Rapid Digital Architecture Design of Computationally Complex Algorithms

A Dissertation Accepted by the Faculty of
Physics and Electrical Engineering of the University of Bremen
in Partial Fulfilment of the Requirements for the Degree of

DOKTOR-INGENIEUR (Dr.-Ing.)

By

Benjamin Andreas Knoop

Referee:	Prof. Dr.-Ing. Steffen Paul
Co-Referee:	Prof. Dr. Sc. Techn. Andreas Burg
3rd Examiner:	Prof. Dr.-Ing. Alberto García-Ortiz
4th Examiner:	Prof. Dr. Angelika Bunse-Gerstner

Date of Dissertation:	June 15, 2018
Date of Colloquium:	February 7, 2019

Summary

The growing availability of computing power allows for the implementation of algorithms of increasing computational complexity. While this is definitely true for general purpose computers, this also applies to the very-large-scale integration (VLSI) design targeting Field-Programmable Gate Arrays (FPGAs). Yet, traditional digital design techniques hardly keep up with the pace of technological advancements and the rising abundance of programmable circuitry found on such devices.

To this end, this work presents the novel Rapid Data Type-Agnostic Digital Design Methodology (RDAM) to elevate the design perspective of digital design engineers away from the register-transfer level to a higher level of abstraction—the algorithmic level. The proposed methodology is founded on the enormous capabilities of High-Level Synthesis (HLS), which basically is a synthesis step to compile concurrently operating VLSI architectures from sequentially coded algorithms specified, e.g., in C++. By consequently working with data type-agnostic source codes, the RDAM brings significant simplifications to the fixed-point conversion of algorithms and the creation of complex-valued arithmetic.

Throughout this dissertation, signal processing applications from the field of Compressed Sensing (CS) will illustrate the efficacy of the RDAM. Algorithmic modifications and improvements are described to incorporate the notion of sparse-coded signals and their recovery in the context of multi-user wireless communications in a wireless sensor network. For instance, a complex-valued digital architecture for the Orthogonal Matching Pursuit (OMP) algorithm with rank-1 updating has successfully been implemented and tested, which can be utilised for the combined multi-user wireless channel estimation and activity detection of sporadically transmitting sensor nodes. Further, sparsity-regularised tree search algorithms will be examined as well as the multi-user data frame detection with a priori unknown user activities based on specifically designed user codes derived from Zadoff–Chu sequences.

Zusammenfassung

Die zunehmende Verfügbarkeit von Rechenleistung erlaubt die Implementierung von immer rechenintensiveren Algorithmen. Dies trifft sicherlich auf Rechner im Allgemeinen zu, doch ebenso auch auf den Very-Large-Scale-Integration (VLSI) Entwurf für Field-Programmable Gate Arrays (FPGAs). Jedoch können traditionelle Entwurfsmethoden kaum Schritt halten mit der Geschwindigkeit technologischen Fortschritts und der weiter wachsenden Verfügbarkeit von programmierbarer Logik, die solche Chips bieten.

Diese Arbeit präsentiert als Antwort darauf die Rapid Data Type-Agnostic Digital Design Methodology (RDAM), um den Digitalentwurf von der Register-Transfer-Ebene weg auf ein höheres Abstraktionsniveau anzuheben – der algorithmischen Ebene. Die vorgeschlagene Methodik ist auf der Leistungsfähigkeit von High-Level Synthesis (HLS) gegründet, was prinzipiell einen Syntheseschritt darstellt, der aus einem sequentiell programmierten Algorithmus, z.B. mit C++, eine nebenläufige VLSI-Architektur kompiliert. Aufgrund konsequent datentyp-agnostischer Quellcodes führt die RDAM zu deutlichen Vereinfachungen bei der Festkommaarithmetik als auch bei komplexwertigen Berechnungen.

