problem with custom on-chip inductors is that they are made up of wires and are therefore not properly represented in the schematics. Every other component in chip-design has a variety of PCells provided with the PDK, e.g. transistors, diodes, resistors, capacitors, etc. These PCells serve two main aspects: they provide a model for simulation, and they contain the corresponding layout. Inductors can be modeled with the help of ideal schematic devices from the Virtuoso "analogLib", but there is no layout available as they are not part of the PDK. Having non-PDK devices in a schematic causes the Layout Versus Schematic (LVS) check to fail because they are not found in the layout. Thus, it is mandatory to provide *multiple schematic views* of a custom inductor for different purposes. For fast simulation and design, "analogLib"-devices are the best way to go. Inductors appear only as metal traces in the layout, but the devices connected to them are logically not connected to the same net so for LVS checks, *LVS-resistors* are necessary. They are part of the PDK-PCells and act as net separators for the LVS check but do not alter the physical properties of a wire. At least one additional third view is often needed, which contains an "nport" to include an S-parameter file with the "actual" behavior of the inductor. S-parameters and the question how to simulate inductors are discussed in the next section. In this section, the modeling with "analogLib"-devices, the influence of metal fill, skin effect, and process corners are discussed, based on the example of a two-terminal square spiral inductor, which is shown in Fig. 3.13. This gives an estimation on the magnitude of the aforementioned effects, and the results can be transferred to the T-coil relatively easy.

### 3.3.1 Segmented Circuit Models

The term "*segmented circuit models*" originates from Mohan [40]. Long et al. [58] called this kind of models "scalable inductor models", while Galal et al. [8] used the term "distributed model", even if the model is not a distributed-element circuit like ,e.g., a transmission line. Regardless of the name, the idea behind this kind of inductor modeling is to model segments of the inductor, e.g. each turn or even each side of a turn, with dedicated circuits. Multiple versions of differently sized segment models are then combined according to the number of segments the inductor comprises. To reflect capacitive coupling and mutual inductance between the segments, different approaches are taken. Long et al. [58] shows a dependent current source in each segment, which is intended to model the mutual inductance. Another approach is taken by Galal et

al. [8] by adding a coupling branch between neighboring segments but neglecting further interactions. Fig. 3.12 shows the model used by Long et al. [58] to describe a single inductor segment. The `SEG_i_0` and `SEG_i_1` pins are used to build a series circuit according to the number of segments. The size of the dependent current source is given by $I_{\text{seg},i} = I_j M_{\text{seg},ji}/L_{\text{seg},i}$, where $I_j$ is the current flowing through $L_{\text{seg},j}$ [58].

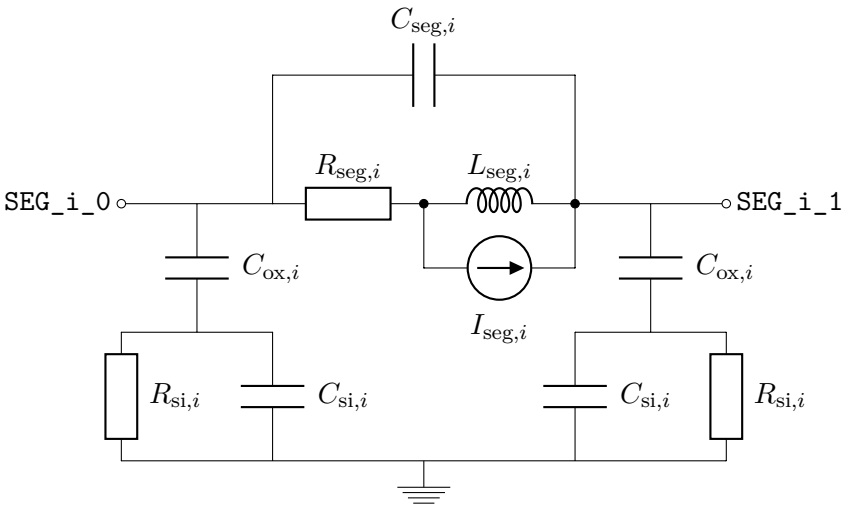



**Fig. 3.12:** $\pi$-model of an inductor segment $i$ used in [58].

These segmented models have several drawbacks. Firstly, they comprise a lot of segments for inductors with multiple turns. The size of such a circuit can easily surpass that of the rest of the design in terms of components [40]. This variability in circuit size can be tolerated to model a specific inductor but is impractical for circuit design because the large number of components makes transfer functions too complicated for optimization, and the circuit needs to be updated every time the number of segments changes. Secondly, it is very difficult to accurately estimate the large number of parameters or derive them from simulation of single segments. Long et al. [58] suggests to calculate the model parameters on a per segment basis, i.e. the self-inductance with a mean distance approximation for rectangular cross-sections and the capacitors and resistors via simple expressions. Such formulas are given by Yue et al. [59], [60]. However, for the mutual inductance this would involve a sum with complexity $\mathcal{O}(n^2)$, or adequate approximations. This could of course be solved by writing a kind of custom "field solver" that computes all mutual inductances numerically with the Neumann integral, but this is no feasible solution.

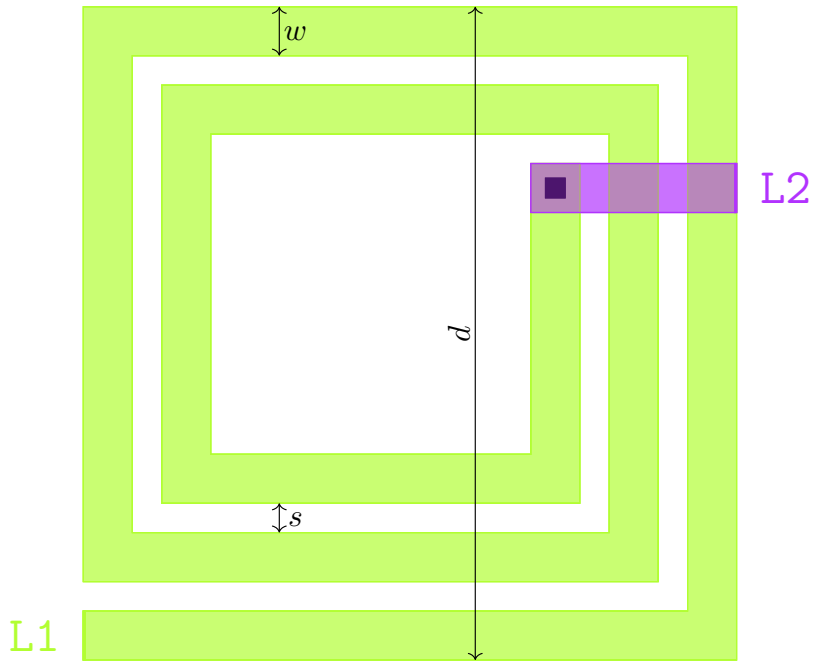



**Fig. 3.13:** Layout of the 2-terminal square spiral inductor analyzed in this section ($n = 2.5$ turns; fill shapes are not shown; the purple layer is on top of the green layer; vias are black).

### 3.3.2 Lumped Circuit Models

As an alternative to the segmented circuit models, so-called *lumped circuit models* are widely used, e.g. in [61], [62]. They are simpler and their structure is fixed, so they are independent of the inductor geometry. The contributions of different segments of the inductor are *lumped* into only a few elements covering the first order properties. While this is not as accurate compared to segmented models or even S-parameters, it is this simplicity that makes them feasible for circuit design and optimization. These models are usually accurate up to the self-resonant frequency of the inductor, which is the range of interest for on-chip inductors [40]. What is important to understand is that for verification before tape-out via post-layout simulations, it is perfectly adequate to use S-parameters. However, for design and fast system simulation, it is important to map the complex properties described with the S-parameters to a first order model. This is also why segmented circuit models have no real use case; they are less precise than S-parameters and too complicated to design with. For this reason, the focus of this work lies on the lumped circuit models, sometimes also called *equivalent models*. Fig. 3.14 shows the widely used $\pi$-model for two-terminal on-chip inductors. The properties of the inductor are modeled with a series inductance $L_s$ and series resistance $R_s$, as well as the inter winding capacitance $C_s$. The influence of oxide and substrate is taken into account with the branches to ground.

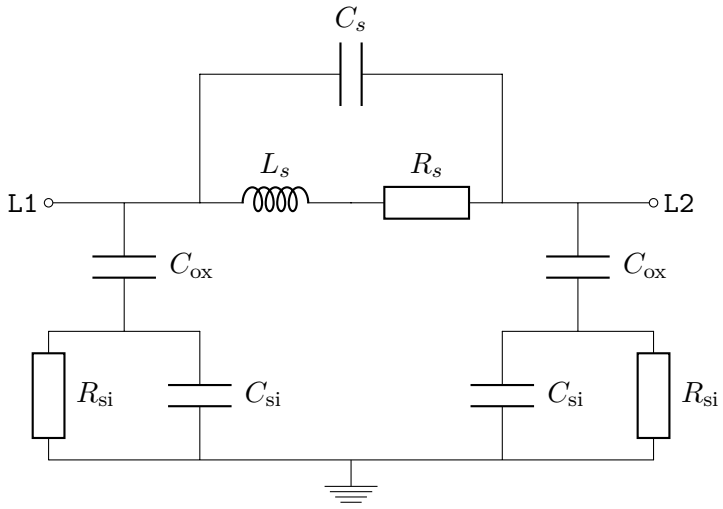



**Fig. 3.14:** The lumped $\pi$-model of a spiral inductor as found in literature. Parameters are the series inductance $L_s$, series resistance $R_s$, inter winding capacitance $C_s$, oxide capacitance $C_{\text{ox}}$, and the substrate properties $C_{\text{si}}$ and $R_{\text{si}}$.

#### Series Inductance – Current Sheet Method

Compared to the resistors and capacitors in the $\pi$-model, it is very difficult to compute the series inductance because an appropriate formula has to take the mutual inductance between all the segments into account. This is technically also true for the inter-winding capacitance, but it is often considered to have its main contribution from the metal under/over-pass, which is then simply computed as a plate capacitor [59]. In modern technologies, the wire thickness is not negligible anymore compared to the width, so technically lateral capacitive coupling between the segments will have a contribution to $C_s$. There have been multiple equations proposed in literature to calculate the inductance of planar inductors. However, to the knowledge of the author, the work of Mohan [40] is the most extensive in this area. He compares other formulas to his own derivations so there is no point in comparing them again in this work. The main proposal of Mohan is the *current sheet method*, which is based on the mean distance method explained in the previous section. The term *current sheet* implies a conductor with one-dimensional cross-section. This work builds upon the results of Mohan in section 3.5, hence an overview of his derivation is given here. For a simple planar spiral, his result is given by Eqs. 3.56 to 3.58 and Tab. 3.2. The permeability of the material is included via $\mu$. On-chip interconnect is typically made of copper, which has a relative permeability of 1.

$$L_s = \frac{d_{\text{avg}} n^2 \mu}{2} \cdot c_1 \cdot \left( \ln\left(\frac{c_2}{\rho}\right) + c_3\rho + c_4\rho^2 \right) \tag{3.56}$$

$$\rho = \frac{nw + (n-1)s}{d_{\text{avg}}} \tag{3.57}$$

$$d_{\text{avg}} = d - nw - (n-1)s \tag{3.58}$$

| Layout | $c_1$ | $c_2$ | $c_3$ | $c_4$ |
|---|---|---|---|---|
| Square | 1.27 | 2.07 | 0.18 | 0.13 |
| Hexagon | 1.09 | 2.23 | 0.00 | 0.17 |
| Octagon | 1.07 | 2.29 | 0.00 | 0.19 |
| Circle | 1.00 | 2.46 | 0.00 | 0.20 |

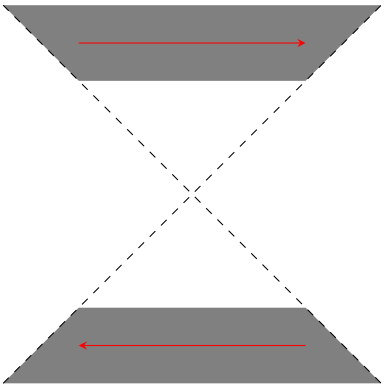
**Tab. 3.2:** Coefficients for the current sheet self-inductance formula given in Eq. 3.56 [40].

With the background from section 3.2, the general structure of the expression looks very familiar. Nevertheless, the following will briefly present the steps performed by Mohan to derive this formula. His derivation is the continuation of the inductance calculations made in the previous section – or as Greenhouse states "*All theoretical equations for calculations involving planar rectangular inductors having one or more turns employ in their derivation the self-inductance of a straight conductor.*" [56]. The steps of the derivation are illustrated in Fig. 3.15. As a first step, Mohan calculates the GMD, AMD, and AMSD for a line cross-section, i.e. a rectangular cross-section with $t = 0$. From this, the self-inductance of a rectangular current sheet is deduced. This is then extended to two parallel current sheets with opposite current directions. The same is done for trapezoidal current sheets because the sides of planar inductors are trapezoidal. The results are then combined to compute the self-inductance of a square current sheet. The last step makes the transition from a single square current sheet to $n$ concentric current sheets. As this would involve the computation of $\mathcal{O}(n^2)$ mutual inductances, Mohan proposes an approximation by defining the ratio of width to length of a conductor $\rho$ and inserting it for $w/l$ in the formula derived in the previous step for the square current sheet. This is of course only exactly valid for $s = 0$, but Mohan validates this approximation by comparison to a summation method from Greenhouse [56], which actually sums up all mutual inductances with a granularity of segments. Furthermore, he computes some corrections for finite spacing and thickness. These terms are presented in section 3.5.

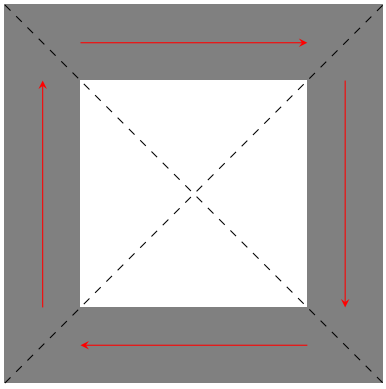
**(a)** Rectangular current sheet.

**(b)** Rectangular current sheets with opposite current directions.

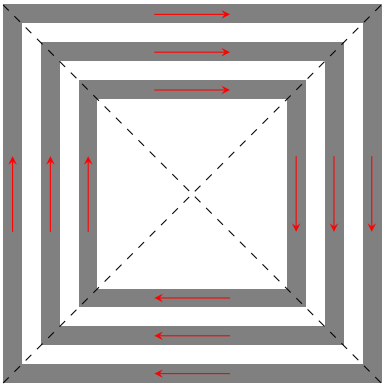
**(c)** Trapezoidal current sheet.



**(d)** Trapezoidal current sheets with opposite current directions.



**(e)** Square current sheet.



**(f)** $n$ concentric, square current sheets.

**Fig. 3.15:** Steps taken by Mohan [40] to derive the current sheet approximation. The red arrows denote the current direction.

### Lumped Parameters

To analyze the effects of metal fill and process corners, the inductor in Fig. 3.13 has been simulated for various settings. Therefore, it is useful to calculate the Y-parameters for this network because they allow to connect simulated or measured S-parameters with the model parameters. The Y-parameter matrix of the lumped $\pi$-model consists of two components, a series admittance $Y_s$ comprised of the three series elements between `L1` and `L2`, and a shunt admittance $Y_{\mathrm{si}}$ dependent on the three components used to model the oxide and substrate. With this, the Y-parameter matrix is given by Eq. 3.61.

$$Y_s = sC_s + \frac{1}{R_s + sL_s} \tag{3.59}$$

$$Y_{\mathrm{si}} = \frac{sC_{\mathrm{ox}} + s^2 C_{\mathrm{ox}} R_{\mathrm{si}} C_{\mathrm{si}}}{1 + sR_{\mathrm{si}}(C_{\mathrm{ox}} + C_{\mathrm{si}})} \tag{3.60}$$

$$\mathbb{Y}_\pi = \begin{pmatrix} Y_s + Y_{\mathrm{si}} & -Y_s \\ -Y_s & Y_s + Y_{\mathrm{si}} \end{pmatrix} \tag{3.61}$$

There are only two independent Y-parameter components, although with a real and imaginary part each but six parameters in the model, so it is not possible to solve for all parameters and plot them as a function of frequency. However, series and shunt admittance can be used to compute four parameters at low frequencies. At higher frequencies, the capacitances $C_s$ and $C_{\mathrm{si}}$ influence these values, as well as not modeled frequency dependent effects like the skin effect. To visualize the influence of different effects, some of the Eqs. 3.62 to 3.65 are plotted in literature. Since the shunt branches are actually not identical for an asymmetric inductor, $R_{\mathrm{si}}^{\times}$ and $C_{\mathrm{ox}}^{\times}$ are averaged here as the lumped $\pi$-model employs identical branches. Note that Eq. 3.63 is, unlike the other three expressions, not identical to $L_s$ at low frequencies but also includes $C_s R_s^2$. However, this value is typically below 1 pH and can be neglected.

$$R_s^{\times}(f) := -\mathrm{Re}\left\{\frac{1}{Y_{\pi,12}}\right\} = R_s + \mathcal{O}(f^2) \tag{3.62}$$

$$L_s^{\times}(f) := -\mathrm{Im}\left\{\frac{1}{\omega Y_{\pi,12}}\right\} = L_s - C_s R_s^2 + \mathcal{O}(f^2) \tag{3.63}$$

$$R_{\mathrm{si}}^{\times}(f) := \mathrm{Re}\left\{\frac{2}{Y_{\pi,11} + Y_{\pi,12} + Y_{\pi,21} + Y_{\pi,22}}\right\} = R_{\mathrm{si}} + \mathcal{O}(f^2) \tag{3.64}$$

$$C_{\mathrm{ox}}^{\times}(f) := \mathrm{Im}\left\{\frac{Y_{\pi,11} + Y_{\pi,12} + Y_{\pi,21} + Y_{\pi,22}}{2\omega}\right\} = C_{\mathrm{ox}} + \mathcal{O}(f^2) \tag{3.65}$$

In addition to the four curves above, often also the single-ended inductance $L^{\times}$ and the quality factor $Q$ are computed and plotted against the frequency. By plotting these, effects of design changes on the performance in typical applications can be understood relatively easy. For, e.g., oscillators it is relevant to achieve high quality factors to reduce

the signal damping. For other applications, like amplifiers, it might be more important to obtain a certain inductance $L^{\times}$ to achieve a desired bandwidth improvement. $L_s^{\times}$ is of less importance in this case because usually one terminal of the inductor is connected to either power or ground. Thus, the corresponding $Y_{\mathrm{si}}$ branch has no effect and at the other pin, the inductor behaves like an inductance $L^{\times}$.

$$L^{\times}(f) := \mathrm{Im}\left\{\frac{2}{\omega Y_{\pi,11} + \omega Y_{\pi,22}}\right\} \tag{3.66}$$

$$Q(f) := -\frac{\mathrm{Im}\left\{Y_{\pi,11} + Y_{\pi,22}\right\}}{\mathrm{Re}\left\{Y_{\pi,11} + Y_{\pi,22}\right\}} \tag{3.67}$$

### 3.3.3 Skin and Proximity Effect

It is well known that alternating currents follow an exponentially decreasing density within conductor cross-sections, which is called *skin effect*. This influences both the resistance and the inductance of a wire and makes them frequency dependent. In case of a constant overall current through the cross-section, the external inductance remains unchanged. However, the scaling with $r^2/R^2$ done to compute the internal magnetic field in Eq. 3.14 would now include an exponential factor, rendering an analytic solution more complicated. Nevertheless, it can be concluded that the internal inductance will decrease as for a given radius $r < R$ the enclosed current is less than for a uniform DC current. Visually, this would cause the graphs in Fig 3.9 to become dented towards lower values in the range $[0, 1]$. Since only the internal inductance is reduced by the skin effect, the overall inductance does not decrease significantly at higher frequencies. This can be seen in Fig. 3.16, which shows the ratio of both inductance parts as a function of wire length versus radius. The length of the wiring used in typical on-chip inductors is much larger than the width or thickness of it so the external inductance is the dominant contribution. Probably for this reason, literature does not spend much effort on modifying inductance calculation due to skin effect. In contrast to the inductance, the series resistance is not comprised of a "field" outside the conductor and is affected significantly by the skin effect. The current density for a round wire is approximately described by Eq. 3.68. The correct solution involves Bessel functions in cylindrical coordinates and can be found in [63]. It is possible to calculate the current density for a rectangular cross-section, as is done by Gerling [64], however the solution is rather complicated and for the purpose of this work, a simpler description is sufficient to visualize the general impact of the skin effect on the series resistance of on-chip inductors.

$$J(r) = J_S \exp\left(-\frac{R-r}{\delta}\right) \text{ and } \delta \approx \sqrt{\frac{2}{\sigma_w \mu_w \omega}} \tag{3.68}$$

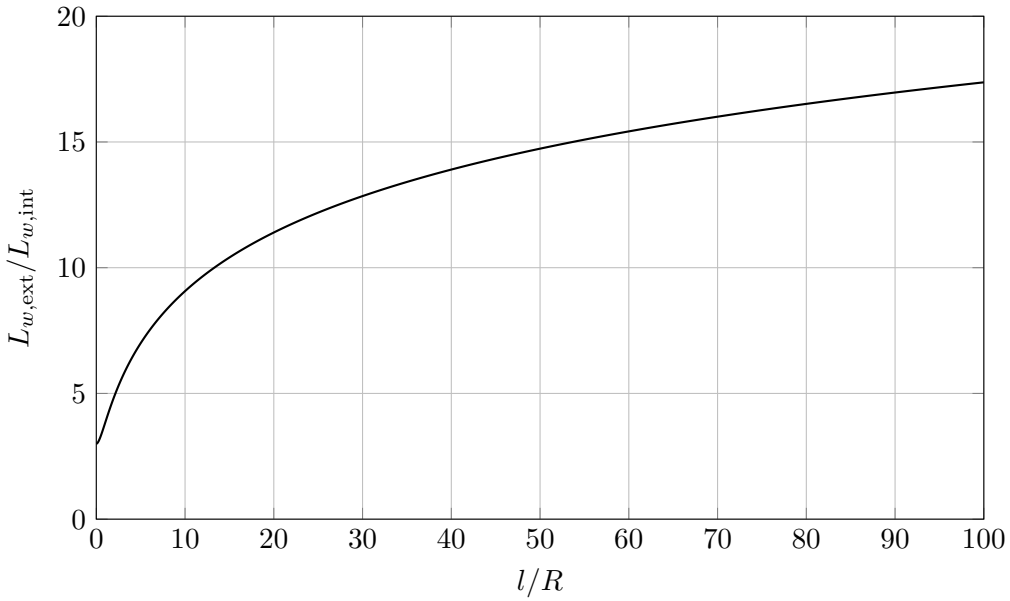



**Fig. 3.16:** Ratio of external and internal inductance of a straight round wire as calculated in Eq. 3.26 for a uniform current.

Here, $J_S$ is the current density at the wire surface, which is identical to the DC current density as it is not affected by the *skin depth* $\delta$. The latter depends solely on the material properties of the wire, i.e. the specific conductivity $\sigma_w$ and the permeability $\mu_w$, and the frequency, but not on the geometry of the wire. Thus, a bunch of thinner wires is usually favored over a single thicker wire in many applications to mitigate the skin effect. This is not a feasible approach for on-chip inductors because spacing rules are increasing the effective width of the turns significantly. Using Eq. 3.68, the simple case of a round wire allows to calculate the ratio of AC to DC resistance.

$$\frac{R(\omega)}{R(0)} = \frac{J_S \pi R^2}{2\pi \int_0^R J(r) r \mathrm{d}r} = \frac{R^2}{2\delta(R - \delta(1 - \exp(-R/\delta)))} \tag{3.69}$$

Fig. 3.17 shows the skin depth for copper and it is well below 1 µm for frequencies above 5 GHz. The red curve shows the resistance increase due to skin effect in a round copper wire with a radius of 1.4 µm. The increase in resistance is not negligible as compared to the decrease in inductance.

While the skin effect alters the current distribution within a wire due to the interaction between the current filaments, a similar effect arises when several wires are closely together, called *proximity effect*. The current in one wire changes the distribution of current in the other wire, and vice versa. Depending on the relative current direction in both wires, the current density is either increased or decreased toward the common center of the wires. The calculations behind this effect are complicated and beyond the

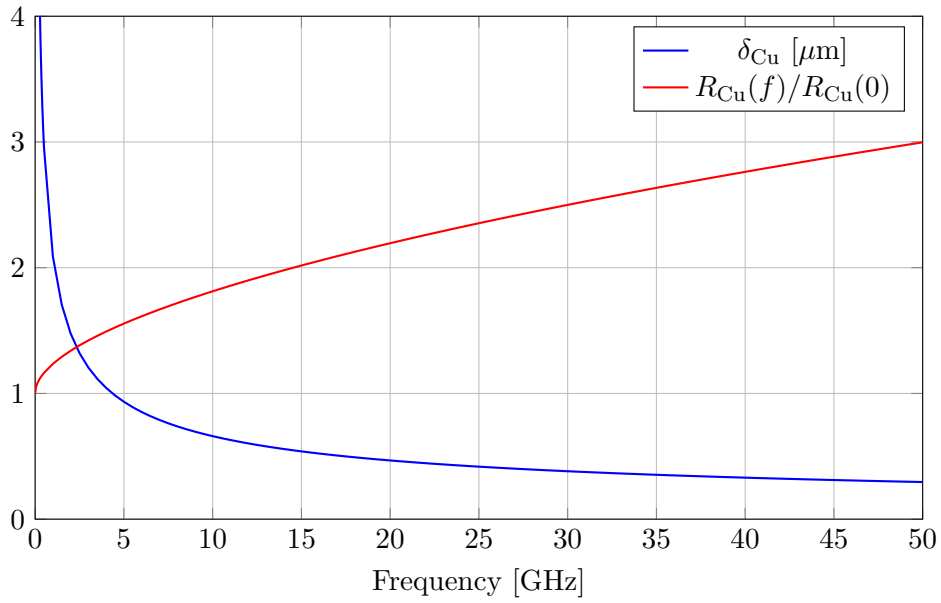



**Fig. 3.17:** Skin depth and resistance increase for a round copper wire with a radius of 1.4 µm. This is representative of a typical wire used for on-chip inductors in advanced nodes.

scope of this work. A detailed analysis is even more involved than the exact skin effect equations and hardly applicable to more complex geometries. The important observation is that the result also describes an increase in series resistance, thus it can be treated similar to the skin effect. The details will be computed by field solvers and can hardly be mapped to lumped circuit models.

### Frequency-Dependent Lumped Models

The question remains on what to do with the frequency dependency of resistance and inductance in lumped circuit models. Making lumped components frequency dependent, severely compromises their simplicity and removes the advantage they provide for circuit design and optimization. Thus, it is not desirable to extend lumped models with a frequency parameter. Instead, fitting can be applied to determine the optimal model parameters resulting in the minimum deviation from the actual S-parameters in the frequency range of interest. This effectively averages the parameters but only over the frequency dependent effects that could not be represented in the lumped model. For the series resistance, the DC value could also be a reasonable approximation because it allows to minimize $S_{11}(0)$, which has the tightest constraint across standards according to Fig. 2.3. Another advantage of the DC resistance value is the computation of DC operating points, which are calculated in most analog circuit simulations before a small signal analysis is performed.

### 3.3.4 Metal Fill

Metal fill is another very important aspect to ensure chip designs can actually be manufactured, and thus it also influences the design of on-chip inductors. The underlying problem concerns density gradients on each process layer and therefore also on the metal layers. The manufacturing steps include the etching of structures using chemicals as well as mechanical polishing steps. Areas with very low density can then become slightly dished, which causes structures, which are to be exposed in subsequent process steps in the same area, to be out of focus. Hence, the yield may be decreased, or it may not be possible to manufacture the design at all. Thus, small fill shapes are scattered in between the shapes drawn by the designer to smooth out the density and meet the design rules. Usually this is done by an automated tool and not manually. This section has presented how a two-terminal inductor is modeled and in this subsection this model is used to study the influence of metal fill on the model parameters.

#### Design of Metal Fill

In the 22nm technology considered in this work, special design rules regarding the metal fill of inductors are provided. Using a marker layer, these rules can be applied to a certain area in the design, which in turn is prohibited to include more than metal interconnect. The fill shapes are allowed to be very small and the overall density to be minimal. It is obvious from the law of induction that reducing the area of the shapes in turn reduces the induced electric field and thus the dissipated energy. The sizes of the fill shapes can be looked up in the design manual and incorporated into the PCells to have them automatically generated. It is not advisable to hand this task to a fill tool as it is better to maintain complete control of the layout within the inductor area and include fill early on into simulations. In addition, it is easy to incorporate the the generation of the fill patterns into PCells so there is not reason to not do so.

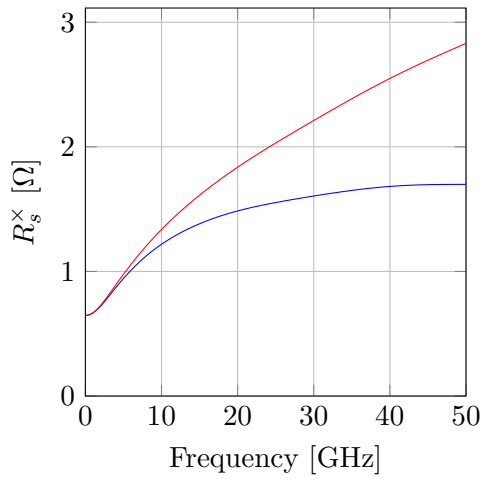
#### Simulation of Metal Fill

Field solvers usually create a mesh to partition the device to be simulated into smaller parts for which a single point of electric and magnetic field strengths is calculated. The smaller the mesh cells, the higher the accuracy of the result, and the longer the runtime of the simulator. For straight wires, e.g. segments of inductors, it is possible to reduce the number of grid points to enable reasonable runtimes. However, the number of mesh cells needed for tens of thousands of fill shapes cannot be reduced below the number of fill shapes as the fill shapes are not a continuous piece of metal. Hence, simplifications have to be employed to include metal fill with these simulators. The SKILL PCell code developed in this work uses the `fill` parameter to not only generate the fill for manufacturing but

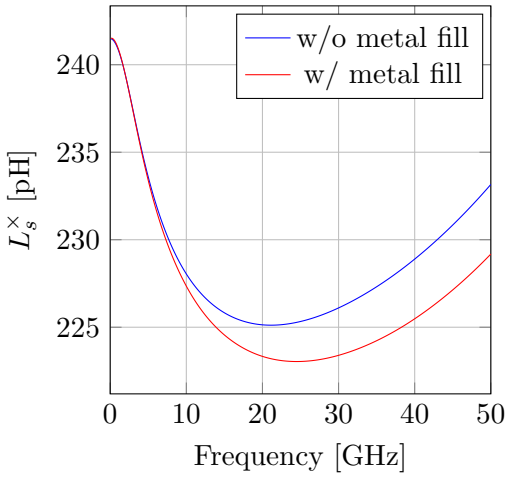
also to up-scale this fill to drastically reduce the number of fill shapes for field solvers. This was not necessary for EMX (see section 3.4), a field solver specifically designed for on-chip structures. Layout extraction engines can be configured to simplify small shapes if required, so the scaling of fill is technically not necessary in this case.

#### Influence of Metal Fill

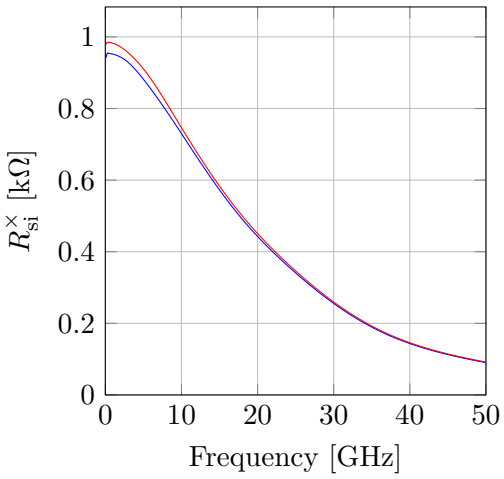
Fig. 3.18 shows the results produced with EMX (full-wave mode) for the inductor shown in Fig. 3.13. Following the law of induction, a changing magnetic field will cause current loops in nearby conductive objects. These *eddy* currents are dissipating energy from the magnetic field, which is primarily reflected in an increased $R_s^\times$. The series inductance $L_s^\times$ is changed only by a few pH. The initial drop is caused by the skin effect, while the rise towards higher frequencies is the result of $C_s$. The substrate resistance is at a plausible value [61] and not altered much by the metal shapes. It decreases towards higher frequencies due to $C_{\mathrm{si}}$. The oxide capacitance is slightly increased due to the coupling to the fill shapes. Towards higher frequencies, it converges to the series connection of oxide and substrate capacitance: $C_{\mathrm{ox}}C_{\mathrm{si}}/(C_{\mathrm{ox}}+C_{\mathrm{si}})$. The single-ended inductance $L^\times$ increases earlier than the series inductance as it involves a higher capacitance. The quality factor is reduced significantly because of the energy loss through eddy currents in the fill shapes.



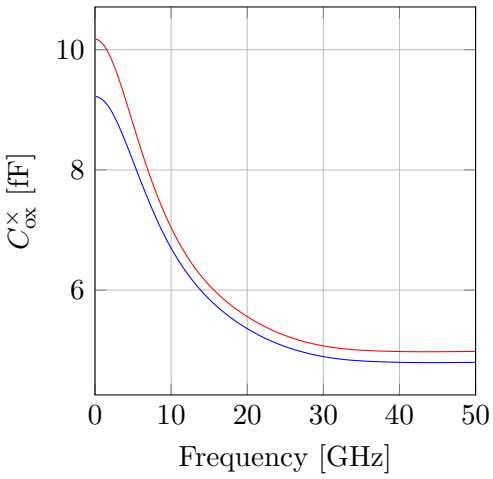

**(a)** Series resistance.



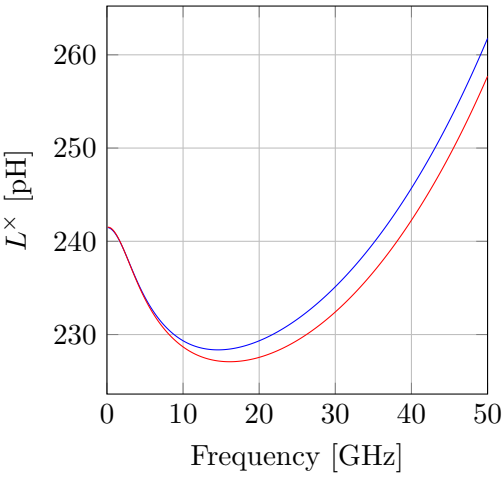

**(b)** Series inductance.



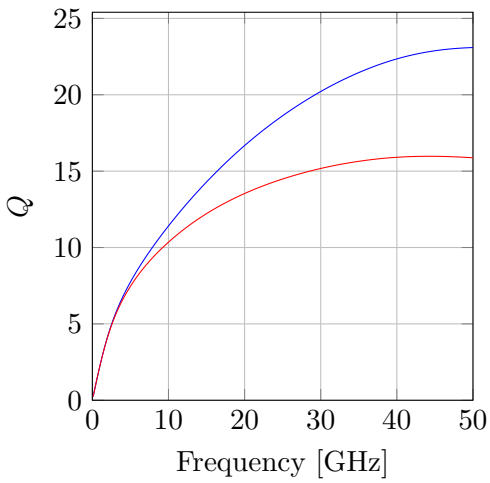

**(c)** Shunt resistance.




**(d)** Shunt capacitance.




**(e)** Single-ended inductance.



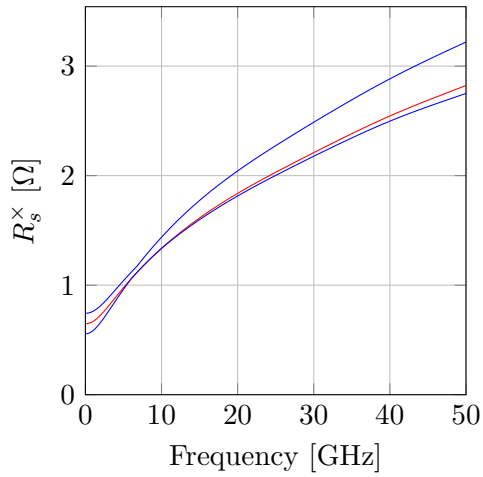

**(f)** Quality factor.

**Fig. 3.18:** Influence of metal fill.

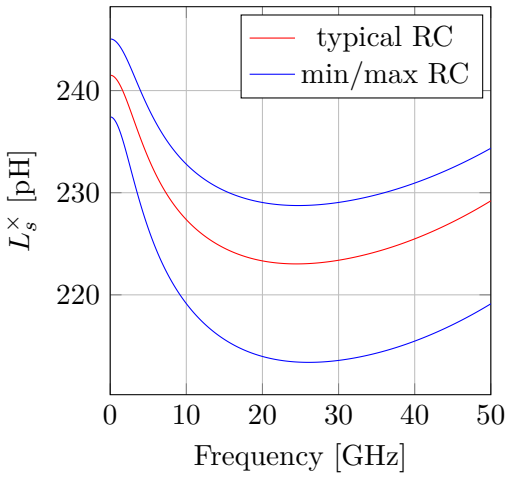
### 3.3.5 Process Corners

Process corners are used to model the effects of manufacturing variations on devices and wires. As inductors are basically just wires and fill, only the interconnect corners (RC-corners) are required to simulate the variation of their electrical properties. Temperature, voltage, and doping concentrations are usually not considered because their influence is minimal. For each RC-corner, there is a stack-up file that contains the specific changes to geometry and material properties. These files are provided in multiple formats for different tools (see section 3.4). The two-terminal inductor used as an example in this section (Fig. 3.13) has been simulated with EMX (full-wave mode) for each RC-corner, including metal fill. The results are shown in Fig. 3.19 and serve to get an estimate of the magnitude of variation. The min/max-values are not singular process corners but simply the highest and lowest value across all corners at each frequency. Thus, they present the absolute highest deviations at all frequencies, but this is enough to get an idea of the order of magnitude by which a typical inductor may vary across chips. The deviation of the inductance is below ten percent, while the resistance varies a little more, especially towards higher values, which is probably due to the influence of metal fill. As a result, the quality factor also exhibits a larger deviation towards lower values. The consequence from this is that, if it is possible to simulate the design for different RC-corners, which is not always the case as will be discussed in the next section, it should be done. However, it seems unrealistic to find an inductor for a given application and then alter its geometry such that the electrical performance stays the same but the variation over RC-corners is made more favorable. Hence, the corner analysis is basically just used for verification of on-chip inductors but not taken into account in the design phase.

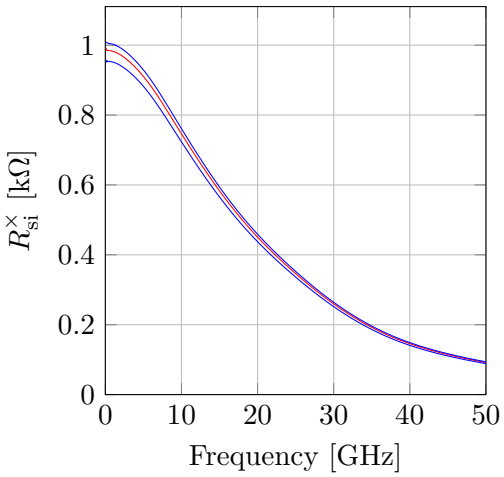
Now the series inductance $L_s$ was simulated for the example inductor and can be compared with the current sheet formula, which estimates an inductance of 279 pH. Including spacing and thickness corrections, the value lowers to 271 pH. Overall, this is a very good result, especially considering that no complicated tools, no expensive licenses, and only a couple of minutes are needed to do the calculation. However, there seems to be room for improvement, so an attempt on reducing this error with the help of field solver results is presented in section 3.5.



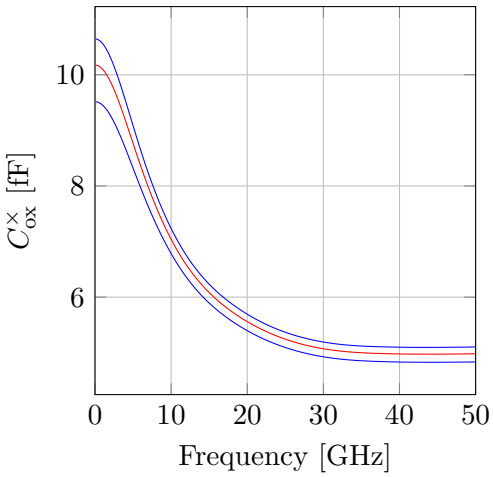

**(a)** Series resistance.



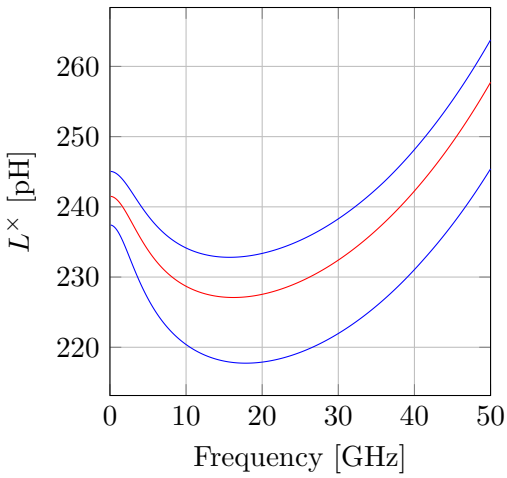

**(b)** Series inductance.



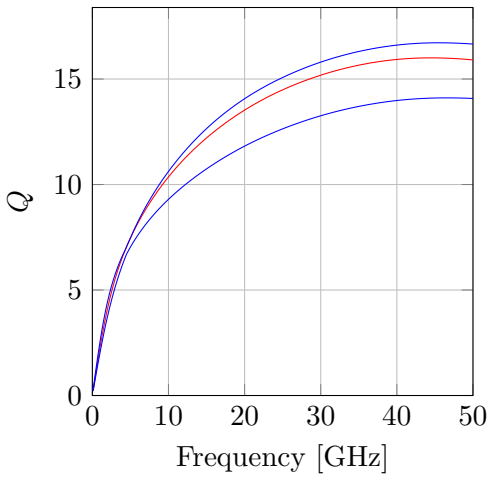

**(c)** Shunt resistance.




**(d)** Shunt capacitance.




**(e)** Single-ended inductance.




**(f)** Quality factor.

**Fig. 3.19:** Influence of process corners, including metal fill.

## 3.4 Simulation of Inductors

To complement the previous section, this section discusses how on-chip inductors can be simulated. Using the current sheet formula, which calculates the DC inductance, is not enough for post-layout verification. It is necessary to properly simulate the inductor layout to ensure it has the expected properties. This section is split into two parts. The first part focuses on the layout extraction engine Quantus QRC from Cadence. In the second part, field solvers are discussed, which have been used throughout the course of this work.

Before diving into the details of this section, it is important to adjust the expectations for it. While there are many tools, like Advanced Design System (ADS) from Keysight Technologies or Sigrity from Cadence, that are well suited to simulate PCBs, it is much more complicated to obtain reliable results for on-chip structures. Field simulations are *not* easily set up in a couple of hours and are *not* a one time procedure that is check marked in the verification plan. This is mainly due to two reasons: the availability of stack-up files, and the lack of the correct inductor layout to simulate. In chip design, the Back End of Line (BEOL), i.e. the metal layers used to interconnect the devices, is described with a *stack-up*, usually defined in a so-called *technology file*. It contains the material properties and thicknesses of the layers available in the corresponding technology, as well as their ordering and additional information. Since these files contain explicit information about the technology, they are often encrypted, especially for advanced semiconductor processes, and can only be read by the appropriate tool. Furthermore, there is a significant amount of different formats available. The "qrcTechFile" is used by Cadence tools to read in information for layout extraction – it is encrypted in all technologies this work dealt with. There is also the interconnect technology (.ict) format, which is an ASCII-based text file with a syntax to describe semiconductor technologies and also holds some information for Cadence tools. It can be viewed with a simple text editor or the Cadence tool ViewICT, which translates it into a graphical representation. However, these files do not contain all stack-up information necessary to simulate inductors. Therefore, foundries sometimes provide separate files for field solvers that are actually intended for this use case. More details on these files are presented in this section.

Even if an inductor layout has been correctly simulated, this does of course not imply that it has the intended properties, nor that it is an optimal solution given the large space of possible inductors. In the likely case it does not have the intended properties, it has to be optimized in an iterative and time-consuming *trial and error* process. This problem is addressed partly in this section and extensively in the next section.