Signalverarbeitungsapplikationen aus dem Bereich des Compressed Sensings (CS) dienen als Beispiel um die Effektivität der RDAM zu demonstrieren. Algorithmische Anpassungen und Verbesserungen binden Ideen zur Mehrnutzer-Detektion dünn besetzter Signale in drahtlosen Sensornetzwerken ein. Beispielsweise wurde eine komplexwertige Architektur für Orthogonal Matching Pursuit (OMP) mit Rank-1 Updates erfolgreich implementiert und getestet, die zur gemeinsamen Schätzung von Nutzerkanälen und der Aktivität sporadisch sendender Sensorknoten herangezogen werden kann. Außerdem werden Baumsuchverfahren untersucht, sowie eine Mehrnutzer-Rahmendetektion mit a priori unbekannten Nutzeraktivitäten basierend auf speziell entworfenen Zadoff-Chu-Sequenzen.

Acknowledgments

There are many persons who influenced and supported this work, whose contributions I would like to acknowledge gratefully.

To begin with, I would like to thank my doctorate thesis supervisor Prof. Dr.-Ing. Steffen Paul (Institute of Electrodynamics and Microelectronics, University of Bremen). His interest in my person and abilities, as early as during my undergraduate studies, set me on the way towards this dissertation, and he generously provided continuous funding for my employment as a research assistant and all the journeys to national and international conferences. What is more, he consistently left enough room to follow one's own scientific creativity.

I would also like to thank Prof. Dr. Sc. Techn. Andreas Burg (École Polytechnique Fédérale de Lausanne) for his outright willingness to serve as an external reviewer of this thesis. With or without knowing him, his scientific work influenced mine beginning with my Diploma thesis, supervised by Dr.-Ing. Till Wiegand. His keen interest in optimised hardware architectures for algorithms such as Sphere Decoding, K-Best detection or Orthogonal Matching Pursuit will find more than one parallel on the pages to come.

My thanks also go to my long-standing office colleague, the soon-to-become Dr.-Ing. Sebastian Schmale. I enjoyed our time together, experiencing the ups and downs of being doctorate students, and I happily like to look back on our journeys to scientific conferences and the frequent discussions we have had. We certainly share our passion for signal processing algorithms.

Similar things can be said with regard to our postdoc, Dr.-Ing. Jochen Rust. He supported me with his digital design expertise, e.g. relating to the Logarithmic Number System, and I feel that a deep professional relationship has grown over time and during countless discussions accompanied by not only one mug of coffee.

Furthermore, I would like to thank Dr.-Ing. Carsten Bockelmann and Dr.-Ing. Fabian Monsees of the Department of Communications Engineering for the many helpful insights and joint project work.

I would also like to pay tribute to the contributions of all the students listed at the end of this dissertation, who investigated and implemented things I could not find time for.

However, special thanks must be awarded to my family, wife and children, for supporting me throughout this project and especially during the final stages of writing this document.

Contents

Chapter 1

Introduction and Motivation

Technological advancements in wireless communications in conjunction with the increased availability of computing power lead to profound changes in how things and human beings communicate with each other. While mobile radio networks and mobile phones can already be found almost everywhere on this globe, the number of communicating machines is on the rise. This development is best described by concepts like the Internet of Things (IoT) or Industry 4.0 [AlF+15; DH14].

However, the anticipated artificially intelligent, autonomously acting and self-organising applications exhibit a principally different communication behaviour, called Machine-Type Communications (MTC), which requires to re-think existing wireless communication schemes and networks.

Currently, efforts are made, therefore, to standardise the future fifth generation of digital mobile networks (5G), also known as IMT-2020 by the International Telecommunication Union (ITU) [BB17; 3GPP18]. It shall encompass and support at least three different usage scenarios: massive MTC, ultra-reliable low-latency communications and classical mobile broadband services for human users. A couple of disruptive technologies are being proposed to achieve this goal, e.g. millimetre waves together with small cells, or very large multi-antenna multiple-input multiple-output (MIMO) systems (also called Massive MIMO) together with beam forming [And+14; Boc+14]. This will eventually lead to an improved performance by an order of magnitude for link latency, single user data throughput or the number of devices per area (see Fig. 1.1).

The enhanced bandwidth efficiency and flexibility to accommodate various types of communication services naturally and inevitably result in higher design complexity with regard to the signal processing algorithms

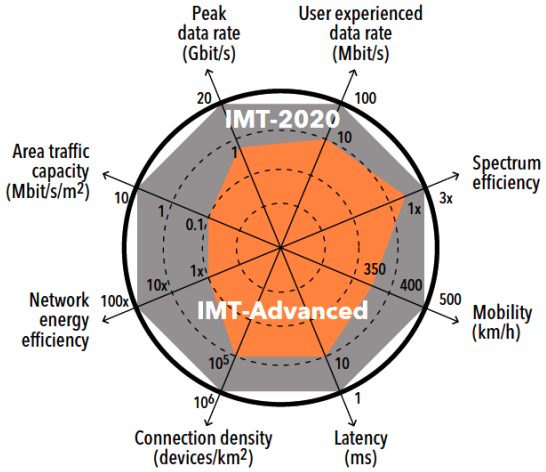


Figure 1.1: The expected enhancement of key capabilities of 5G mobile networks (IMT-2020) in comparison to LTE-Advanced (IMT-Advanced) [BB17].

and digital hardware resources needed to enable such high-tech mobile networks in the first place—the complexity increase from one generation to another is a factor of about ten as [HJ09] observes.

Another area completely unrelated to wireless communications where computationally complex algorithms are typically to be encountered are data centres for cloud services offering machine learning with deep neural networks or big data analytics [HM17]. Recently, Field-Programmable Gate Arrays (FPGAs) have become employed in vast numbers in such computing centres as a substantial workforce to speed up computations.

With potent digital circuitry on the one hand, and complex digital signal processing algorithms and applications on the other hand, two worlds come together here.

FPGAs are digital logic devices which operate inherently parallel and concurrently; their programming follows the paradigm of digital hardware design, traditionally with specialised hardware description languages (HDLs), namely VHDL and Verilog. In contrast, software to implement algorithms is usually written with high-level languages (HLLs),

e.g. C/C++, which are sequential in nature. The OpenCL framework, for instance, tries to exploit parallelisms and target heterogeneous computing systems such as graphics processing units (GPUs) or even FPGAs in addition to central processing units (CPUs) [Khr17]. Major FPGA vendors hence are eager to deploy tools for an easy transition from OpenCL to HDL code.

In fact, there is quite a powerful concept involved between the two antipodes of "software" and "hardware": High-Level Synthesis (HLS). HLS denotes the automatic generation or compilation of HDL code from an HLL description. All commercially successful and functionally mature tools support the family of C languages, i.e. C/C++ with support for SystemC, to be precise.

This thesis, therefore, investigates how the capabilities of HLS can be utilised for the creation of sophisticated digital architectures for complex signal processing applications. It consists of two parts, as indicated by its title "Rapid Digital Architecture Design of Computationally Complex Algorithms":

- Firstly, a novel design methodology, called the Rapid Data Type-Agnostic Digital Design Methodology (RDAM), is proposed. It accelerates digital design time and achieves great simplifications with regard to the fixed-point conversion of algorithms and the implementation of complex-valued operations. For instance, one baseline fixed-point format can be applied throughout a digital architecture, which is a rather astonishing result.

- And secondly, a wireless communications scenario supplies examples for computationally complex algorithms. It touches the aforementioned topics such as IoT, Industry 4.0 or 5G. The assumed communication system is a wireless sensor network (WSN) with sporadically active sensor devices (or "users"), which exhibit MTC behaviour. Ideas from the Compressed Sensing (CS) theory are incorporated for a cross-layer optimisation to avoid signalling overhead concerning user activities on higher layers of the protocol stack. This in turn, however, requires the execution of rather complex algorithms for signal recovery.

1.1 Contributions to the State of the Art

Both parts of this thesis contain original contributions to the state of the art in their respective fields of study.

1.1.1 Digital Architecture Design

The aforementioned observable increase in computational complexity challenges traditional digital design flows. The technological evolution and transistor integration densities scale quite well to accommodate this growth in complexity, but the design capabilities of hardware engineers do not, which is known as the productivity design gap [MS13].