### 3.4.1 Layout Extraction

So far, this chapter has presented the background on analytical inductance calculation. However, this is hardly the approach someone starting out with on-chip inductor design would be looking into right from the start, without background knowledge and experience. Literature on ESD compensation and the large number of seemingly available tools hint strongly towards simulation as the way to go. The same was the case for this work, and the most straight forward design process is to simply draw an inductor in the layout editor and then try to simulate it. Hence, the initial attempts to create a fitting inductor for ESD compensation were inevitably based on trial and error. Consequently, the layout procedure was automated as described in section 3.1, but still a lot more steps had to be performed manually to assess the electrical parameters via simulation. The amount of manual work required per inductor was in fact too large to realistically find a satisfying solution within a reasonable time frame. Furthermore, trial and error approaches always have that uncertainty about how long it takes to find a solution and if one is found, there is a bitter taste associated with it because there is little indication how close the result is to an optimal solution. This led to the idea to take the automation even further in this work, with the goal of specifying only the geometrical parameters as an input and obtaining the electrical parameters as a result. To this end, an automated flow using the Quantus QRC extraction engine from Cadence was implemented. The results of this work were presented at CDNLive EMEA 2018 [16]. This section will detail the flow up to the generation of S-parameters. The transformation of S-parameters into electrical parameters is presented in the next section. Fig. 3.20 shows an overview of the entire flow. The process of creating and extracting an inductor involves a lot of steps. Starting with the geometrical parameters, it requires to create a layout view, run the LVS check, extract the layout with QRC, and generate the S-parameters with a simulator. Manual execution of these steps for all inductors consumes a lot of time and is hardly conducted consistently. The flow is controlled by a bash script that accepts the required input via command line parameters. An example command line would look like this:

```
./inductor.bash --process cmos22 --inductor IND_SQUARE_SPIRAL \
--diameter 40.0u --width 3.0u --spacing 1.8u --turns 2.5 --layer MX
```

Since layer names change from technology to technology, some one-time work is needed to extend the flow. If new inductors are to be supported, new PCells have to be created. Virtuoso executes a SKILL file called `libInit.il` when entering a library in the library manager if it is present in the library folder on the file system. So the SKILL PCells that are intended to be located in this library can be loaded in this file via the SKILL load function, `(load <pcell.il>)`. This ensures the OpenAccess PCells in the library are always in sync with the code but causes some delay in accessing the library. Therefore, loading the PCells should probably only be done when the code is updated. It is most

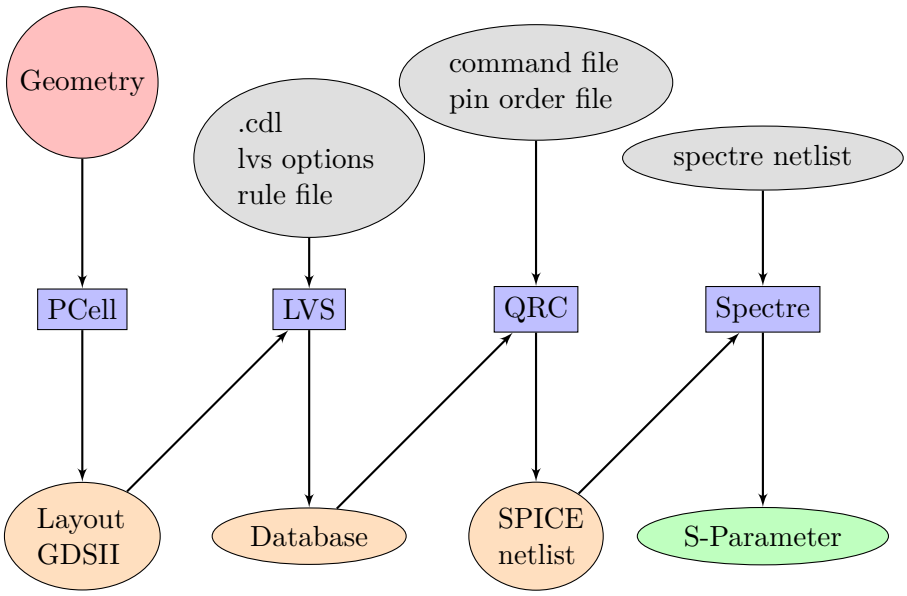



**Fig. 3.20:** Block diagram of the automated extraction flow. A command line with geometry parameters is translated into S-parameters.

feasible to include the generated OpenAccess PCells into version control, too.

The user specifies the inductor geometry via parameters to the flow script. Then the first step executed by the flow is to create a cell containing a layout view of the inductor corresponding to the geometry specified on the command line. To do this, a short code is generated with Python containing SKILL commands that can be executed without the Virtuoso GUI by passing it to Virtuoso with the `-replay` flag. This functionality is intended to replay a series of commands that has been executed in the Virtuoso GUI. The SKILL code opens the layout view, places a PCell instance with the given parameters, and saves it afterwards. In the next step, the `strmout` command is used to convert the layout view into the GDSII-format to hand it over to the `cdnspvs` tool, which conducts an LVS check required for layout extraction. However, `cdnspvs` needs more input files. Obviously, a schematic is required; it can be provided by a text-based .cdl-file, which is easier to create than an OpenAccess schematic view of the inductor cell. The .cdl-file contains metal resistors to logically separate the nets used by the inductor. These resistors are needed for the LVS check to correctly identify the inductor but do not have any physical relevance. The final two files to run `cdnspvs` are an options and a rule file containing settings normally supplied via the GUI – most importantly the layout and schematic to be compared. `cdnspvs` is advised to create the database required by QRC with the `-qrc_data` flag. The extraction tool `qrc` is again controlled with a command file, which contains the settings that are otherwise specified in the GUI. As the name suggests, extraction is normally performed to include post-layout parasitics in the form of resistance (R) and capacitance (C). However, it is possible to let QRC extract

inductance (L) of and coupling (k) between nets when it is executed in "RLCk" mode. Capacitance extraction is set to coupled to properly account for capacitances between neighboring shapes. The accuracy of capacitance extraction is enhanced by using the Quantus 3D Field Solver (Quantus FS). It computes capacitances with the actual design geometry, instead of using simpler capacitance models defined in the technology file. The runtime is not significantly longer, even at the high accuracy settings, as inductor layouts do not comprise a lot of shapes. However, this field solver is quite simple and does not provide the same feature for inductances. Furthermore, to extract the "Lk" correctly, a region, metal layers, and net names have to be supplied, and only the nets meeting these criteria have their inductance and inductive coupling extracted by QRC. The output format for the extracted netlist is SPICE. Along with the output format, it is necessary to specify the `-pin_order_file` option. It is provided with yet another file that defines the order of the pins in the SPICE netlist; otherwise this order can vary, which causes problems in the remainder of the flow. The S-parameters are simulated by calling the circuit simulator Spectre. The testbench is described in the .scs-format (Spectre Circuit Simulation Netlist) and contains three S-parameter sources with $50\,\Omega$ reference impedance, the extracted netlist, and simulation settings, including an output path for the S-parameters. The last step, the transformation of the S-parameters to lumped circuit parameters, is described in section 3.5.

The main drawback of this flow is that QRC does perform certain simplifications during inductance extraction as it is not a field solver. This has been tested with a PDK inductor, which a foundry provided model was available for, in order to rule out potential simulation errors as a reliable field solver was not available at this time. However, the PDK inductor also uses some special design rules, which the extraction might not work very well for. QRC matches segments of the layout to known structures to annotate parasitics, which might produce better results for the custom inductors presented in this work. Although, Cadence would not spend effort on integrating field solvers into its Virtuoso environment if QRC could already properly handle inductance extraction accurately at all frequencies.

### 3.4.2 Field Solvers

Field solvers are the most accurate way to simulate inductors and distributed electrical structures in general. They numerically solve the Maxwell equations for given boundary conditions, e.g. for the geometry of an inductor. Several tools exist to do this, but there are technical and licensing barriers to overcome to simulate on-chip inductors. "*Furthermore, [...] considerable experience is required on the part of the user to simulate on-chip inductors*" [40]. Field solvers have a variety of flags for tweaking them, and it requires time to get an overview and analyze their effect on the results. The work

with some tools is described in this subsection. This is not intended as a comprehensive overview and there are many more tools available, which could not be evaluated due to licensing costs and the sheer amount of effort required to do so. The tools used in this work are ADS 2012 from Keysight Technologies, and Sigrity and EMX from Cadence.

The analog chip design environment used in this work was Cadence Virtuoso, so particular attention was paid to tools from Cadence that enable inductor simulation. Over the last years, there has been increasing effort from Cadence to incorporate various field solvers into the Virtuoso environment. Keysight Technologies offers a tool called GoldenGate, which integrates with Virtuoso and allows to simulate on-chip inductors. It has been only briefly tested at the beginning of this work, but it requires a separate license. Unfortunately, no noteworthy results have been documented from this. Then Cadence integrated their in-house field solver Sigrity into the Virtuoso layout editor, which was primarily marketed for PCB and package simulation before. This integration has been tested in this work with the help of a "beta" evaluation license, with mediocre success. When the evaluation license ran out, a custom stack-up file was created for use with ADS 2012, again with mediocre success. In 2020, Cadence bought Integrand Software Inc., a company offering the field solver EMX, which was previously not available for universities but mainly for foundries. Hence, few months prior to the end of this work, EMX could also be evaluated.

A field solver usually needs two different inputs to simulate a device, materials and geometry. The former defines which electromagnetic properties, i.e. conductivity, permittivity, and permeability, the materials used to build and enclose the device have. The latter describes the shape of the device and its environment. Foundries tend to provide stack-up files that include this information for various tools. Unfortunately, there is no standardized format available so the correct tool has to be purchased to utilize the available files. While it is sometimes possible to manually convert or create such files, as has been done in this work, it is strongly advisable to select the tool depending on the files available and not the other way around.

#### Sigrity

The company Sigrity was acquired by Cadence in 2012 [65]. Since then, the Sigrity software was mainly distributed with PCB and package design tools offered by Cadence. However, it was integrated into the "Electromagnetic" workspace of the Virtuoso layout editor in 2018, to enable EM-simulation of passive on-chip components. In the context of this thesis, an evaluation license could be used to do some tests of this integration while it was still in an early stage. Thus, this is no description of the actual software that can be licensed from Cadence today, but only intends to motivate the decisions taken in this work. The plug-in allows to read in the "qrcTechFile" to load the stack-up

information. In addition, the locations of the ports have to be specified. However, the ports could only be properly created via the automatic placement and if it did not work, the manual alternative was not functional. In case the configuration was done successfully, it could then be saved into the binary .clf-format. Unfortunately, the creation of this configuration file could not be automated, e.g. by using SKILL, due to the necessary graphical interaction. Starting the external Sigrity software directly from the Virtuoso layout editor was also not yet functional, but the .clf-file could be imported into Sigrity manually. However, the "Conformal Outer Box" setting, which is important for specifying the boundary conditions correctly in Sigrity, was quietly deselected during the import process but could be re-selected manually afterwards. The main difficulty with Sigrity was the import of the stack-up, which showed partly wrong material properties, especially for the vias. After changing these to "realistic" values, some inductors could be successfully simulated, meaning they showed reasonable results, but of course there was no measure on how accurate they were.

#### Advanced Design System

Another field simulator available for this work was Advanced Design System (ADS) from Keysight Technologies. It is usually used to simulate all kinds of different electromagnetic designs like PCBs or antennas. Therefore, it is theoretically also possible to simulate on-chip structures with ADS. It was considered for this work as the license for the Sigrity plug-in described before had expired. However, only an older version from 2012 was available. There are stack-up files in the 22nm PDK used in this work available for ADS but only for the "millimeter wave" stack-ups, which did not include the stack-up in use. Some technologies provide multiple stack-up variants for different application areas, but not all files are available for each variant. Furthermore, the foundries use recent versions of ADS to generate the stack-up files in an encrypted format, which in turn requires an up-to-date version of ADS to be decrypted. Thus, the stack-up could not be easily imported into the ADS version from 2012. In fact, trying to do so results in program crashes. However, the most important stack-up information could be extracted from the stack-up viewer in Sigrity, which displayed the decrypted data (at least partly). Hence, a new stack-up file could be built manually for ADS. For the text-based .ltd-format, it is necessary to specify the material properties of metal layers, dielectrics, and vias. The stack-up is then built by a sequence of statements that defines the order, thicknesses, and materials of the layers, as well as their layer number. The latter is needed in conjunction with a layer mapping file so that the layout in the GDSII-format, which has been exported from Virtuoso, can be imported into ADS. The ports are created manually in the ADS layout editor, but it does not provide a command line interface to script all these manual steps to facilitate an automated flow. The runtime of the simulations is in the order of

several hours to days, especially if fill shapes are included. This was one of the reasons to include scalable fill into the PCells described in section 3.1. Several settings have been compared to study their influence on the simulation results, and acceptable results could be achieved in this work. An "expert user" might have found more optimized settings, but this shows again that such tools require considerable experience. The simulation results shown in the previous section were produced with Cadence EMX, which is discussed next. The same two-terminal inductor has been simulated with ADS, though with a scaled-up metal fill and a lower conductor thickness due to the different stack-up. The results show an increased series resistance and inductance but decreased inter-winding capacitance, which is expected. This 22nm technology allows for an increased substrate resistance below inductors, which was not modeled into the manually created stack-up for ADS and thus, $R_{\mathrm{si}}$ was significantly lower. Last but not least, the oxide capacitance was very similar. It can be concluded that the manually created stack-up can produce results that look reasonable but for expensive tape-outs, it is worthwhile to invest in a tool that can actually read one of the supplied stack-up files in the given PDK.

#### EMX

Unlike the previously presented solvers, EMX (ElectroMagentic eXtraction) is specifically targeting semiconductor applications. While originally developed by Integrand Software Inc. and almost exclusively provided to semiconductor manufacturers, it was acquired by Cadence in 2020 and integrated into their layout editor. EMX seems to be the "golden" standard with regards to on-chip EM-simulation. Most foundry-provided inductor models are created with this software. Therefore, also the stack-up files (.proc) required by EMX are provided with some PDKs but again not necessarily for all stack-up variants. EMX was evaluated in the last few months of this work with a license provided by Cadence. Unfortunately, the .proc-files were not available for the stack-up that was used to implement the ESD protection. So the ADS results could not be directly validated with EMX. Cadence also offers Clarity, a massively parallel, fully 3D solver that makes use of the finite element method. It could also be used for on-chip structures, but it is more targeted towards system level simulation, e.g. connectors, boards, bond wires, and packages [66]. On-chip structures are mostly planar and their simulation can be significantly accelerated by solving the integral form of Maxwell's equations, which is called the *method of moments*. EMX makes use of the *fast multipole method*, which again increases the speed of the method of moments. This gives EMX a significant speed advantage versus full-fledged 3D solvers in the special case of on-chip inductors [65]. Furthermore, the accuracy of EMX has been "*extensively verified against silicon measurements*" [67] as it is used by all major foundries to model and characterize their passive components.

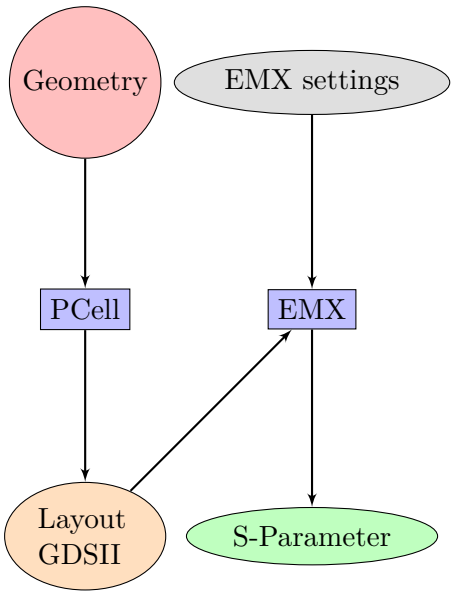
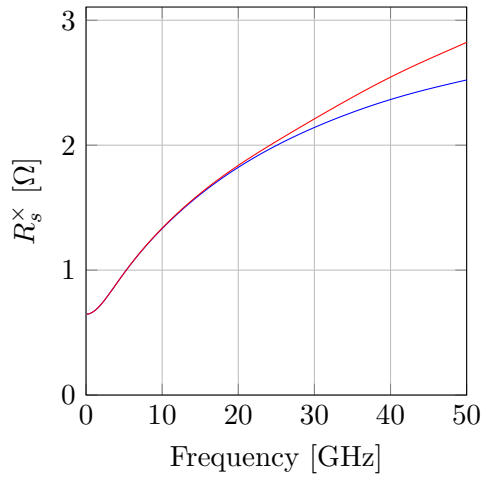
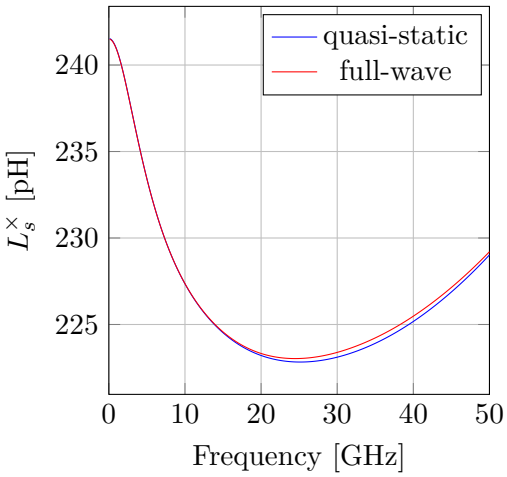



**Fig. 3.21:** Block diagram of the automated EMX-based flow.

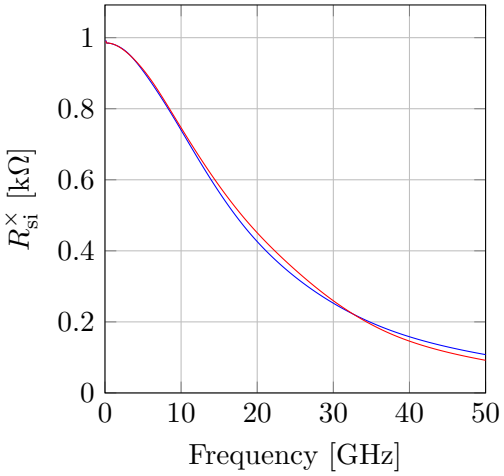
The automated layout extraction flow presented before was eventually discontinued due to the limited capability of QRC to extract inductance. In addition to that, the available field solvers could not be operated in a feasible manner via a command-line interface. Finally, the long simulation times of field solvers put the last nail into the coffin of this approach. However, the availability of EMX suddenly revived the idea again as it possesses all three aforementioned properties: a fast simulation (albeit without metal fill), a command-line interface, and EM-simulation capability. The usage of EMX is much simpler compared to ADS or Sigrity due to its more focused application area. It is invoked directly from the Virtuoso layout editor and recognizes pin labels as ports, and those are already generated by the PCells. Furthermore, a shell script containing the EMX command line is automatically generated when the simulator is started. Thus, the procedure of the QRC-based flow could be vastly simplified as the layout can be directly simulated (see Fig. 3.21). However, without the GUI it is still necessary to manually generate the GDSII file. The runtime for a single inductor is only one to two minutes depending on the complexity and size of the inductor. Quite a portion of this time is consumed to start Virtuoso in replay mode and check out licenses. When metal fill is included, the runtime increases to a few hours, but EMX is the only field solver considered in this work that could simulate the actual metal fill with its tens of thousands of shapes. Fig. 3.22 shows the difference between the less accurate “quasi-static” and the “full-wave” mode for the two-terminal inductor discussed in the previous section, including metal fill. The differences are negligible and the runtimes are similar so the other results presented in this thesis were simulated with the full-wave mode.



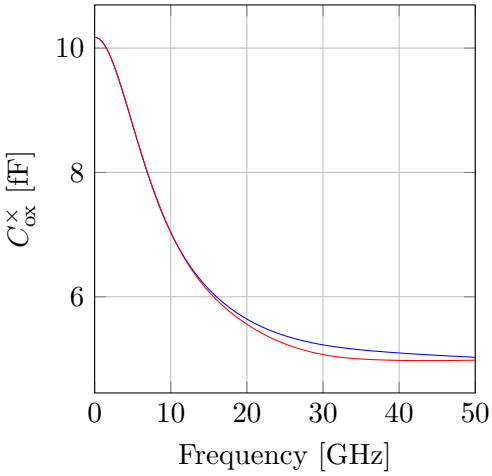

**(a)** Series resistance.



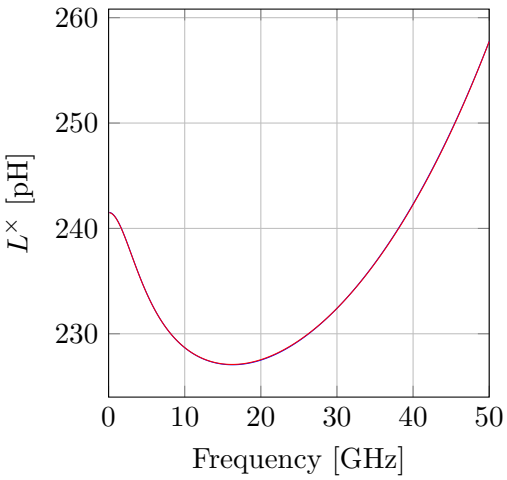

**(b)** Series inductance.



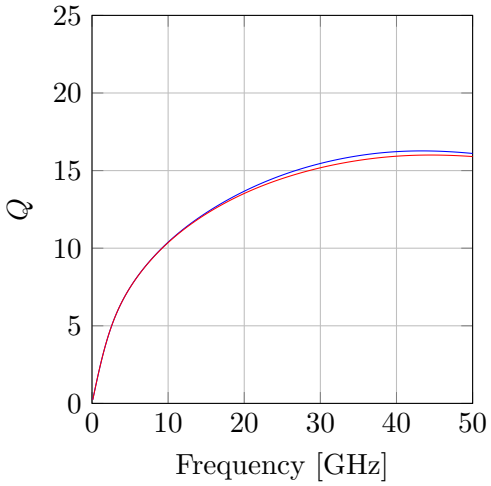

**(c)** Shunt resistance.




**(d)** Shunt capacitance.




**(e)** Single-ended inductance.




**(f)** Quality factor.

**Fig. 3.22:** EMX “full-wave” vs. “quasi-static”.

## 3.5 Synthesis of Inductors

So far, it has been shown that the lumped models are well suited for circuit design and optimization, but their components or parameters are not constant over frequency due to various effects. Furthermore, not all parameters can be computed from the S-parameters independently. Both problems can be alleviated by fitting the lumped model to the simulated S-parameters, which effectively averages the parameters over frequency.

The current sheet approximation is good but still has room for improvement. Mohan [40] tried to mitigate this by adding correction terms for finite conductor thickness and spacing. An analytic derivation considering thickness and spacing right from the start is at least very complicated if possible at all, and not pursued in this work. Instead, an updated formula is derived for the stacked T-coil with the help of simulation results.

Up to this point, most aspects were discussed on two-terminal inductors for simplicity but for ESD compensation, the focus now shifts towards T-coils. This section aims to combine the aspects discussed so far for on-chip inductors to provide a methodology for an educated design instead of having to resort on trial and error. Therefore, the section is split into four parts. The first part motivates the lumped T-coil model used for the remainder of this section. Then, the second part presents a fitting method to extract the model parameters from simulated S-parameters. In the third part, the current sheet formula from Mohan, including its correction terms, is discussed and fitted to a set of automatically generated and simulated T-coils. The resulting model is then used in the fourth part to derive a T-coil layout that is optimized with respect to this model and a few simple constraints.

### 3.5.1 Lumped T-Coil Model

This section uses the lumped T-coil model shown in Fig. 2.29. It has been used in section 2.3 to design the ESD compensation. Compared to the $\pi$-model of the two-terminal inductor, it neglects the substrate branches. Mohan also provided formulas for the stacked T-coil, for which he used the lumped circuit model shown in Fig. 3.23. The top and bottom inductors, $L_{s,t}$ and $L_{s,b}$, are again modeled with the current sheet formula and a constant coupling factor of $k = 0.9$, which is relatively high compared to the results shown later in this section. The bridge capacitance $C_{\text{ov}}$ is calculated with the oxide thickness between the inductors and their overlapping area. Mohan reduced the substrate branches to single capacitors. Technically, the substrate capacitors could be incorporated into the diode or pad capacitance, but that adds complexity to the analysis presented in this section and does not provide additional insight. This section will show

that compared to the influence of wires connecting to the T-coil, the substrate is less important. Especially the center-tap connection needs to be considered, which was not done in Mohan's work.

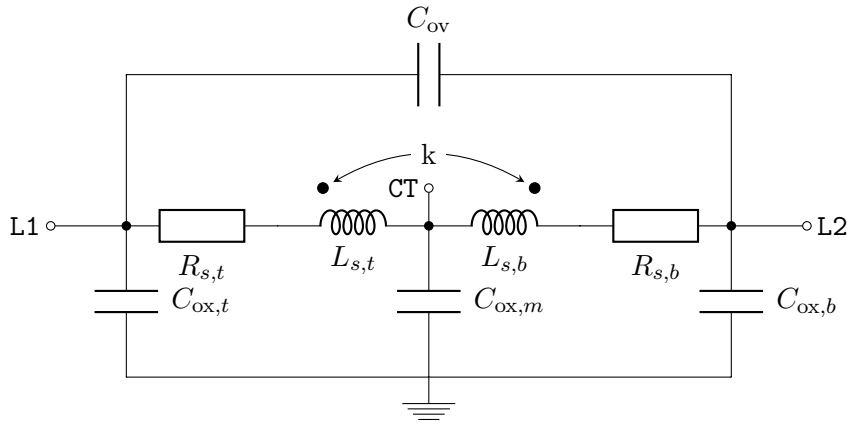



**Fig. 3.23:** Lumped circuit model for the stacked T-coil used by Mohan [40].

### 3.5.2 Parameter Estimation

Structures like transmission lines and on-chip inductors are commonly described with S-parameters obtained from EM-simulation tools. The previously presented QRC flow also generates them as the output format. S-parameters are typically stored in the text-based *Touchstone* file format. These files can be used in simulation at the schematic level by using "nports" from the Virtuoso standard library. However, the simulation with these is more time-consuming and they are less expressive compared to lumped circuit models. Hence, there is value in mapping the simulated S-parameters to lumped models. EMX does also offer such a feature, however the supported models are more complex compared to the ones discussed in this section and seem to be intended more for modeling an existing inductor instead of designing a new one. A brief test revealed that EMX fitted "strange" values for some parameters. This might not be problematic for the overall circuit but shows that fitting to lumped models is not a trivial task. A custom fitting solution was built in this work with Python, which can use either a least squares or an MCMC algorithm to estimate the electrical parameters from simulated S-parameters.

#### Touchstone Reader

As a first step, a Python class `SParameterFile` was developed to read in the S-parameters stored in Touchstone files. These can contain them either as real and imaginary parts or magnitude and phase pairs. The latter is automatically transformed into the former as the value range of real and imaginary parts of S-parameters is confined to the interval

$[-1, 1]$. This is better suited for fitting routines because magnitude and phase have different value ranges so it is more difficult to weight them equally. The `SParameterFile` object stores the read data as a list of `SParameter` objects. Each can hold a frequency value and the corresponding S-parameter matrix. The stored data can be converted to Y- and Z-parameters provided a reference impedance.

### Lumped Model S-parameters

A Python class called `InductorTapped` defines the lumped model, for which the parameters have to be fitted, as shown in Fig. 2.29. It provides a function to compute the 3-port S-parameters for given model parameters. This is done by transforming the circuit into star topology and computing the Y-parameters. From this, the S-parameters can be computed via a matrix multiplication. $\mathbb{1}$ is the unity matrix and $Z_0$ the port impedance, usually $50\,\Omega$. Fitting directly to Y-parameters is not feasible as their value range is not bounded.

$$\mathbb{S} = (\mathbb{1} - Z_0\mathbb{Y}) \cdot (\mathbb{1} + Z_0\mathbb{Y})^{-1} \tag{3.70}$$

Y-parameters are preferred over Z-parameters because the latter are not properly defined without a substrate branch to ground. Even with a capacitive branch to ground, they are infinite at DC. This can be also seen in Fig. 3.25, which shows a 3D view of a T-coil with the ports represented by red arrows. The reference plane is located below the inductor, on the opposite end of the arrows. There is no conductive path from any inductor pin to this reference plane. However, the definition of Z-parameter matrix elements given in Eq. 3.71 demands this conductive path.

$$Z_{ij} = \left.\frac{V_i}{I_j}\right|_{\forall k \neq j : I_k = 0}, Y_{ij} = \left.\frac{I_i}{V_j}\right|_{\forall k \neq j : V_k = 0} \tag{3.71}$$

The condition $\forall k \neq j : I_k = 0$ means all ports except for port $j$ are open, and at port $j$, a current $I_j$ is flowing. Then the voltage at port $i$ is divided by this current, but since all other ports are open, the voltage $V_i$ tends towards infinity, resulting in infinite Z-parameters. Y-parameters, on the other hand, are well defined. All ports except for $j$ are shorted to the reference plane and a voltage $V_j$ is applied to port $j$. The current flowing through port $i$ is finite and divided by this voltage.

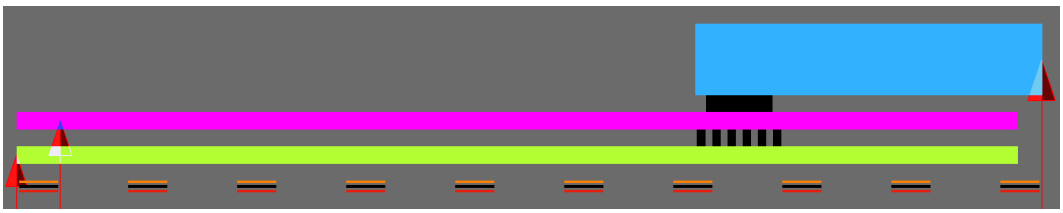


**Fig. 3.24:** Projection of a T-coil (created with Keysight ADS 2012).

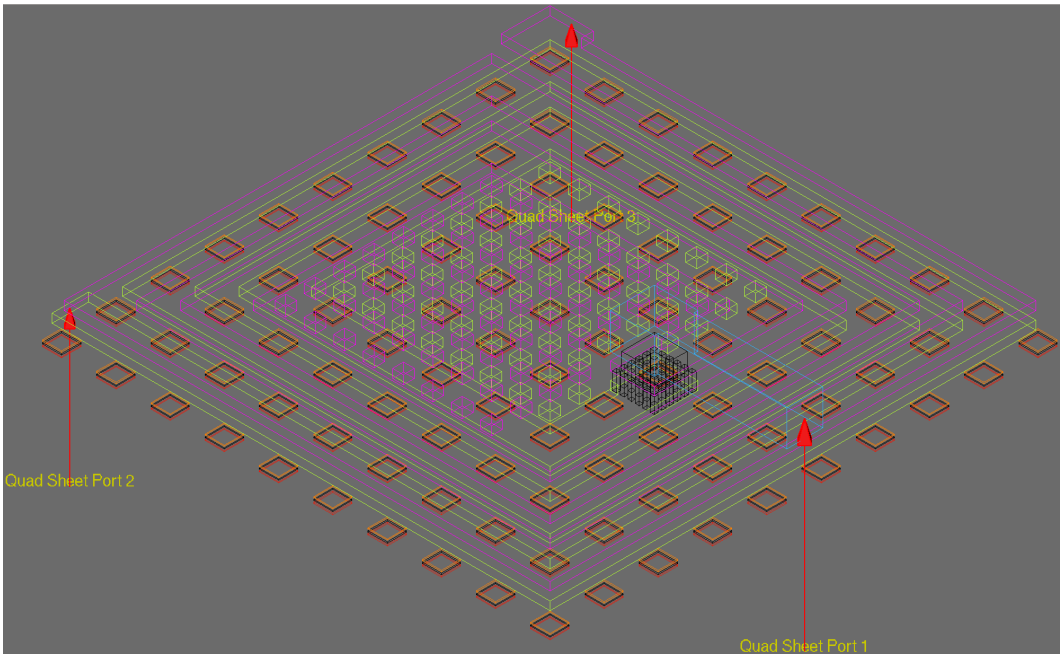



**Fig. 3.25:** 3D-view of a T-coil (created with Keysight ADS 2012).

### MCMC Fitting

The Python code developed to fit the lumped model to the S-parameters can do this with two different methods. The first is a least squares fit conducted with the `scipy` module [68]. This was fine for the S-parameters produced by QRC, which could be approximated very well by the lumped circuit. As QRC uses a lot of simplifications, it was expected that EM-simulation results would be much harder to fit. So as an alternative to the classical least squares fit, another method was evaluated to cover possibly more complex S-parameters and to avoid getting stuck in local minima, which can happen during least squares fits. A popular method to fit models with tens or hundreds of parameters is to use Markov-Chain Monte-Carlo algorithms. These are used to sample a posterior Probability Density Function (PDF) as it appears in *Bayesian Inference*. An overview of the underlying principles will be presented here to explain the various terms as these algorithms are advanced statistical methods. First is the term *Bayesian*, which refers to Bayes' theorem (Eq. 3.72), that can be used to compute dependent probabilities.

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} \tag{3.72}$$

This can be illustrated with a six sided dice. An experiment with three dice rolls is done and there are two possible results $A$, 'at least two rolls are 4', and $B$, 'first roll is not 4'. Before rolling the dice, the probability to roll at least two 4s is $P(A) = 2/27$. The probability of result $B$ is $P(B) = 5/6$. The conditional probability to roll at least two 4s

if the first roll is not a 4, is given by the conditional probability $P(A|B) = 1/36$. Bayes' theorem can now be used to compute the probability $P(B|A) = 5/16$, i.e. how likely it is that the first dice roll was not a 4 when it is known that overall at least two 4s were rolled.

Transferring this to the problem of fitting a model with parameters $\Theta$ to data $D$, this results in Eq. 3.73.

$$P(\Theta|D) = \frac{P(D|\Theta)P(\Theta)}{P(D)} \propto P(D|\Theta)P(\Theta) \tag{3.73}$$

The left side is the probability of the model given the data, also called *posterior* probability. So this is the PDF that has to be computed to derive the estimated model parameters. This is also a fundamental difference to least squares, which computes parameter estimates with errors instead of PDFs over the parameter space. On the right side, there is the probability $P(D|\Theta)$ of the data given a set of parameters times the probability of the parameters $P(\Theta)$. The former is called the *likelihood* and has to be implemented as a function that calculates how well a given set of parameters represents the data. The latter is called *prior*, i.e. a function that contains all a-priori knowledge available on the parameters. In most cases, it is reasonable to assume a *flat prior*, which makes all parameter values equally likely. However, the prior is also the place to provide this algorithm with boundaries. Thus, the prior is chosen to be flat within a certain range of parameters but zero outside to exclude unreasonably large or small values. The third part is the probability of the data $P(D)$, which is unknown, but constant. It is called *marginalized likelihood* or sometimes *model evidence* and is equal to the product of likelihood and prior marginalized, i.e. integrated, over all parameters.

$$P(D) = \int P(D|\Theta)P(\Theta)\mathrm{d}\Theta \tag{3.74}$$

If there are multiple models to be fitted to the same data, this value can be used to determine which model fits best, hence the name evidence. In cases with a single model, it presents a simple constant that does not depend on the parameters $\Theta$, so only the proportionality from Eq. 3.73 is considered. To summarize, the basic procedure conducted by this approach is to transform the prior into a posterior PDF by using the likelihood function and the data. If the same inductor would be simulated again, e.g. at different frequency points, the posterior distribution could be used as the new prior to compute an updated posterior. This is called *Bayesian updating*.

In most cases, this cannot be solved analytically and numerical methods are applied. A common method is to sample a chain of random numbers from the posterior PDF. If this is done a large number of times, the resulting distribution approximates the posterior.

A relatively old MCMC algorithm to create such a chain is the Metropolis-Hastings (M-H) algorithm. An initial version was presented in 1953 by Metropolis et al. [69], which was later generalized by Hastings [70] in 1970. The details are beyond the scope of this section, but this algorithm needs to be tuned by setting $N(N+1)/2$ parameters to work correctly [71]. Here $N$ is the dimension of $\Theta$. This makes the M-H algorithm complicated to apply as there is no foolproof method to set these parameters for different problems [71]. An improved algorithm was developed by Goodman & Weare [72]. It *"significantly outperforms"* [71] M-H based algorithms through various computational advantages and much less free parameters. Instead of a single Markov-Chain, an *ensemble* of *walkers* creates multiple chains. During initialization the walkers are distributed randomly across the parameter space. In every step, the position of each walker $\Theta_i(t)$ is updated with the position of another, randomly selected, walker $\Theta_j(t)$ as shown in Eq. 3.75. $r$ is a random value drawn from a specific distribution.

$$\Theta_i(t) \rightarrow \Theta_i(t+1) = \Theta_j + r(\Theta_i(t) - \Theta_j) \tag{3.75}$$

For algorithmic reasons, this has to be done in series, i.e. $\Theta_i(t)$ is replaced immediately with $\Theta_i(t+1)$, and then the next $\Theta_j$ is drawn from the updated ensemble. However, the new position is only accepted with a certain probability based on the ratio of the posterior probability of the new and old position. As the initial distribution of the walkers is not related to the posterior PDF, there is a certain amount of "burn-in" steps required. The length of this phase depends on the model and data, but can easily be determined and excluded from the result. As the details are beyond the scope of this work, this is only a coarse description of the algorithm.

Foreman et al. [71] developed a Python module called `emcee`, which allows to use the Goodman & Weare algorithm as a well-proven MCMC Python code and with relatively low effort. The prior and likelihood functions have to be user-defined and handed to the `emcee.EnsembleSampler` method. In case of the T-coil, the model $\mathbb{S}_\Theta$ is given by the function in Eq. 3.76, where $\Theta = (L_1, L_2, L_3, C_B)$ and the ports are numbered in the order `CT`, `L1`, `L2`. The resistors are not fitted but simply calculated at $f = 0$ to get proper DC operating points with the resulting model. Each of the nine S-parameter components has a real and imaginary part so there are 18 single-valued functions, which are all weighted equally in the likelihood function. It could be considered to weight the off-diagonal element with 1/2 because the T-coil is a linear device and the matrix is

symmetric, but this was not done for the results presented in this thesis.

$$\mathbb{S}_\Theta(f,\Theta) = \begin{pmatrix} S_{\Theta,11}(f,\Theta) & S_{\Theta,12}(f,\Theta) & S_{\Theta,13}(f,\Theta) \\ S_{\Theta,21}(f,\Theta) & S_{\Theta,22}(f,\Theta) & S_{\Theta,23}(f,\Theta) \\ S_{\Theta,31}(f,\Theta) & S_{\Theta,32}(f,\Theta) & S_{\Theta,33}(f,\Theta) \end{pmatrix} \tag{3.76}$$

The data is given by $\mathbb{S}_D(f_i)$. Here, the $f_i$ are the simulated frequency points (in total $n_\mathrm{f}$). A common approach is to define a Gaussian likelihood function. `emcee` works with the natural logarithm of the likelihood, the log-likelihood $\mathcal{L}$.

$$\mathcal{L} = \ln(P(D|\Theta)) = -\frac{1}{2}\sum_{i=0}^{n_\mathrm{f}-1}\left(\frac{\Delta^2_{D\Theta,i}}{\sigma_i^2} + \ln\left(2\pi\sigma_i^2\right)\right) \tag{3.77}$$

The difference of model and data for the $i$-th frequency $f_i$ is defined by Eq. 3.78. This is the sum of squares for each of the 18 S-parameter functions.

$$\Delta^2_{D\Theta,i} = \sum_{j=1}^{3}\sum_{k=1}^{3}\left(\mathrm{Re}\left(S_{D,jk}(f_i) - S_{\Theta,jk}(f_i)\right)\right)^2 + \left(\mathrm{Im}\left(S_{D,jk}(f_i) - S_{\Theta,jk}(f_i)\right)\right)^2 \tag{3.78}$$

The error of each frequency point $\sigma_i$ is unknown as there is no indication of a simulator error. Obviously, it cannot be set to zero, but it can be included into the model as $\sigma_i = \sigma$ for all data points. There is no reason to assume different error bars for each data point. Furthermore, it would lead to overfitting. Including $\sigma$ is important to get valid credibility intervals for the remaining parameters as their errors are scaled with $\sigma$. Compared to the other parameters, $\sigma$ does not describe the T-coil and is called a *nuisance* parameter, which can be marginalized over. The results of an `emcee` run are presented with a *corner* plot, which can be created with the `corner` module [73]. The corner plot shows a histogram for each parameter, which simply presents the samples drawn for the specific parameter. For a large enough number of samples drawn, this is equivalent to the marginalized PDFs of the parameters. This is given by Eq. 3.79 for $L_1$ and is analogous for the other parameters.

$$P(L_1|D) = \iiint P(\Theta,\sigma|D)\mathrm{d}L_2\mathrm{d}L_3\mathrm{d}C_B\mathrm{d}\sigma \tag{3.79}$$

The two-dimensional plots leave two parameters remaining after marginalization. This can be used to analyze correlations between the parameters. $\sigma$ for example is uncorrelated to every other parameter, which is expected; whereas the others show an elliptic PDF indicating some correlation. Finally, the parameters are estimated by taking the 16th, 50th, and 84th percentile of the sorted sample vector. For a large enough sample size, this is equivalent to "$\mu \pm \sigma$" if the maginalized PDFs can be assumed as Gaussian, which

is reasonable in this case. Fig. 3.26 shows the marginalized PDFs for the T-Coil shown in Fig. 3.25. However, it should be noted that for multi-modal likelihood functions the parameters that maximize this likelihood are not identical to the estimates extracted from the marginalized PDFs. In such a case, it is much more complicated to extract parameter estimates and the corresponding procedure has to be specifically decided on according to the problem at hand.

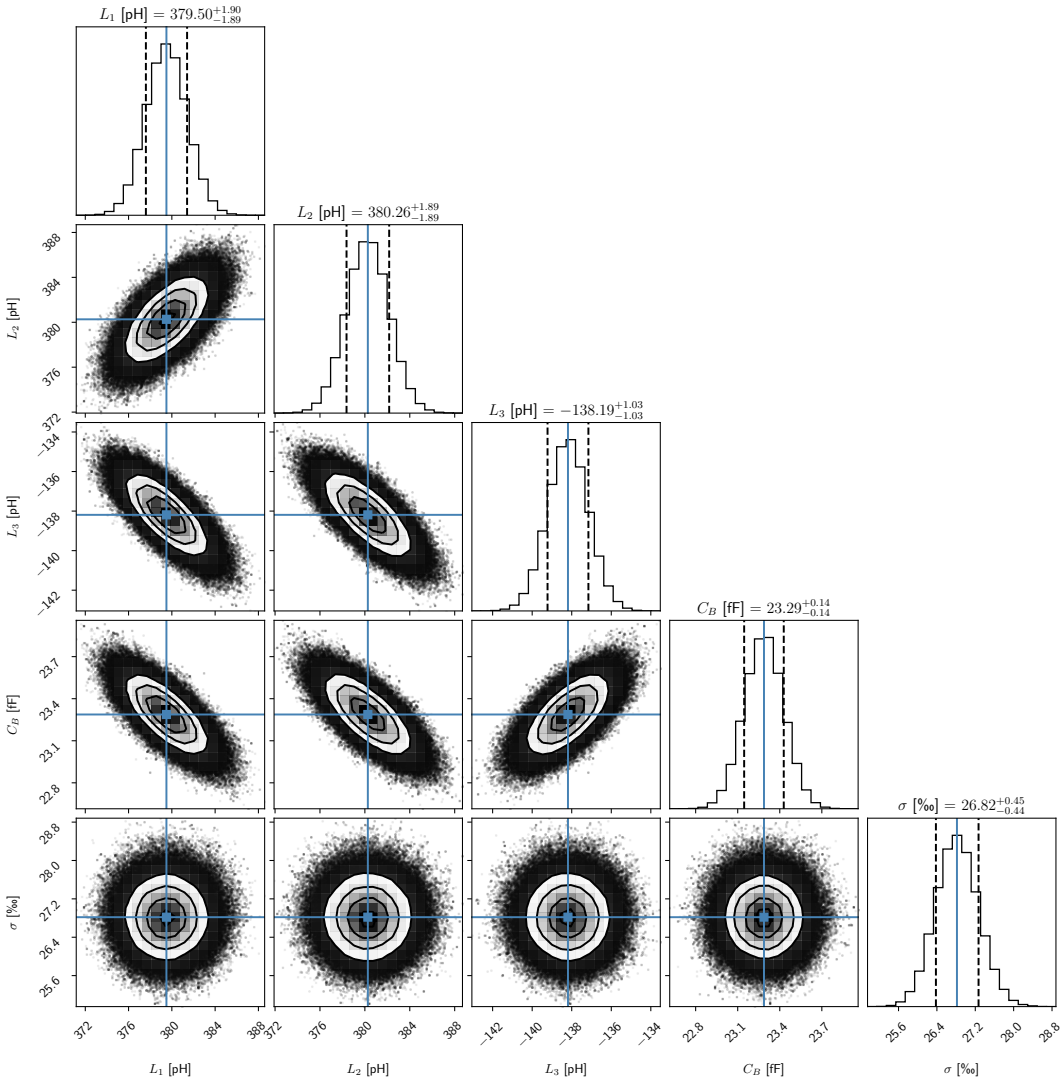



**Fig. 3.26:** Example corner plot of parameters $(\Theta, \sigma)$. Shown are the marginalized posterior PDFs for each model parameter as a histogram, along with the 16th, 50th (median), and 84th percentile.

### 3.5.3 Analytic T-Coil Model

In this subsection the results of this chapter are combined to derive a model that maps geometry parameters to lumped model parameters, i.e. a function $M_{\text{T-coil}}$.

$$M_{\text{T-coil}} : (d, w, s, n) \mapsto (R_1, R_2, R_3, L_1, L_2, L_3, C_B) \tag{3.80}$$

This is possible because many inductors can be simulated automatically using EMX and PCells. Furthermore, a viable fitting technique has been found to automatically estimate the lumped model parameters of these inductors. Last but not least, the formulas provided by Mohan [40] are a good foundation to build on. The resulting model will of course be specific to this type of T-coil and this process, but the insight it provides is actually independent of that and the methodology can be transferred to similar problems. Therefore, it allows to build inductors in a forward design flow instead of an iterative "trial and error" approach. This subsection is split into three parts. Firstly, the distribution of the generated inductors in the geometry space is described. Secondly, the error of the inductance from Mohan's T-coil model is analyzed. In the last part, the parameters in this model are optimized to best describe the simulated inductors.

#### Geometry Space

Before the large number of inductors was generated, a few constraints had to be applied to obtain a feasible set of inductors. This chapter has shown that the layout of each type of on-chip inductor is described by a certain set of geometric parameters $(d, w, s, n)$. However, the stacked T-coil PCell has further parameters, e.g. to set metal layers or draw metal fill. It is enough to focus only on a fixed layer combination because for ESD compensation, the pair of thicker metals below the redistribution layer is the optimal choice. The other layers can be neglected as they have much larger parasitic resistance, which compromises ESD performance and signal integrity. This will certainly be similar for other applications as series resistance in inductors is usually not desirable. The center tap was placed on the redistribution layer for the generated T-coils. Furthermore, the metal fill parameter has to be deactivated to reduce the runtime of the EMX simulation from hours to minutes. So the remaining parameters are the aforementioned four that a designer actually wants to tune: outer diameter $d$, turn width $w$, turn spacing $s$, and the number of turns $n$. A simple approach would be to sweep each parameter, but this has to be done more sophisticatedly. The four variables cannot be chosen independently as this will result in invalid geometries. A small inductor cannot accommodate as many turns as a larger one. So the following condition is checked to ensure enough space is kept free in

the center to reasonably place the vias to connect all three parts of the inductor.

$$\max(6\,\mu\text{m}, d/3) \leq d - 2\lceil n \rceil w - 2(\lceil n \rceil - 1)s \tag{3.81}$$

A significant drawback of this is that it results in considerably more high than low diameter T-coils to be generated. So the accuracy of the resulting model will be heavily biased towards larger diameters. This is especially critical since the diameter is usually the primary constraint imposed on the inductor design, while the other parameters can be chosen almost independent of the surrounding structures. So to avoid this, constraint-random generation was applied for $w$, $s$, and $n$, but the diameter was stepped equally from 20 µm to 60 µm in steps of 4 µm.

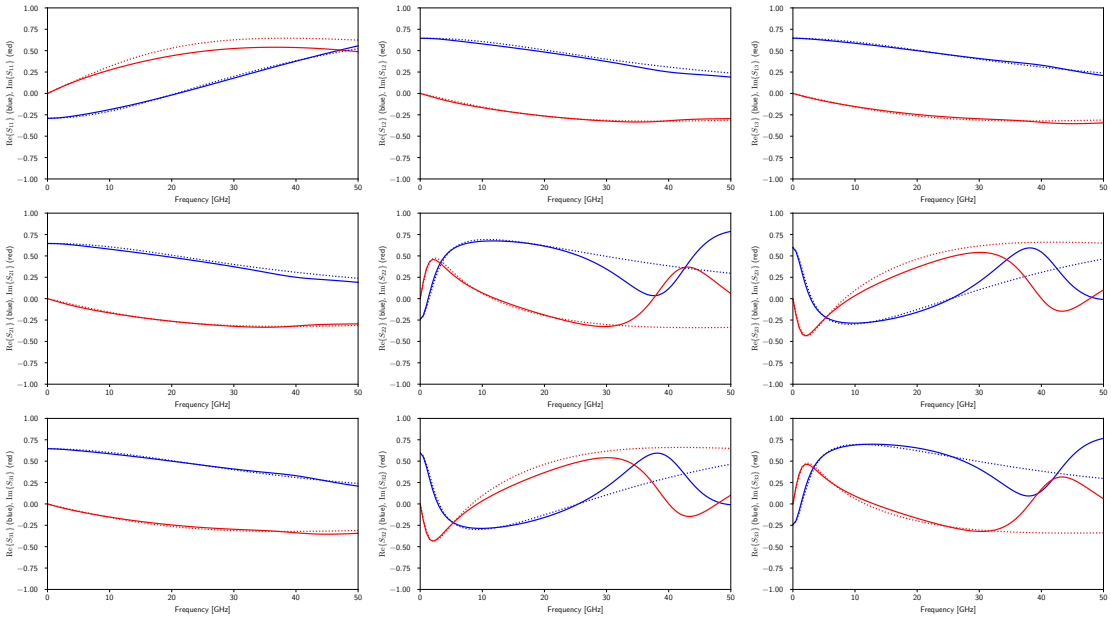



**Fig. 3.27:** S-parameters of the (60 µm, 1.6 µm, 1.3 µm, 7.0) T-coil and fitted lumped circuit model (dashed). The real parts are blue and the imaginary parts red.

Larger inductors start to inhibit many more second and higher order effects for five and more turns already at frequencies below 50 GHz as their distributed nature becomes relevant. Fig. 3.27 shows the S-parameters of the most extreme case found in the generated inductors. The lumped circuit model can certainly not approximate the inductor at more than 25 GHz. This should not impede the model much as the fraction of these inductors is around 2% and their influence on the model likelihood is diminished by their fit error, which is significantly higher than that of other inductors (ten to hundred times).

The second constraint is $s \leq w$ because the model from Mohan even assumes that the spacing is truly smaller than the width. In total, 1089 inductors have been generated and fitted with `emcee`. The fit takes the most time but can also be easily parallelized.

### Mohan's Inductance Model

Before updating the model proposed by Mohan, it should be checked how big the error to the fitted parameters actually is. Here, this analysis is limited to the inductance as it is the interesting part. The expression proposed by Mohan, including the correction terms, is shown in Eq. 3.82 [40]. The last term in the first line corrects the GMD term for finite thickness, while the second line comprises the corrections for finite spacing, for all three terms, GMD, AMD, and AMSD.

$$\begin{aligned} L_{\mathrm{nsq,cor}} = {} & \frac{2\mu_0 n^2 d_{\mathrm{avg}}}{\pi} \cdot \left( \ln\left(\frac{2.067}{\rho}\right) + 0.178\rho + 0.125\rho^2 - \frac{1}{n}\ln\left(\frac{w+t_{\mathrm{ct}}}{w}\right)\right) \\ & + \frac{2\mu_0 n^2 d_{\mathrm{avg}}}{\pi} \left( 0.5\frac{(n-1)s^2}{(\rho d_{\mathrm{avg}})^2} + 0.178\frac{(n-1)s}{n d_{\mathrm{avg}}} + 0.0833\frac{(n-1)s(s+w)}{d_{\mathrm{avg}}^2}\right) \end{aligned} \tag{3.82}$$

$$L_{1,\mathrm{Mohan}} = 1.9 \cdot L_{\mathrm{nsq,cor}} \tag{3.83}$$

$$L_{2,\mathrm{Mohan}} = 1.9 \cdot L_{\mathrm{nsq,cor}} \tag{3.84}$$

$$L_{3,\mathrm{Mohan}} = -0.9 \cdot L_{\mathrm{nsq,cor}} \tag{3.85}$$

Mohan approximates the coupling factor to be $k = 0.9$ for all inductors. He only lowers $k$ if the inductors are not exactly stacked above each other and neglects the influence of the vertical extension of the inductors as *"The thicknesses of the top and bottom metal layers and the thickness of the oxide between them has only a second order effect on k"* [40]. With the MCMC fit results for each inductor, the different variations of the above expression can be easily compared to study the influence of the various correction terms. To visualize this, the cumulative distribution of inductors over the relative error between Mohan's model and the fitted inductance is plotted: $|(L_{\mathrm{model}} - L_{\mathrm{fit}})/L_{\mathrm{fit}}|$. A steep slope and a graph located at the left side means a higher prediction quality of the corresponding formula. At first glance, there are a lot of possible combinations of terms that could be compared, but this number can be reduced by a fair amount. While it is interesting to compare the effect of the mean distance terms, it is not meaningful to apply the AMSD correction for spacing when only the GMD term of the current sheet part is used. This results in the following interesting combinations. The current sheet terms are either used up to the GMD, AMD, or AMSD. Then, these three variants double by adding the thickness correction and then double again by adding the spacing correction terms. As noted before, the spacing correction is always applied up to the corresponding current sheet terms. Note that Mohan did not derive AMD and AMSD corrections for finite thickness. So this results in twelve interesting formulas to be analyzed. Two plots are needed for each model, one for $L_3$, and only one for $L_1$ and $L_2$ as the inductors are assumed to be symmetric.

The plots show a poor correlation to the simulated inductors. While $L_1 + L_2$ shows an error between 20 to 120 percent, $L_3$ is even larger with a range of 50 to 600 percent. This renders the formulas unusable for the inductors considered in this work but does by no means imply that Mohan's model is bad as several assumptions made by Mohan are not valid in this case. The three mean distance approximations are grouped by color in Figs. 3.28 and 3.29. It is easily visible that using only the GMD terms (red) is generally better than adding the AMD (green) or even AMSD (blue) terms. Thus, the deviation from the current sheet assumptions are large enough that higher order terms actually do more harm than good. Considering the correction for finite thickness, it is clearly visible that it improves the error a lot as it basically splits the graphs into two groups. The finite spacing correction is more subtle but again leads into the wrong direction as it always increases the error. The conclusion of these findings is that the thickness correction is extremely important, which is reasonable as the thickness of the used layers is in the same order of magnitude as the turn width of the T-coils. To explain the increase of error by adding the other terms, it is important to look at the differences between Mohan's inductors and the ones created in this work. Firstly, there is the size, with Mohan considering inductors with an outer diameter roughly an order of magnitude larger, approximately in the range of 150 µm to 700 µm [40]. This also implies wider conductors, which reduces the influence of the finite thickness. Larger conductors also reduce the error caused by offsets like vias. In particular for very small T-coils in the simulated set, i.e. for around 20 µm, the via used to connect to the redistribution layer and the associated additional metal is relatively large and adds a substantial offset. This already leads to the next important point, the additional wire to route the center-tap terminal to the border of the T-coil. Its self-inductance renders $L_3$ more positive than it would be without it and thus decreases the effective $k$. Last but not least, it is to be noted that the fitted inductance is an average across the frequency, which is – according to section 3.3 – lower than the DC inductance computed in Mohan's model. This shows that analytic results should always be cross-checked with simulation because it is very difficult to extract and understand the assumptions made in such formulas in literature. It is likely that the specific design at hand deviates in at least one aspect that was not anticipated by the designer.

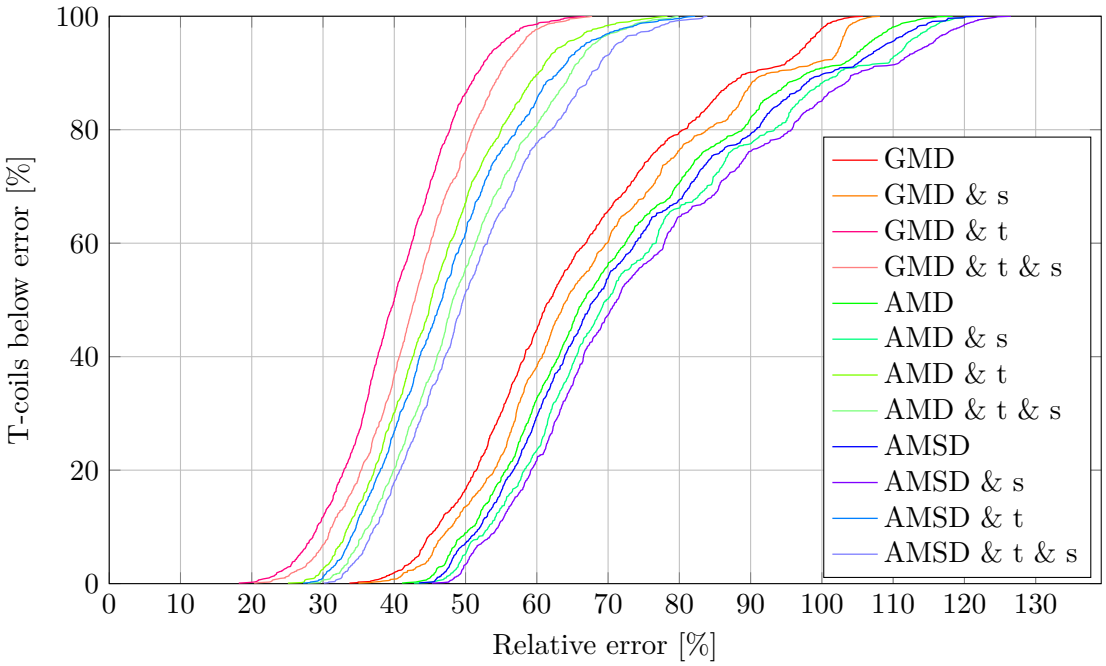



**Fig. 3.28:** Cumulative distribution of the relative error for $L_{1,\text{Mohan}} + L_{2,\text{Mohan}}$.

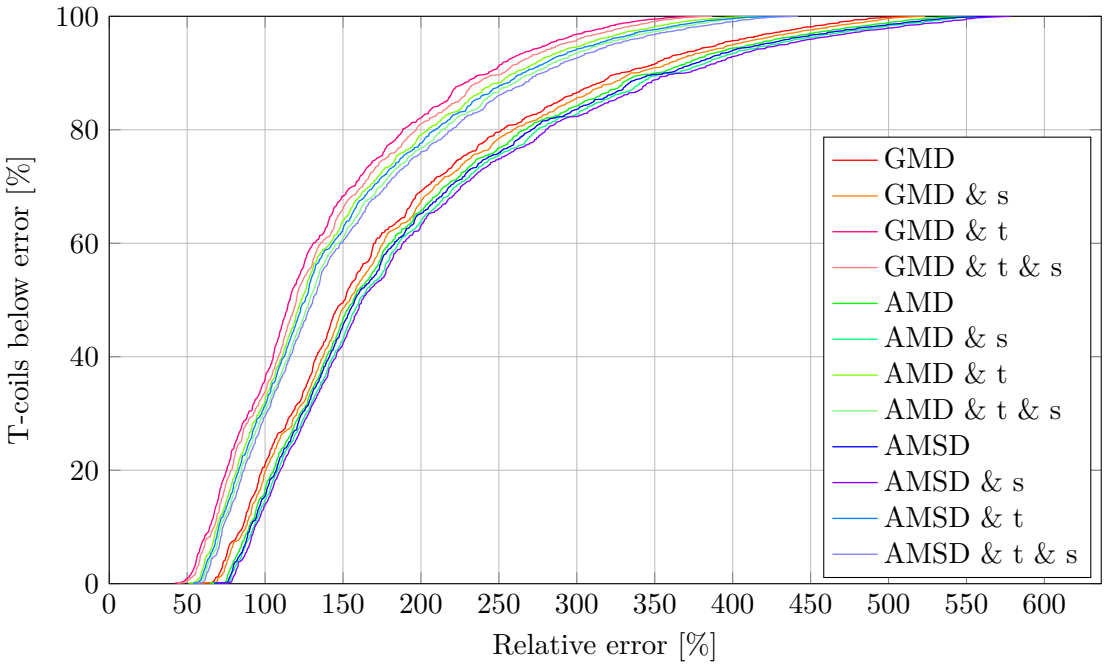



**Fig. 3.29:** Cumulative distribution of the relative error for $L_{3,\text{Mohan}}$.

### Updated Inductance Model

As demonstrated, the model proposed by Mohan cannot be used effectively for the type of T-coil to be designed in this work. However, deriving more accurate analytic formulas is very difficult because the integrals for the mean distances get very complicated when finite thickness is included. So instead, this work proposes an updated model that makes use of the knowledge presented so far in this work: firstly Mohan's corrected model for the square current sheet, secondly the mean distance formula for the inductance of a straight wire with rectangular cross-section, and thirdly the MCMC fitting technique. This results in Eqs. 3.86 to 3.91. There are two main differences to Mohan's model: The numeric constants are changed into fitting parameters, one for each term and one for $k$; and the self-inductance of the center-tap wire $L_{\mathrm{ct}}$ is added to $L_3$ with the approximated formula of the self-inductance of a straight wire with rectangular cross-section. The length $l_{\mathrm{ct}}$ of this wire varies depending on the number of turns, for whole turns it is longer and for half turns shorter, and also its thickness $t_{\mathrm{ct}}$ is different from the thickness of the windings $t$. So in addition to the twelve models plotted previously, the center-tap correction adds another twelve variations to be analyzed.

$$\Theta_L = (k, m_0, m_1, m_2, t_0, s_0, s_1, s_2, c_0, c_1, c_2) \tag{3.86}$$

$$\begin{aligned} L_{\mathrm{nsq,fit}} = {} & \frac{2\mu_0 n^2 d_{\mathrm{avg}}}{\pi} \cdot \left( \ln\left(\frac{m_0}{\rho}\right) + m_1\rho + m_2\rho^2 + \frac{t_0}{n}\ln\left(\frac{w+t}{w}\right)\right) \\ & + \frac{2\mu_0 n^2 d_{\mathrm{avg}}}{\pi}\left( s_0 \frac{(n-1)s^2}{(\rho d_{\mathrm{avg}})^2} + s_1 \frac{(n-1)s}{n d_{\mathrm{avg}}} + s_2 \frac{(n-1)s(s+w)}{d_{\mathrm{avg}}^2}\right) \end{aligned} \tag{3.87}$$

$$L_1 = L_2 = (1+k)\cdot L_{\mathrm{nsq,fit}} \tag{3.88}$$

$$L_{\mathrm{ct}} = \frac{\mu_0 l_{\mathrm{ct}}}{2\pi} \cdot \left( \ln\left(\frac{c_0 l_{\mathrm{ct}}}{w+t_{\mathrm{ct}}}\right) + c_1 \cdot \frac{\sqrt{w^2 + t_{\mathrm{ct}}^2 + 0.46 w t_{\mathrm{ct}}}}{l_{\mathrm{ct}}} + c_2 \cdot \frac{w^2 + t_{\mathrm{ct}}^2}{l_{\mathrm{ct}}^2}\right) \tag{3.89}$$

$$l_{\mathrm{ct}} = \begin{cases} d_{\mathrm{avg}}, & n \in \mathbb{N} \\ \rho d_{\mathrm{avg}}, & (n+0.5) \in \mathbb{N} \end{cases} \tag{3.90}$$

$$L_3 = -kL_{\mathrm{nsq,fit}} + L_{\mathrm{ct}} \tag{3.91}$$

Instead of plotting all 24 curves, it is more efficient to look at Tab. 3.3 first and then plot only the interesting cases. It shows the parameters obtained from `emcee` as well as the Bayesian Information Criterion (BIC) of each model. The BIC is a model selection criterion that favors the model with the lowest value. It is defined in Eq. 3.92

$$\mathrm{BIC} := N_{\mathrm{params}} \cdot \ln(N_{\mathrm{data}}) - 2\ln(\mathcal{L}_{\max}) \approx N_{\mathrm{params}} \cdot \ln(N_{\mathrm{data}}) - 2\ln(\mathcal{L}(\Theta_{\mathrm{opt}})) \tag{3.92}$$

The first term penalizes adding more and more parameters and aims to prevent overfitting, while the second term favors large likelihood values. $N_{\text{params}}$ and $N_{\text{data}}$ are the number of parameters and data points, respectively. The approximation assumes that the likelihood for the optimal parameters is equal to the global maximum of the likelihood function. Since the resulting marginalized posteriors seem well defined and narrow, this should be a valid simplification. `emcee` does not provide the global maximum, so this is also a necessary choice.

Starting with the first column for $k$, every second entry shows a value close to 0.46, while the next one is always much larger. This also coincides with the center-tap correction and demonstrates how $k$ is underestimated by neglecting the self-inductance of that wire segment. This also indicates that the influence of other wires connecting to the T-coil is most likely substantial, even if it is neglected in this work. The value of the BIC can be used to quickly estimate the relevance of the three corrections. The BIC is reduced significantly stronger when the thickness and center-tap corrections are applied, compared to the spacing correction (bold entries). In line with to the observation that the AMD and AMSD terms tend to increase the error as the extracted inductances are quite far from valid current sheet assumptions, the BIC does also not decrease much when adding these terms. In addition, these terms exhibit values very different from their "normal" values when compared to Mohan's model, which further indicates that the AMD and AMSD terms are artificial here.

Fig. 3.30 and 3.31 show again the cumulative distribution of the relative error. The effects described above are most visible in the plot for $L_3$. The error was dramatically reduced compared to the raw Mohan model. Despite its simplicity, the "GMD & t & c" model already achieves low errors. The corner plot for this model is shown in Fig. 3.32.

Note that BIC and relative error do not necessarily correlate because the log-likelihood function did not minimize the relative error but instead a Gaussian expression (Eq. 3.93).

$$-\frac{1}{2}\left(\left(\frac{L_{\text{model}}-L_{\text{fit}}}{\sigma\cdot\Delta L_{\text{fit}}}\right)^2+\ln\left(2\pi(\sigma\cdot\Delta L_{\text{fit}})^2\right)\right) \tag{3.93}$$

Theoretically, an alternative would be to fit the model directly to the set of S-parameters, i.e. without an intermediate fit of each T-coil to the lumped model. Unfortunately, this is computationally very expensive and practically not a feasible solution. Thus, to still consider a bad fit of a T-coil to the lumped model (see Fig. 3.27), the above Gaussian likelihood function has been chosen over the relative error in this work. Nevertheless, plotting the latter is much more instructive.

| $k$ | $m_0$ | $m_1$ | $m_2$ | $t_0$ | $s_0$ | $s_1$ | $s_2$ | $c_0$ | $c_1$ | $c_2$ | BIC |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0.45 | 1.41 | | | | | | | | | | 20068 |
| 0.58 | 1.26 | | | | | | | 7.72 | | | **19631** |
| 0.45 | 1.61 | | | | -2.22 | | | | | | **19880** |
| 0.59 | 1.43 | | | | -2.03 | | | 8.07 | | | 19373 |
| 0.47 | 2.17 | | | -1.62 | | | | | | | **18705** |
| 0.64 | 1.86 | | | -1.55 | | | | 14.83 | | | 16999 |
| 0.47 | 2.15 | | | -1.92 | 1.55 | | | | | | 18600 |
| 0.64 | 1.83 | | | -1.88 | 1.70 | | | 16.58 | | | 16567 |
| 0.46 | 1.26 | 0.29 | | | | | | | | | 20041 |
| 0.72 | 0.77 | 1.08 | | | | | | 0.17 | 29.40 | | **18621** |
| 0.46 | 1.23 | 1.13 | | | 0.74 | -7.66 | | | | | **19839** |
| 0.71 | 0.87 | 1.03 | | | -1.24 | -0.41 | | 0.16 | 28.61 | | 18395 |
| 0.46 | 2.60 | -0.36 | | -1.77 | | | | | | | **18630** |
| 0.68 | 1.43 | 0.45 | | -1.29 | | | | 0.55 | 20.19 | | 15979 |
| 0.46 | 2.10 | 0.71 | | -2.14 | 5.12 | -9.07 | | | | | 18399 |
| 0.68 | 1.37 | 0.71 | | -1.60 | 2.34 | -2.57 | | 0.74 | 18.34 | | 15461 |
| 0.46 | 0.99 | 1.80 | -2.23 | | | | | | | | 20024 |
| 0.80 | 0.54 | 3.02 | -2.79 | | | | | 11.06 | -2.26 | 83.93 | **18434** |
| 0.46 | 0.86 | 3.89 | -3.81 | | 2.54 | -17.97 | 10.59 | | | | **19824** |
| 0.78 | 0.55 | 4.19 | -4.40 | | 0.77 | -12.15 | 12.83 | 6.10 | 1.48 | 71.47 | 18224 |
| 0.46 | 2.82 | -0.82 | 0.68 | -1.79 | | | | | | | **18633** |
| 0.72 | 1.27 | 0.82 | -0.49 | -1.22 | | | | 12.87 | -3.86 | 60.94 | 15844 |
| 0.46 | 2.60 | -1.92 | 3.79 | -2.22 | 2.42 | 11.97 | -24.72 | | | | 18358 |
| 0.72 | 1.34 | -0.06 | 1.13 | -1.56 | 1.08 | 7.04 | -11.27 | 21.27 | -7.52 | 64.96 | 15183 |

**Tab. 3.3:** Fitted parameters for the inductance model.

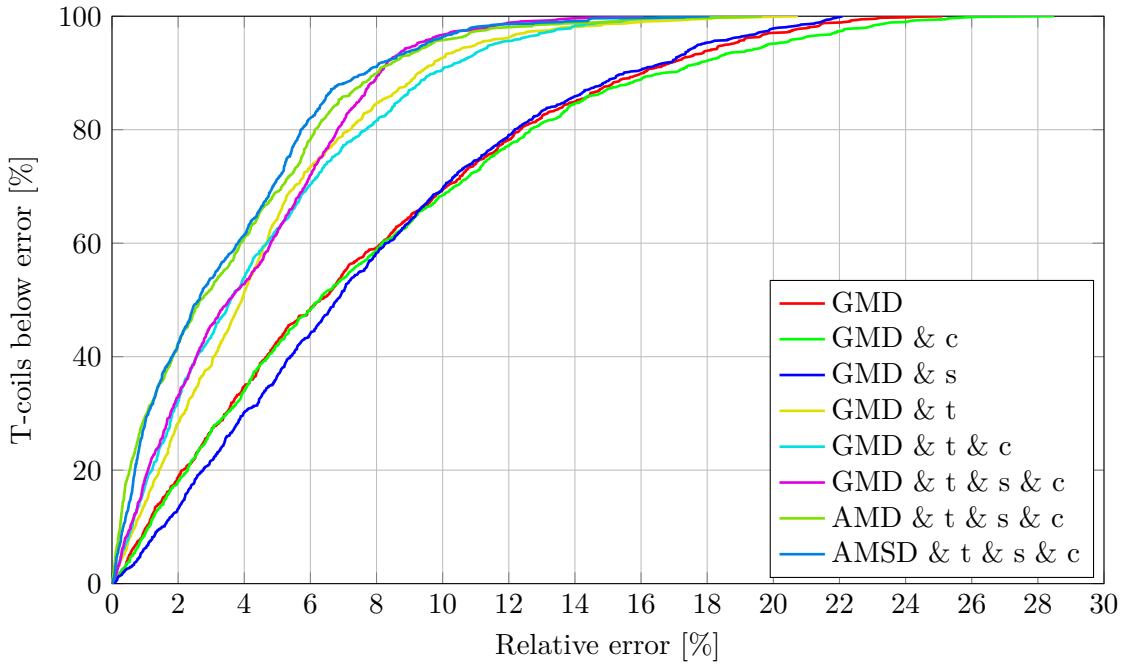



**Fig. 3.30:** Cumulative distribution of the relative error for $L_1 + L_2$.

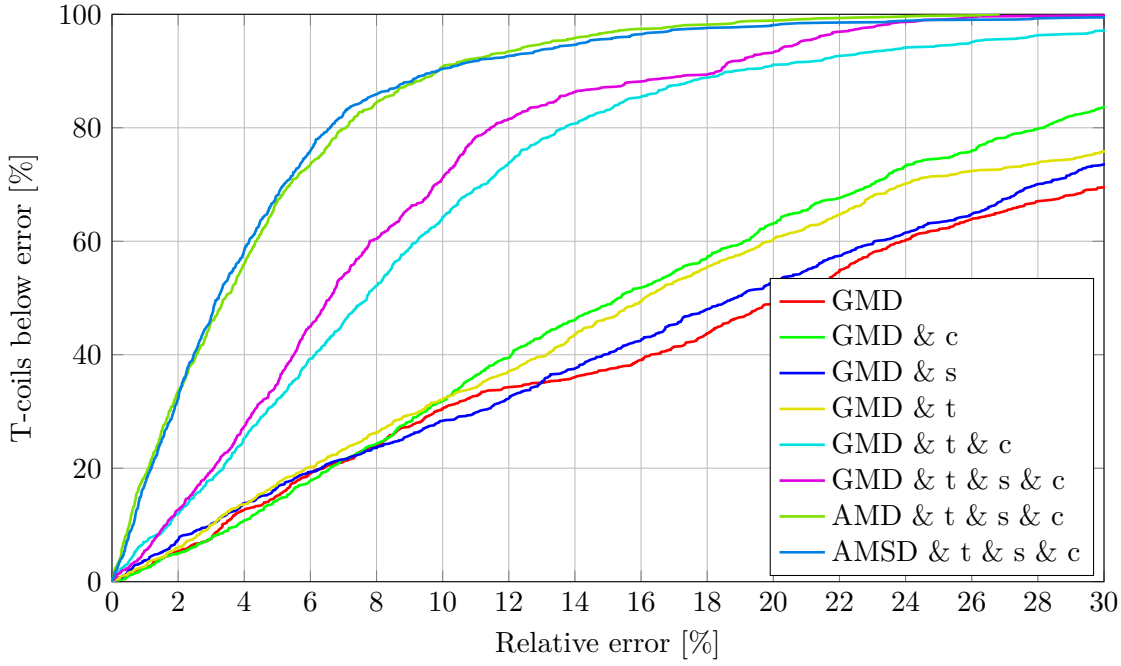



**Fig. 3.31:** Cumulative distribution of the relative error for $L_3$.

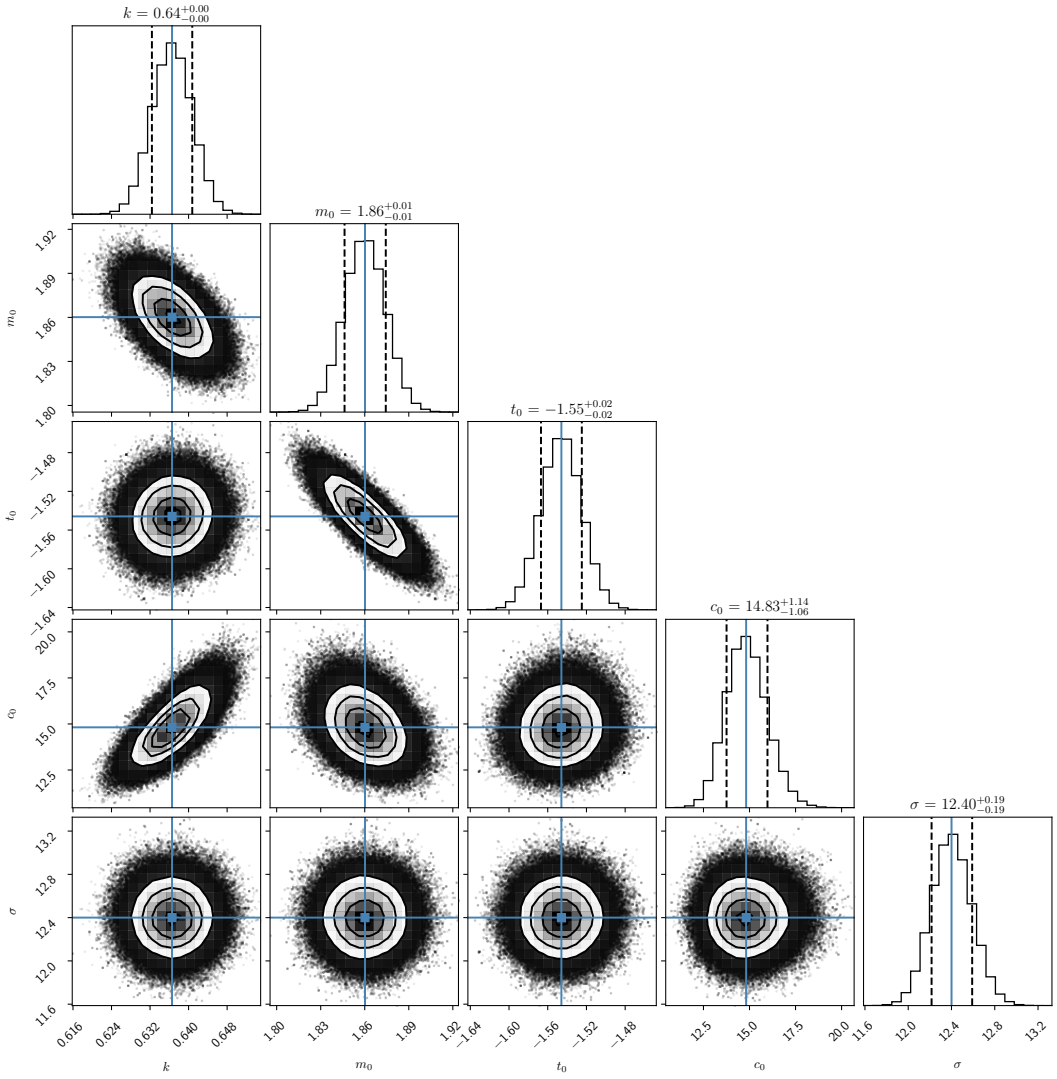



**Fig. 3.32:** Corner plot for the "GMD & t & c" model.

### Bridge Capacitance Model

The bridge capacitance is as important for the ESD compensation as the inductances. Therefore, some model formula needs to be found for it as well. Unfortunately, much more than a simple model proportional to the area of the windings is not easily derived or found. However, extending the proportionality parameter $a_0$ to a monomial of the geometry parameters is quite potent. The six variants shown in Eqs. 3.94 to 3.99 have been fitted to the data, and the results are shown in Fig. 3.33 and Tab. 3.4. Note that the length of the inductor is proportional to $nd_{\mathrm{avg}}$. The models are generally less accurate compared to the ones for the inductance.

One would expect the simple formula with an additional offset to be at least equally

accurate to the one without that offset. However, this is not the case here because as mentioned before, the log-likelihood function did not minimize the relative error but an expression as shown in Eq. 3.93. This weights every T-coil according to its error, while the relative error treats every T-coil equally. A simple check has revealed that indeed the ones with a large deviation from the model are much more likely to also have a larger error on their fit result.

$$C_{B,\text{simple}} = a_0 \cdot n d_{\text{avg}} w \tag{3.94}$$

$$C_{B,\text{monom}} = a_0 d^{a_1} w^{a_2} s^{a_3} n^{a_4} \cdot n d_{\text{avg}} w \tag{3.95}$$

$$C_{B,\text{simple,offset}} = a_0 \cdot n d_{\text{avg}} w + b_0 \tag{3.96}$$

$$C_{B,\text{monom,offset}} = a_0 d^{a_1} w^{a_2} s^{a_3} n^{a_4} d_{\text{avg}} + b_0 d^{b_1} w^{b_2} s^{b_3} n^{b_4} \tag{3.97}$$

$$C_{B,\text{monom,simple}} = a_0 d^{a_1} w^{a_2} s^{a_3} n^{a_4} \tag{3.98}$$

$$C_{B,\text{monom,avg}} = a_0 d^{a_1} w^{a_2} s^{a_3} n^{a_4} d_{\text{avg}}^{a_5} \tag{3.99}$$

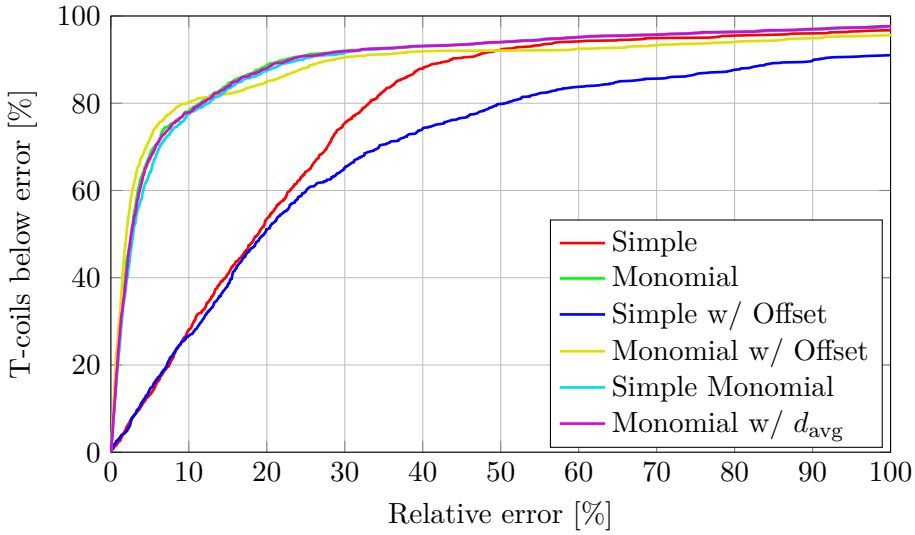



**Fig. 3.33:** Cumulative distribution of the relative error for $C_B$.

| $a_0$ | $a_1$ | $a_2$ | $a_3$ | $a_4$ | $a_5$ | $b_0$ | $b_1$ | $b_2$ | $b_3$ | $b_4$ | BIC |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0.0567 | | | | | | | | | | | 5871 |
| 0.0335 | 0.2538 | -0.451 | -0.014 | 0.0632 | | | | | | | 2306 |
| 0.0471 | | | | | | 2.9314 | | | | | 5599 |
| 0.0035 | 0.5711 | 0.7385 | 0.1111 | 1.4267 | | 0.1756 | 0.9422 | -0.028 | -0.375 | -0.012 | 1746 |
| 0.0160 | 1.5179 | 0.3560 | -0.080 | 0.7515 | | | | | | | 2344 |
| 0.0250 | 0.7545 | 0.4723 | -0.040 | 0.9396 | 0.6038 | | | | | | 2284 |

**Tab. 3.4:** Fitted parameters for the bridge capacitance models. Note the unit of $d$, $w$, $s$, and $d_{\text{avg}}$ is µm, while $C_B$ is in fF, so the units of $a_0$ and $b_0$ follow from this but are different depending on the model.

### Resistance Model

The resistances are the most prominent parasitics associated with the T-coils. For the simulated set of T-coils, only the DC value has been determined and it is basically a function of the inductor length divided by the turn width. The length is proportional to $nd_{\text{avg}}$ for the turns and a piecewise function identical to the one used for the center-tap correction of the inductance. A simple first order model would therefore be given by Eqs. 3.100 to 3.101.

$$R_{1,\text{simple}} = R_{2,\text{simple}} = a_1 \cdot \frac{nd_{\text{avg}}}{w} \tag{3.100}$$

$$R_{3,\text{simple}} = a_3 \cdot \frac{d_{\text{avg}}}{w} \begin{cases} 1, & n \in \mathbb{N} \\ \rho, & (n+0.5) \in \mathbb{N} \end{cases} \tag{3.101}$$

Fig. 3.34 shows that the accuracy is good enough but with some slight modifications, it can be improved substantially. Since the T-coils are not perfectly symmetric, $R_2$ is handled separately in this "complex" model. Also an offset parameter is added to account for vias. For $R_1$, the offset is only applied below a certain turn width $w_v$. Without this a "jump" in error for the last ten percent of inductors can be noticed. This is a side effect of via spacing rules, which suddenly increases the number of vias connecting the spirals above a certain turn width. It occurs in $R_1$ and not $R_2$ because the index 1 is the lower spiral. In cases where the number of vias grows more smoothly, this could of course be proportional to the total via area, e.g. $w^2$. The results in Tab. 3.5 show that the "complex" model also has a much lower BIC.

$$R_{1,\text{complex}} = a_1 \cdot \frac{nd_{\text{avg}}}{w} + \begin{cases} b_1, & w < w_v \\ 0, & w \geq w_v \end{cases} \tag{3.102}$$

$$R_{2,\text{complex}} = a_2 \cdot \frac{nd_{\text{avg}}}{w} + b_2 \tag{3.103}$$

$$R_{3,\text{complex}} = b_3 + a_3 \cdot \frac{d_{\text{avg}}}{w} \begin{cases} 1, & n \in \mathbb{N} \\ \rho, & (n+0.5) \in \mathbb{N} \end{cases} \tag{3.104}$$

| $a_1$[mΩ] | $b_1$[mΩ] | $a_2$[mΩ] | $b_2$[mΩ] | $a_3$[mΩ] | $b_3$[mΩ] | BIC |
|---|---|---|---|---|---|---|
| 23.737 | 0 | 0 | 0 | 13.315 | 0 | -14881 |
| 23.769 | 24.533 | 23.853 | -30.226 | 10.577 | 32.100 | -20879 |

**Tab. 3.5:** Fitted parameters for the resistance models.

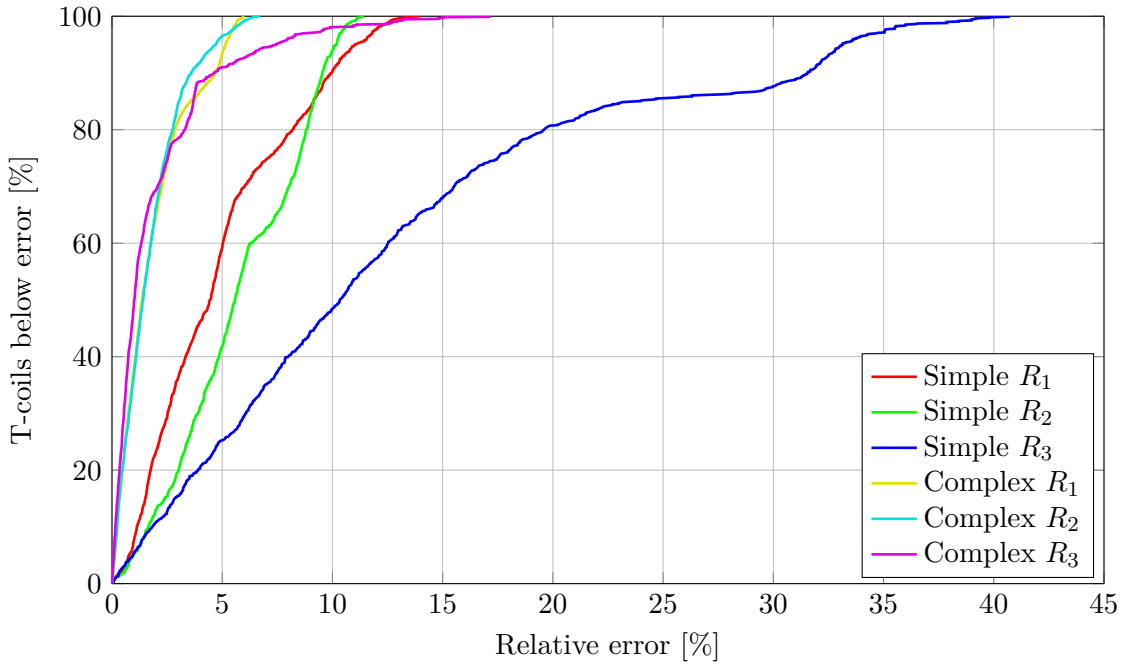



**Fig. 3.34:** Cumulative distribution of the relative error for $R_1$, $R_2$, and $R_3$.

### 3.5.4 Inductor Synthesis

The $M_{\text{T-coil}}$ function cannot be inverted to obtain the inverse function mapping electrical parameters to geometry. However, it can be evaluated extremely fast compared to EM-simulation. It is therefore possible to do something like "inductor synthesis", i.e. automatically find the "optimal" inductor for a given problem. This subsection demonstrates the idea on an example. In this case, an ESD capacitance $C_L = 300\,\text{fF}$ and a termination resistor $R_T = Z_0 - R_1 - R_2$ are connected to the T-coil as shown in Fig. 2.25. The "optimal" T-coil is defined as the one that minimizes the average reflection in the range from 0 to 50 GHz. In case of an actual termination circuit, this could be done in a more sophisticated way, e.g. by applying a weight to favor low reflection at low frequencies, but it is enough to demonstrate the general idea. The optimization has been implemented in Python and delivers a result within in a few minutes. Fig. 3.35 shows the simulated reflection with the lumped model calculated by the optimizer, the actual S-parameters of the T-coil with and without metal fill, and the fitted lumped models derived from the S-parameters.