One answer to this challenge certainly is HLS, which accelerates digital architecture design by allowing hardware engineers to work on a higher level of abstraction [Cou+09]. Instead of describing digital logic on the behavioural register-transfer level (RTL), HLS elevates hardware design to the algorithmic layer of the Gajski–Kuhn chart [GK83]. HLS is especially well-suited for FPGAs as target devices, because they allow for fast design and test cycles. Its capability to create even complicated architectures has already been demonstrated [Con+11]. And to cite another example, an application-specific integrated circuit (ASIC) developed with HLS for the pre-coding and detection in a 128×8 Massive MIMO system has been reported on in [Pra17].

This thesis introduces a novel design technique called the Rapid Data Type-Agnostic Digital Design Methodology (RDAM), which drives design abstraction even further by requiring data type-agnostic HLS design entry source codes written in C++ to leverage the polymorphism of that programming language. The concept of data type agnosticism (DTA) itself is not new and has been applied by the Mathworks Fixed-Point Designer for Matlab to switch between normal computations and fixed-point operation, with fixed-point objects being substituted for the default double-precision floating-point data type; no code modifications are needed. And following a different line of reasoning, namely to optimise synthesis results, the HLS tutorial of Xilinx mentions that standard C types can be updated in favour of specialised fixed-point types, requiring some code modifications [Xil-UG871].

These elementary ideas are brought together by the RDAM which produces a synergistic effect. Firstly, it eases fixed-point design and the

conversion of algorithms to fixed-point arithmetic, especially with regard to the proper sizing of the finite word lengths. Secondly, it also applies to complex-valued algorithms and can map these to digital logic without any additional manual design steps.

The RDAM has been described for the first time in [Kno+16c], where Mathworks Matlab and Xilinx' HLS tool are additionally linked with each other to allow for semi-automatic parametric sweeps to explore the design space. That work also presents, to the best knowledge of the author, the first digital design of the Approximate Conjugate Gradient Pursuit (ACGP) algorithm [BDR12].

The RDAM was also applied to the well-known Orthogonal Matching Pursuit (OMP), [TG07]. The resulting digital design is compared with other architectures taken from the literature [Kno+16a]. Furthermore, rank updates to the QR matrix decomposition (QRD) as well as the Moore–Penrose pseudoinverse were proposed to solve the least squares (LS) optimisation problem within OMP. To the best knowledge of the author, [Kno+16a] presents the first complex-valued digital architecture for OMP as an important benefit of the RDAM.

1.1.2 Compressed Sensing in Wireless Communications

Following up on the seminal work of Candès, Donoho and others, Compressed Sensing (CS) gained much attention in the signal processing community [CT05a; Don06]. Basically, CS allows for the sub-sampling of a signal below the classical Nyquist–Shannon sampling rate, if and only if the signal of interest is sparse, either directly or in some basis transform domain, e.g. the frequency domain. Hence, it was proposed to incorporate the CS framework into communication applications where sparsity is present, e.g. to exploit sporadic user activities in WSNs to facilitate a joint activity and data estimation [TLL09].

In this work, three closely related tree search algorithms are modified according to the CS sparsity constraint and explored with regard to their detection performance. Results for optimal sparsity-aware Sphere Decoding (SD) have been reported in [KWP12] and a digital design thereof has been published in [Kno+17]. The observed run time behaviour of the constrained algorithm led to the exploration of sparsity-aware Successive Interference Cancellation (SIC) detection in [Kno+13]. The novel combination of SIC detection with Sorted QRD (SQRD), as it had been invented

for the improved performance in multi-antenna MIMO communication systems [Wüb+01], exhibits almost optimal detection performance. Alternatively, the constrained K-Best algorithm was investigated in [Kno+14], which constitutes a hardware-friendly trade-off between SIC and SD.

However, it became obvious that, for a practical technology demonstration with Software-Defined Radios (SDRs), the assumed sparsity within the communication system can already (and needs to) be exploited during user-specific channel equalisation [Kno+16b]. The cited work, therefore, successfully applies OMP to the derived channel state information (CSI) estimation problem.