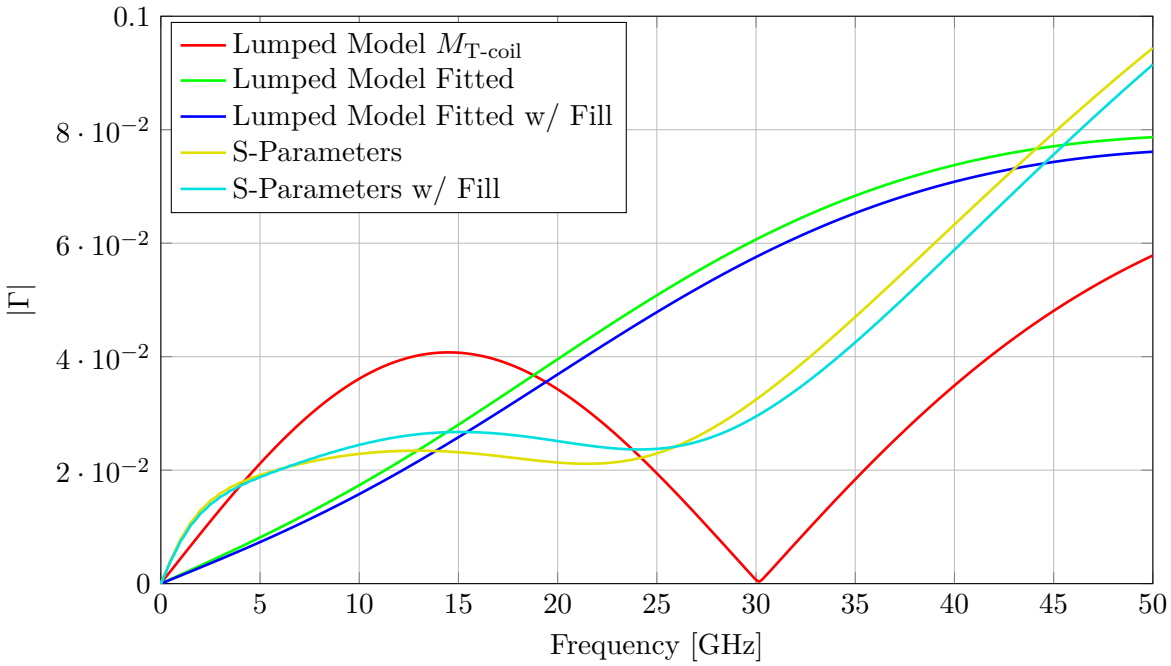



**Fig. 3.35:** Comparison of the different descriptions of the "synthesized" T-coil (53.1 µm, 5.1 µm, 1.2 µm, 2.5).

At first glance, the different curves seem not to be correlated at all, especially the lumped model with the calculated parameters. However, the actual difference is not really significant. The magnitude of the reflection is a value between zero and one, and all five curves are below 0.1. This similarity between the curves becomes especially obvious when compared to the uncompensated circuit shown in Fig. 3.36. It can also be observed that the reflection is not influenced much by the metal fill, so it can well be ignored during the design phase and should not result in surprises when it is included for verification. In fact, including up-scaled metal fill only hurts simulation performance and does at best not introduce additional errors. The lumped model chosen for the T-coil also seems to describe the layout closely enough, despite its simplicity.

Through various tests with the optimizer code it has been observed that, given a constant diameter, the optimal T-coil always was the one with the lowest spacing (here 1.2 µm). This was not anticipated but is logical in retrospect. The larger the T-coil becomes, the higher the series resistance. Thus, if the required inductance can be achieved with a smaller T-coil with fewer and wider turns, it performs better than a larger T-coil that delivers the same inductance. Closer turns increase the mutual inductance between the segments and thus the self-inductance of the T-coil.

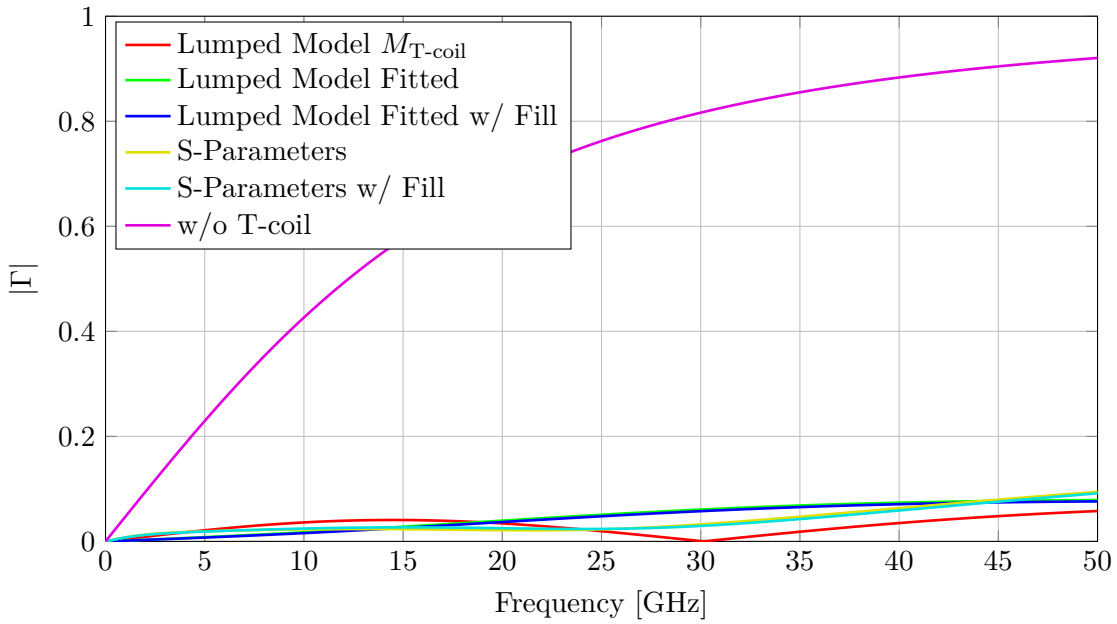



**Fig. 3.36:** A different perspective on the quality of the T-coil model.

### 3.5.5 Conclusion

This chapter, and this section in particular, demonstrated on the example of T-coils, how custom passive on-chip structures built from metal interconnect can be analyzed, modeled, and designed. Simple formulas exist already in literature, but these are always subject to approximations and assumptions that can be easily misinterpreted or overlooked. So relying solely on those is very dangerous and lacks any verification. The only way to verify such structures are field solvers; layout extraction is not suited for frequencies of multiple ten gigahertz. These field solvers are complicated to configure correctly and are also often quite expensive. In this regard, the main conclusion from the experience collected in this work is that a lot of difficulties and uncertainties can be avoided if a field solver is selected that can accurately read in the stack-up descriptions provided in the PDK. Writing custom stack-up files is tedious, error prone, time consuming, and not necessarily possible at all due to encrypted PDK files. However, in the case simulations of such structures can be performed, this section detailed a way to map the resulting S-parameters to lumped models, which are often used for circuit design because they facilitate optimization. Depending on the complexity of the structure that is to be designed, a smaller number of simulated S-parameters may already be enough to interpolate the design space and create a model that can be used to determine the optimized layout. This is a much more directed approach than “trial and error” iterations. It also provides the designer with insight on the problem and a sense for the quality of the results.

## 3.6 Test Structures

This section describes the layout of the RX termination and measurement results on silicon test structures, including some lessons learned.

### 3.6.1 Layout

The schematic of the RX termination is presented in Fig. 3.37. The Analog Front-End (AFE), the secondary diodes, and the CDM protection resistor are connected to the all-pass point behind $L_2$, so only the primary diodes are compensated. The termination resistors can be tuned and the input voltage of the AFE is adjusted with an operational amplifier in feedback configuration. A simplified version of the corresponding layout is shown in Fig. 3.38. Only the upper metal layers are present and the lower parts of the building blocks are abstracted with a red box. An exception to this is the power grid of the AFE, which is not present, and the excluded metal fill.

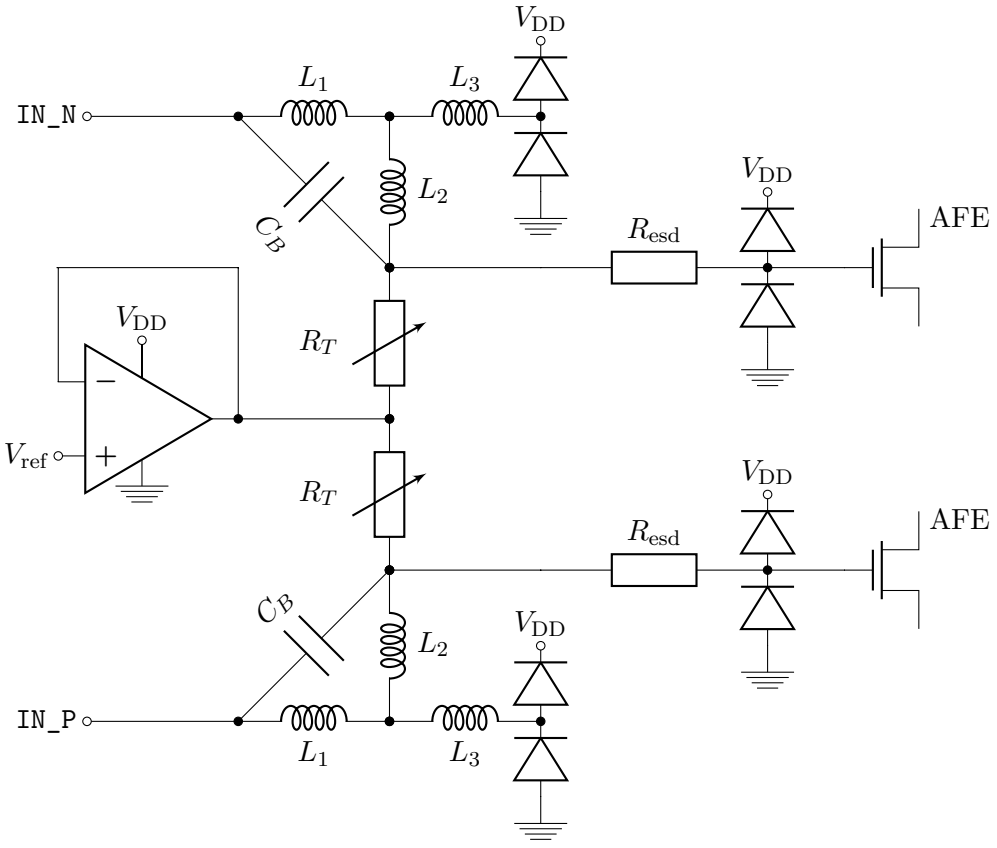



**Fig. 3.37:** Simplified schematic of the RX termination and ESD protection (w/o power clamp).

The power is supplied through the wide tracks on the left ($V_{DD}$) and the right ($V_{SS}$),

and the octagonal pads are `IN_N` and `IN_P`. To reduce the capacitive coupling to the pads, the T-coils are placed in between them. The area above them is exclusively used for circuits regarding termination and ESD protection, while the AFE and the remainder of the RX (not shown here) is placed below them. The power clamps are located at the top to leave room for the other blocks and to prevent too much power grid below the upper pad. To reduce the deviation from the lumped model as much as possible, the primary diodes and the termination resistors were placed close to the T-coil. However, the diodes also need a low impedance connection to the power clamps via the power and ground rails. The one-sided power rails complicate this matter as only one connection can make use of the thick redistribution layer on each side, while the other connection has to be provided via the power grid on lower metal layers. The center node of the termination resistors is connected to the operational amplifier, which receives its reference voltage from a dedicated digital-to-analog converter. The thin wires through the CDM resistor are leading to the AFE, which has the secondary ESD-diodes in close proximity.

A major challenge in the design of this structure was the fact that it could not be simulated in its entirety with a field solver. This is theoretically possible with EMX, however EMX was not available at this time, and it is currently still impossible due to missing EMX files for this particular stack-up. Thus, the design process was simplified to only simulating the T-coil with ADS and Sigrity, neglecting further interactions to surrounding structures and contributions of wires connecting to the T-coil. In addition, the pad capacitance was not included in the design process for simplicity as the hassle with the field solvers and technology file did consume a lot of time. As discussed in section 2.3, this certainly leaves a lot of performance on the table. As a consequence of the missing attention to the pads, the power grids below them differ as the AFE is not a part of the termination top level cell, which causes an unknown skew in capacitance. Furthermore, the remaining free space was filled automatically, introducing further uncertainties.

Considering the significant impact of, e.g., the pad and the center-tap wire, it is obvious that optimizing only the T-coil is simply not enough. The structures below the pad and the wires connecting the pad, the diodes, and the termination resistor to the T-coil also have to be carefully designed. With the tools and experience from this work, it might be promising to pursue a more monolithic approach in the future. The circuit and layout described in this subsection is currently split into several cells in the Virtuoso library. It would be much more convenient in terms of controlling parasitic effects if most structures on the three uppermost layers of the entire area shown in Fig. 3.38 were designed and EM-simulated as a single cell. This could include an identical power grid and manually placed metal fill below both pads to lower their capacitance skew. The structures placed by the AFE designer as well as the other termination circuits below the upper pad should then only contribute a second order effect, leaving much less room

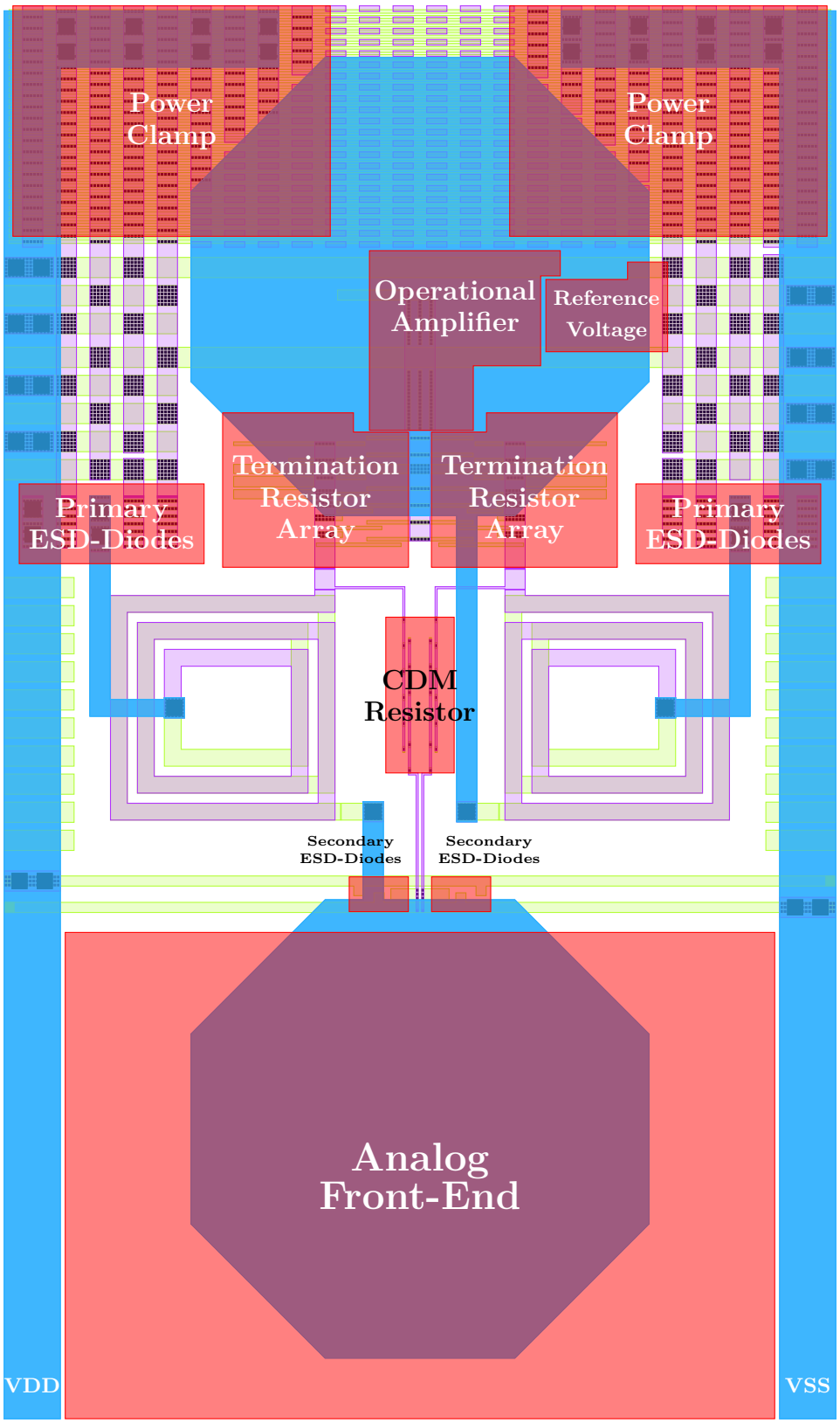



**Fig. 3.38:** Simplified layout of the RX termination and ESD protection (AFE power grid is not shown).

for post-layout “surprises”.

As an outlook: this monolithic cell could be optimized by first finding the minimum possible pad capacitance (lower is always better in this context) and then building a PCell that can automatically adjust and connect the T-coil. In case only a slow and complicated field solver solution exists, the approach of the previous section might be most useful, i.e. fitting lumped models to a set of simulated designs to interpolate the design space. However, with a fast simulator like EMX, it could actually be more feasible to simulate a large number of such structures and sweep over the set of S-parameters in the Virtuoso simulator environment. In any case, there is much to be gained from further optimization in future designs.

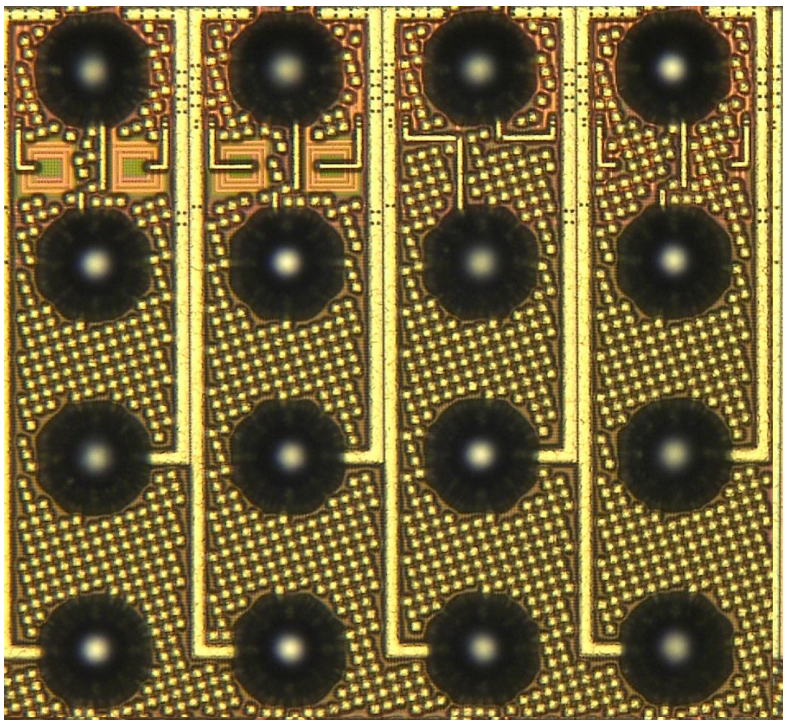
### 3.6.2 Measurements

To measure the effects of different T-coil layouts a small set of test structures was manufactured in a 22nm node. They consist of variations of the layout shown in Fig. 3.38 and are all simplified versions of the RX with everything but the termination and ESD devices removed. The AFE has been replaced with an equivalent metal capacitor. This was mainly done for simplicity as it also removed the need to connect all the configuration signals of the RX, which could have interfered with the productive part of the chip.

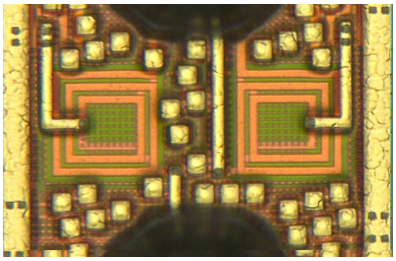
Fig. 3.39 shows micrographs of the four variants. The first is the same structure that was also included into the productive SerDes lanes – but with the RX removed, as mentioned before. This aimed at enabling a comparison of how much this changes the termination. The second version is a T-coil with narrower turns and an additional half turn, which results in a higher inductance. The third option is designed simply without a T-coil so the pads are directly connected to the primary diodes. Last but not least, the fourth version again uses the T-coil that was also included into the productive lanes but without special inductor fill rules, and therefore an increased amount of fill. Additionally, it also comprises a grid of decoupling capacitors (decap) placed below the inductors. These four structures were mainly intended to assess how much care has to be taken when designing T-coils.

The measurements were conducted with a TDR. A vector network analyzer could have provided more accurate results but was not available at the time. Before the actual measurement, the differential DC termination resistance of each lane was tuned with the termination resistor $R_T$ to be as close as possible to $100\,\Omega$. Then, the impedance of the test structures and a productive SerDes lane was measured. The raw results are plotted in Fig. 3.40.

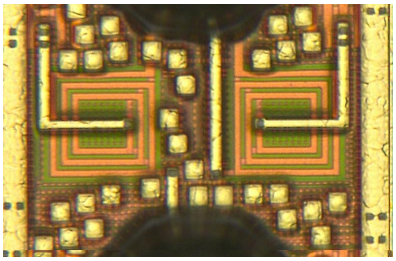
The first two spikes are aligned for all curves and are caused by the connector between the cables and the test PCB. The dips between 2.5 and 3.0 nanoseconds are caused by the package and on-chip termination. To facilitate a comparison, Fig. 3.41 shows a



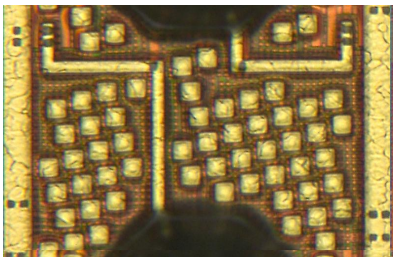
**(a)** Complete view of the test lanes.



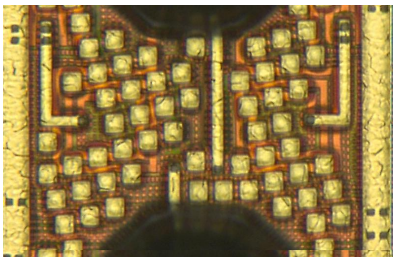
**(b)** RX T-coil.



**(c)** Alternative T-coil.



**(d)** Without T-coil.



**(e)** RX T-coil with decap and metal fill.

**Fig. 3.39:** Micrographs of all four termination test structures. The RX AFE was replaced with a metal capacitor and the remaining RX was replaced with metal fill.

zoomed-in version. All five curves are aligned on the falling edge of the package dip. This was necessary because the trace length on the PCB is different for each lane. Without a T-coil, the termination resistor needs to be set to a higher value – as expected. Note that the values enclosed by the brackets are the typical values of $R_T$ expected at the corresponding settings, not the measured values. $R_T$ is even above $50\,\Omega$ without a T-coil, which is probably due to process variations. The alternative T-coil requires a lower $R_T$ because it has a higher series resistance, while the remaining three are identical because they use the same T-coil. It is clearly visible that the variant without a T-coil (yellow) performs worst by far in terms of reflection. Next is the T-coil surrounded by regular metal fill and decoupling capacitors (light blue). Placing such structures near the T-coil is obviously affecting the termination quality. Surprisingly, the alternative T-coil (dark blue) with its higher inductance performs better than the T-coil that has also been applied in the productive SerDes lanes (green). The reason for this is that the T-coil for this tape-out was designed to compensate only the primary ESD diodes and the pad capacitance was completely neglected. In section 2.3, it is shown that the inductance of a symmetric T-coil has to be larger when a pad capacitance is present. This also explains why the same T-coil performs differently in the productive SerDes lane (red) than within the test structure. By reducing the RX layout to a metal capacitor and metal fill, the capacitance of the pad above the RX AFE changed significantly. These results strongly support the previous findings that structures below the pad have to be planned with much more care in future designs.

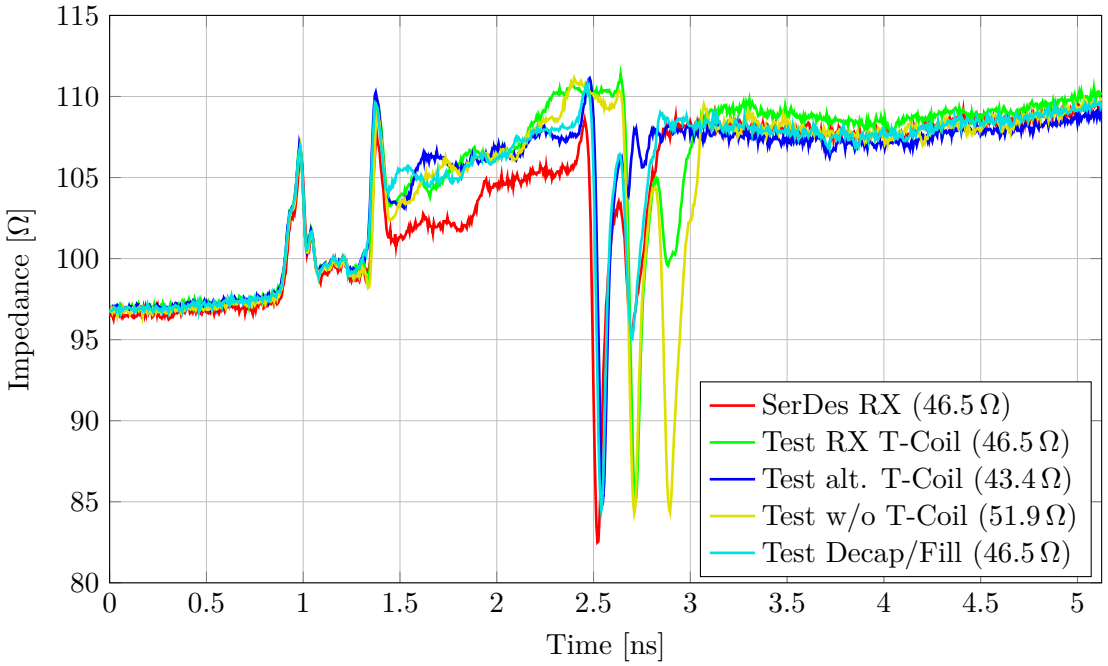



**Fig. 3.40:** Full impedance profiles. The expected value of $R_T$ is given in brackets.

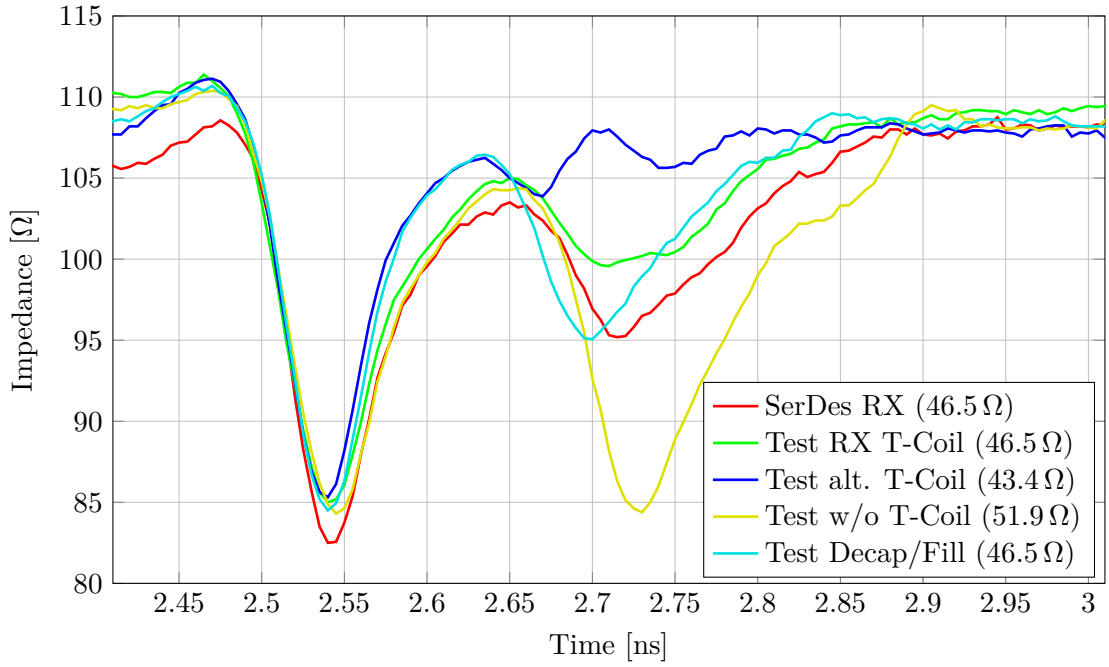



**Fig. 3.41:** Aligned impedance profiles to compare the on-chip termination. The expected value of $R_T$ is given in brackets.

Chapter 4

# Floating-Point Arithmetic

High Performance Computing systems increase their performance over time due to technology enhancements, larger systems, and more efficient architectures. This performance increase is displayed in the TOP500 list [74], a ranking of the world's most powerful HPC systems. The unit of performance is TFlop/s, i.e. floating-point operations per second. These performance values are assessed with the LINPACK benchmark [75], a linear algebra code written in Fortran. This demonstrates that floating-point calculations are at the center of high-performance computing.

This chapter now switches the focus from I/O technology to the second part covering computing. The first section provides an introduction to floating-point numbers as a preparation of the following sections. In the second section, an overview of a FMA floating-point unit developed by Kaiser at the CAG at Heidelberg University is presented [7]. Kaiser also implemented a Universal Verification Methodology (UVM) based verification environment to ensure correctness of the design. This thesis extends the work done by Kaiser in two ways: Firstly, with an additional verification testbench that executes test patterns generated by the TestFloat software [76], [77], and secondly, with a Power-Performance-Area (PPA) analysis of the FMA unit. Most of the content of the second section, including the results, has been published in an issue of *Supercomputing Frontiers and Innovations* [15].

## 4.1 Number Formats

This section aims to introduce the basics on binary number formats needed for the remainder of this chapter. As the name "computer" suggests, the primary purpose of digital circuits is to perform computations, which requires a representation of numbers.

This representation will always be a finite set of values, thus the fundamental issue that needs to be solved is to define an efficient encoding that contains the numbers commonly used and that is "hardware-friendly" to ensure fast execution of calculations. The simplest mapping is found for natural numbers as their binary representation, extended to a certain number of bits, is directly used as encoding scheme. These are called *unsigned integers*. Whole numbers can be defined by using the *two's complement*, called *signed integers*. Most numeric applications, e.g. simulation codes, are using real numbers that include a decimal point. However, real numbers are not countable and as such it is not possible to map them in a continuous way on a finite set of values as it is done with natural numbers. There will always be an infinite amount of real numbers between each pair of representations. This is usually not an issue but opens the possibility to make trade-offs in terms of accuracy, dynamic range, and computation performance. An easy way to implement real numbers are *fixed-point* numbers. They are integers with an implicit fixed scaling factor that is only applied when the number representation is to be interpreted. Their main drawback is the limited dynamic range, which depends only on the number of bits used for their representation. In contrast to this, there are *floating-point* numbers, which have an explicit variable scaling factor, i.e. one encoded into the binary representation. This allows to define the dynamic range without increasing the number of bits used to represent those values. The discussion in this section is mostly limited to IEEE 754 and Posit floating-point numbers.

### 4.1.1 IEEE 754

The arguably most commonly known floating-point number format is described by the IEEE 754 standard, with its latest iteration being from 2019 [78]. This number representation is derived from the normalized scientific notation, which is shown in Eq. 4.1. Here $\mathcal{B}$ is the corresponding base of the numeral system. The IEEE 754 standard defines binary and decimal flaoting-point representations, this section, however, will focus only on binary numbers.

$$x = a \cdot \mathcal{B}^b \text{ and } 1 \leq |a| < \mathcal{B} \tag{4.1}$$

The IEEE 754 format splits the representation of a number into three parts. A single bit encodes the sign of $a$, a certain number of bits the exponent $b$, and the remaining bits the absolute value of $a$.

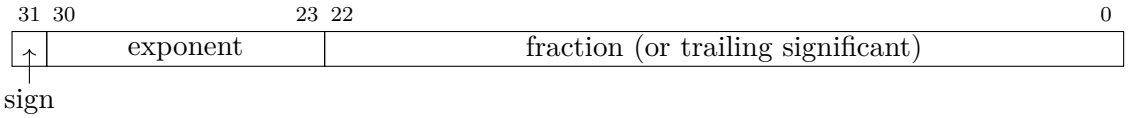



**Fig. 4.1:** 32-bit IEEE 754 binary floating-point number representation.

The *normalized numbers*, i.e. not the special cases, represented in the bit pattern shown in Fig. 4.1, are interpreted as shown in Eq. 4.2.

$$\text{represented value} = (-1)^{\text{sign}} \cdot \mathcal{B}^{\text{exponent}-\text{bias}} \cdot 1.\text{fraction} \tag{4.2}$$

The exponent is encoded as an unsigned integer in this format, so it is possible to compare numbers by comparing their representation (ignoring the sign bit). Thus, the bias is only needed when the representation is interpreted. It is defined as $\text{bias} = 2^{\text{size(exponent)}-1} - 1$ for an almost symmetrical exponent range around zero. Since the numbers are normalized to avoid duplicate representations for the same number, the "1" before the fraction is actually a constant for binary numbers. Hence it is not encoded, which allows to store an additional fraction bit. Due to not being encoded but implicitly assumed, it is called the *hidden bit*. Together with the fraction it is called the *significant*.

There are quite a few special values besides these normalized numbers. They are described by two special exponent values. If the exponent is zero, the number is either positive or negative "0", or a *subnormal number*, depending on whether the fraction is zero or not. The subnormal numbers are number smaller than what can be achieved via the smallest exponent. They provide a tapered way towards zero by changing the hidden bit to "0" and are needed in particular to avoid underflows from additions and subtractions. An *underflow* occurs if the result of an operation is smaller than the smallest representable number. This is avoided as the smallest subnormal number is equal to the smallest difference of two numbers.

If the exponent is equal to $2^{\text{size(exponent)}} - 1$, it either represents positive or negative infinity, or a *not a number* (NaN), again depending on whether the fraction is zero or not. NaNs are needed to represent results of operations that are outside the representable range, e.g. imaginary numbers. If the Most Significant Bit (MSB) of the fraction is "1", then the NaN is "quiet" (qNaN), else it is a "signaling" NaN (sNaN). The latter will cause an exception to signal the occurrence of a NaN, while the former will continue operations and keep quiet about the NaN. The standard suggests to encode diagnostic information into qNaNs [78].

| Total Size [bit] | Exponent Size [bit] | Fraction Size [bit] |
|---|---|---|
| 16 | 5 | 10 |
| 32 | 8 | 23 |
| 64 | 11 | 52 |
| 128 | 15 | 112 |

**Tab. 4.1:** IEEE 754 binary floating-point formats [78].

Tab. 4.1 shows the various sizes defined for binary floating-point numbers in the IEEE 754 standard. The parameters for larger sizes can be derived by a formula given in the

standard but are not shown here as they have less practical relevance.

### 4.1.2 Posits

The IEEE 754 format uses fixed sizes for the exponent and fraction fields, which largely decouples magnitude and precision. Independent of the relative size of two numbers in this format, their precision is identical until the borders of the format are reached, i.e. the subnormal numbers or infinity. An alternative to this is a *tapered* floating-point format, which reduces precision gradually for very large and very small numbers by including a scheme to vary exponent and fraction size. The idea is to provide higher precision at values close to one. A tapered format has been first proposed by Morris in 1971 [79]. In 2017, Gustafson proposed a new tapered floating-point format called *posits* [80]. Except for the sign, the remaining parts of the format can vary in position and size as the *regime* field can grow to the right. This is exactly what makes these numbers tapered floating-point numbers. An example is shown in Fig. 4.2.

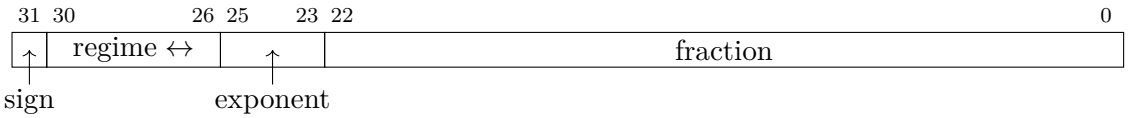



**Fig. 4.2:** 32-bit posit floating-point number representation ($es = 3$). Note that the regime field can grow to the right.

Exponent and fraction are interpreted identical to IEEE 754 floating-point numbers, except that there is no exponent bias and that they can be pushed to the right (partly and even entirely) to make room for a growing regime. The regime is a unary coded (thermometer code) "super exponent", i.e. it is always presented as a sequence of identical bits terminated with the opposite bit. This causes the minimal regime length to be two bits for posits with a total of more than two bits and only one bit for 2-bit posits. This shows that if the regime grows up to the Least Significant Bit (LSB), the implicit bit at position "-1" is assumed to terminate the regime. Two parameters are needed to describe a posit format. Gustafson chose the total width $N$ and the exponent size $es$. The behavior of the regime can be understood relatively easily with the following procedure: The value 1.0 is encoded in the posit format with a "1" at position $N - 2$. When it gets incremented, the exponent field will start to fill up until it saturates. This is when the regime starts to grow by one bit to the right, decreasing the size of the fraction by one, and the exponent field is "reset" to zero. Therefore, an additional regime bit actually scales the represented value by $useed = 2^{2^{es}}$. The unary coding of the regime sometimes leads to the first impression that the format is very wasteful. However, the opposite is true as it uses every available bit pattern for a meaningful number and does

not even include NaNs. There are only two special values, 0 and *not a real* (NaR). The NaR also represents $\pm\infty$. Gustafson often presents the posit encoding scheme mapped to the projective reals, a circular real axis. A 5-bit version ($es = 1$) is shown in Fig. 4.3.

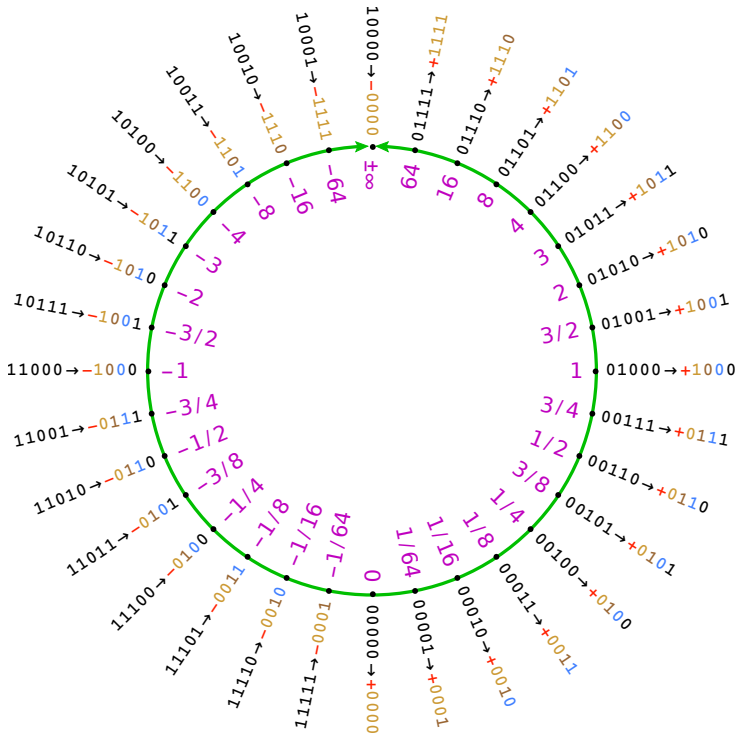



**Fig. 4.3:** 5-bit posits with $es = 1$ visualized on the projective reals. Taken from [81].

The visualization shows that the posits have the same ordering as signed integers using the two's complement. Furthermore, a posit is negated by taking the two's complement. The posits originated from *universal numbers* (unum) proposed by Gustafson [82]. In fact, posits are also called Unum Type III. Unum Type I is a variable size floating-point format for interval arithmetic and thus not very suitable for hardware implementations. Unum Type II on the other hand uses a constant size and an encoding similar to posits. However, there are no fields for regime, exponent, and fraction. Instead, a set of $2^{N-3} - 1$ real numbers $x_i$ is chosen to define how the type II unums are interpreted. As shown in Fig. 4.4, this has the advantage that division can be conducted by mirroring the number on the horizontal axis. However, multiplication of these numbers is now very expensive as the value of a number cannot be extracted by interpreting a fraction like an integer anymore. Instead, lookup tables for the $x_i$ are needed. As this is only feasible for low sizes (below 20 bit [80]), Gustafson invented the posits that provide a linear fraction, which can be utilized by existing multipliers but therefore have to use more expensive division. Nevertheless, this trade-off is extremely interesting.

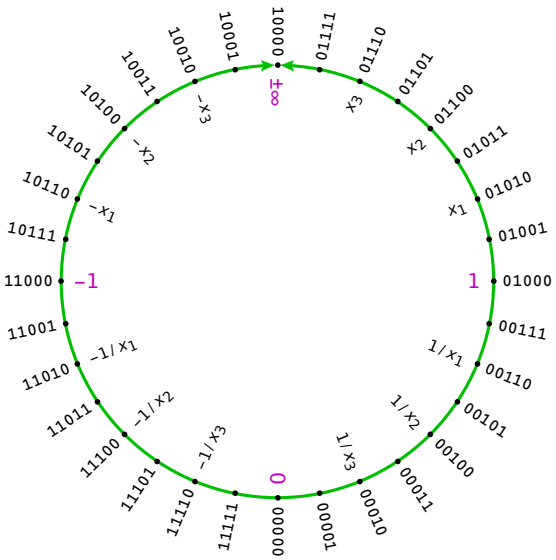



**Fig. 4.4:** 5-bit unum type II defined with $x_1$, $x_2$, and $x_3$, visualized on the projective reals. Based on [80].

In the last years, a debate started on whether posits are superior to IEEE 754 floating-point numbers or not. The discussion has spread on hardware efficiency, low level math routines, algorithms, special corner cases, and many more, where either one or the other expose deficiencies or shows strengths. Quite a few of these have been discussed by Dinechin et al. in [83]. It is beyond the scope of this work to take a closer look on each of these aspects, but a comment on the hardware effort should be given here. The decoding of special values is often attributed to being a drawback of IEEE 754 floating-point numbers. However, posits on the other hand do require a leading zero counter to decode the regime size. Once decoded, the internal data-path for both formats is actually rather similar. The draft posit standard [84] defines $es = \log_2(N) - 2$ for $N \in \{8, 16, 32, 64\}$, which leads to a maximum significant size of 58 bit for $N = 64$. This is 5 bit larger compared to 64-bit IEEE 754 numbers, rendering the multipliers and adders slightly larger and therefore also slower. Concerning rounding, posits are expected to be somewhat easier on hardware resources as the IEEE 754 requires more complicated rounding modes. Overall, hardware effort should be similar for both formats although this has not actually been demonstrated in this work. An FMA posit unit including support for the *quire*, an accumulator register, has been designed and verified by Melzer in a master's thesis at the CAG [85].

### 4.1.3 Machine Learning

Machine learning and artificial intelligence have attracted a lot of attention and research resources in the last decade. The training of neural networks is a demanding task for

commodity hardware like Central Processing Units (CPUs) and Graphics Processing Units (GPUs). A major reason for this is the overhead associated with the floating-point format used to calculate the weights and structure of neural networks. It turns out that 64-bit or 32-bit floating-point values offer way more precision than needed in many cases. By lowering the number size further, significant speed-ups can be achieved at negligible accuracy loss. Therefore, new formats have been developed in the last years to reduce the memory bandwidth and arithmetic hardware size. NVIDIA derived a 19-bit format called TensorFloat-32 (TF32) by using the exponent size of IEEE 754 32-bit numbers and the fraction size of IEEE 754 16-bit numbers [86]. Another, more popular format is Google's *bfloat16* format, which is basically the upper half of the IEEE 754 32-bit format [87]. It is already applied in production use in Google Tensor Processing Units (TPUs). The format has gained relatively high attention and even papers on hardware implementations have been published, e.g. Hutchins et al. present a FMA bfloat16 unit [88]. A common function used in neural networks is the *sigmoid* function calculating $1/(1+\exp(-x))$, which is extremely expensive to compute. Therefore, simpler approximations like rectifier functions are often used. Gustafson points out that 8-bit posits with $es = 0$ are perfectly suited for neural network training and especially to compute the sigmoid function. Flipping the first bit of these posits and shifting them right by two places approximates the result of a sigmoid function extremely well [80].

## 4.2 Fused Multiply-Add

The results presented in this section have been published in vol. 6 no. 2 of *Supercomputing Frontiers and Innovations* in 2019 [15].

### 4.2.1 Introduction

The RISC-V ISA developed at the UC Berkeley has become extremely popular since its inception in 2010. Large technology companies have started to support RISC-V, and first implementations and software support are now available. The open license of RISC-V differentiates it fundamentally from other ISAs like x86 or ARM. Traditionally, high costs associated with licensing have prevented open source processor designs for these ISAs, especially at universities. Therefore, RISC-V is said to spawn a "new era for computer architecture". Thus, a major motivation for the work presented in this section was the goal to eventually develop a RISC-V processor core at the CAG at Heidelberg University. Other companies and universities have also developed RISC-V processors in the last years. Most prominently SiFive, a spin-off of the UC Berkeley, has already produced RISC-V silicon, released on its HiFive prototyping boards [89]. Furthermore, there are several variants of Western Digital's SweRV core targeting primarily embedded tasks, like state machines and flash controllers [6]. The most notable university projects in this area are the PULP Platform (Parallel Ultra Low Power) at the ETH Zurich and the Berkeley Out-Of-Order Machine (BOOM), as well as the Rocket Chip Generator at the UC Berkeley. SiFive is offering first high performance designs today, but a few years ago, the focus in the RISC-V community was primarily on low-power embedded applications. Therefore, development at the CAG was aiming towards the HPC direction, and a key step to establishing RISC-V for HPC is to provide fast floating-point arithmetic in hardware. This is underlined by the most prominent ranking of HPC systems, the TOP500 list [74]. The metric used to compile this list is the floating-point performance based on the LINPACK benchmark [75]. This is a linear algebra code that performs matrix multiplications, which involve scalar multiplications with subsequent additions. To improve the execution of such operations, so called Fused Multiply-Add units are commonly found in HPC processors. The fusion of both operations poses an advantage in speed and accuracy over two independent operations. Kaiser has designed and verified such an FMA unit in the context of a master's thesis at the CAG [7]. It supports 64-bit IEEE 754 floating-point numbers. The IEEE 754 standard leaves some implementation details open to the designer, e.g. which rounding modes to implemented or the behavior when *tinyness* is detected before or after rounding [90]. Hence, results computed on different systems can vary, which is often criticized by supporters of the posit format. The RISC-V standard avoids such differences in functionality between different RISC-V

compliant Floating-Point Units (FPUs) by fixing these decisions. However, they are not identical to, e.g., Intel FPUs.

At the time when Kaiser started the development of the FMA unit, there was no high-speed open-source hardware implementation of the IEEE 754 floating-point standard available. The PULP Platform only contained a preliminary 32-bit FPU [91], which has now been replaced by a design called "fpnew" [92]. The old design was also missing the rounding mode *roundTiesToAway*, which is mandatory for RISC-V. The arguably most widely known implementation of a RISC-V FPU is the HardFloat developed by Hauser at the UC Berkeley [93], [94]. It is used within different cores, or core generators, like the SOC generator Rocket [95] and the out-of-order core Berkeley Out-of-Order Machine (BOOM) [96]. The fastest Rocket implementation (in 2017) is SiFive's U54 Rocket on the TSMC 28nm HPC process with 1.5 GHz [97]. HardFloat can be expected to be a reliable implementation due to its appearance in multiple projects and existing instances in silicon. However, it is developed using the high-level hardware generation language *Chisel* (**C**onstructing **h**ardware **i**n a **S**cala **e**mbedded **l**anguage) [98]. Chisel is built on Scala, a language based on the Java virtual machine. It was created at the UC Berkeley in 2010 with the intention to increase the design productivity and reusability of code. Therefore, Chisel offers the possibility to write hardware generators. A drawback of a new language is the compatibility with back-end tools. The current solution of this problem is an automated translation to Verilog if the design is to be synthesized. This process does contain a lot of "expert knowledge". We decided not to go the Chisel route and used SystemVerilog instead, in order to make potential low level optimizations easier to add later on.

### 4.2.2 FMA Unit Design

This subsection will provide a brief overview of the FMA design. As shown in Tab. 4.2, the FMA unit supports all four double-precision fused operations defined in the RISC-V ISA, as well as add, subtract, and multiply. It supports all rounding modes required by the IEEE 754-2008 standard and additionally roundTiesToAway, which is mandatory for RISC-V. Division and square root have been designed and verified by Henger at the CAG in a master's project [99]. The division and square root use a Goldschmidt and Newton-Raphson algorithm, respectively – i.e. iterative approaches, calculating multiple intermediate results with the FMA unit.

Fig. 4.5 shows the interface and architecture of the FMA unit, which is based on [100]. It comprises the three 64-bit inputs `port_a`, `port_b`, and `port_c` for the operands, and the 64-bit wide output `port_res` for the result. The type of operation is determined by `op`, the rounding mode by `rm`, and exceptions are signaled at the output `exception_flags`.

| Instruction | Description | Operation |
|---|---|---|
| FADD | Add | $A + C$ |
| FSUB | Subtract | $A - C$ |
| FMUL | Multiply | $A \cdot B$ |
| FMADD | Fused Multiply-Add | $A \cdot B + C$ |
| FMSUB | Fused Multiply-Subtract | $A \cdot B - C$ |
| FNMSUB | Negative Fused Multiply-Subtract | $-A \cdot B + C$ |
| FNMADD | Negative Fused Multiply-Add | $-A \cdot B - C$ |

**Tab. 4.2:** RISC-V floating-point instructions supported by the FMA design.

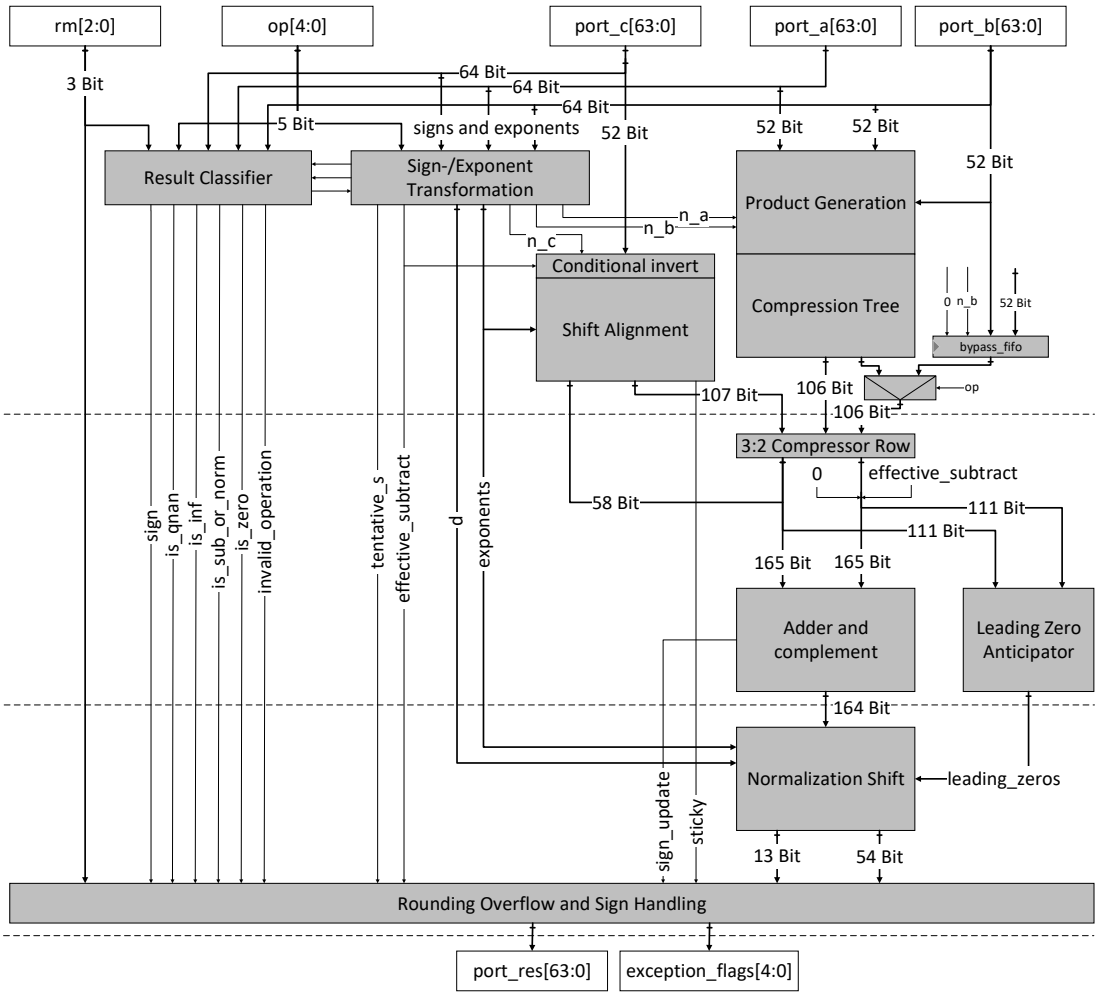



**Fig. 4.5:** FMA unit architecture, including three pipeline stages (dashed lines) [15].

A forward flow control (not shown here) is implemented via `valid_in` and `valid_out`. `valid_in` can also be used for clock-gating inside the FMA unit. In the following, the main components of the design are described in more detail. The Sign-/Exponent Transformation transforms the operand exponents from the biased representation into 2's

complement and checks if the operands are normal, i.e. no special values. Furthermore, it calculates the difference between the product's exponent ($e_A + e_B$) and the addends exponent ($e_C$). It also generates an effective subtraction bit indicating if the absolute values of $A \cdot B$ and $C$ are added or subtracted. The Result Classifier handles operations with special values, such as qNaN, sNaN, zero, infinity, and subnormal numbers. Product Generation and Compression Tree perform the main part of the multiplication. Therefore, they take the significants of the operands $A$ and $B$ and provide the product in carry-save representation, i.e. a carry and a sum vector, which need still to be added. This allows to take the additional 3:2 Compressor Row to add another operand (significant of $C$) at the cost of a single full-adder delay. This is the reason why fusing both operations is more energy efficient. A more detailed explanation of this is given in chapter 5. For a floating-point addition, it is necessary to align the addends according to their exponent by shifting one of them relative to the other. This is done by the Shift Alignment in parallel to the compression tree. There is the case where one addend is much larger than the other, so the smaller one is completely absorbed and doesn't change the result. The Adder and Complement resolves the carry-save representation as well as the following 2's complement into a 1's complement intermediate result. In parallel to the Adder, a so called Leading Zero Anticipator estimates the leading zeros of the intermediate result for the normalization. The latter is then done by the Normalization Shift. Afterwards, the result is finalized by the Rounding, Overflow and Sign Handling unit, which determines if there is an overflow and performs rounding based on this information.

### 4.2.3 FMA Unit Verification

The FMA unit does not have a complex state, but it can still not be verified by iterating through all possible inputs. There are two methods for verification, formal or simulation-based. Although there have been formal verifications of FPUs in the last years [101], [102], the proposed design is verified in a simulation-based way due to the larger amount of time needed for the corner cases in the execution of the formal methods [103], [104]. Since a verification environment for an FMA unit does not have to react to its internal state, stimuli for it can be generated statically. It can also be generated offline, which both increases the performance of the tests. Such a stimuli generator is presented by Aharoni et al. and works using a constraint solver [105]. Unfortunately, the actual code is not open-source, only a set of pre-generated single-precision inputs. Consequently, we decided to use the industry standard simulation-based UVM utilizing the Specman $e$ hardware verification language.

Key challenges of verification are the reference model and the generation of test patterns that actually reveal hidden bugs, since not all input combinations can be tested. The latter is addressed by the use of the UVM, which performs constrained-random online

stimuli generation and thus also facilitates automation. To keep track of which parts have been tested, code coverage as well as functional coverage are collected. The next aspect is checking the behavior of the Design Under Verification (DUV), which a reference model is needed for. It allows to automatically generate the answer to the question whether a transaction has been executed correctly by DUV or not. Behavioral models are the common way to solve this issue and are usually developed by a verification engineer for the specific design. Fortunately, there are already existing implementations of the IEEE 754 standard. However, executing the FMA operation simply in *e* is not a valid solution. The abstraction layers between *e* and the actual instruction executed prohibit any control on how the multiplication and addition are actually executed. Separate instructions produce different results compared to fused operations. To circumvent this, we integrated the FPU of an Intel processor using the Function Level Interface (FLI) provided by Specman *e*. This interface is used to execute C code containing Intel Intrinsics [106], i.e. essentially assembler code to execute the exact FMA instruction that the DUV is executing. The physical Intel FMA unit can likely be assumed as correct and is also an extremely fast reference model.

However, this C code is not sufficient as a reference model for a RISC-V-conform FPU. As mentioned before, the implementation details are slightly different between Intel and RISC-V. To also verify these differences, another reference model has been integrated. It was developed by Hauser to verify the HardFloat FPU and is a software implementation of the IEEE 754 standard written in C, called SoftFloat [107], [108]. Instead of just filling the holes using this model, it was integrated in parallel to the Intel model and if possible, both models were also checked against each other. A simplified view of the UVM monitor is shown in Fig. 4.6.

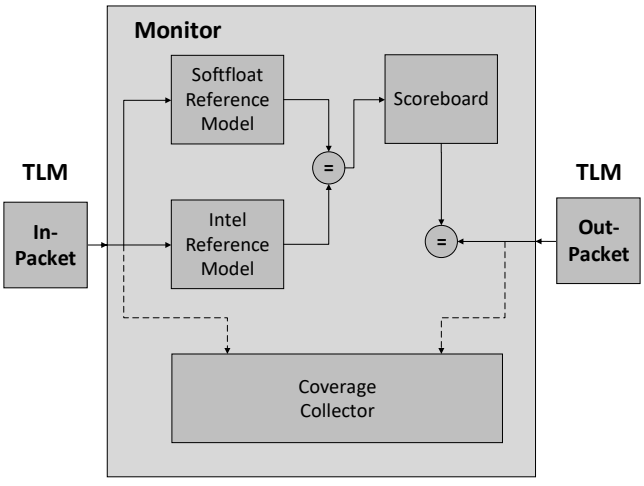



**Fig. 4.6:** UVM monitor using the Intel Intrinsics and the SoftFloat reference model [15].

#### TestFloat

Hauser also created a tool that generates tests for IEEE 754 implementations, called TestFloat [76], [77]. A small testbench that executes the TestFloat sequences has been developed in the present work to complement the UVM-based verification environment created by Kaiser [7]. TestFloat is a command line tool and simply prints lines into the terminal (stdout), each containing input and expected output values for a given type of floating-point operation as hexadecimal values. The expected result and exception flags are computed by TestFloat via SoftFloat. Of course the design has already been verified with SoftFloat, but the reason for also performing verification with TestFloat is mainly to make use of the different generation algorithm. Thus, it is possible to use it for offline generation. However, redirecting all the generated tests into a file and reading them back during simulation is difficult due to the large file sizes. To circumvent this, the read-in of the tests has been converted into a pseudo-online generation by reading directly from stdin – so each test is completed before the next test can start and no expensive file I/O is needed. The code of the driver and checker module is shown in Lst. 4.1. It reads the test values from stdin and drives them to the DUV. It also keeps a small queue of the last five results and exceptions to compare them to the delayed output of the FMA pipeline. TestFloat specifies two levels of testing with a different number of tests. However, simulator speed currently limits testing to the around six million tests in level 1, which needs around 38 min. Level 2 comprises around 180 billion tests, which would demand spending effort on parallelization. (The verification with TestFloat is not included in [15].)

### 4.2.4 FMA Unit Back-End

The back-end implementation in this work was done in a recent 22nm Fully-Depleted Silicon-On-Insulator (FDSOI) technology. A silicon-on-insulator process applies an additional insulator layer to remove the diodes between drain/source and the substrate. Furthermore, the channel is fully depleted, i.e. not weakly doped, to reduce leakage current. The insulator layer acts like a back-gate, which can be used to modify the transistor's threshold voltage. A back-gate bias voltage generator can later be used to apply a voltage to the back-gate, called Body Bias (BB). This allows to tune the circuit for either more performance, or lower leakage, or to compensate process corners. The latter already emphasizes that this bias needs to be treated like process, voltage, and temperature in static timing analysis. The GF22FDX process offers four types of transistors for different applications: high (HVT), regular (RVT), low (LVT), and super low (SLVT) threshold voltage devices. Since no low power implementation was intended but instead a performance analysis, the SLVT standard cell library was chosen.

```
module testfloat_stdin #(
  parameter MANTISSA_WIDTH = 53,
  parameter EXP_WIDTH      = 11
)(
  input  logic res_n,
  input  logic clk,
  output logic [EXP_WIDTH+MANTISSA_WIDTH-1:0] port_a,
  output logic [EXP_WIDTH+MANTISSA_WIDTH-1:0] port_b,
  output logic [EXP_WIDTH+MANTISSA_WIDTH-1:0] port_c,
  output logic valid_in,
  input  logic [EXP_WIDTH+MANTISSA_WIDTH-1:0] port_res,
  input  logic [7:0] exception_flags
);

localparam STDIN = 32'h8000_0000;
localparam N = EXP_WIDTH+MANTISSA_WIDTH-1; // = 63
logic [N:0] r_queue [4:0]; // "queue" of last 5 expected results
logic [7:0] e_queue [4:0]; // "queue" of last 5 expected exceptions
int i;
logic res_n_delay;
assign #8 res_n_delay = res_n; // starts assertions when first result is ready

always @(posedge clk or negedge res_n) begin
  if(~res_n) begin
    i <= 0; valid_in <= 0; port_a <= 0; port_b <= 0; port_c <= 0;
  end else begin
    valid_in <= 1;
    if(!$feof(STDIN)) begin
      $fscanf(STDIN,"%h␣%h␣%h␣%h␣%h",port_a,port_b,port_c,r_queue[i],e_queue[i]);
      // correct for different NaN encoding of TestFloat
      r_queue[i] = (&r_queue[i][N-1:MANTISSA_WIDTH-2]) ? 64'h7ff8000000000000 : r_queue[i];
      i = (i+1)%5;
    end else begin
      $finish;
    end
    if(res_n_delay) begin
      assert (r_queue[i] == port_res) else begin
        $display("Expected:␣%h,␣DUV:␣%h", r_queue[i], port_res); $finish;
      end
      assert (e_queue[i] == exception_flags) else begin
        $display("Expected:␣%h,␣DUV:␣%h", e_queue[i], exception_flags); $finish;
      end
    end
  end
end
```

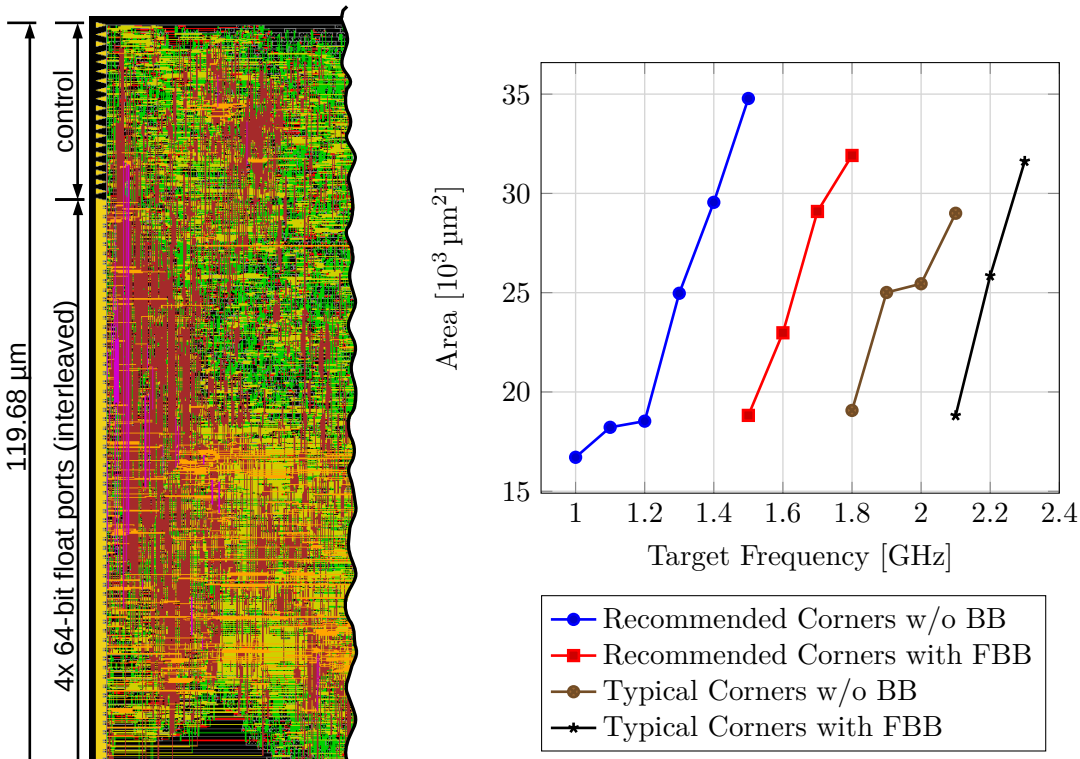
**Listing 4.1:** TestFloat driver and checker module.

### Synthesis Results

The floorplan was kept rather simple for this first implementation. Only the height was defined to be 119.68 µm. The length was adjusted to yield a utilization of 80%. A more detailed placement will be part of future work when more submodules are described

at a lower abstraction level, allowing for more control on what is synthesized. Possible optimizations have been researched in this work after the publication of these results in *Supercomputing Frontiers and Innovations*. They are discussed in chapter 5. Pin placement was done with a later application in a RISC-V processor implementation in mind. RISC-V suggests a dedicated floating-point register file, which will need three read- and one write-port to provide the operands for fused operations. Assuming the register file will be located left of the FPU in a pipeline, the operand and result pins were placed on the left side in an interleaved manner using metal layers 3 to 6 and a spacing of 0.35 µm. The remaining pins are also placed on the left side with a spacing of 1.4 µm on metal 3 following the pins of `port_a`. This is shown in Fig. 4.7a. Depending on the exact register file size, this spacing may need to be changed in the future. To get a realistic timing, scan insertion was conducted even for this first synthesis run. This replaces all Flip-Flops (FFs) with Scan Flip-Flops (SFFs) that have a multiplexer in the data path to switch between the regular input (D) and the scan input (SI). The additional multiplexer delay reduces the time available for other logic, but a scan chain is needed for chip testing. The effect of the clock distribution was also considered by performing Clock Tree Synthesis (CTS). This adds a buffer tree to the design to distribute the clock to all clock inputs and assures that the rising clock edge reaches every FF within a defined time window. Subsequent to CTS, the design was routed. After routing, the timing was met for a cycle time of 666 ps, i.e. 1.5 GHz, over all recommended implementation corners without using Forward Body Bias (FBB). Fig. 4.7b shows the area over target frequency.

The synthesis results for the lower frequencies (blue curve) were obtained using the recommended corners for setup and hold analysis: slow and fast process, 10% voltage deviation around the nominal voltage of 0.8 V, and a temperature of −40 °C and 125 °C. From these recommended corners the tools identified the combination (SS, 0.72 V, 125 °C, RC max) to be most critical for timing. The currently only partially optimized design suffers from a significant area increase with rising target frequency. Fig. 4.7b shows that the area roughly doubles from 1.2 GHz to 1.5 GHz. From there, the corners were adjusted for higher frequencies. It is interesting to observe how much a FBB can increase performance. For this design, it allows to reach frequencies up to 1.8 GHz (red curve). To see how the design performs typically, it was synthesized only with typical corners (brown curve) and also with their forward body biased versions (black curve). Typical corners are available for 25 °C and 85 °C. Here, the tools identified (TT, 0.8 V, 85 °C, RC nominal) to be most critical for timing. This allowed us to reach up to 2.3 GHz. The tools used were Cadence Genus and Innovus. Note that with only typical corners, higher frequencies can be reached than with the recommended corners using FBB. This shows that it is not possible, at least for this design, to compensate a worst-case corner completely by using FBB.




**(a)** Pin placement.

**(b)** Area over target frequency for different corner setups.

**Fig. 4.7:** FMA unit synthesis results.

### Power Analysis

For all points presented in Fig. 4.7b, a power analysis based on a value change dump (vcd) file containing stimuli was conducted. The stimuli were generated with a modified test from the verification environment using Cadence Xcelium. The testbench contained the netlist, derived after all the previously described synthesis steps where executed, and applied a clock signal of the corresponding target frequency. The test itself applied new operands every clock cycle and performed a random operation with a random rounding mode. The total power consumption for each synthesized design is shown in Fig. 4.8. The values were calculated for a typical corner (TT, 0.8 V, 85 °C, RC nominal), with the ones implemented with FBB having their power determined with the corresponding FBB corner. This corner was chosen as it best resembles the most likely operation condition. The clock tree makes up between 0.77 % and 1.56 % of the total power from high to low target frequencies, which seems plausible for a small design. The power analysis was done with Cadence Innovus/Voltus.

Fig. 4.8 shows that the power increases with area and frequency as expected. The

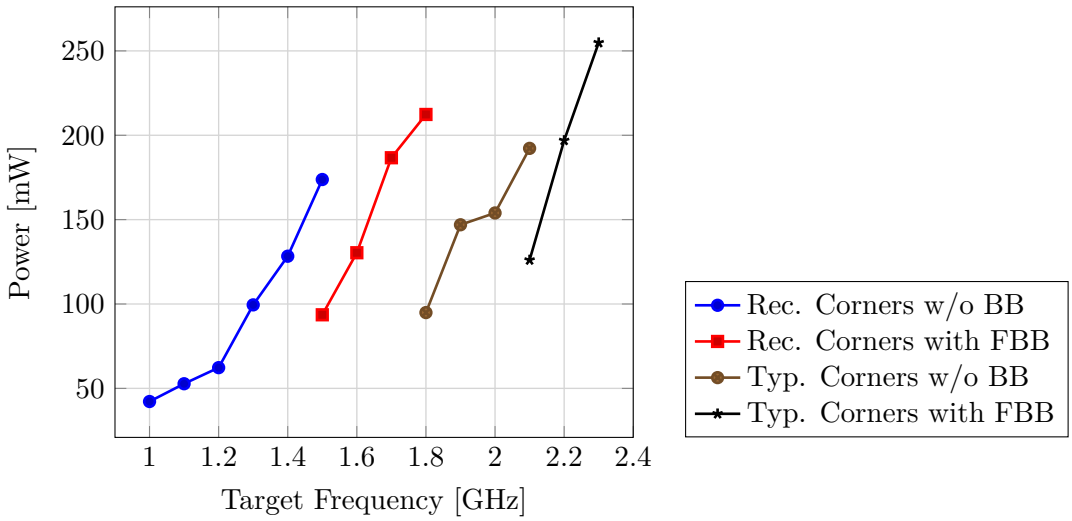



**Fig. 4.8:** Power over target frequency for different corner setups.

power consumed by the 2.3 GHz implementation at different operating frequencies is shown in Fig. 4.9. The linear rise of power with frequency is expected, but the values also show that a faster design, which uses more area, also consumes more power at lower operating frequencies than a design implemented for that particular target frequency – which is important to consider when dynamic frequency scaling is applied. Besides the total power (black curve), Fig. 4.9 also shows the three parts which make up the total power. These are the switching power representing the loading/unloading of nets, and the power used internally in the standard cells. They make up the linear part. The third part (brown curve) is the leakage power, which is constant at 33.6 mW over the operating frequency. This high leakage current is caused by using SLVT standard cells and FBB.

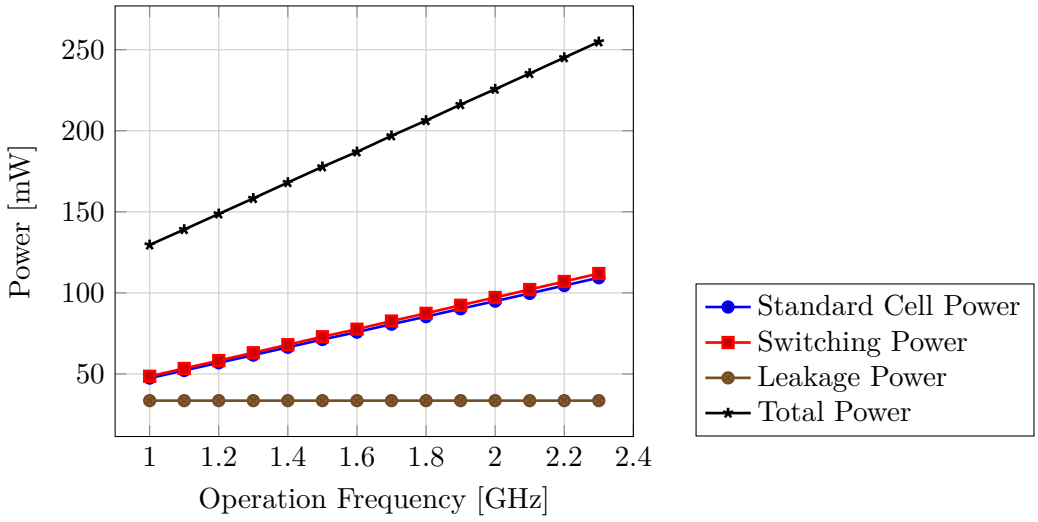



**Fig. 4.9:** Power of the 2.3 GHz implementation over operation frequency.

Tab. 4.3 compares the two most similar FMA implementations from this work and from [109] in terms of the metrics used in [109]. We calculated the performance of our design by assuming two floating-point operations per clock cycle, i.e. the maximum throughput possible. This was also done in [109]. Their design runs with 1.81 GHz in a 45nm technology using low threshold devices and six pipeline stages. Compared to our synthesis result at 1.8 GHz for a typical process with super-low threshold devices, we have a similar power per performance but are roughly a factor of three smaller in terms of area per performance. The latter is contributed to technology scaling. The former is probably due to the low pipeline depth of three versus six. A lower number of pipeline stages makes it harder to achieve timing and thus requires the synthesis tool to use additional logic to fit the combinational logic into the cycle time. This fast area increase was observed in Fig. 4.7b. Furthermore, our design is not optimized for power.

| Property | 45nm FMA | Our 22nm FMA |
|---|---|---|
| $V_{th}$ | low | super low |
| $V_{DD}$ in V | 0.9 | 0.8 |
| Pipeline Depth | 6 | 3 |
| Frequency in GHz | 1.81 | 1.8 |
| Area in µm$^2$ | 49839 | 19066 |
| W/GFLOPS | 0.0253 | 0.0264 |
| mm$^2$/GFLOPS | 0.0145 | 0.0053 |
| W/mm$^2$ | 1.75 | 4.98 |

**Tab. 4.3:** Comparison of our synthesis results with [109].

Another interesting aspect can be seen in Fig. 4.10, which shows power density over target frequency. Power density scales linearly with frequency as expected, but the second observation is that using FBB keeps the power density constant, whereas going from slow to typical corners reduces power more than one would expect from the area shrink alone. This shows again that FBB is not enough to compensate worst-case corners.

### 4.2.5 Conclusion

A RISC-V- and IEEE 754-conform FMA unit which passed the complete ASIC design flow was designed and verified using different techniques. The PPA analysis demonstrates that the design is comparable to existing implementations like HardFloat and results presented in [109]. However, many more aspects need to be researched in order to reach optimum performance. The timing reports have identified that the critical paths are almost always ending at registers behind the Compression Tree. Hence, it might be useful to consider a new placement of the pipeline stages. This could also be automated by using the *retime* functionality of the Genus synthesis tool. It allows to automatically position registers within a cloud of combinational logic to optimize timing and/or power.

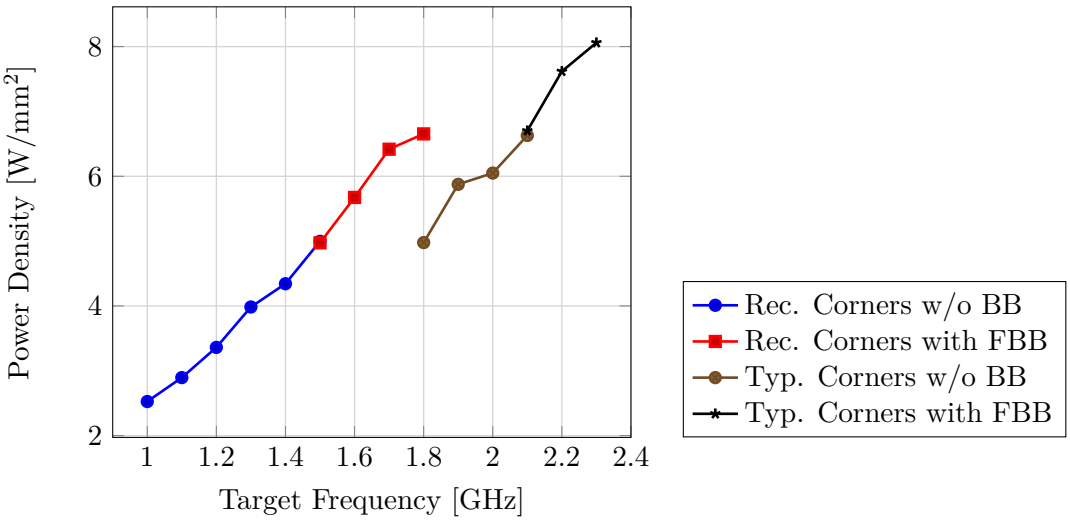



**Fig. 4.10:** Power/Area over target frequency for different corner setups.

This is a trade-off as the FMA unit has less outputs than inputs, so moving registers towards the inputs tends to increase their number and therefore power. The general viability of re-timing registers is limited by the capability of the front-end tool (Genus) to estimate interconnect and placement because the back-end (Innovus) cannot reposition registers anymore. Simple experiments resulted in Genus underestimating wire delay in spite of using *physical layout estimation*. Cadence has since presented new tool support allowing for closer integration of front-end and back-end. This shows that EDA software is going to be able to achieve the quite significant savings possible through automated pipelining. Therefore, manual effort spend on placing register stages will become less important.

Another possible aspect could be to optimize the multiplication and addition through gate-level implementations, instead of relying on the synthesis tool. Kaiser already added a structural (not gate-level) description of the Compression Tree with the intention to access the partial products that allow to "sneak" in the addition via the 3:2 Compressor Row. However, the synthesis tool was still able to translate it to whatever standard cells it saw fit. Thus, the resulting netlist did not contain the intended tree structure. This led to the idea to investigate whether actual gate-level implementations of arithmetic hardware are more efficient than simple '`*`' and '`+`' operators. Chapter 5 presents the results of this research.

Chapter 5

# Hardware Arithmetic

Chapter 4 has pointed out that arithmetic circuits are a potential target for optimizing the performance floating-point hardware. This chapter takes a closer look on how multipliers and adders are built and whether manually designed implementations have any benefits over the logic generated by synthesis tools. A major motivation for revisiting these components in this thesis is that the performance of many common applications like machine learning, digital signal processing including video compression, and of course HPC applications, heavily depends on the speed of basic hardware arithmetic functions used in specialized and general purpose hardware. From a power efficiency point of view, it is debatable if such an analysis can provide much benefits since data movement consumes much more power than the actual computation. Nevertheless, saving power where possible cannot be wrong.

This chapter is structured as follows: the first section will present the basics of binary multiplication and discuss different tree structures used for multipliers. As these multipliers fundamentally rely on additions, the chapter continues with an in-depth view of adder circuits in the second section. The goal of this effort is to understand which trade-offs can be made and if the design of custom structures is worth the effort considering the capabilities of recent synthesis tools. The third section deals with the verification process of these custom structures to ensure that the results are based on correct designs. Section 5.4 then comes back to the initial problem of performance and power consumption. The adder and multiplier variations are synthesized, including place & route, and a detailed PPA analysis is presented.

## 5.1 Multipliers

This section presents the principles of multiplier circuits and starts with terminology. An expression $A \cdot B$ is called a *product*, while $A$ is called *multiplier* and $B$ *multiplicand*. Both are also *factors*. For integer multiplication, the simplest way to compute the product is by adding copies of the multiplicand: $A \cdot B = \sum_{i=1}^{A} B$. However, this is a very slow and inefficient method, both manually and especially for hardware implementations, since the number of steps, i.e. cycles, does depend on the value of $A$. Another method comes from the following observation, where $\mathcal{B}$ is the base or radix of the numeral system and $a_i$ and $b_j$ are the digits of $A$ and $B$.

$$P = \sum_{i=0}^{n-1} \sum_{j=0}^{m-1} a_i b_j \mathcal{B}^{i+j} \tag{5.1}$$

So depending on the number of digits $n$ and $m$, there are $nm$ so-called *partial products* $a_i b_j$, weighted with the corresponding power of the base. This is often represented in a rhombus-like figure as shown in Tab. 5.1. These partial products need to be added with consideration of possible carries. Note that there is no dependency of the number of partial products on the operand values. The terms $a_i B$ are called *summands*.

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | $a_0b_{m-1}$ | $a_0b_{m-2}$ | $\cdots$ | $a_0b_1$ | $a_0b_0$ | $a_0B$ |
| $+$ | | | | $a_1b_{m-1}$ | $a_1b_{m-2}$ | $\cdots$ | $a_1b_1$ | $a_1b_0$ | | $a_1B$ |
| $+$ | | | $\iddots$ | $\iddots$ | $\iddots$ | $\iddots$ | $\iddots$ | | | $\vdots$ |
| $+$ | | $a_{n-2}b_{m-1}$ | $a_{n-2}b_{m-2}$ | $\cdots$ | $a_{n-2}b_1$ | $a_{n-2}b_0$ | | | | $a_{n-2}B$ |
| $+$ | $a_{n-1}b_{m-1}$ | $a_{n-1}b_{m-2}$ | $\cdots$ | $a_{n-1}b_1$ | $a_{n-1}b_0$ | | | | | $a_{n-1}B$ |
| $p_{m+n-1}$ | $p_{m+n-2}$ | | | $\cdots$ | | | $p_1$ | $p_0$ | $P$ | |

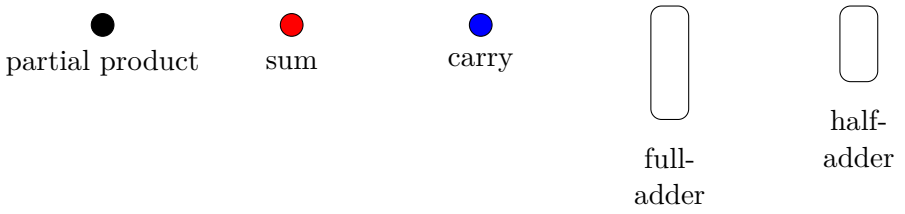
**Tab. 5.1:** Partial products and summands.

For binary numbers ($\mathcal{B} = 2$), the partial products of the operand digits can be computed by a two-input logical AND. This is exactly what is done in hardware multipliers in a first stage called *partial product generation*. In the next stage, these partial products need to be summed up, where $m = n$ is now assumed as this is commonly the case in hardware. A simple sequential method to do this is given by adding the summands one by one with an adder, e.g. a Carry-Ripple Adder (CRA), which results in a delay of $\mathcal{O}(n^2)$. This can be improved by using adders with logarithmic time like the Ccarry-Lookahead Adder (CLA) to $\mathcal{O}(n \log(n))$. Using multiple CLAs composed in a tree structure, the delay can be further reduced to $\mathcal{O}\left(\log(n)^2\right)$. However, it is possible to reduce the time complexity of multiplication to $\mathcal{O}(\log(n))$ through special tree structures called *carry-save trees*. It is cumbersome to describe them in structural SystemVerilog but much easier in other

languages like Python. Thus, a Python code has been developed for this work to generate the SystemVerilog code for these tree structures. Furthermore, the tool also generates the TikZ code to visualize the trees in LaTeX as shown in this section. The TikZ code also serves as a useful tool to debug the tree generation.

### 5.1.1 Wallace Tree

C. S. Wallace was the first to describe a multiplier circuit with delay $\mathcal{O}(\log(n))$ in 1964. His paper *"A Suggestion for a Fast Multiplier"* [110] basically made the observation that Carry-Save Addition (CSA) can be done in $\mathcal{O}(1)$. CSA means the carries are not propagated, and the result of a CSA has therefore two components, a sum vector and a carry vector. A half-adder does actually perform a CSA, and the same is true for a full-adder whose carry-in is used as a third input. Hence, a row of $n$ full-adders is an $n$-bit three-operand carry-save adder. Wallace further proposed to group the summands – not the partial products as is sometimes assumed or not described precisely – into groups of three to add them in parallel. With respect to their weight, they are fed into rows of full-adders. Since the summands in each group are shifted relative to each other, half-adders are added left and right as "padding" where only two partial products remain per column. Due to the CSA, every group of three summands is reduced to two results by each stage. The remaining summands ("number of summands modulo 3") from the grouping phase are just passed to the next stage and grouped along with the results from the previous stage. Thus, the number of summands is reduced to at most 2/3 in each stage. When this process is finished, the remaining two summands are added with a conventional adder. The delay of a CSA is $\mathcal{O}(1)$, but the tree is $\mathcal{O}(\log(n))$ deep and the final addition also takes $\mathcal{O}(\log(n))$. Hence, the overall delay is $\mathcal{O}(\log(n))$ – a remarkable result as it shows multiplication and addition have the same time complexity. However, this comes at a price in terms of a gate count complexity of $\mathcal{O}(n^2 \log(n))$. An example for 12-bit operands is shown in Fig. 5.1. Each black dot is a partial product, each red one a sum, and each blue one a carry. The rectangles with rounded corners are full- and half-adders, taking the enclosed dots as an input. Their outputs are presented in the next stage as sum in the same and as carry in the next higher column.



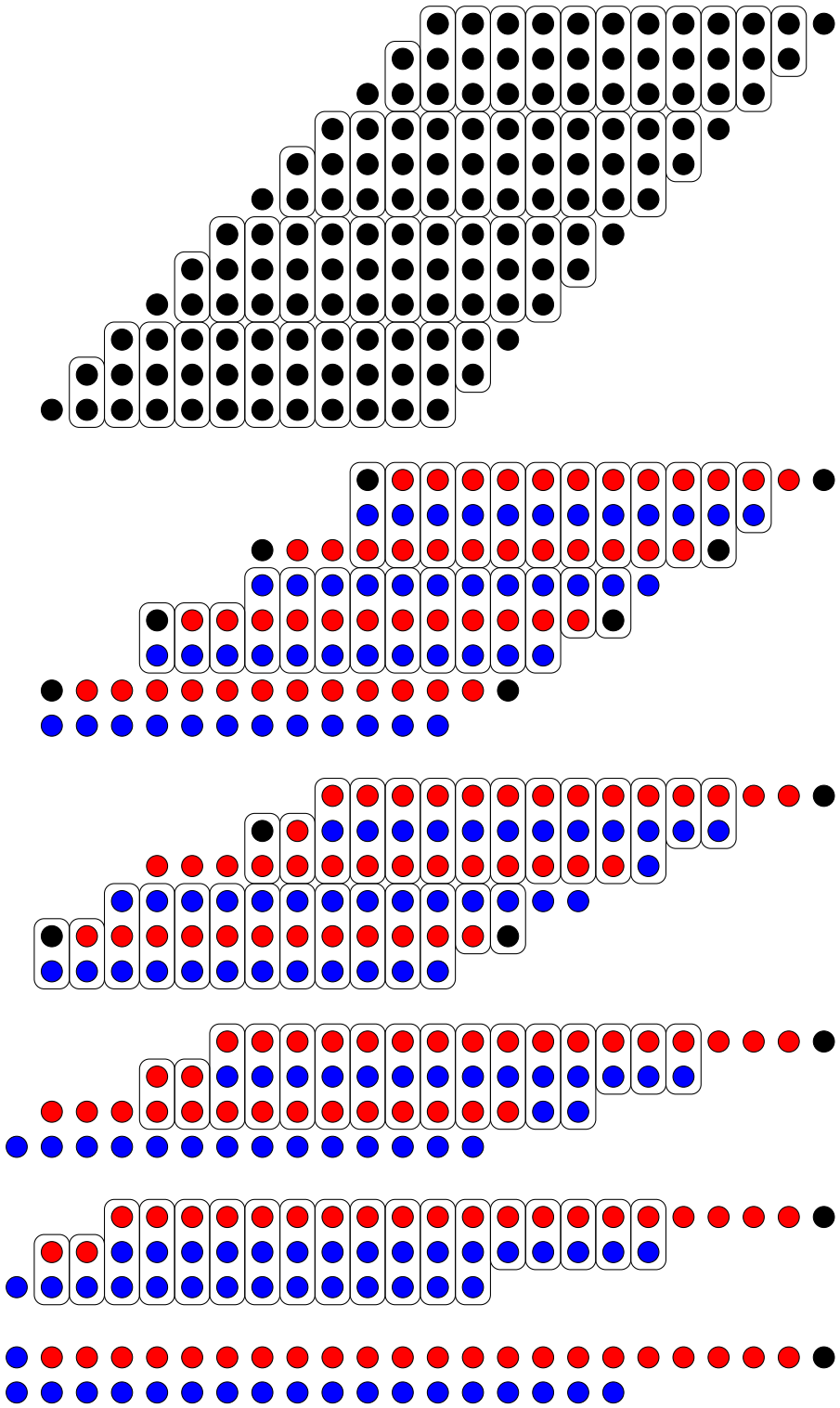


**Fig. 5.1:** 12-bit Wallace tree: 102 full-adders, 34 half-adders.

### 5.1.2 Dadda Tree

Based on the work of Wallace, L. Dadda published a paper called *"Some Schemes for Parallel Multipliers"* [111] in 1965, which approaches the additions of the partial products in a more general way by omitting the implicit and unnecessary grouping caused by thinking of summands. This change in the way of thinking is represented by regrouping the partial products from the rhombus structure to a triangular one. Comparing Fig. 5.1 and Fig. 5.3 illustrates this. Note, however, that Dadda kept the rhombus structure for the first stage in his drawings in [111], unlike presented here. It can be observed that the order in which the partial products within each column are summed can be completely arbitrary.