Furthermore, the sparse activity pattern of the users in the WSN has direct impact on the frame synchronisation of the received superimposed data frames of the active users. This was investigated in [Zed+17], wherein a special multi-user synchronisation preamble is proposed. This preamble is based on Zadoff–Chu sequences (ZCSs) and utilises their ideal cyclic autocorrelation property to construct an orthogonal basis of user-specific codes.

Starting with an idea for joint activity and symbol detection at the end of the baseband signal processing chain of an (uncoded) digital wireless transceiver, the above summarised incremental research steps brought the notion of sparsity closer to the radio frequency (RF) front-end and within reach for a practical implementation.

1.2 Outline

The following chapters are grouped into two parts, according to the two areas of research this thesis contributes to.

The first part describes the fundamentals to motivate and explain the novel RDAM in detail:

- Chapter 2 clarifies terminology with regard to “complexity” and gives a brief introduction to contemporary HLS and how it evolved. Alternative digital design techniques are compared and current HLS tools examined for their support of the RDAM.
- Chapter 3 surveys various common data types for digital signal processing applications and explains why data types can be abstracted in general. Furthermore, the principal advantages and limitations

of the RDAM are discussed. Its climax surely is Sec. 3.4, which lists the necessary methodological design steps belonging to the RDAM.

- Chapter 4 describes the DTA and gives an in-depth illustration thereof. It is further discussed how the RDAM applies to complex-valued algorithms and how it abstracts and simplifies fixed-point arithmetic. The chapter concludes with a section on function approximation to obtain optimised (fixed-point) HLS compilation results for the implementation of mathematically complicated functions, e.g. the square root or logarithm.
- Chapter 5 at the end of Part I lists and discusses synthesis results for two design examples: a simple scalar vector product and OMP, representing a computationally complex algorithm. It concludes with a hardware-in-the-loop (HIL) simulation and embedded test of the devised architectures.

Part II deals with CS and sparse coding in a WSN:

- Chapter 6 summarises the theoretical foundations of CS and explains the two steps involved: compressive measurement and reconstruction. A survey of algorithms for sparse signal recovery is given. Afterwards, in Sec. 6.3, the CS framework is applied to digital communication systems with regard to their particular constraints. A multi-user uplink in a WSN is described as a baseline system model for the chapters to come.
- Chapter 7 presents sparsity-constrained tree search algorithms, namely SD, K-Best and SIC, and evaluates their performance for joint user activity and data detection.
- Chapter 8 shifts the attention to combined multi-user wireless channel and user activity estimation. An extended system model with data frames is introduced to enable channel estimation and timing synchronisation at the receiver, which takes sporadic activities of the users into account.

Chapter 9 finally summarises and concludes this thesis.

Part I

Rapid Digital Architecture Design Based on High-Level Synthesis

Chapter 2

Fundamentals of High-Level Synthesis (HLS)

The Rapid Data Type-Agnostic Digital Design Methodology (RDAM), which forms the essence of this work, is founded on the power of High-Level Synthesis (HLS) tools.

The following therefore briefly sketches the idea of HLS before the advantages and disadvantages of the added DTA are discussed in the next chapter. This chapter furthermore motivates the need for an elevated digital design perspective and also discusses some alternatives to HLS. It will conclude with an overview of contemporary commercial and academic HLS design tools.

2.1 Increasing Design Complexity as a Driver for High-Level Synthesis

The design of faster and more bandwidth-efficient communication systems goes hand in hand with the development of more than ever sophisticated signal processing algorithms. This has direct impact on the quantity of needed hardware resources for practical implementations. For ASIC designs this translates to silicon area and for FPGA designs to the number of utilised logic cells to create adders, multipliers, etc.

As [HJ09] formulates it, "the tremendous complexity increase from one technology generation to its successor…is roughly a factor of ten [in wireless communications,] while time-to-market has to be preserved for the sake of competitiveness." This statement directly addresses the central problem which has to be solved—to increase design productivity and

thereby enable digital hardware designs for complex signal processing applications.