Dadda deduces several schemes as the title of his paper suggests, until he concludes with what is nowadays called the Dadda tree. This subsection presents his deduction in a shortened version because it provides some insight. He starts with so-called parallel $(N, M)$-counters as the basic elements needed for calculating the sum of each column. These are simply logic functions with $N$ inputs and $M = \lceil \log_2(N+1) \rceil$ outputs. The number of ones among the inputs is presented binary-coded at the output. Since the number of output bits is smaller than the number of input bits for $N > 2$, the output is considered a "compressed" version of the input. Therefore, the parallel counters are also called *compression cells*. The two smallest, i.e. the $(2, 2)$- and $(3, 2)$-counter, are most commonly known as half- and full-adder. After presenting the first two schemes applying $(N, M)$-counters with $N$ chosen according to the column size, he realizes that for arbitrary $(N, M)$, the counters are complex logic functions not available in most cases. This is still true today as there are no standard cells for the complex counters in modern technologies but only for half- and full-adders. It is interesting to note that even using only half-adders (as they are the smallest counters possible) still yields a functional tree structure. Even though they are not obviously reducing the number of bits to be summed, they shift a carry to the left, i.e. to a higher weight. A central observation regarding these structures is that bits in columns with a corresponding weight of $2^{2n}$ or greater are always zero and can be neglected in subsequent stages. A product of $n$-bit numbers cannot be wider than $2n$ bit. Unfortunately, using only half-adders is a bad choice as Dadda found the number of stages is no longer $\mathcal{O}(\log(n))$ but instead $\mathcal{O}(2^n)$.

So after realizing structures composed purely of half-adders are not feasible, he also allows full-adders. In each stage, the maximum amount of full-adders is applied to each column to provide a maximum reduction and consequently a minimum number of stages. If necessary, an additional half-adder is applied in a column as well. However, it is not necessary to apply a half-adder if every column to the right has less than three elements because these columns will remain untouched from any carry propagation and can be directly transferred into the results. As in the first stage, each group of three partial

products in a column is applied to a full-adder; this is sometimes confused with the description of the Wallace tree because the partial products are confused with summands. As a reference for a "naive" implementation synthesis, results are also shown for this tree, called "MaxReduce" in this thesis, in absence of a better name. This tree applies the maximum number of full-adders per column plus a possible half-adder. An example with 12-bit operands is shown in Fig. 5.2.

The 12-bit MaxReduce tree shows that the twelve rows of partial products are reduced to eight rows by the first stage. Dadda noticed that this amount of reduction is a waste of adders in the first stage because besides the eight rows, also a nine row configuration could be reduced to six rows by the second stage. This is because $\lceil 8/1.5 \rceil = 6 = \lceil 9/1.5 \rceil$. Thus, starting from the output of the last stage, for which he knew it has two rows, Dadda deduced the optimal number of rows per stage. If $r_i$ is the row count of the $i$-th stage looking from the end, where $r_1 = 2$, the maximum number of rows that can be reduced to $r_i$ is $r_{i+1} = \lfloor 1.5 \cdot r_i \rfloor$. This sequence basically defines the Dadda tree algorithm:

2, 3, 4, 6, 9, 13, 19, 28, 42, 63, ...

The first stage of the Dadda tree is built by reducing the $n$ rows of partial products to the largest number smaller than $n$ in this sequence by the first stage – so that under the consideration that full- and half-adders on the right of a column increase the number of its elements by propagating a carry, each column contains not more rows than defined by the sequence. To consider this carry propagation, it is most practical to start with the column on the right and move to the left as carries are also propagated from right to left. This is also what the Python tree generator developed in this work does. It contains a list filled with partial products for each column and starts its loop with column zero. If an adder needs to be placed in the tree, the corresponding carry bit is appended to the list representing the next higher column. It might be worth noting that this sequential approach does not lead to chained adders within one stage, which means every stage still has the delay of one full-adder only. An example Dadda tree is shown in Fig. 5.3. Dadda does not provide a proof but assumes that this is the variant using the lowest number of adders.

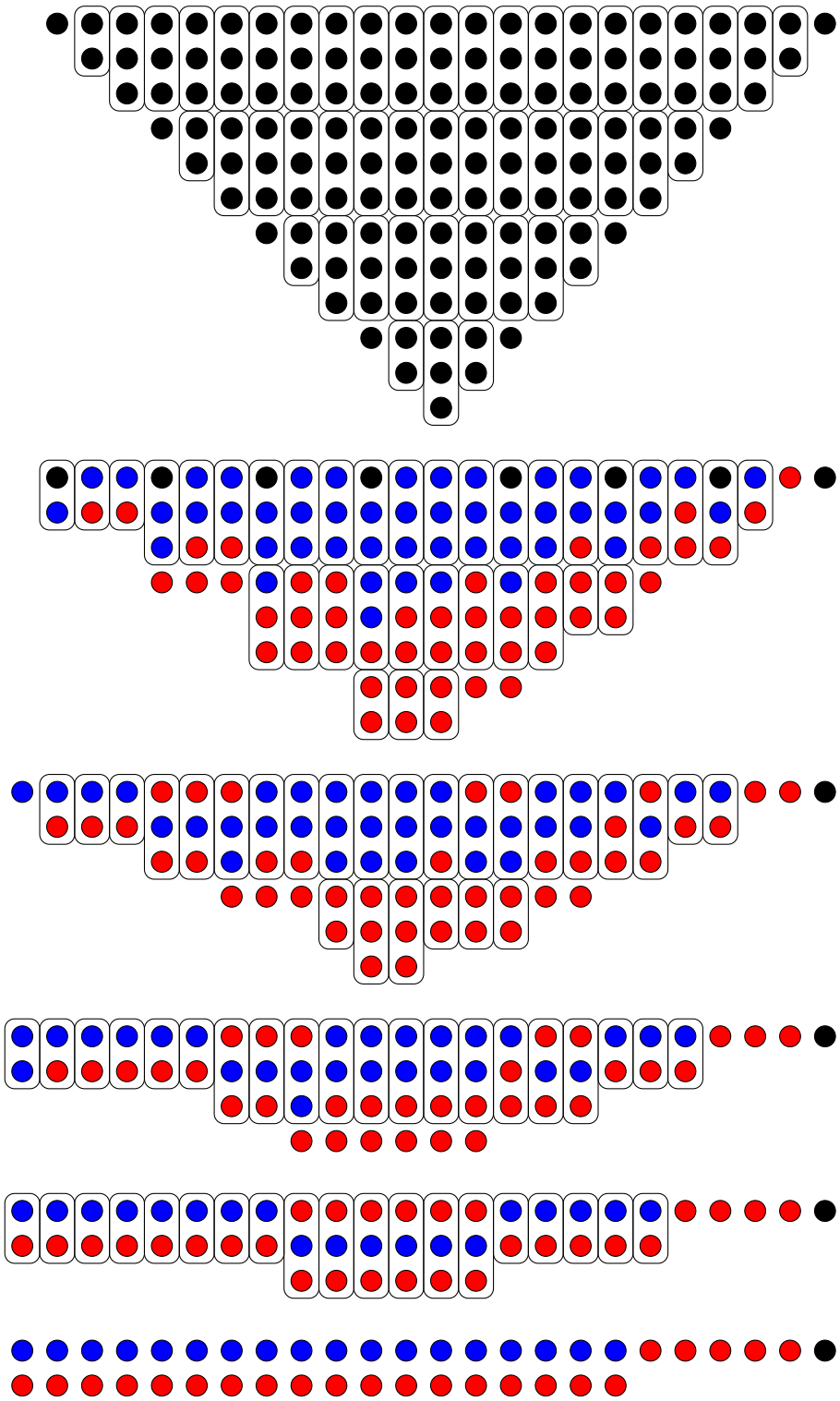


**Fig. 5.2:** 12-bit MaxReduce tree: 100 full-adders, 48 half-adders.

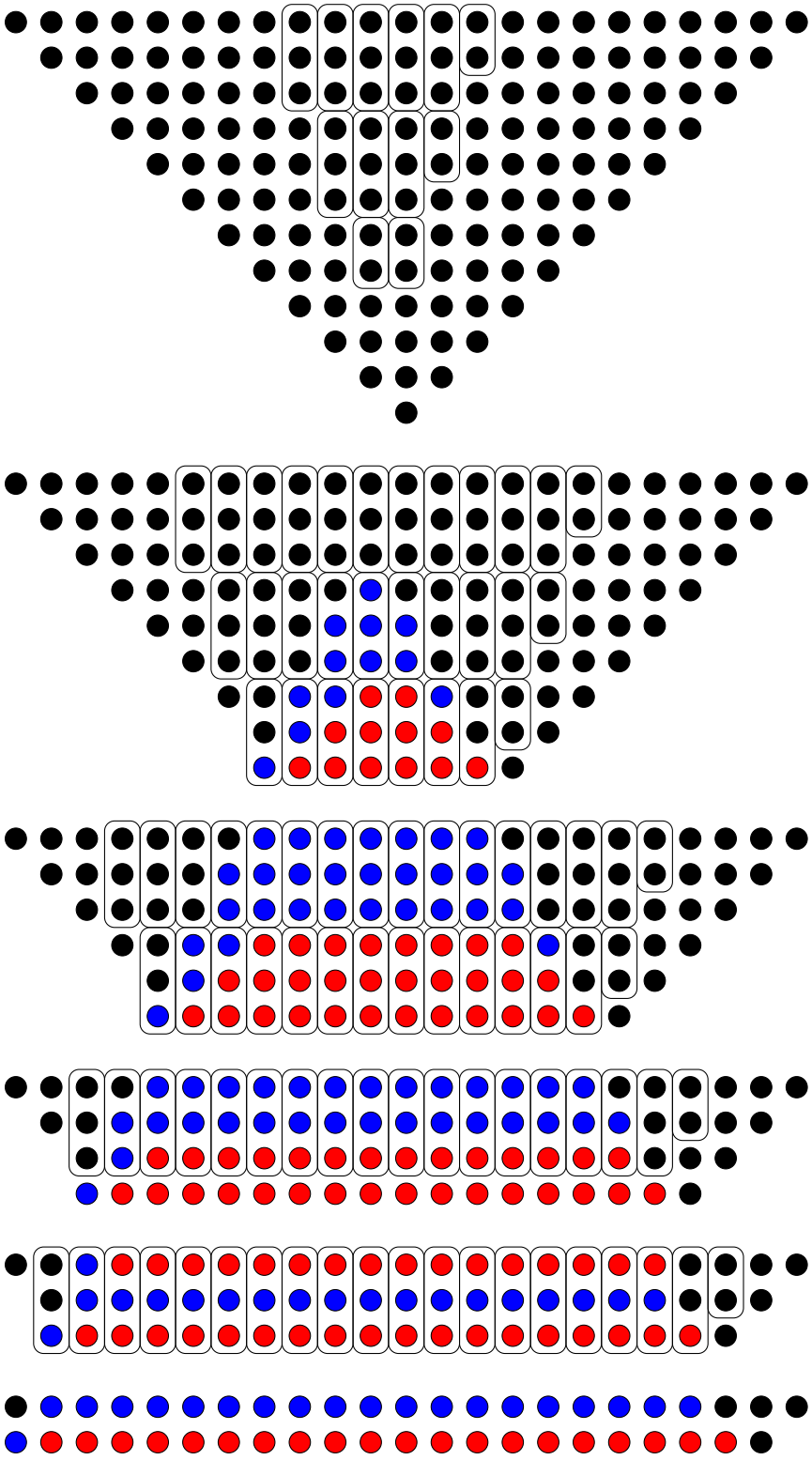


**Fig. 5.3:** 12-bit Dadda tree: 99 full-adders, 11 half-adders.

### 5.1.3 Comparison

The Wallace and Dadda trees have been compared on a theoretical level by Townsend et al. [112]. They also derived the analytical expressions for the number of full- and half-adders, as well as the width of the subsequent Carry-Propagating Adder (CPA) needed to add the final two rows. The formulas are given in Tab. 5.2.

| Property | Wallace ($n > 5$) | Dadda |
|---|---|---|
| full-adder count | $n^2 - 4n + 1 + \#\text{stages}$ | $n^2 - 4n + 3$ |
| half-adder count | $\geq n$ | $n - 1$ |
| CPA width | $2n - 1 - \#\text{stages}$ | $2n - 2$ |

**Tab. 5.2:** Scaling of component count for Wallace and Dadda multipliers [112].

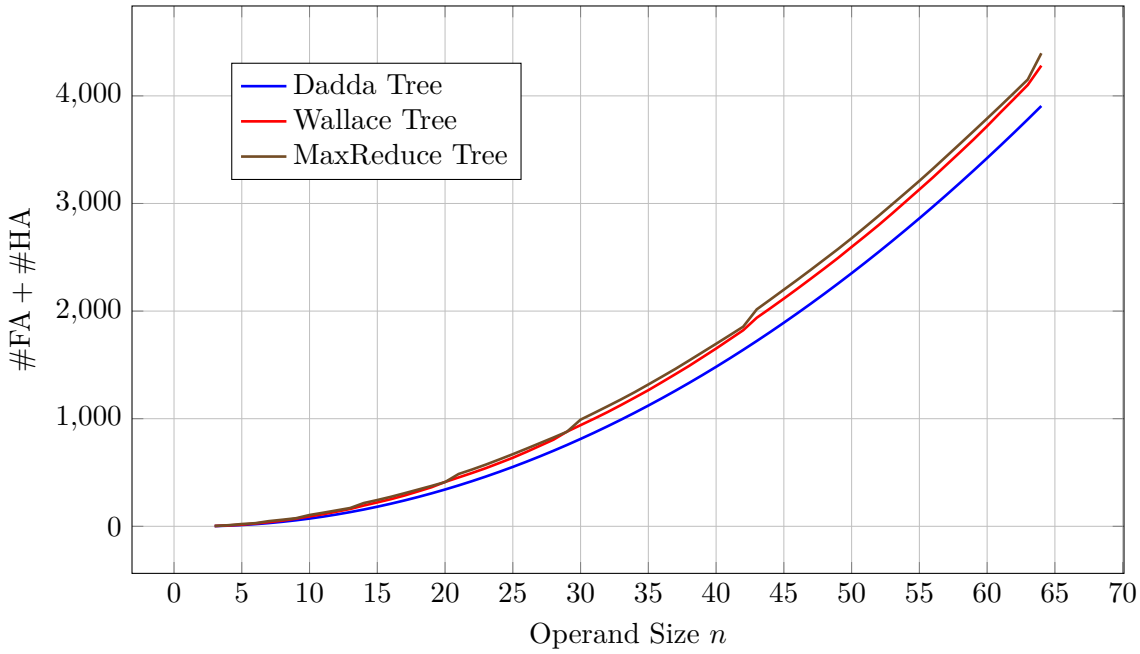



**Fig. 5.4:** Combined number of full- and half-adders for different trees.

## 5.2 Adders

The previous section has shown that the speed advantage of the multiplier trees is based on delaying the carry-propagation by using Carry-Save Addition (CSA). This delay saves the carry-propagation, hence the name Carry-Save Addition. The two remaining summands still need a final Carry-Propagating Adder (CPA). An interesting way to look at multipliers is to see them as multi-operand adders with the Wallace tree designed on word-level and the Dadda tree on bit-level [113]. From this perspective, it is obvious why multiplication has the same time complexity as addition. Up to today, a large variety of adders has been developed. This section gives an overview on this topic, to explore the design space for architectures to be evaluated in the context of modern synthesis tools. A master's thesis by Lynch from 1996 [114] on binary adders provides a lot of insight. An addition is comprised of two summands $A = \{a_{n-1}, \ldots, a_0\}$ and $B = \{b_{n-1}, \ldots, b_0\}$ resulting in a sum $S = \{s_n, \ldots, s_0\}$. The calculation of the sum bits is usually expressed with Eqs. 5.2 and 5.3.

$$s_i = \begin{cases} \text{xor}(a_i, b_i, c_i), & i < n \\ c_n, & i = n \end{cases} \tag{5.2}$$

$$c_{i+1} = \text{majority}(a_i, b_i, c_i) \text{ and } c_0 = c_{\text{in}} \tag{5.3}$$

The dependency of the sum bit at index $i$ on all operand bits with index lower or equal to $i$ is expressed using a new bitvector $C = \{c_n, \ldots, c_0\}$, called carry. This carry vector appears as a third operand and limits the speed of addition [114]. The simplest method for addition is called serial by Lynch but should rather be named sequential as it uses a result register where each cycle one bit of the result is computed and the carry bit is fed back for the next cycle. It is obviously impractical for high-speed computations and not even area or energy efficient as the operand bits need to be multiplexed for the single full-adder instance. If an $n$-bit result register is used over a single one, even the output has to multiplexed. The combinational version of this is called Carry-Ripple Adder (CRA). It is a chain of $n-1$ full-adders starting with a half-adder connected with their carry in- and outputs. This is very area efficient but has linear time complexity as the carry generated by the half-adder may ripple to the end of the chain.

### 5.2.1 Carry-Select Adder

As more transistors could be crammed onto chips, higher performance became a reasonable optimization criterion, and logic was added to improve speed by parallelism. A first variant of such a parallel adder is the Carry-Select Adder (CSL), a divide-and-conquer approach [114], i.e. the operands are split in a lower and upper half, both added separately

by a CRA. However, the upper half is replicated once to compute the result for both potential outcomes of the lower half's carry-out bit. The result is then multiplexed and the time delay approximately cut in half for large enough $n$. As Lynch [114] considers a fixed number of sub-adders, he concludes that the CSL has a linear time complexity. It can however be easily shown that if the number of sub-adders, $m_{\text{csl}}$, is allowed to change with $\sqrt{n}$, a time complexity of $\mathcal{O}(\sqrt{n})$ is achieved (Eq. 5.4). For simplicity, the second step assumes full-adder delay $t_{\text{d,fa}}$ and multiplexer delay $t_{\text{d,mux}}$ to be equal to 1.

$$t_{\text{d,csl,fix}} = \frac{n}{m_{\text{csl}}} \cdot t_{\text{d,fa}} + (m_{\text{csl}} - 1) \cdot t_{\text{d,mux}} \Rightarrow t_{\text{d,csl,fix}} = 2\sqrt{n} - 1 \tag{5.4}$$

The absolute time can be further optimized by using variable block sizes. The optimal layout would be a constant increase in sub-adder size from $n_{\text{csl}}$ towards the MSB in steps of $\lceil t_{\text{d,mux}}/t_{\text{d,fa}} \rceil$. This ratio should be around 1 in most technologies as the multiplexer delay is usually smaller than a full-adder delay. Thus if a block completes addition, the carry from the previous block is already available. Eq. 5.5 connects $n_{\text{csl}}$ with $n$, and Eq. 5.6 is the delay with full-adder and multiplexer delays again set to 1. Inserting Eq. 5.5 in Eq. 5.6 via $n_{\text{csl}}$ results in a optimal delay for $m_{\text{csl}} = \sqrt{2n}$ – lower compared to the version with fixed-size blocks but with the same general time complexity.

$$n = \sum_{i=0}^{m_{\text{csl}}-1} n_{\text{csl}} + i = m_{\text{csl}} n_{\text{csl}} + \frac{m_{\text{csl}}(m_{\text{csl}} - 1)}{2} \tag{5.5}$$

$$t_{\text{d,csl,var}} = n_{\text{csl}} + m_{\text{csl}} - 1 \tag{5.6}$$

$$\Rightarrow t_{\text{d,csl,var}} = \sqrt{2n} - 1/2 \tag{5.7}$$

Of course, both calculations did not consider any remainders occurring when dividing the adder into sub-adders but capture the general idea.

Another way to use the divide-and-conquer strategy is the conditional sum addition. It is similar to the CSL but uses a block size of only one bit. Except for the LSB, each block "speculatively" computes the addition for both possible carries. The results are then fed into a multiplexer tree, which is also "speculative", i.e. there are two multiplexers at each stage to propagate both possible results for the corresponding block. The carry-out of the LSB full-adder steers the multiplexer of the next higher bit to choose the correct result. This result is then used for the next two bits, then the next four bits, the next eight bits and so on. From this, it is obvious that the path from carry-in to carry-out contains $\mathcal{O}(\log(n))$ multiplexer delays. Note that the time complexity compared to the CSL is not reduced as a consequence of adding speculation to almost every bit but instead by adding a "speculative" multiplexer tree. A carry-select with block size of one would actually have time complexity $\mathcal{O}(n)$. A nice schematic of both adders can be found in [113]. The conditional sum adder is not analyzed further in this work, but the next subsection will

show the parallel prefix version of it, called Sklansky adder [115], [116].

### 5.2.2 Parallel Prefix Adders

Logarithmic time complexity is the target delay scaling to be achieved for many arithmetic/combinational circuits. The previous methods for addition are possible solutions, but their exact trade-offs "remain in the dark". The fundamental problem adders aim to solve is the carry propagation (Eq. 5.3). Such a problem is called a *prefix computation*. A prefix function $X \mapsto Y = f_{\circ,n}(X)$ for an associative two-input Boolean operator $\circ$ takes an $n$-bit input vector $X = \{x_{n-1}, \ldots, x_0\}$ and computes the result $Y = \{y_{n-1}, \ldots, y_0\}$ as follows.

$$\begin{aligned} y_0 &= x_0 \\ y_1 &= x_1 \circ y_0 = x_1 \circ x_0 \\ y_2 &= x_2 \circ y_1 = x_2 \circ x_1 \circ x_0 \\ &\vdots \\ y_{n-1} &= x_{n-1} \circ y_{n-2} = x_{n-1} \circ \cdots \circ x_2 \circ x_1 \circ x_0 \end{aligned} \tag{5.8}$$

The dependency of the sum at index $i$ on all operand bits with lower or equal index, which is generally abstracted with the carry, is such a prefix function with a slightly more complicated operator, called Fundamental Carry Operator (FCO). This operator is associative and can therefore be parallelized to achieve high-speed addition. Hence, adders built based on this idea are called Parallel Prefix Adders (PPrAs). In a wider context, they are categorized together with the CLA as *propagate-generate adders* by Lynch [114]. The majority function of the carry in Eq. 5.3 can be rewritten into the form shown in Eq. 5.9. To simplify the notation, "$\cdot$" and "$+$" are used for the Boolean AND and OR, respectively.

$$c_{i+1} = \text{majority}(a_i, b_i, c_i) = a_i b_i + (a_i + b_i)c_i = g_i + p_i c_i \tag{5.9}$$

The bits $g_i$ and $p_i$, implicitly defined by this equation, are commonly called *generate* and *propagate*, respectively; hence the name propagate-generate adders used by Lynch. When the generate bit is set, a carry will be produced at index $i$ and passed to the higher index regardless of the incoming carry. In contrast, a set propagate bit only says that an incoming carry is passed to the next higher index. However for this meaning, $p_i$ should be rather computed with an XOR to exclude the generate case. Indeed, most literature uses this interpretation of propagate (instead of the one mentioned previously) as this does not change the value of $c_{i+1}$. Lynch [114] actually calls the OR-based signal *transfer* instead of propagate. In this work, the XOR-version is ignored and the OR-version is

called propagate as this is the common name, and the following derivations do not come across the XOR-version in any way.

Based on the two previously discussed signals, the FCO is defined by Eq. 5.10. The $\circ$ from the introduction of the prefix function is now used for the FCO as this is a common symbol used in literature.

$$(g_1, p_1) \circ (g_0, p_0) \coloneqq (g_1 + p_1 g_0, p_1 p_0) \tag{5.10}$$

Using the FCO, the carry equation can be rewritten into Eq. 5.11.

$$(c_{i+1}, 0) = (g_i, p_i) \circ (c_i, 0) = (g_i + p_i c_i, 0) \tag{5.11}$$

It is now obvious how the carry vector is the result of a prefix computation.

$$\begin{aligned}
(c_0, 0) &= (c_{\text{in}}, 0) \\
(c_1, 0) &= (g_0, p_0) \circ (c_{\text{in}}, 0) \\
(c_2, 0) &= (g_1, p_1) \circ (g_0, p_0) \circ (c_{\text{in}}, 0) \\
&\vdots \\
(c_n, 0) &= (g_{n-1}, p_{n-1}) \circ \cdots \circ (g_0, p_0) \circ (c_{\text{in}}, 0)
\end{aligned} \tag{5.12}$$

Compared to Eq. 5.8, the carry-in $(c_{\text{in}}, 0)$ is somewhat impractical for an elegant notation, and it is convenient to redefine this as shown in Eq. 5.13.

$$(G_{i:i}, P_{i:i}) \coloneqq (g_{i-1}, p_{i-1}) \text{ and } (G_{0:0}, P_{0:0}) \coloneqq (c_{\text{in}}, 0) \tag{5.13}$$

The colon notation facilitates the following range relation.

$$(G_{i:j}, P_{i:j}) : = (G_{i:i}, P_{i:i}) \circ \cdots \circ (G_{j:j}, P_{j:j}) \tag{5.14}$$

$$= (G_{i:k}, P_{i:k}) \circ (G_{k-1:j}, P_{k-1:j}) \; \forall k \in \mathbb{N} \; : \; j \leq k \leq i \tag{5.15}$$

$$\Rightarrow c_i = G_{i:0} \tag{5.16}$$

Since the FCO is associative, it is possible to parallelize the logic function for each $G_{i:0}$ by using a binary tree structure. However, this results in many separate trees that need to be arranged in a way to share most of their logic to save FCO instances. This is exactly the place where variation comes into the design and several variants with different trade-offs have been proposed in literature. A three-dimensional taxonomy using variables $(l, f, t)$ was provided by Harris [116], and is shown in Fig. 5.5. The dashed lines in Fig. 5.5 mark the plane $f + t + l = \log_2(n) - 1$, where Harris locates all PPrA variants published before his paper [116]. He concludes the inherent trade-off between these networks are those

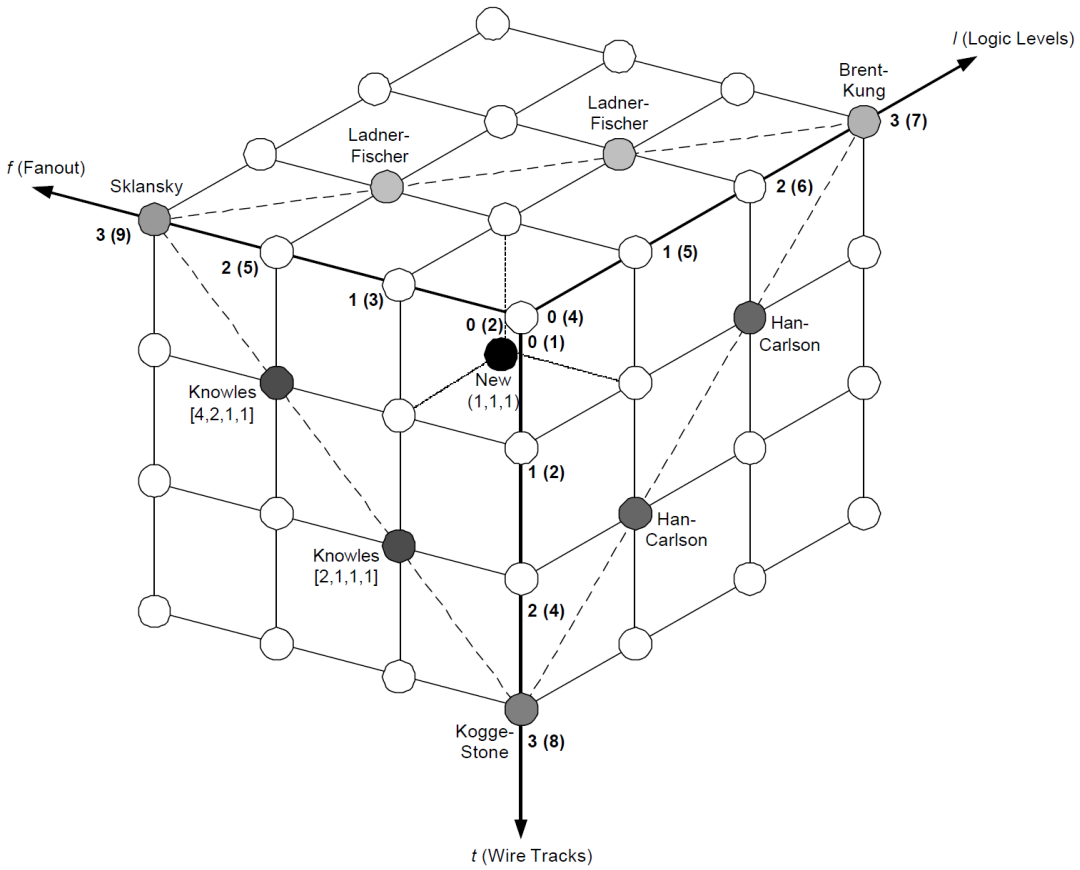



**Fig. 5.5:** Taxonomy of Parallel Prefix Adders with $n = 16$. The three dimensions are logic levels $\log_2(n) + l$, fan-out $2^f + 1$, and horizontal wire tracks $2^t$ with $l, f, t \in \{0, \ldots, \log_2(n) - 1\}$. Picture taken from [116].

three dimensions. This is extremely useful for understanding the relation between the PPrA architectures from several decades but does not necessarily point to an optimum in terms of the Power-Performance-Area metric. The taxonomy shows the Brent & Kung [117], the Kogge & Stone [118], and Sklansky [115] adders are on the extreme ends regarding logic levels, wiring tracks, and fan-out, respectively. On the other hand, Ladner & Fischer, Han & Carlson, and Knowles are representing trade-offs between pairs of the extremes. To limit the number of PPrA variants to analyze, this work focuses on the architectures located at the corners of this triangle as these should be the interesting cases.

### Prefix Graphs

A common graphical representation of those adders is called a *prefix graph*. It consists of prefix cells, i.e. the hardware structure to compute the FCO for two inputs. To

improve readability of these graphs in this work, the cells are not simply shaped or colored differently as it is often done in literature, e.g. [114], [116], but labeled with the index, indicating the operand bit range over which the FCO has been computed at the output of this cell. Fig. 5.6 has an overview of the different prefix cells used. The blue cell transforms the operand bits $(a_i, b_i)$ into the generate and propagate signals $(G_{i+1:i+1}, P_{i+1:i+1})$. A red cell performs a full FCO operation on the two inputs provided whereas a yellow cell does not compute the propagate part. Since $P_{0:0}$ is always zero, $P_{i:0}$ is zero for all $i$ because it is computed via an AND which includes $P_{0:0}$. Hence, the yellow cells save an AND gate compared to the red cells. The green cells are simple buffers. For better visibility the ranges inside the prefix cells including index zero are written in a bold font.

Note that $P_{0:0}$ can be set to zero even when no carry input is needed. This can be seen in two ways: Firstly, it is true for an adder with carry-in set to zero, which has to behave like an adder without carry-in; thus leaving $(c_{\text{in}} = 0, 0)$ out of the prefix calculation has to be identical to not doing so. Secondly, as no carry input exists, a carry can only be generated at index zero. Therefore, all information about a carry from index zero is already captured in $g_0$. To avoid the synthesis of unnecessary gates for a static $c_{\text{in}} = 0$, Eq. 5.13 can be redefined to yield Eq. 5.17. This is important as all gates in the code will be set to "preserve" for the synthesis tool.

$$c_0 = c_{\text{in}} = 0 \Rightarrow (G_{i:i}, P_{i:i}) \coloneqq (g_i, p_i) \Rightarrow c_i := G_{i+1:0} \tag{5.17}$$

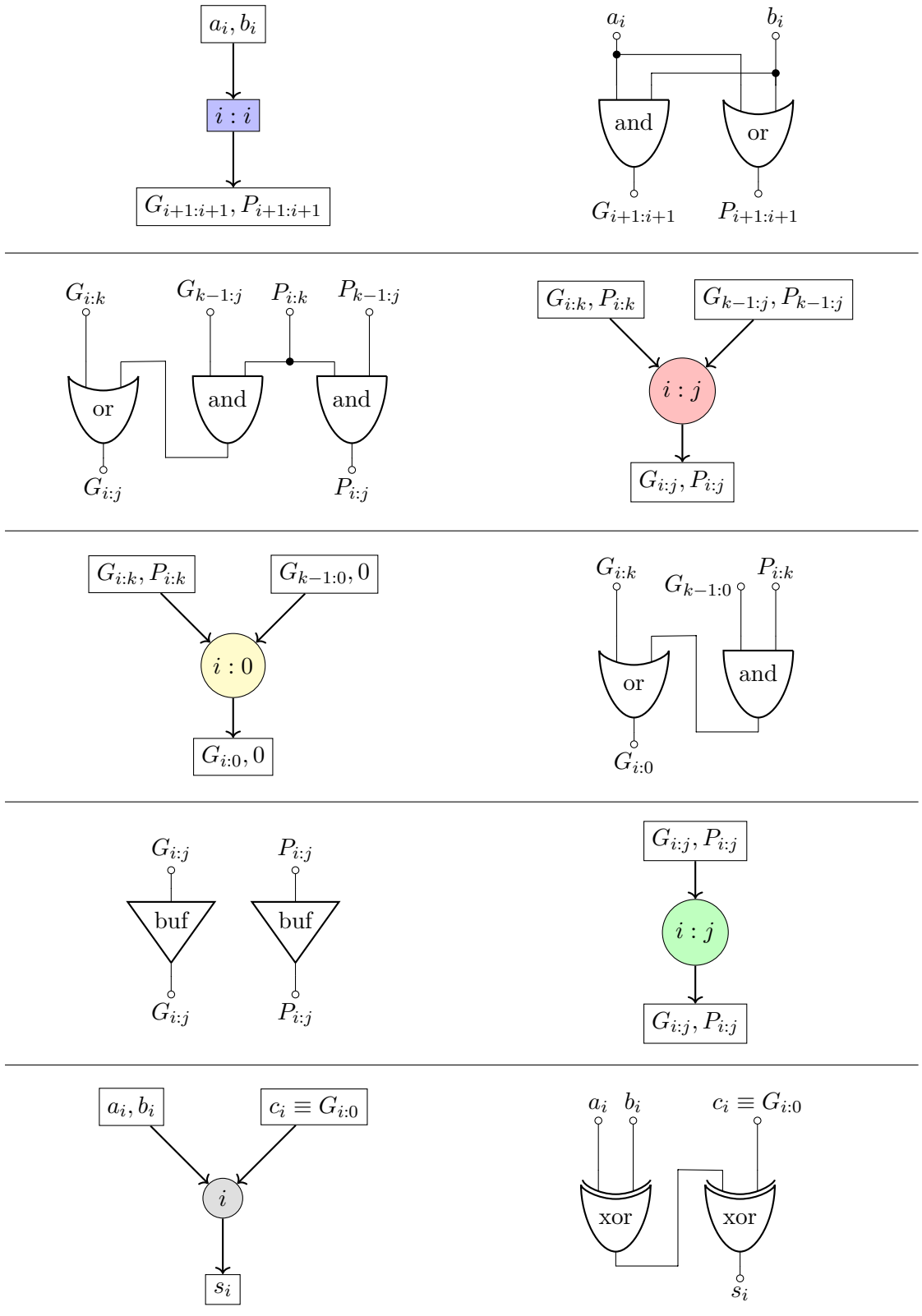



**Fig. 5.6:** Propagate and generate calculation cell (blue), prefix cells (red, yellow, green), and sum cell (gray).

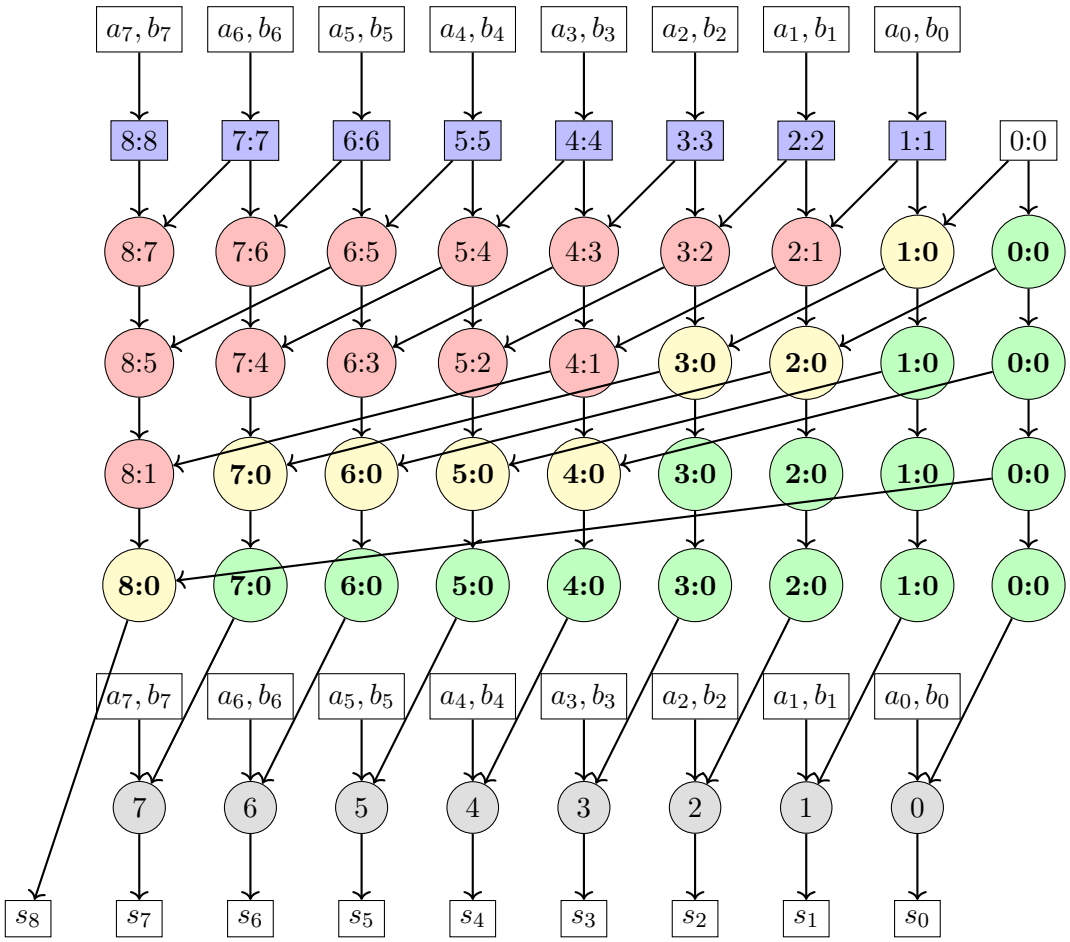



**Fig. 5.7:** 8-bit Parallel Prefix Adder with carry input, expressed in the cells from Fig. 5.6.

Fig. 5.7 shows a whole adder in the graphical notation defined by Fig. 5.6. Since only the arrangement of the prefix cells (red, yellow, green) changes from adder to adder, the other cells are mostly omitted for the rest of this chapter. Furthermore, only adders using the carry input are considered for synthesis. This increases the number of operands in the prefix network by one. However, the prefix graphs later in this section can either be interpreted as graphs for $(n-1)$-bit operands plus a carry-in or graphs for $n$-bit operands without a carry-in. The structure does not change. It is worth to comment on one additional aspect regarding the carry input. In case $n$ is a power of two like in the example above, it seems especially inefficient to provide the carry-in as it causes the prefix tree for $c_n \equiv G_{n:0}$ to contain at least $\log_2(n)+1$ FCO stages. However, this additional delay for $c_n$ is saved in the summation stage because $s_n \equiv c_n$. The use of $c_{\text{in}}$ is not equivalent to increasing $n$ as it does not increase the number of sum bits.

### Brent & Kung Adder

The prefix graph of the adder proposed by Brent & Kung [117] arranges the FCOs in a binary tree. This leaves many columns unfinished and requires a second inverse binary tree to compute all prefixes. As such, its delay is $t_{\mathrm{d,bk}} = (2\log_2(n) - 2) \cdot t_{\mathrm{d,fco}} + t_{\mathrm{d,buf}}$ if $n$ is a power of two. For arbitrary $n$, the formula is more difficult. Note that Harris [116] treats FCO and buffer delay equally for his taxonomy as he only considers the depth in terms of prefix cells (red, yellow, green) and does not distinguish between them. The structural SystemVerilog description developed in this work to synthesize the Brent & Kung adder, and the other PPrAs, does not include the buffer prefix cells. This actually violates the maximum fan-out of two, as given by the taxonomy for the Brent & Kung adder, in some places and also reduces its delay by the before mentioned buffer delay, i.e. by one prefix cell delay. However, buffers are something the synthesis tool can insert automatically when needed and it turns out that a fan-out of three is perfectly fine in a 22nm technology. Otherwise, if all buffers shown in this prefix graph would be synthesized, possible advantages of these adders would be immediately burned.

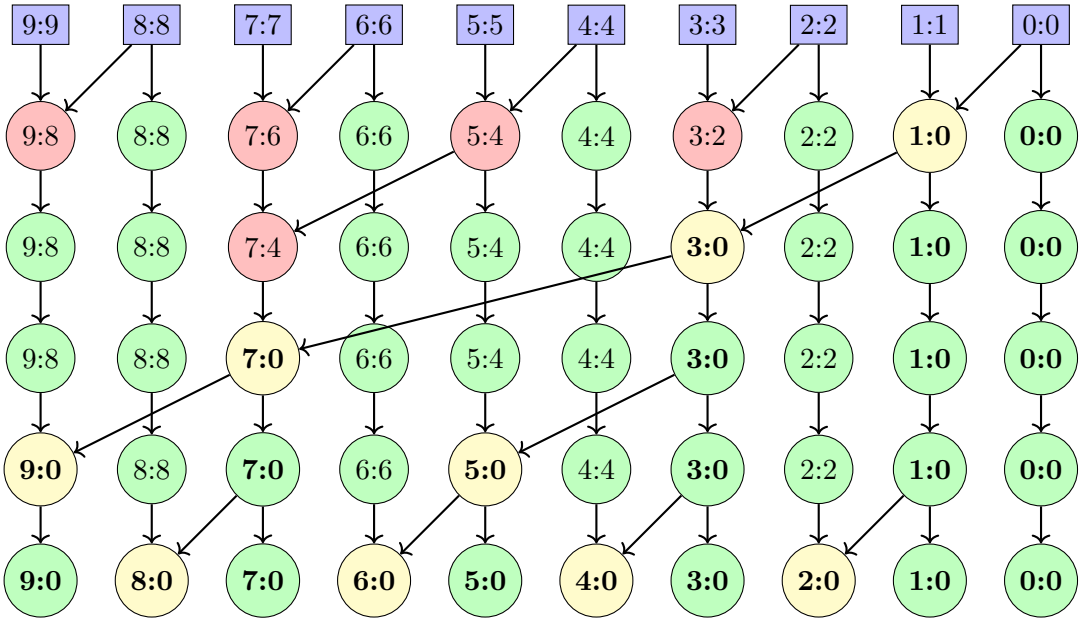



**Fig. 5.8:** 10-bit Brent & Kung prefix graph.

### Kogge & Stone Adder

The adder described by Kogge & Stone [118] includes a prefix graph with the lowest possible delay of $t_{\mathrm{d,ks}} = \lceil \log_2(n) \rceil \cdot t_{\mathrm{d,fco}}$, and a minimum fan-out. If $n$ is a power of two and the carry input is used, the depth of the prefix graph is actually increased by one. However, as explained before, this additional delay is absorbed in the summation stage.