Before that can be elaborated in detail, it must be explained what exactly "complex" digital architectures and algorithms are.

2.1.1 The Definition of Complexity

The word "complexity" has ambiguous meanings which makes it necessary to clarify terminology within this dissertation.

2.1.1.1 Computational Complexity

First and foremost, there is the *computational complexity* which can be expressed as the number of required arithmetic operations.

Although such a definition is short and clear, it is quite vague in practice. For floating-point designs, there is the count of floating-point operations (FLOPs) which is accurate as long as all computations are performed with floating-point arithmetic. HLS explicitly supports and allows for floating-point architectures (see Sec. 2.2), but even though, digital designs are commonly created with fixed-point arithmetic for reasons of efficiency, at least until nowadays. A FLOP count cannot represent actually needed hardware resources for such systems, because not all arithmetic operations cause the same computational impact. A hardware multiplier is of course larger than an adder, not to mention the square root operation which would probably be implemented within a design of its own.

A very important part in digital signal processing applications play multiply-and-accumulate operations (MACs), and modern FPGAs have plentiful of DSP slices for those available. The Xilinx DSP48E1 slice of the 7 series devices (shown in Fig. 2.1), e.g., consists of a full-custom 18 $\times$ 25 bit two's-complement multiplier, a 48 bit accumulator and a powerful arithmetic logic unit (ALU) to compute ten different logic functions of two operands and dual 24 bit additions or subtractions [Xil-UG479]. Therefore, most of the fixed-point arithmetic will preferably be mapped to such DSP slices and, hence, the actual DSP slice count will constitute a somewhat invariant and thus reliable estimate of the computational complexity for digital FPGA designs.

The design examples presented in Sec. 5.2.4 will demonstrate that the DSP slice utilisation is the only invariant figure between pre-RTL

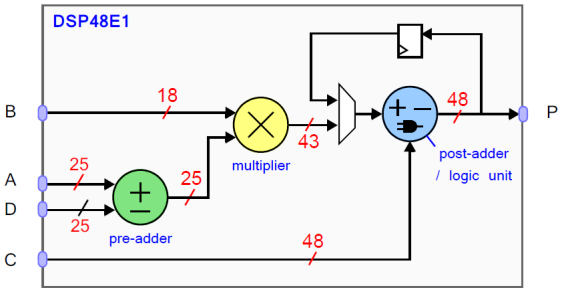


Figure 2.1: Simplified block diagram highlighting the arithmetic capabilities of the Xilinx DSP48E1 slice [Cro+14].

synthesis, i.e. the HLS estimate, and post-RTL synthesis with Xilinx Vivado.

2.1.1.2 Algorithmic Complexity

The computational complexity is closely related to the *algorithmic complexity*. The latter is well-defined and can be expressed by the so-called big O notation, which classifies algorithms according to their run time or memory storage requirements.

If these are expressed by functions $f, g : \mathbb{N} \to \mathbb{R}^+$, then

$$\mathcal{O}(g(n)) = \left\{ f \mid \exists\, c \in \mathbb{R}^+, \exists\, n_0 \in \mathbb{N}, \forall\, n \geq n_0 : f(n) \leq c \cdot g(n) \right\} \tag{2.1}$$

formally defines the set $\mathcal{O}(g(n))$, which contains all functions with a growth rate less than $g(n)$, [CLC09]. n is a parameter of the algorithm, e.g. the size of an input vector or a list to be sorted. $g(n)$ is an upper bound scaled by some positive constant c. That means, the big O notation makes only a case for the behaviour $n \to \infty$, i.e. it is an asymptotic upper bound.

Depending on the choice of g, algorithms can be categorised into classes of complexity. These are among others and in order of increasing growth rate:

- constant complexity, i.e. no dependence on n, $\mathcal{O}(1)$,

- logarithmic complexity, $\mathcal{O}(\log(n))$,
- linear complexity, $\mathcal{O}(n)$,
- polynomial complexity, $\mathcal{O}\left(n^k\right)$ for some $k > 0$, and
- exponential complexity, $\mathcal{O}(k^n)$, $k > 0$.