The same is true for the Sklansky adder. The low delay and fan-out are achieved by connecting every two neighboring cells with an FCO in the first stage, then every cell with its neighbor two columns apart in the second stage, then the ones four columns apart, and so on. The cost of this is a large cell count and many horizontal wire tracks. Harris defines the number of wire tracks to be the highest number of parallel wires that cross and/or start/end on the same column in a single stage. An example can be seen in column "4:4" in Fig. 5.9 where three parallel "arrows" with different starting points are crossing this column and one starts at "4:1" – thus this adder is considered to have four horizontal wire tracks. The Brent & Kung adder only has single wires crossing columns in each stage, while the Sklansky adder (see Fig. 5.10) might seem to have multiple crossings but the arrows all start at the same cell and are therefore only a single wire that is tapped multiple times. Hence, both Brent & Kung and Sklansky are located at $t = 1$ in the taxonomy by Harris. The explanation of how these wire tracks are counted is actual very short or almost non-existent in [116]. Furthermore, the concept might be suited to classify the PPrAs, but it might be more expressive to use the prefix cell count instead. Especially in modern technologies, routing congestion in an adder circuit is rather unlikely to cause trouble. However, the overall size of a module influences performance and power significantly (see section 5.4).

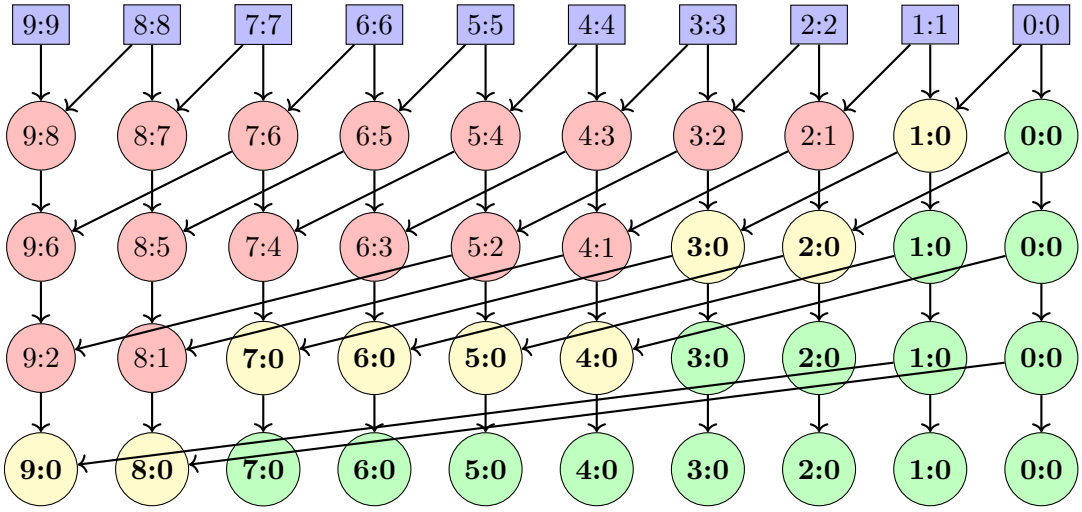



**Fig. 5.9:** 10-bit Kogge & Stone prefix graph.

### Sklansky Adder

The third variant to be considered in this work is the Sklansky adder [115]. It actually originates from a paper on conditional sum addition, but the corresponding prefix graph is sometimes called Sklansky adder, e.g. by Harris [116]. It has minimal delay, identical to the Kogge & Stone adder, but in comparison far fewer cells at the cost of fan-outs as

high as $n/2$. Similar to the multiplexer tree in the conditional sum adder, the carry-in determines the first bit, then the next two, then the next four, and so on. The high fan-out of $n/2$ is of course also found in the conditional sum adder in the last multiplexer stage.

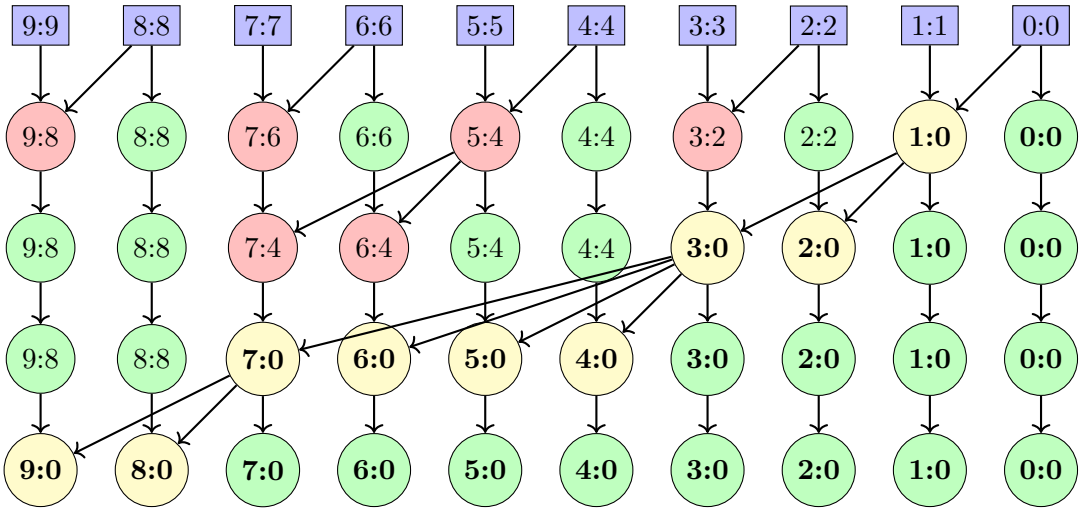



**Fig. 5.10:** 10-bit Sklansky prefix graph.

### 5.2.3 Carry-Lookahead Adder

The term Ccarry-Lookahead Adder (CLA) is difficult to define exactly and needs a historical perspective to differentiate it from the previously discussed adders. Exactly like the PPrAs, it includes a first stage, which translates operand bits into propagate and generate signals. However, instead of feeding these signals into a prefix graph, the disjunctive normal form of each carry bit is implemented as shown in Eqs. 5.18 [119]. Thus, the CLA is considered to be computing each carry bit in parallel.

$$\begin{aligned}
(c_0, 0) &= (c_{\text{in}}, 0) \\
(c_1, 0) &= (g_0, 0) \circ (c_{\text{in}}, 0) \\
(c_2, 0) &= (g_1 + g_0 p_1, 0) \circ (c_{\text{in}}, 0) \\
(c_3, 0) &= (g_2 + g_1 p_2 + g_0 p_1 p_2, 0) \circ (c_{\text{in}}, 0) \\
(c_4, 0) &= (g_3 + g_2 p_3 + g_1 p_2 p_3 + g_0 p_1 p_2 p_3, 0) \circ (c_{\text{in}}, 0) =: (g_G, p_G) \circ (c_{\text{in}}, 0) \\
&\vdots
\end{aligned} \tag{5.18}$$

Since no terms are shared, this method does not scale and gates with fan-in of $\mathcal{O}(n)$ are required [120], or trees of gates with a lower fan-in. If a single gate is to be used, this carry-lookahead logic is limited to at most four or five bits [114], [119]. To alleviate the problem in this case, smaller groups of operand bits, e.g. four bits, are then augmented

with a group generate signal, $g_G$, as well as a group propagate signal, $p_G$. Both are defined via Eq. 5.18. These group signals are then combined using the exact same carry-lookahead logic used in the 4-bit group. This scheme implements term sharing through a tree structure and allows to reach logarithmic time complexity. The carry-lookahead logic essentially includes multi-valency FCO cells. Since the term sharing can be done in many ways within the CLA, the term Ccarry-Lookahead Adder can be and sometimes is actually seen as an umbrella term for Parallel Prefix Adders. While the latter includes only prefix graphs using 2-input FCOs, the former term can comprise multi-valency operators. When the group size is lowered to two bit, the group propagate and generate signals are then simply computed with the FCO, which therefore results in the PPrA version of the group-based CLA described before. This variant is presented in [121] as *the* CLA, and is also called Ccarry-Lookahead Adder in the remainder of this work even though the umbrella term Ccarry-Lookahead Adder is a valid definition. The prefix graph of this variant is shown in Fig. 5.11.

An important distinction to the previously presented PPrAs is how the carry-in is handled. The prefix graph of the CLA is built by grouping the operand bits into pairs of two within the first stage, regardless if the carry-in is used or not. This leads to an inefficient handling of the carry-in and causes an additional FCO stage, independent of $n$. The graph without the carry-in, or rather including the carry-in as an additional operand like the other PPrAs, is derived from the one in Fig. 5.11 by shifting the yellow cells one row towards the top and adjusting the connections made to the carry-in column and the indices properly. The delay of this CLA is similar to the Brent & Kung prefix graph (without the one buffer delay) if the carry-in is included as an additional operand: $t_{\mathrm{d,cla}} = (2\log_2(n) - 2) \cdot t_{\mathrm{d,fco}}$. Otherwise, it is one FCO delay greater. Tab. 5.3 attempts to place both variants of the CLA in the taxonomy of Harris [116]. The number of wire tracks is 2 when the carry-in is treated separately. This can be understood with Fig. 5.11 where the connection from "0:0" to "8:0" runs in parallel to the wire from "4:0" to "6:0" and "5:0". In case the carry-in is included in the operands, the former connection would also start at "4:0" thus leaving only a single track. The fan-out on the cell driving this single wire is therefore increased by one. To the knowledge of the author, there has been no previously reported attempt that puts the CLA into this taxonomy.

| Version | Logic Levels $\log_2(n) + l$ | Fan-out $2^f + 1$ | Wire Tracks $2^t$ |
|---|---|---|---|
| CLA (included $c_{\mathrm{in}}$) | $2\log_2(n) - 2$ | $\log_2(n)$ | 1 |
| CLA (separate $c_{\mathrm{in}}$) | $2\log_2(n) - 1$ | $\log_2(n) - 1$ | 2 |
| Brent & Kung | $2\log_2(n) - 1$ | 2 | 1 |
| Kogge & Stone | $\log_2(n)$ | 2 | $2^{\log_2(n)-1}$ |
| Sklansky | $\log_2(n)$ | $2^{\log_2(n)-1}$ | 1 |

**Tab. 5.3:** Placing the Ccarry-Lookahead Adder in the taxonomy suggested by Harris.

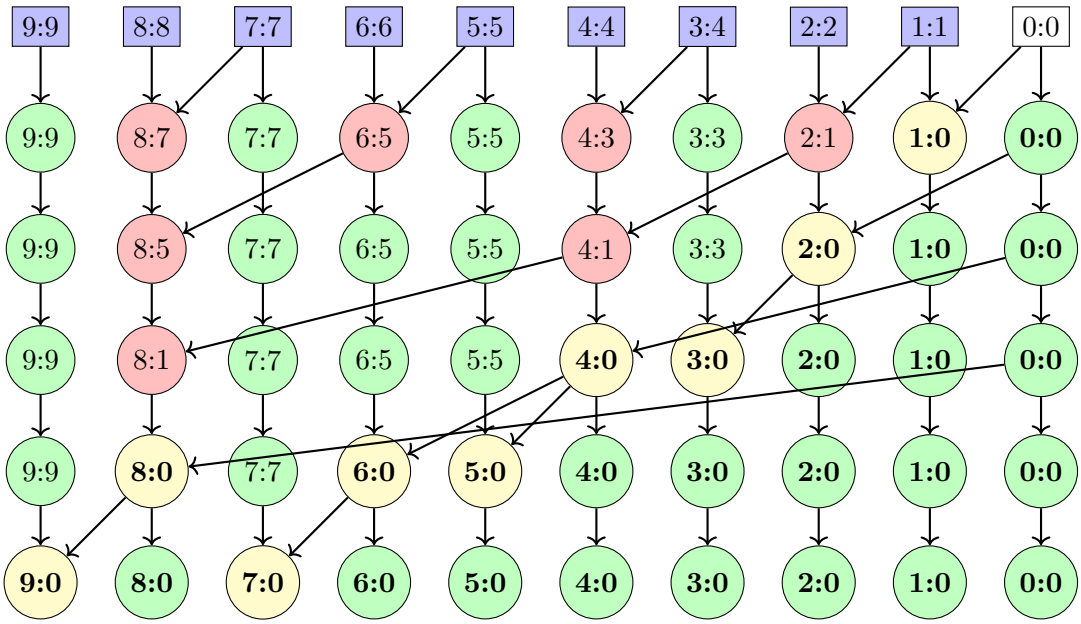



**Fig. 5.11:** 10-bit carry-lookahead prefix graph (separate $c_{\text{in}}$).

## 5.3 Verification

The SystemVerilog code describing the multipliers and adders discussed in the previous sections had to be verified to assure they are logically equivalent to a "+" and "∗" operator. This was done using formal verification tools. Formal verification defines correct behavior to be equivalent to non-violated assertions. This requires correct and exhaustive assertions, and that all valid stimuli are applied to the DUV to ensure the assertions are correct in all cases. However, the verification of simple arithmetic functions is much simpler. A single assertion checking for equivalence of the structured code and the "+" and "∗" operators is sufficient. There is also no sequential complexity because all modules are combinational functions. JasperGold from Cadence has been used for this job as it provides a Tcl interface, which allows to script verification across the operand size $n$. The verification itself is done by so-called *engines*, of which JasperGold provides approximately thirty. These engines apply different algorithms to prove the correct behavior of the DUV, and are therefore suited for different types of designs and assertions. Some are looking for errors with very deep sequential traces, while others test more input combinations but only a few cycles each. However, the engine descriptions and names did not reveal an obvious choice for arithmetic functions so a test run was performed with every engine using an 11-bit Wallace multiplier. Fig. 5.12 shows that the engine G2, a variant of the G engine, is optimal. With the exception of the Tri engine, all other engines had significantly longer runtimes.

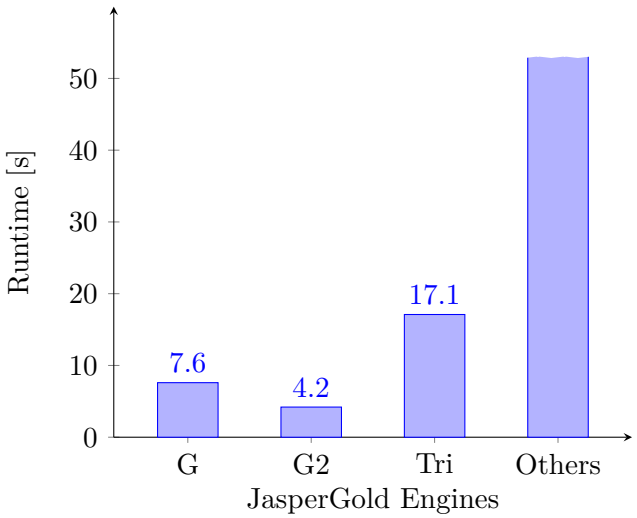



**Fig. 5.12:** Runtime comparison of JasperGold engines for an 11-bit Wallace multiplier.

Fig. 5.13 shows the runtimes needed to verify the multipliers with the corresponding tree for different operand widths $n$. The exponential increase is a result of the $2^{2n}$

possible input values. A runtime of above one hour is reached at $n = 17$ already. The Wallace multiplier seems to need less time and reaches $n = 18$. This result shows the limit of formal verification when the tool cannot make simplifying assumptions since it does not recognize the structural code to represent a multiplier. The operand sizes that have been synthesized were therefore verified using simulation – but of course not exhaustively. It should be noted that the adder implementations can be proven for huge operand size, e.g. 512 bit, in a couple of seconds using the Tri engine. A possible reason for this could be that the adders are written using loops, while the multipliers are simply a generated list of full- and half-adder instances. JasperGold seems to be able to make certain simplifications for loops, which greatly speeds up the verification.

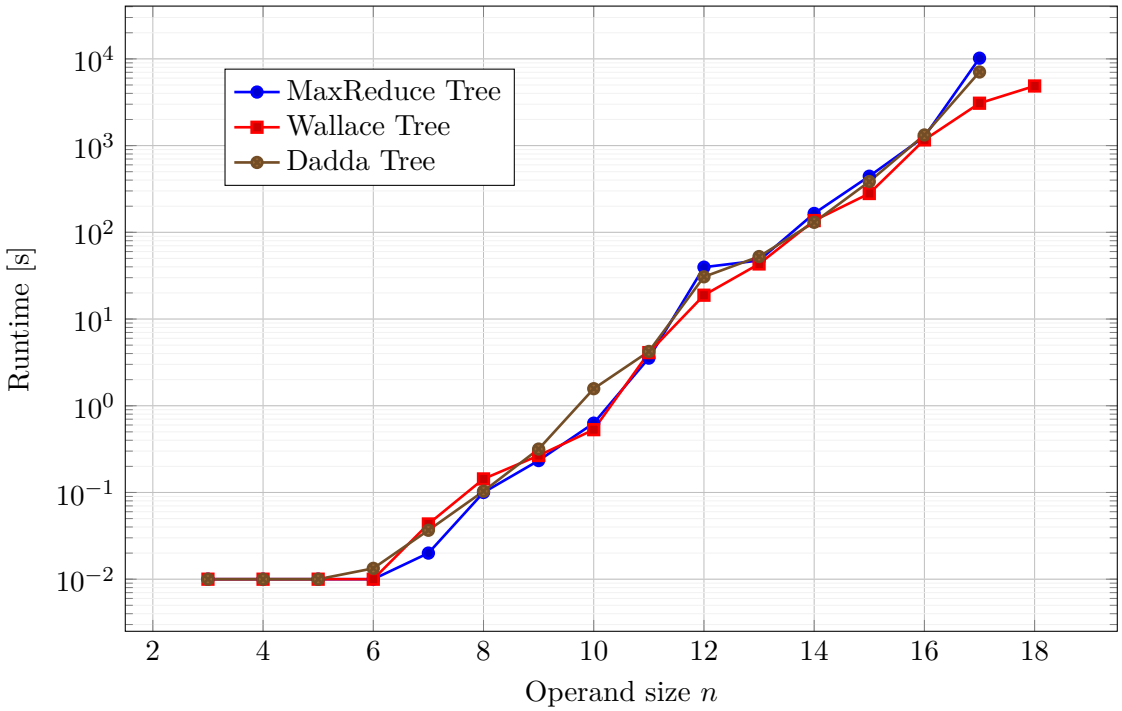



**Fig. 5.13:** Runtime for different multipliers with JasperGold's G2 engine.

In addition to JasperGold, this work also verified some of the arithmetic modules using the open-source tool SymbiYosys [122]. Similar to JasperGold, it provides multiple engines. However, a more detailed analysis is beyond the scope of this document.

## 5.4 Synthesis

The multipliers and adders presented in this chapter are analyzed in terms of the Power-Performance-Area metric in this section. The first subsection presents the methodology used to produce the results shown in the second subsection.

### 5.4.1 Methodology

Comparing synthesized designs has a huge potential to compare "apples and oranges". Therefore, this first subsection is dedicated to describe the methodology in more detail.

#### Gate Transfer Level

All arithmetic modules have been expressed with standard cell instances using a specific instance name, which allows to filter them in the Cadence Genus Tcl script interface. This enables preserving these instances, i.e. to prohibit the synthesis to switch them with other standard cells during optimization steps. Only the driving strength is allowed to be modified. This conserves the architecture that is synthesized, which is crucial because otherwise only the starting point for the synthesis might be a Kogge & Stone adder, but the actual synthesized design is just some circuit that behaves like an adder. For the counters within the multiplier trees, full- and half-adder standard cells have been employed. The final adder is left for the synthesis tool to decide. It is just a simple "+". The prefix cells used by the PPrAs are comprised of an AO21 complex-gate for the reduced (yellow) FCO and an additional two-input AND gate for the full (red) FCO. Despite being a complex-gate, it is available in multiple technologies. The other prefix cells are implemented as shown in Fig. 5.6, except for the buffer, which is simply ignored and can be inserted by the synthesis tool if needed. The sum bit is not computed via a three-input XOR because the operand bits can already be processed in parallel to the prefix graph.

#### Synthesis Flow

The target technology is a 22nm FDSOI process. A set of shell scripts was used to execute the flow for the different modules and provided a simplified command line to ensure all synthesis steps are conducted consistently between runs. Simplified, the basic steps for logic synthesis are the following.

- Generic Synthesis: express HDL code with generic gates
- Mapping: map generic gates to standard cells
- Placement: place standard cells and pins in floorplan

- Clock-Tree Synthesis: insert clock tree starting from clock pin
- Routing: connect standard cells with wires
- Signoff: final checks for timing and Design for Manufacturing (DFM)

The first two steps are commonly known as the *front-end*, which transforms the HDL code into a Gate Transfer Level (GTL) netlist. A GTL netlist expresses the design with the cells found in the corresponding standard cell library of the target technology. During this process it also performs logic optimization to ensure the GTL-netlist achieves timing, i.e. is free of setup- and hold-violations. The netlist is then handed over to the *back-end*, which executes the remaining four steps. The back-end tool places the standard cells into a defined floorplan area, creates a clock-tree, and routes the wires to connect all cells. As a final step, a detailed RC-extraction is performed to get an accurate timing information including all parasitics.

This is the traditional flow, which has obvious drawbacks. The first is its forward direction that strictly separates the single steps from HDL to layout. As a result, this flow cannot reliably find the optimum physical implementation of the HDL design. There is no way for the front-end to know which challenges the back-end has to face to place and route the netlist. In literature, the 250nm node is often considered to be the turning point where the gate delay has become smaller than the wire delay [123]. Thus, the contribution to total path delay that the front-end considers (gate delay) has turned from dominant to small, while the reversed situation is true for the back-end (wire delay). Therefore, this flow has become extremely inaccurate with shrinking feature sizes. While there have been changes "on both ends", the overall structure remained unchanged until today.

The front-end has been improved with *physical-awareness*, and the back-end can do *in-place optimization*. The former allows to add various levels of physical information to the logic synthesis, e.g. to provide a floorplan and call the back-end tool to trial-place the current netlist to estimate wire lengths, which are then combined with RC-data from the technology file to improve the accuracy of the netlist. Conversely, the back-end tool has been given some methods to alter the netlist. Most prominently, it can insert buffers or duplicate logic to match driving strength requirements. However, it cannot do logic transformations, so it is still dependent on a certain quality of the front-end flow.

It is an important step to realize this as it explicitly presents a drawback of high-level HDL code. Precisely, if the implementation of a "+"-operator is decided by the front-end tool to contain long "ripple"-paths because the target clock frequency is rather low, the back-end may still not be able to implement the design. It has no chance to split this long path in two parts if it is necessary due to wire delay.

### Floorplan

Synthesis adds the additional complexity of physical extension to the logical description of a design. Two designs can now be different, even when the overall logical behavior and HDL description do not change. So to ensure comparability between the results, several aspects need to be considered. The first aspect to address is the constraint file (.sdc), which specifies the target clock frequency. It usually also includes input and output delays, as well as the load assumed to be present at output pins. However, arithmetic modules are usually found within pipelines, i.e. they are preceded and succeeded by a register stage, and input and output loading are determined by the size of these registers. Therefore, to avoid estimating (arbitrary) values for these constraint parameters, the actual synthesized design includes a register stage before and after the combinational logic.

Another possible inconsistency may be the summation order within the multiplier trees, which is logically not important but could influence the synthesis results. This problem is similar to changing the order of the pin placement. Since every pair of operand pins is needed for the partial product generation, the place & route algorithm has enough opportunities to distribute the partial products regardless of the pin order. This has been tested and timing differences were negligible. Hence, it should be save to assume that the order of partial product summation is also not significant.

The next issue is the floorplan, especially the pin placement. The floorplan has been made $(200\,\mu m)^2$ in size, which is enough to easily accommodate all circuits that have been synthesized. All pins were placed centered on the left side because placing them on opposite sides requires to scale the floorplan to provide a fair comparison over different operand widths. To argue for a certain scaling factor as a function of $n$ for every design would be difficult and time consuming. The pins themselves are located on metal 4 as this the first horizontal layer excluding the smallest layers metal 1 and metal 2. The standard cell library is an eight track library and the spacing between pins has been set to two tracks for the adders, and six tracks for the multipliers. Lower and greater spacings were tested but did not significantly impact timing. Operand and result bits are interleaved in a `{a[i],b[i],product[2*i],product[2*i+1]}` and `{sum[i],a[i],b[i]}` manner, respectively. The clock pin is centered in between.

### Timing Analysis

To measure if the clock period constraint is met, timing analysis is needed. This is more complicated than simple sums of gate delays. As with every manufactured good, there are some tolerances between instances. For semiconductor chips, these tolerances cannot be pushed to a low enough level to be considered insignificant. Hence, they have to be accounted for during the design phase to avoid low yield through many defective devices.

Commonly, these variations are accounted for by using so called PVT-corners. Historically, there were process (P), voltage (V), and temperature (T). The former describes variations of the semiconductor properties, e.g. due to differences in doping concentration. The latter two are operating conditions. As mentioned before, the delay contribution of wires rose from insignificant to dominant compared to gate-delay with shrinking process nodes. As a consequence, RC-corners were added to include variations in metal geometry affecting resistance and capacitance. PVT- and RC-corner combinations, where the combined delays are especially high or low, are then used to model global variations. However, there are also local variations along paths, which is called *on-chip variation*. To increase or decrease the delay globally, i.e. independent of the path length, is either too pessimistic or too optimistic – for example it is unlikely that all devices along a long path are all slow (or fast). Rather a statistical approach is desired, which provides a probability distribution of the path delay [124]. This can be done with Statistical On-Chip Variation (SOCV), which is supported by the Cadence synthesis flow and was used in this work.

#### Power Analysis

Power consumption was determined in this work with the GTL-netlist of the design after the post-route optimization run. This netlist was simulated within a testbench, which applied randomized operands to it and dumped all waveform data into a value change dump file (.vcd). This waveform data can then be read by the synthesis tools and used to sum up the number of charge/discharge cycles for each net in the extracted physical layout. By simulating thousands of calculations, a stable average power consumption is achieved. The power is calculated for typical operating conditions, but the synthesis is done with slow and fast corners.

#### Register Retiming

The placement of register stages, which was done manually for the FMA unit, can actually be automated with Genus. This feature is called *register retiming*. Genus can analyze the logic and move registers within combinational logic to optimize the timing. However, the quality seems to heavily depend on the amount of physical information Genus can use during optimization. It is not possible to transfer the retime property into Innovus. Therefore, the position of the retimed registers is fixed for the back-end flow. This makes the approach very rigid as the most complicated delay part, the wires, are not considered for retiming properly. Nevertheless, with future tool improvements this feature will likely make manual register placement obsolete. Retiming has been investigated at the CAG but has not been applied to the results presented in this section.

### 5.4.2 Results

The power consumption of Complementary Metal-Oxide-Semiconductor (CMOS) circuits is commonly described in literature by Eq. 5.19. It is split into two parts, the first term describes the dynamic power consumed through charging and discharging wires, gates, and other parasitic capacitances. The overall consumed power is then determined by the number of such charge/discharge cycles per time. In Eq. 5.19, the capacitances for the whole circuit are summed up into $C_{\text{total}}$ and multiplied with an average switching factor $\alpha$. The second part is called static power because it does not depend on the clock frequency $f_{\text{clk}}$ and is caused by a continuous leakage current $I_{\text{leak}}$, which is the sum over all standard cells in the circuit.

$$P_{\text{cmos}} = \alpha f_{\text{clk}} C_{\text{total}} V_{\text{DD}}^2 + I_{\text{leak}} V_{\text{DD}} \tag{5.19}$$

Both $C_{\text{total}}$ and $I_{\text{leak}}$ are proportional to the number of gates and nets in the circuit, which in turn are proportional to the area required by the circuit. For a PPA analysis, it is more convenient to use Eq. 5.20. The operating voltage used for the presented results is constant, $V_{\text{DD}} = 0.8\,\text{V}$.

$$\frac{P_{\text{cmos}}}{A_{\text{cmos}}} = \beta f_{\text{clk}} + \gamma \tag{5.20}$$

Thus, power density is expected to be a linear function of clock frequency. The plots in this section only show the area and power of the combinational logic, i.e. without the register stages. Only an operand width of 64 bit is considered because the scaling with the operand width is similar between all structures and does not provide much insight.

#### Adders

The three PPrAs discussed in section 5.2 as well as the CLA have been analyzed from 500 MHz up to their maximum frequency in steps of approximately 100 MHz. As a reference, an adder using the "+"-operator of SystemVerilog has been included. The Kogge & Stone adder needs significantly more power and area compared to all the other adders, which was expected from the prefix graphs. Together with the Sklansky adder, it reaches a much higher maximum frequency compared to the CLA and Brent & Kung adder. This is also not very surprising due to the lower number of FCO stages. Nevertheless, the "+"-adder performs much better in regard to all metrics. It can use all standard cells available, which is clearly utilized by Genus for a more optimized structure. Unfortunately but expectedly, the netlist produced by Genus is not easily interpreted. From the gates report of the fastest variant it can be observed that a large number of gates used by Genus have an inverted output, i.e. NAND, NOR, or complex-gates like OAI21. These are of course cheaper in CMOS in terms of area and delay compared to

their counterparts used in the structured code.

Fig. 5.16 shows that the linear scaling with target clock frequency from Eq. 5.20 can be observed. The area increase with rising target frequency in Fig. 5.15 is due to the higher driving strengths needed to meet timing. It varies more for the "+"-adder because it is also allowed to add more logic to speed up the circuit.

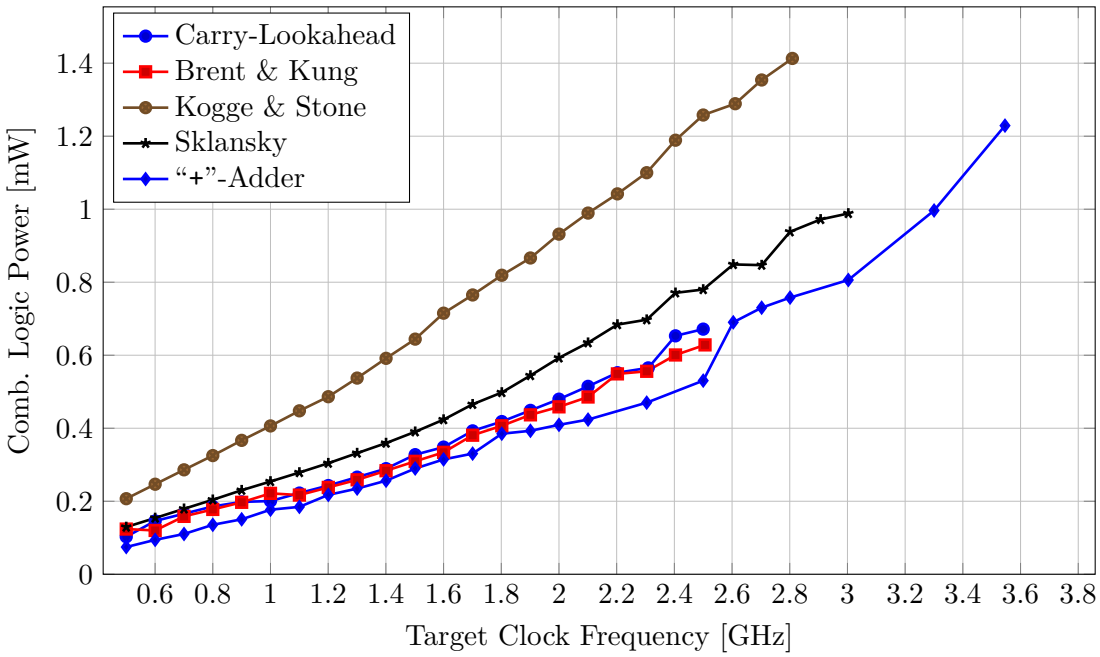



**Fig. 5.14:** Power consumption of the combinational logic over target clock frequency for 64-bit operand adders including a carry input.

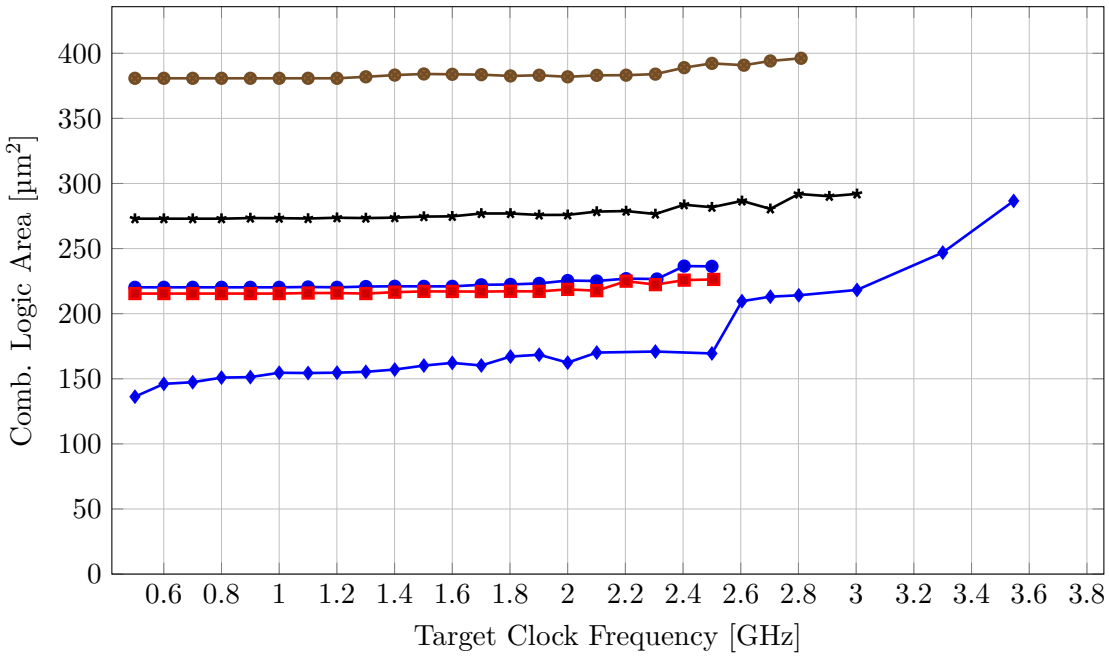



**Fig. 5.15:** Standard cell area of the combinational logic over target clock frequency for 64-bit operand adders including a carry input.

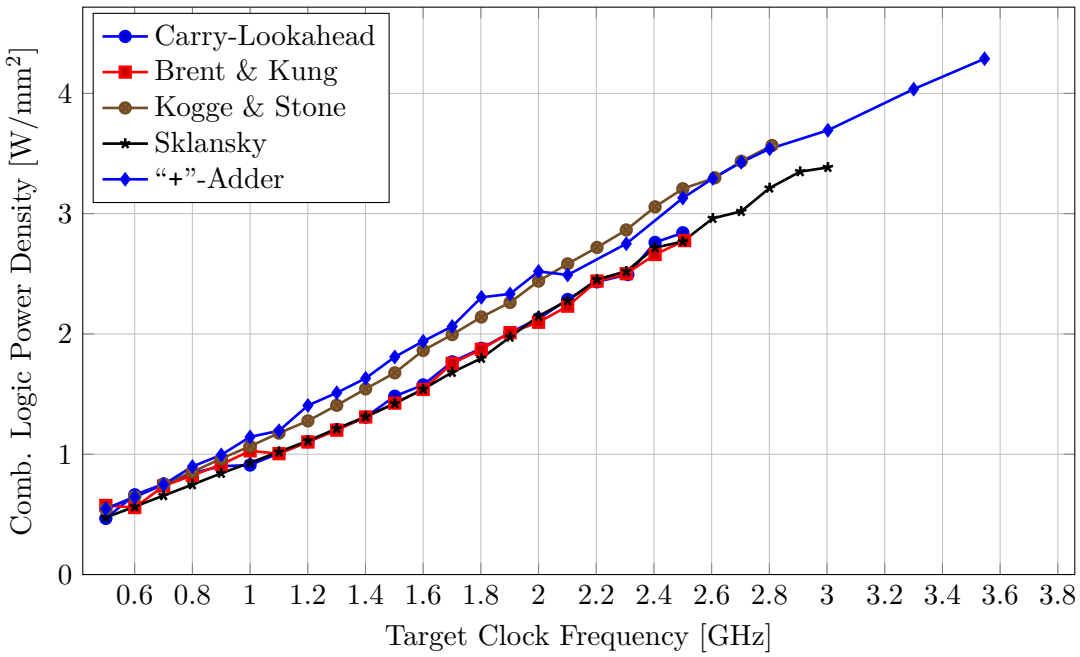



**Fig. 5.16:** Power density of the combinational logic over target clock frequency for 64-bit operand adders including a carry input.

### Multipliers

The three multiplier trees presented in section 5.1 and a multiplier implemented with the "*"-operator of SystemVerilog have been synthesized. Up to 1 GHz, all structured variants behave rather similar. Beyond that, the Dadda tree is the optimal solution, which could be expected because it uses the least amount of standard cells and is basically an optimized version of the other trees. The "*"-multiplier uses roughly one fourth less area compared to the other multipliers. The gate report reveals that among full-adders a large number of multiplexers and OAI22 complex-gates (two two-input OR gates connected with a NAND gate) are used. This indicates that *booth recoding* is used, a technique to lower the amount of partial products, which has not been covered in this thesis but can explain the reduced area. However, the gate report of the fastest variant shows a much broader mix of gates, indicating no particular structure. In summary, the custom structures are once again not worth the engineering overhead – in fact, they perform worse. One advantage of fixed structures, which had less weight with the adders, is that the synthesis of the SystemVerilog operators takes considerably more time. This chapter only analyzed the tree structures with full- and half-adders, but as Dadda [111] explained, the usage of larger counter cells can decrease the number of stages. However, implementing these with existing standard cells does not result in a speed advantage. This is only possible through standard cells explicitly designed for this use case. The most common larger counter appearing in literature is the *4:2-compressor*, a (5,3)-counter with both carries having the same weight. It is available in some standard cells libraries but not in the one used for the results presented in this section. It remains for future work to analyze if synthesis tools can make proper usage of it, or if a custom tree comprised of such cells could actually outperform their results.

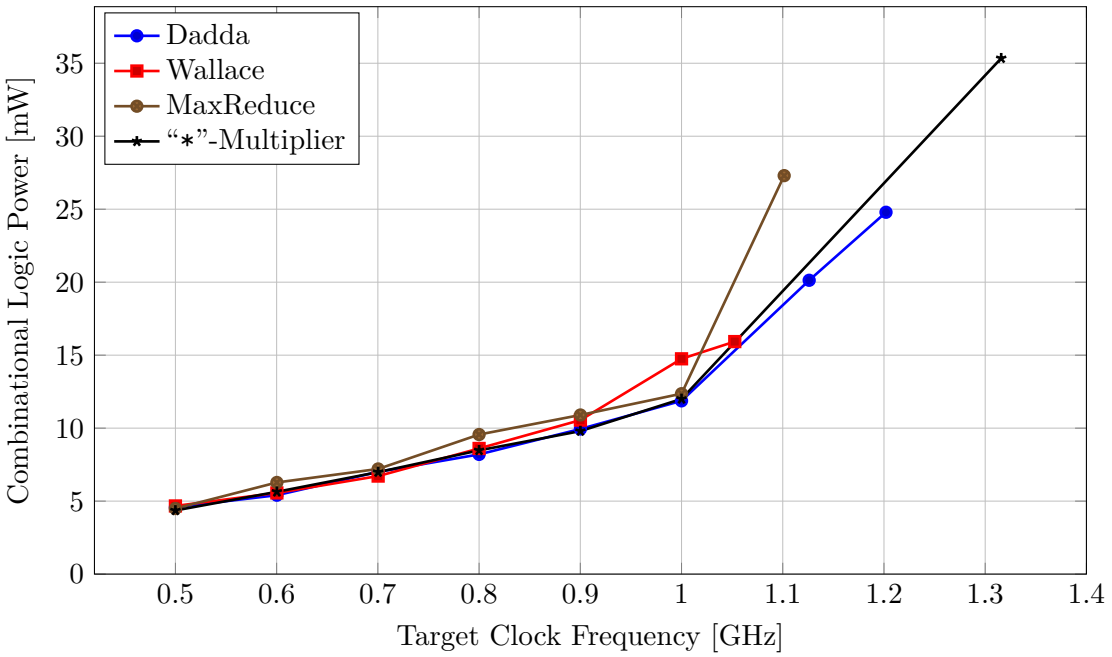



**Fig. 5.17:** Power consumption of the combinational logic over target clock frequency for 64-bit operand multipliers.

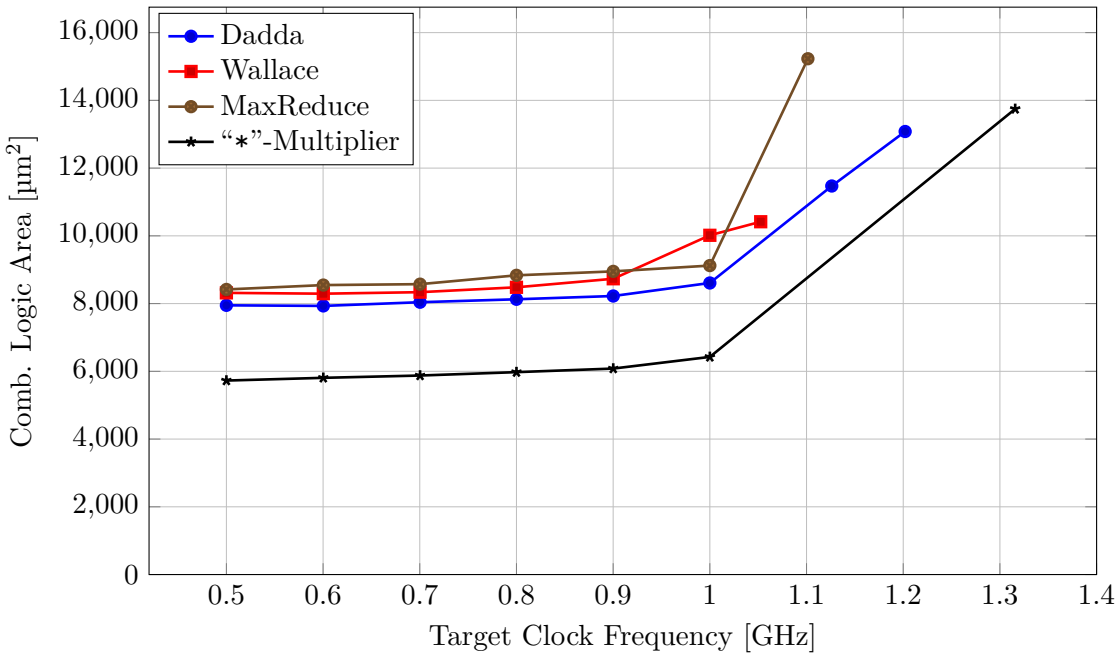



**Fig. 5.18:** Standard cell area of the combinational logic over target clock frequency for 64-bit operand multipliers.

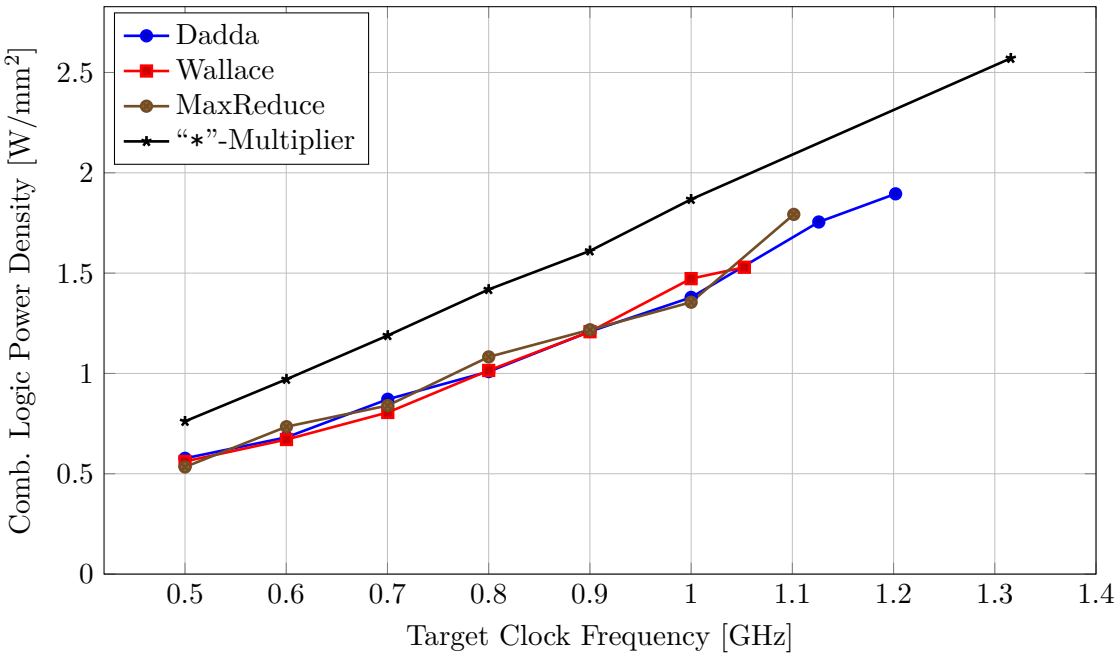



**Fig. 5.19:** Power density of the combinational logic over target clock frequency for 64-bit operand multipliers.

### 5.4.3 Conclusion

Literature on FMA unit design usually presents an architecture similar to Fig. 4.5, e.g. in [100]. By doing so, the advantage of "sneaking in" a third operand after the partial products have been compressed by a carry-save tree, with only a single full-adder delay, is highlighted. This led Kaiser [7] to implement such a tree to follow the design proposed in literature. However, as the results in this section have shown, the effort spent on designing customized arithmetic logic is not increasing performance, nor is it decreasing power or area compared to state-of-the-art synthesis tools. However, a small advantage of structured code is that the synthesis algorithms are more likely to hit the timing if it is possible. With the "`+`" and "`*`" operators the timing is sometimes missed by just a few picoseconds after the final optimization, even when higher target frequencies are definitely possible – which may be due to the fact that the final optimization step has limited capabilities in what it can do. This starts to occur also for the structural code but only when getting close to the maximum frequency. It may be possible to circumvent this by using a higher degree of physical awareness in the front-end tool. Another advantage of structural code is a reduced runtime, especially for multipliers and higher target frequencies. A test has revealed that a simple "`a*b+c`" with 64-bit operands achieves almost the same speed as an "`a*b`", which shows that the manual effort of a 3:2-compressor row is not necessary.

A look at open-source projects, e.g. in the RISC-V context, reveals that these projects do not implement custom arithmetic. However, it is not documented if this is due to coding efficiency reasons, i.e. time constraints, or because an analysis comparable to this work has been done. An example would be the "fpnew" of the PULP Platform, which includes an FMA unit that uses the SystemVerilog operators for multiplication of the significands and addition of the third operand [92]. Western Digital Corp.'s 32-bit RISC-V SweRV cores do not support floating-point instructions, but the integer multiplier is also implemented with a "$*$"-operator.

Fig. 4.8 has shown the power consumption of the FMA unit, which included sequential cells. At 1 GHz, the power of only the combinational logic is 85% of this value, i.e. 36 mW. From additional analysis done in this work, it has been found that the multiplier tree plus the subsequent adder are only making up for around one fourth of this power. This highlights that the large amount of shifting necessary to align the mantissas of the operands in floating-point formats has a significant contribution to the overall power consumption.

The design of arithmetic units comprises many more interesting aspects. For example, it can be shown that the average length of the longest carry propagation is $\log_2(n)$ [119] (carry-completion), or that the CPA following a carry-save tree can be optimized with respect to the different path delay of each bit [125]. And there are many more functions within an execution unit of a processor. While this is beyond the scope of the present work, future work may investigate if it is worth to implement these functions in a customized fashion – although the results of this chapter suggest that this will not be the case.

Chapter 6

# Conclusion

In order to keep the performance of HPC systems increasing, it is crucial to augment the I/O-bandwidth and power efficiency of each node. Both aspects were addressed in this thesis – in the form of on-chip inductors and arithmetic circuits.

## 6.1 Summary

High-speed I/O is usually realized using SerDes technology to circumvent the pin limitation of chips and packages. The transmitted serial data contains frequencies in the multi-gigahertz range, which demands treating the channel as a transmission line, and hence also on-chip signal termination is required. However, this is a goal conflicting with ESD protection, which is commonly realized using diodes. Their parasitic capacitance increases the signal reflection back into the channel, but at the same time, they need to be large enough to deflect high discharge currents during an ESD event. This thesis systematically examined possible circuits using on-chip inductors to improve impedance matching. The effects of series-resistance and the capacitance of the pad were analyzed. It was found that the environment of the pad has a very strong impact on its capacitance: Not only is the pad capacitance in the same order of magnitude as the ESD diode capacitance, it will also cause signal skew if the structures below the P- and N-pad are not designed similarly. Hence as a major result, the necessity to carefully revisit the power grid and metal fill below the pad for future designs was demonstrated. The first layer below the pad should not contain power grid at all but only manually created metal fill for an accurate treatment of its contribution to the overall capacitance, and to avoid post-layout changes through the automatic fill tool. Furthermore, the power grid beneath the fill should be reduced as much as possible to still allow the necessary power delivery. Fortunately, the

first two layers below the pad are shielding it from further structures on lower metal layers, so the designs below are not significantly impacted in their degrees of freedom.

Today, inductors are rarely found in PDKs and if they are, their intended use cases are oscillators and wireless applications, leading to very large area footprints. Thus unlike for PDK devices, there is neither a model nor layout generation available for use during circuit design and optimization if custom inductors are required. Literature on ESD compensation typically only presents the performance of one particular inductor design, not how the results – especially the layout – were devised. In fact, the process seems to be heavily experience-based with trial-and-error being the only – time-consuming – way to find a layout matching the schematic design. In order to accelerate this iterative process, it is necessary to automate the layout procedure. With this goal in mind, this thesis evaluated different techniques to create parameterized cells, namely SKILL PCells, PyCells, and XCells. SKILL PCells turned out to be feasible and were used for the most part throughout this work as they are perfectly integrated into the Cadence Virtuoso environment. PyCells were also evaluated, but difficulties with technology file formats ruled them out. And last but not least, XCells were considered, which were developed at the Computer Architecture Group at Heidelberg University. They are based on a Python to SKILL translation and are applied to PDK devices via the constraint editor in Virtuoso. Although they are promising, they were only briefly evaluated in this work.

Another contribution made by this work is a detailed overview of analytical inductance calculation. As a starting point, the inductance of the straight round wire was considered, and a new expression for its internal inductance was derived, which was shown to actually depend on the wire radius – unlike the much simpler expression found throughout literature. Furthermore, the mean distance approximation derived from the Neumann integral was explained, and various inductance formulas from literature based on this method were compared to the previously derived exact formula for the straight round wire. The section concluded with the result that the Neumann formula becomes inaccurate for very short wires. Unfortunately, no satisfactory explanation for this could be found in the context of this work.

This work identified lumped circuit models as being superior to segmented circuit models, due to their simplicity. If high accuracy is required, S-parameters can be used to outperform the latter, and for circuit design and optimization lumped models are necessary. Using a two-terminal inductor as example, the influence of oxide and substrate, skin effect, metal fill, and process corners was studied. It was demonstrated that skin effect and metal fill have a lower effect on inductance than on resistance. Nevertheless, metal fill for inductors should be included into the PCells and not left for the automatic fill tool in order to properly include it into simulations. Mohan [40] applied the mean distance method to derive an expression for the series inductance of planar spiral inductors.

However, this formula is only valid at DC and can not be applied blindly for today's multi-gigahertz designs, without the verification through field solvers.

On these grounds, layout extraction and different field solvers were analyzed throughout the course of this work. In order to accelerate feedback on the performance of a new inductor layout, the entire design flow was automated – from entering an inductor's geometry parameters to writing an S-parameter file with the simulation results. This utilized Cadence Quantus QRC for layout extraction and Cadence EMX as a field solver. A portion of this part of the thesis was presented at CDNLive 2018 [16]. Layout extraction proved to be unreliable at higher frequencies and was thus not further evaluated. However, EMX is accurate and also very fast. Although, layout extraction still has the major advantage that the technology file is always available, while for field solvers stack-up files are not comprehensively supplied. If they are provided, a plethora of different formats requires the correct field solver to utilize them. In the context of this thesis, a stack-up file was manually created to simulate T-coils on a stack-up without support for ADS, which was tedious, time-consuming, and presents a potential source of errors. Therefore, a major conclusion of this thesis is that the field solver should be chosen in accordance to the stack-up descriptions available in the PDK, if possible. Again, similar to the PyCells, the technology file format jungle causes significant problems highlighting the dire need of more standardization.

Moreover, this thesis presented a method to close the design gap of custom on-chip inductors from schematic to layout by making use of an MCMC fitting technique to map S-parameters to lumped models. In conjunction with the aforementioned results, namely the analytic inductance formulas and the automated simulation flow, a model function for T-coils that maps geometry to lumped model parameters could be created. With this, the optimal layout for a given inductor can be calculated with reasonable accuracy. While the model is only accurate for one type of T-coils and this 22nm technology, the way it was created can be transferred to other structures, even to ones that are not inductors. For example, the pad capacitance in chapter 2 was extracted using the same fitting technique. By creating such a model instead of relying solely on formulas from literature, the designer ensures no assumptions and approximations that could lead to an erroneous design have been missed. And compared to the trial-and-error approach relying only on simulations, the presented methodology is less time-consuming and identifies the optimal layout in a straightforward way.

The RISC-V ISA has spawned many processor developments throughout industry and universities in the recent years. The CAG at Heidelberg University also started to develop a RISC-V-based processor core. However, in order to enter the realm of HPC, fast arithmetic circuits are needed. Therefore, Kaiser [7] developed an FMA unit for

IEEE 754 floating-point numbers at the CAG. This work analyzed the design in regard to PPA, and complemented the verification of Kaiser with a driver and checker module that employs the TestFloat tool developed by Hauser [76], [77]. Kaiser followed the design principle advertised in literature that suggests building a multiplier tree and adding the third operand with a 3:2-compressor row. The synthesis results revealed that many critical paths are located within this tree, thus a more detailed analysis was conducted in this work to research potential gains through custom gate-level arithmetic circuits.

Therefore, this work contributes a comprehensive overview on circuits for multiplication and addition. These structures were implemented on the gate-level and preserved in the synthesis flow to accurately analyze them with the metrics of PPA. However, the results clearly show that the abstract descriptions via "`*`" and "`+`" do result in more efficient and faster designs. Their synthesis needs a longer runtime but produces much more optimized results because it can simultaneously consider all available gate types. Even the delay-optimized addition of a third operand, which is typically done manually with a 3:2-compressor row, is implemented efficiently by the synthesis tools. Hence, it can be concluded that the effort spend on custom arithmetic circuits is better invested on higher level design challenges.

## 6.2 Outlook

As with almost any research ever performed, open questions remain and call for further investigations. Hence, this thesis concludes by listing two starting points for interesting future work.

The latest iteration of the SerDes' termination design presented in section 3.6 has not yet been optimized via a fitted model as developed in section 3.5, instead more or less a trial-and-error approach was used. As already pointed out before, a monolithic design including the structures on the upper metal layers, i.e. the pads, T-coils, power grid, and metal fill should allow for a much cleaner optimization of the whole structure. Therefore, a lumped model should be designed for it, and fitted to the S-parameters in order to analyze whether all relevant effects are considered and to obtain a simple model for simulation. In case a fast field solver is available, also a brute-force sweep over a large number of layouts could be conducted. In any case, a PCell description of the whole layout would be beneficial. This should be done with XCells as they provide many comfort functions and integrate easily with other Python code.

Regarding the FMA unit, it would be interesting to research the potential performance gains with the full-blown physical retime flow. Although the multiplier tree constructed

by Kaiser was not preserved during the synthesis process so far, it may be beneficial for the tools if it is replaced with an abstract operator. The same analysis has yet to be done for the Posit FMA unit developed by Melzer [85] at the CAG, in order to obtain an actual comparison of the hardware resources for both number formats. Though, it is to be expected that they will be comparable due to their similar internal data paths. Further research could be conducted on an efficient implementation of the Posit accumulator register (Quire), whose integration into a processor pipeline poses an architectural challenge due to its large size.

# List of Abbreviations

**ADS** Advanced Design System

**AFE** Analog Front-End

**ALU** Arithmetic Logic Unit

**AMD** Arithmetic Mean Distance

**AMSD** Arithmetic Mean Square Distance

**ANSI** American National Standards Institute

**BB** Body Bias

**BEOL** Back End of Line

**BER** Bit Error Rate

**BIC** Bayesian Information Criterion

**BOOM** Berkeley Out-of-Order Machine

**CAG** Computer Architecture Group

**CDM** Charged Device Model

**CDR** Clock Data Recovery

**CLA** Ccarry-Lookahead Adder

**CML** Current Mode Logic

**CMOS** Complementary Metal-Oxide-Semiconductor

**CPA** Carry-Propagating Adder

**CPU** Central Processing Unit

**CRA** Carry-Ripple Adder

**CSA** Carry-Save Addition

**CSL** Carry-Select Adder

**CTLE** Continuous-Time Linear Equalizer

**CTS** Clock Tree Synthesis

**DFE** Decision Feedback Equalizer

**DFM** Design for Manufacturing

**DIMM** Dual Inline Memory Module

**DRAM** Dynamic Random Access Memory

**DRC** Design Rule Check

**DUT** Device Under Test

**DUV** Design Under Verification

**EDA** Electronic Design Automation

**ESD** Electrostatic Discharge

**ESDA** Electrostatic Discharge Association

**FBB** Forward Body Bias

**FCLGA** Flip-Chip Land Grid Array

**FCO** Fundamental Carry Operator

**FDSOI** Fully-Depleted Silicon-On-Insulator

**FF** Flip-Flop

**FFE** Feed-Forward Equalizer

**FICDM** Field-Induced Charged Device Model

**FIR** Finite Impulse Response

**FLI** Function Level Interface

**FMA** Fused Multiply-Add

**FPU** Floating-Point Unit

**GMD** Geometric Mean Distance

**GPU** Graphics Processing Unit

**GTL** Gate Transfer Level

**HBM** Human Body Model

**HDL** Hardware Description Language

**HPC** High Performance Computing

**IEC** International Electrotechnical Commission

**IIR** Infinite Impulse Response

**IP** Intellectual Property

**ISA** Instruction Set Architecture

**ISI** Intersymbol Interference

**JEDEC** JEDEC Solid State Technology Association

**LFSR** Linear-Feedback Shift Register

**LSB** Least Significant Bit

**LVS** Layout Versus Schematic

**MAC** Media Access Layer

**MCMC** Markov-Chain Monte-Carlo

**MM** Machine Model

**MSB** Most Significant Bit

**PCB** Printed Circuit Board

**PCIe** Peripheral Component Interconnect Express

**PCS** Physical Coding Sublayer

**PDF** Probability Density Function

**PDK** Process Design Kit

**PIPE** PHY Interface for PCI Express Architectures

**PLL** Phase-Locked Loop

**PMA** Physical Media Attachment Layer

**PPA** Power-Performance-Area

**PPrA** Parallel Prefix Adder

**PRBS** Pseudorandom Binary Sequence

**RF** Radio-Frequency

**RISC** Reduced Instruction Set Computer

**SBR** Single-Bit Response

**SFF** Scan Flip-Flop

**SMT** Simultaneous Multithreading

**SNR** Signal-to-Noise Ratio

**SOC** System-on-Chip

**SOCV** Statistical On-Chip Variation

**SSTL** Stub Series Terminated Logic

**TDR** Time-Domain Reflectometer

**TLP** Transmission Line Pulse

**TPU** Tensor Processing Unit

**UC Berkeley** University of California, Berkeley

**UI** Unit Interval

**UVM** Universal Verification Methodology

**VFTLP** Very Fast Transmission Line Pulse

# List of Figures

# List of Tables

# List of Listings

# References


[1] Wikipedia. (2020). CPU socket, [Online]. Available: https://en.wikipedia.org/wiki/CPU_socket (visited on 01/16/2021).

[2] Intel. (2020). Intel i9-10900K, [Online]. Available: https://ark.intel.com/content/www/de/de/ark/products/199332/intel-core-i9-10900k-processor-20m-cache-up-to-5-30-ghz.html (visited on 01/16/2021).

[3] S. Naffziger, K. Lepak, M. Paraschou, and M. Subramony, "AMD Chiplet Architecture for High-Performance Server and Desktop Products", in *2020 IEEE International Solid- State Circuits Conference - (ISSCC)*, 2020, pp. 44–45. DOI: 10.1109/ISSCC19947.2020.9063103.

[4] A. S. Waterman, "Design of the RISC-V Instruction Set Architecture", PhD thesis, EECS Department, University of California, Berkeley, Jan. 2016. [Online]. Available: http://www2.eecs.berkeley.edu/Pubs/TechRpts/2016/EECS-2016-1.html.

[5] A. Shilov. (Dec. 2018). Western Digital Reveals SweRV RISC-V Core, Cache Coherency over Ethernet Initiative, [Online]. Available: https://www.anandtech.com/show/13678/western-digital-reveals-swerv-risc-v-core-and-omnixtend-coherency-tech (visited on 01/18/2021).

[6] Western Digital Corp. (2021). EH1 SweRV RISC-V CoreTM 1.8 from Western Digital, [Online]. Available: https://github.com/chipsalliance/Cores-SweRV (visited on 01/18/2021).

[7] F. Y. Kaiser, "Design and Verification of a RISC-V Conform, Double-Precision Fused Multiply-Add Unit", Master's thesis, Heidelberg University, 2018.

[8] S. Galal and B. Razavi, "Broadband ESD protection circuits in CMOS technology", *IEEE Journal of Solid-State Circuits*, vol. 38, no. 12, pp. 2334–2340, Dec. 2003, ISSN: 0018-9200. DOI: 10.1109/JSSC.2003.818568.

[9] M. Kossel, C. Menolfi, J. Weiss, P. Buchmann, G. von Bueren, L. Rodoni, T. Morf, T. Toifl, and M. Schmatz, "A T-Coil-Enhanced 8.5 Gb/s High-Swing SST Transmitter in 65 nm Bulk CMOS With −16 dB Return Loss Over 10 GHz

Bandwidth", *IEEE Journal of Solid-State Circuits*, vol. 43, no. 12, pp. 2905–2920, 2008. DOI: 10.1109/JSSC.2008.2006230.

[10] J. Kim, A. Balankutty, R. K. Dokania, A. Elshazly, H. S. Kim, S. Kundu, D. Shi, S. Weaver, K. Yu, and F. O'Mahony, "A 112 Gb/s PAM-4 56 Gb/s NRZ Reconfigurable Transmitter With Three-Tap FFE in 10-nm FinFET", *IEEE Journal of Solid-State Circuits*, vol. 54, no. 1, pp. 29–42, 2019. DOI: 10.1109/JSSC.2018.2874040.

[11] A. Ruehli, "Inductance Calculations in a Complex Integrated Circuit Environment", *IBM Journal of Research and Development*, vol. 9, pp. 470–481, Oct. 1972. DOI: 10.1147/rd.165.0470.

[12] U. Brüning *et al.*, *Skalierbare Energieeffiziente Prozessoren für Intelligente Anwendungen – RISC-V für IoT bis HPC*, Jul. 2019.

[13] T. Bühler, "Pipeline Control, Scoreboard and Commit Strategy of a RISC-V Microprocessor", Master's thesis, Heidelberg University, 2021.

[14] J. Philipp, "Development of an Instruction Fetch Unit with Branch Prediction for a RISC-V Processor", Master's thesis, Heidelberg University, 2021.

[15] F. Kaiser, S. Kosnac, and U. Brüning, "Development of a RISC-V-Conform Fused Multiply-Add Floating-Point Unit", *Supercomputing Frontiers and Innovations*, vol. 6, no. 2, 2019, ISSN: 2313-8734. [Online]. Available: https://superfri.org/superfri/article/view/274.

[16] S. Kosnac and U. Brüning, "Design Flow Automation for On-Chip Inductors", May 2018. [Online]. Available: https://www.cadence.com/en_US/home/cdnlive/emea-2018/proceedings.html.

[17] M. R. Müller, "Digital Centric Multi-Gigabit SerDes Design and Verification", PhD thesis, Heidelberg University, 2018. DOI: https://doi.org/10.11588/heidok.00023972.

[18] M. Thürmer, "Modelling and performance analysis of multigigabit serial interconnects using real number based analog verification methods", PhD thesis, Heidelberg University, 2018. DOI: 10.11588/heidok.00023838.

[19] F. Yuan, A. AL-Taee, A. Ye, and S. Sadr, "Design techniques for decision feedback equalisation of multi-giga-bit-per-second serial data links: A state-of-the-art review", *Circuits, Devices & Systems, IET*, vol. 8, pp. 118–130, Mar. 2014. DOI: 10.1049/iet-cds.2013.0159.

[20] S. A. C. M. Schatral, "Design of Multi-Gigabit Network Interconnect Elements and Protocols for a Data Acquisition System in Radiation Environments", PhD thesis, Heidelberg University, 2018. DOI: https://doi.org/10.11588/heidok.00024533.

[21] S. Kosnac, "Design-Aspects of a Decision Feedback Equalizer in a 28 nm Technology", Master's thesis, Heidelberg University, 2016.

[22] P. Moller, "Electric Fish", *BioScience*, vol. 41, no. 11, pp. 794–796, Dec. 1991, ISSN: 0006-3568. DOI: 10.2307/1311732. eprint: https://academic.oup.com/bioscience/article-pdf/41/11/794/816768/41-11-794.pdf. [Online]. Available: https://doi.org/10.2307/1311732.

[23] (Aug. 2018). Socket TR4 (sTR4) - Packages - AMD, [Online]. Available: https://en.wikichip.org/wiki/amd/packages/socket_tr4 (visited on 04/06/2020).

[24] (2010). Fundamentals of Electrostatic Discharge – Part Six – ESD Standards, [Online]. Available: https://www.esda.org/esd-overview/esd-fundamentals/part-6-esd-standards/ (visited on 04/07/2020).

[25] J. S. Mamaradlo. (Jul. 2016). Simple Yet Effective ESD Testing Methods for Higher Reliability, [Online]. Available: https://www.electronicdesign.com/blogs/guest-blogger/article/21802829/simple-yet-effective-esd-testing-methods-for-higher-reliability (visited on 04/03/2020).

[26] *22FDX® ESD Reference Guide*.

[27] A. Righter and B. Carn, "A Look at the New ANSI/ESDA/JEDEC JS-002 CDM Test Standard", *Analog Dialogue*, vol. 51, no. 4, Oct. 2017.

[28] S. H. Voldman, *ESD: RF Technology and Circuits*. John Wiley & Sons Ltd, 2006.

[29] ——, *ESD: Failure Mechanisms and Models*. John Wiley & Sons Ltd, 2009.

[30] ——, *ESD: Physics and Devices*. John Wiley & Sons Ltd, 2004.

[31] ——, *ESD: Design and Synthesis*. John Wiley & Sons Ltd, 2011.

[32] Seungyoung Ahn, Seungyong Baek, Junho Lee, and Joungho Kim, "Compensation of ESD and device input capacitance by using embedded inductor on PCB substrate for 3 Gbps SerDes applications", in *2004 International Symposium on Electromagnetic Compatibility (IEEE Cat. No.04CH37559)*, vol. 2, 2004, 499–504 vol.2. DOI: 10.1109/ISEMC.2004.1349847.

[33] W. Soldner, M. Kim, M. Streibl, H. Gossner, T. H. Lee, and D. Schmitt-Landsiedel, "A 10GHz Broadband Amplifier with Bootstrapped 2kV ESD Protection", in *2007 IEEE International Solid-State Circuits Conference. Digest of Technical Papers*, 2007, pp. 550–551. DOI: 10.1109/ISSCC.2007.373538.

[34] B. Kleveland, T. J. Maloney, I. Morgan, L. Madden, T. H. Lee, and S. S. Wong, "Distributed ESD protection for high-speed integrated circuits", *IEEE Electron Device Letters*, vol. 21, no. 8, pp. 390–392, Aug. 2000, ISSN: 1558-0563. DOI: 10.1109/55.852960.

[35] D. M. Pozar, *Microwave Engineering*. John Wiley & Sons, 2009.

[36] S. Kim, K. Shinae, J. Goeun, K.-W. Kwon, and J.-H. Chun, "Design of a Reliable Broadband I/O Employing T-coil", *JSTS:Journal of Semiconductor Technology and Science*, vol. 9, Dec. 2009. DOI: 10.5573/JSTS.2009.9.4.198.

[37] S. C. D. Roy, "Comments on "Analysis of the Bridged T-coil Circuit Using the Extra-Element Theorem", *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 54, no. 8, pp. 673–674, Aug. 2007, ISSN: 1549-7747. DOI: 10.1109/TCSII.2007.899834.

[38] J. Paramesh and D. J. Allstot, "Analysis of the Bridged T-Coil Circuit Using the Extra-Element Theorem", *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 53, no. 12, pp. 1408–1412, 2006. DOI: 10.1109/TCSII.2006.885971.

[39] T. Markus, "Circuit Design Automation for High Speed Interconnects in Advanced Nodes", PhD thesis, Heidelberg University, 2021.

[40] S. S. Mohan, "The Design, Modeling and Optimization of On-chip Inductor and Transformer Circuits", PhD thesis, Stanford University, 1999.

[41] S. S. Mohan, M. del Mar Hershenson, S. P. Boyd, and T. H. Lee, "Simple accurate expressions for planar spiral inductances", *IEEE Journal of Solid-State Circuits*, vol. 34, no. 10, pp. 1419–1424, Oct. 1999, ISSN: 1558-173X. DOI: 10.1109/4.792620.

[42] M. Tiner. (2018). What is Open about Si2 OpenAccess?, [Online]. Available: https://si2.org/2018/04/13/what-is-open-about-openaccess/ (visited on 09/11/2020).

[43] M. Guiney and E. Leavitt, "An Introduction to OpenAccess: An Open Source Data Model and API for IC Design", in *Proceedings of the 2006 Asia and South Pacific Design Automation Conference*, ser. ASP-DAC '06, Yokohama, Japan: IEEE Press, 2006, pp. 434–436, ISBN: 0780394518. DOI: 10.1145/1118299.1118405. [Online]. Available: https://doi.org/10.1145/1118299.1118405.

[44] S. Alassi and B. Winter, "PyCells for an Open Semiconductor Industry", *CoRR*, vol. abs/1607.00859, 2016. arXiv: 1607.00859. [Online]. Available: http://arxiv.org/abs/1607.00859.

[45] Synopsys, *Python API Reference Manual*. Jun. 2017.

[46] ——, *Santana™ Technology File Reference Manual*. Jun. 2017.

[47] T. Markus and N. Buwen. (). Python-Skill Bridge, [Online]. Available: https://github.com/unihd-cag/skillbridge (visited on 09/22/2020).

[48] J. C. Maxwell, *A Treatise on Electricity and Magnetism*, ser. A Treatise on Electricity and Magnetism. Clarendon Press, 1873, vol. 1. [Online]. Available: https://books.google.de/books?id=1lwPVLBLj1oC.

[49] L. Chua, "Memristor - The Missing Circuit Element", *IEEE Transactions on Circuit Theory*, vol. 18, no. 5, pp. 507–519, Sep. 1971, ISSN: 2374-9555. DOI: 10.1109/TCT.1971.1083337.

[50] E. B. Rosa, "The Self and Mutual Inductances of Linear Conductors", in, ser. Bulletin of the Bureau of Standards. U.S. Department of Commerce and Labor, Bureau of Standards, 1908, vol. 4, pp. 301–344.

[51] R. Weaver, *Geometric Mean Distance - Its Derivation and Application in Inductance Calculations*, Mar. 2016. [Online]. Available: http://electronbunker.ca/DLpublic/GMD.pdf.

[52] C. Hoer and C. Love, "Exact Inductance Equations for Rectangular Conductors With Applications to More Complicated Geometries", *Journal of Research of the National Bureau of Standards – C. Engineering and Instrumentation*, pp. 127–137, 1965.

[53] J. C. Maxwell, *A Treatise on Electricity and Magnetism*, ser. A Treatise on Electricity and Magnetism. Clarendon Press, 1873, vol. 2. [Online]. Available: https://books.google.de/books?id=OcS4BEiMNwoC.

[54] B. Burgstaller and F. Pillichshammer, "The Average Distance Between Two Points", *Bulletin of the Australian Mathematical Society*, vol. 80, no. 3, pp. 353–359, 2009. DOI: 10.1017/S0004972709000707.

[55] H. A. Aebischer and B. Aebischer, "Improved Formulae for the Inductance of Straight Wires", *Advanced Electromagnetics*, vol. 3, Sep. 2014. DOI: 10.7716/aem.v3i1.254.

[56] H. M. Greenhouse, "Design of Planar Rectangular Microelectronic Inductors", *IEEE Transactions on Parts, Hybrids, and Packaging*, vol. 10, no. 2, pp. 101–109, 1974. DOI: 10.1109/TPHP.1974.1134841.

[57] J. C. Maxwell, "On the Geometrical Mean Distance of Two Figures on a Plane", *Transactions of the Royal Society of Edinburgh*, vol. 26, no. 4, pp. 729–733, 1872. DOI: 10.1017/S008045680002559X.

[58] J. R. Long and M. A. Copeland, "The modeling, characterization, and design of monolithic inductors for silicon RF IC's", *IEEE Journal of Solid-State Circuits*, vol. 32, no. 3, pp. 357–369, 1997. DOI: 10.1109/4.557634.

[59] C. P. Yue, C. Ryu, J. Lau, T. H. Lee, and S. S. Wong, "A physical model for planar spiral inductors on silicon", in *International Electron Devices Meeting. Technical Digest*, 1996, pp. 155–158. DOI: 10.1109/IEDM.1996.553144.

[60] C. P. Yue and S. S. Wong, "Physical modeling of spiral inductors on silicon", *IEEE Transactions on Electron Devices*, vol. 47, no. 3, pp. 560–568, 2000. DOI: 10.1109/16.824729.

[61] L. Nan, K. Mouthaan, Y. Xiong, J. Shi, S. C. Rustagi, and B. Ooi, "Experimental Characterization of the Effect of Metal Dummy Fills on Spiral Inductors", in *2007 IEEE Radio Frequency Integrated Circuits (RFIC) Symposium*, 2007, pp. 307–310. DOI: 10.1109/RFIC.2007.380889.

[62] V. N. Rao Vanukuru, "Impact of Floating Dummy Fill On Q Characteristics of Spiral Inductors", in *2018 IEEE MTT-S International Microwave and RF Conference (IMaRC)*, 2018. DOI: 10.1109/IMaRC.2018.8877365.

[63] G. S. Smith, "A simple derivation for the skin effect in a round wire", *European Journal of Physics*, vol. 35, no. 2, p. 025 002, Jan. 2014. DOI: 10.1088/0143-0807/35/2/025002. [Online]. Available: https://doi.org/10.1088%2F0143-0807%2F35%2F2%2F025002.

[64] D. Gerling, "Approximate analytical calculation of the skin effect in rectangular wires", in *2009 International Conference on Electrical Machines and Systems*, 2009, pp. 1–6. DOI: 10.1109/ICEMS.2009.5382786.

[65] P. McLellan. (Jul. 2020). Clarity, Sigrity, EMX, and AWR: So Many EM Solvers to Choose From…, [Online]. Available: https://community.cadence.com/cadence_blogs_8/b/breakfast-bytes/posts/em-solvers (visited on 03/12/2021).

[66] ——, (Feb. 2020). Designing Radios: Integrand, [Online]. Available: https://community.cadence.com/cadence_blogs_8/b/breakfast-bytes/posts/integrand (visited on 03/12/2021).

[67] Integrand Software Inc., *Foundry relationships*. [Online]. Available: https://integrandsoftware.com/foundries.php (visited on 03/13/2021).

[68] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S. J. van der Walt, M. Brett, J. Wilson, K. J. Millman, N. Mayorov, A. R. J. Nelson, E. Jones, R. Kern, E. Larson, C. J. Carey, İ. Polat, Y. Feng, E. W. Moore, J. VanderPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E. A. Quintero, C. R. Harris, A. M. Archibald, A. H. Ribeiro, F. Pedregosa, P. van Mulbregt, and SciPy 1.0 Contributors, "SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python", *Nature Methods*, vol. 17, pp. 261–272, 2020. DOI: 10.1038/s41592-019-0686-2.

[69] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller, "Equation of State Calculations by Fast Computing Machines", *J. Chem. Phys.*, vol. 21, no. 6, pp. 1087–1092, Jun. 1953. DOI: 10.1063/1.1699114.

[70] W. K. Hastings, "Monte Carlo sampling methods using Markov chains and their applications", *Biometrika*, vol. 57, no. 1, pp. 97–109, Apr. 1970, ISSN: 0006-3444. DOI: 10.1093/biomet/57.1.97. eprint: https://academic.oup.com/biomet/article-pdf/57/1/97/23940249/57-1-97.pdf. [Online]. Available: https://doi.org/10.1093/biomet/57.1.97.

[71] D. Foreman-Mackey, D. W. Hogg, D. Lang, and J. Goodman, "emcee: The MCMC Hammer", *Publications of the Astronomical Society of the Pacific*, vol. 125, no. 925, p. 306, Mar. 2013. DOI: 10.1086/670067. arXiv: 1202.3665 [astro-ph.IM].

[72] J. Goodman and J. Weare, "Ensemble samplers with affine invariance", *Commun. Appl. Math. Comput. Sci.*, vol. 5, no. 1, pp. 65–80, 2010. DOI: 10.2140/camcos.2010.5.65. [Online]. Available: https://doi.org/10.2140/camcos.2010.5.65.

[73] D. Foreman-Mackey, "corner.py: Scatterplot matrices in Python", *The Journal of Open Source Software*, vol. 1, no. 2, p. 24, Jun. 2016. DOI: 10.21105/joss.00024. [Online]. Available: https://doi.org/10.21105/joss.00024.

[74] H. Meuer, E. Strohmaier, J. Dongarra, H. Simon, and M. Meuer. (2018). Top 500 list, [Online]. Available: https://www.top500.org/lists/2018/11/ (visited on 06/21/2019).

[75] J. Dongarra, J. Bunch, C. Moler, and G. Stewart. (1979). LINPACK, [Online]. Available: http://www.netlib.org/linpack/ (visited on 11/14/2019).

[76] J. Hauser. (2018). Berkeley TestFloat, [Online]. Available: http://www.jhauser.us/arithmetic/TestFloat.html (visited on 02/16/2021).

[77] J. Hauser *et al.* (2014). Berkeley TestFloat Release 3e, [Online]. Available: https://github.com/ucb-bar/berkeley-testfloat-3 (visited on 02/16/2021).

[78] "IEEE Standard for Floating-Point Arithmetic", *IEEE Std 754-2019 (Revision of IEEE 754-2008)*, 2019. DOI: 10.1109/IEEESTD.2019.8766229.

[79] R. Morris, "Tapered Floating Point: A New Floating-Point Representation", *IEEE Transactions on Computers*, vol. C-20, no. 12, pp. 1578–1579, 1971. DOI: 10.1109/T-C.1971.223174.

[80] J. L. Gustafson and I. Yonemoto, "Beating Floating Point at its Own Game: Posit Arithmetic", *Supercomputing Frontiers and Innovations*, vol. 4, no. 2, 2017, ISSN: 2313-8734. [Online]. Available: https://superfri.org/superfri/article/view/137.

[81] J. L. Gustafson, *Posit Arithmetic*. Oct. 2017. [Online]. Available: https://posithub.org/docs/Posits4.nb.

[82] ——, *The End of Error: Unum Computing*. Feb. 2015, ISBN: 1482239868. DOI: 10.1201/9781315161532.

[83] F. Dinechin, L. Forget, J.-M. Muller, and Y. Uguen, "Posits: the good, the bad and the ugly", Mar. 2019, pp. 1–10. DOI: 10.1145/3316279.3316285.

[84] P. W. Group, *Posit Standard Documentation Release 3.2-draft*. Jun. 2018. [Online]. Available: https://posithub.org/docs/posit_standard.pdf.

[85] C. Melzer, "Design and Verification of a Parameterizable Posit Unit with Fused Multiply-Add and Quire Support", Master's thesis, Heidelberg University, 2020.

[86] P. Kharya. (May 2020). TensorFloat-32 in the A100 GPU Accelerates AI Training, HPC up to 20x, [Online]. Available: https://blogs.nvidia.com/blog/2020/05/14/tensorfloat-32-precision-format/ (visited on 02/14/2021).

[87] S. Wang and P. Kanwar. (Aug. 2019). TensorFloat-32 in the A100 GPU Accelerates AI Training, HPC up to 20x, [Online]. Available: https://cloud.google.com/blog/products/ai-machine-learning/bfloat16-the-secret-to-high-performance-on-cloud-tpus (visited on 02/14/2021).

[88] S. Hutchins and E. Swartzlander, "A Bfloat16 Fused Multiplier-Adder", in *2020 11th IEEE Annual Ubiquitous Computing, Electronics Mobile Communication Conference (UEMCON)*, 2020, pp. 0052–0055. DOI: 10.1109/UEMCON51285.2020.9298120.

[89] SiFive. (2021). HiFive, [Online]. Available: https://www.sifive.com/boards (visited on 02/14/2021).

[90] "IEEE Standard for Floating-Point Arithmetic", *IEEE Std 754-2008*, 2008. DOI: 10.1109/IEEESTD.2008.4610935.

[91] M. Gautschi *et al.* (2017). PULP-Platform - FPU, [Online]. Available: https://github.com/pulp-platform/fpu (visited on 06/21/2019).

[92] S. Mach *et al.* (2021). PULP-Platform - FPU, [Online]. Available: https://github.com/pulp-platform/fpnew (visited on 02/15/2021).

[93] J. Hauser. (2019). Berkeley HardFloat, [Online]. Available: http://www.jhauser.us/arithmetic/HardFloat.html (visited on 02/16/2021).

[94] J. Hauser *et al.* (2012). Berkeley Hardware Floating-Point Units, [Online]. Available: https://github.com/ucb-bar/berkeley-hardfloat (visited on 02/16/2021).

[95] K. Asanović, R. Avizienis, J. Bachrach, S. Beamer, D. Biancolin, C. Celio, H. Cook, D. Dabbelt, J. Hauser, A. Izraelevitz, S. Karandikar, B. Keller, D. Kim, J. Koenig, Y. Lee, E. Love, M. Maas, A. Magyar, H. Mao, M. Moreto, A. Ou, D. A. Patterson, B. Richards, C. Schmidt, S. Twigg, H. Vo, and A. Waterman, "The Rocket Chip Generator", EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2016-17, Apr. 2016. [Online]. Available: http://www2.eecs.berkeley.edu/Pubs/TechRpts/2016/EECS-2016-17.html.

[96] C. Celio, D. A. Patterson, and K. Asanović, "The Berkeley Out-of-Order Machine (BOOM): An Industry-Competitive, Synthesizable, Parameterized RISC-V Processor", EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2015-167, Jun. 2015. [Online]. Available: http://www2.eecs.berkeley.edu/Pubs/TechRpts/2015/EECS-2015-167.html.

[97] C. Celio, P.-F. Chiu, B. Nikolic, D. Patterson, and K. Asanović. (2017). BOOM v2 an open-source out-of-order RISC-V core, [Online]. Available: https://content.riscv.org/wp-content/uploads/2017/12/Wed0936-BOOM-v2-An-Open-Source-Out-of-Order-RISC-V-Core-Celio.pdf (visited on 06/21/2019).

[98] J. Bachrach, H. Vo, B. Richards, Y. Lee, A. Waterman, R. Avižienis, J. Wawrzynek, and K. Asanović, "Chisel: Constructing hardware in a Scala embedded language", in *The 49th Annual Design Automation Conference 2012, DAC 2012, 3-7 June 2012, San Francisco, California, USA*, 2012, pp. 1212–1221. DOI: 10.1145/2228360.2228584.

[99] L. Henger, "Design and Verification of a RISC-V Conform, Double-Precision Division and Square Root Unit", Master's thesis, Heidelberg University, 2020.

[100] J.-M. Muller, N. Brisebarre, F. de Dinechin, C.-P. Jeannerod, V. Lefèvre, G. Melquiond, N. Revol, D. Stehlé, and S. Torres, *Handbook of Floating-Point Arithmetic*, 1st. Birkhäuser Basel, 2012, ISBN: 978-0-8176-4705-6. DOI: 10.1007/978-0-8176-4705-6.

[101] E. M. Clarke, "The Birth of Model Checking", in *25 Years of Model Checking: History, Achievements, Perspectives*, Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 1–26, ISBN: 978-3-540-69850-0. DOI: 10.1007/978-3-540-69850-0_1. [Online]. Available: https://doi.org/10.1007/97835406985001.

[102] J. Harrison, "Formal Verification of IA-64 Division Algorithms", in *Theorem Proving in Higher Order Logics*, Berlin, Heidelberg: Springer Berlin Heidelberg, 2000, pp. 233–251, ISBN: 978-3-540-44659-0. DOI: 10.1007/354044659115.

[103] E. M. Clarke, S. M. German, and X. Zhao, "Verifying the SRT division algorithm using theorem proving techniques", in *Computer Aided Verification*, Berlin, Heidelberg: Springer Berlin Heidelberg, 1996, pp. 111–122, ISBN: 978-3-540-68599-9. DOI: 10.1007/354061474562.

[104] S. Beyer, C. Jacobi, D. Kröning, D. Leinenbach, and W. J. Paul, "Putting it all together – Formal verification of the VAMP", *International Journal on Software Tools for Technology Transfer*, vol. 8, no. 4, pp. 411–430, 2006, ISSN: 1433-2787. DOI: 10.1007/s1000900602046. [Online]. Available: https://doi.org/10.1007/s10009-006-0204-6.

[105] M. Aharoni, S. Asaf, L. Fournier, A. Koifman, and R. Nagel, "FPgen - a test generation framework for datapath floating-point verification", in *Eighth IEEE International High-Level Design Validation and Test Workshop 2003, 12-14 November 2003, San Francisco, California, USA*, 2003, pp. 17–22. DOI: 10.1109/HLDVT.2003.1252469.

[106] *Intel C++ Intrinsics Reference*, 2007. [Online]. Available: http://www.info.univ-angers.fr/pub/richer/ens/l3info/ao/intel_intrinsics.pdf (visited on 06/21/2019).

[107] J. Hauser. (2018). Berkeley SoftFloat, [Online]. Available: http://www.jhauser.us/arithmetic/SoftFloat.html (visited on 02/16/2021).

[108] J. Hauser *et al.* (2014). Berkeley SoftFloat Release 3e, [Online]. Available: https://github.com/ucb-bar/berkeley-softfloat-3 (visited on 02/16/2021).

[109] S. Galal and M. Horowitz, "Energy-Efficient Floating-Point Unit Design", *IEEE Transactions on Computers*, vol. 60, no. 7, pp. 913–922, 2011, ISSN: 0018-9340. DOI: 10.1109/TC.2010.121.

[110] C. S. Wallace, "A Suggestion for a Fast Multiplier", *IEEE Transactions on Electronic Computers*, vol. EC-13, no. 1, pp. 14–17, 1964. DOI: 10.1109/PGEC.1964.263830.

[111] L. Dadda, "Some schemes for parallel multipliers", *Alta Frequenza*, no. 34, pp. 349–356, 1965.

[112] W. Townsend, E. Jr, and J. Abraham, "A comparison of Dadda and Wallace multiplier delays", *Proceedings of SPIE - The International Society for Optical Engineering*, vol. 5205, Dec. 2003. DOI: 10.1117/12.507012.

[113] *Hardware algorithms for arithmetic modules*. [Online]. Available: http://www.aoki.ecei.tohoku.ac.jp/arith/mg/algorithm.html (visited on 07/02/2020).

[114] T. W. Lynch, "Binary Adders", Master's thesis, University of Texas at Austin, 1996.

[115] J. Sklansky, “Conditional-Sum Addition Logic”, *IRE Transactions on Electronic Computers*, vol. EC-9, no. 2, pp. 226–231, 1960.

[116] D. Harris, “A Taxonomy of Parallel Prefix Networks”, in *The Thrity-Seventh Asilomar Conference on Signals, Systems Computers, 2003*, vol. 2, 2003, pp. 2213–2217.

[117] R. P. Brent and H. T. Kung, “A Regular Layout for Parallel Adders”, *IEEE Transactions on Computers*, vol. C-31, no. 3, pp. 260–264, 1982.

[118] P. M. Kogge and H. S. Stone, “A Parallel Algorithm for the Efficient Solution of a General Class of Recurrence Equations”, *IEEE Transactions on Computers*, vol. C-22, no. 8, pp. 786–793, 1973.

[119] O. L. MacSorley, “High-Speed Arithmetic in Binary Computers”, *Proceedings of the IRE*, vol. 49, no. 1, pp. 67–91, 1961.

[120] T. Rhyne, “Limitations on Carry Lookahead Networks”, *IEEE Transactions on Computers*, vol. C-33, no. 4, pp. 373–374, 1984. DOI: 10.1109/TC.1984.1676445.

[121] H. W. Lang. (2018). Carry-Lookahead-Addierer, [Online]. Available: https://www.inf.hs-flensburg.de/lang/algorithmen/arithmetik/cla.htm (visited on 06/25/2020).

[122] YosysHQ. (2021). SymbiYosys (sby) Documentation, [Online]. Available: https://symbiyosys.readthedocs.io/en/latest/ (visited on 02/22/2021).

[123] L.-R. Zheng and H. Tenhunen, “Wires as Interconnects”, in. Jan. 2005, pp. 25–54. DOI: 10.1007/1-4020-7836-6_2.

[124] R. Goering. (Nov. 2013). Signoff Summit: An Update on OCV, AOCV, SOCV, and Statistical Timing, [Online]. Available: https://community.cadence.com/cadence_blogs_8/b/ii/posts/signoff-summit-an-update-on-ocv-aocv-socv-and-statistical-timing (visited on 02/26/2021).

[125] Y. Choi, “Parallel Prefix Adder Design”, PhD thesis, University of Texas at Austin, 2004.