Concerning the digital design of algorithms, the limited availability of resources has to be taken into account to judge upon feasibility. Algorithms belonging to any class of complexity will of course be technically feasible for a sufficiently small n, although n might be larger for smaller growth rates. At all times the generally unknown constant c has to be factored in.

An exact count of arithmetic operations differs from the algorithmic complexity such that it is a tight bound with known constant, at least approximately. And it gives reliable information even for small $n \ll n_0$. The asymptotically tight bound is formally defined as

$$\Theta(g(n)) = \big\{ f \mid \exists\, c_1, c_2 \in \mathbb{R}^+, \exists\, n_0 \in \mathbb{N}, \forall\, n \geq n_0 : \\ 0 \leq c_1 g(n) \leq f(n) \leq c_2 g(n) \big\} \ . \tag{2.2}$$

There are also an asymptotic lower bound $\Omega(\cdot)$ and asymptotically tight upper/lower bounds, $o(\cdot)$ resp. $\omega(\cdot)$, [CLC09].

2.1.1.3 Mathematical Complexity

The extension of the set of real numbers $\mathbb{R}$ by imaginary numbers with the help of the imaginary unit j creates the set of *complex numbers* $\mathbb{C}$. Often, DSP applications make extensive use of complex numbers, e.g. in wireless communications. If that is the case, the adjective "complex-valued" will be used within this text. A complex-valued digital design utilises complex numbers and complex-valued additions, multiplications, etc., but can be fairly simple at the same time because it must not be a computationally complex system.

Additionally, there are *mathematically complex functions* from the perspective of a fixed-point digital design. Such functions can very well be real-valued but complicated to be computed with binary logic. The square root or trigonometric functions are good examples, since their implementation usually requires an algorithm of its own, e.g. the Coordinate Rotation Digital Computer (CORDIC) [Vol59; Wal71].

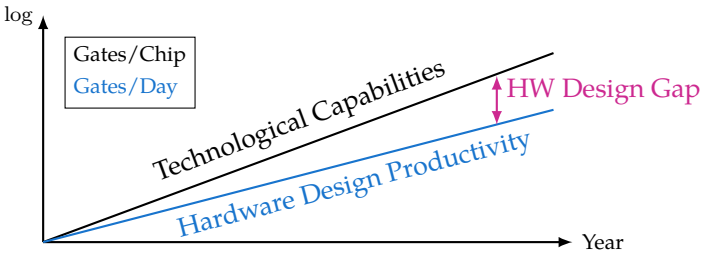


Figure 2.2: The productivity design gap [EMD09].

2.1.2 The Productivity Design Gap

The technological evolution scales quite well to accommodate the above mentioned growth in computational complexity. The famous Moore's law, that the number of transistors which can be integrated on a single chip doubles every 18 months, has been valid for decades by now although being questioned more frequently recently [Moo65; Moo98]. This leads nonetheless to an increased transistor density on the die area and the availability of more logic resources with every latest semiconductor process technology node. Contemporary FPGAs thus can offer hundreds (some even thousands) of dedicated DSP slices for the fast and efficient implementation of MACs and other operations.

But the engineer's design capabilities do not scale at the same rate. This is called the *productivity design gap* and is illustrated in Fig. 2.2. In contrast to the increasing technological capabilities according to Moore's law, the hardware design productivity increases only by a factor of 1.6 every 18 months [EMD09]. This leads to a gap which can only be closed by improved design performance.

Recently, the pace of the technological development is slowing, because the miniaturisation is approaching some physical limits and effects on the nanometre scale lead to a decreased return of investment and rising costs. Far from signaling an end to progress, this gradual "end of Moore's law" will open a new era in information technology as the focus of research and development shifts from miniaturisation of long-established technologies to the coordinated introduction of new devices, new integration technologies, and new architectures for computing [TW17]. This will

definitely have an impact on how devices and digital architectures are going to be programmed and HLS will presumably play a part here as well.

2.1.2.1 Design Entropy

In [MS13], the view has been advanced that the productivity design gap is caused by the structure of an algorithm itself, which can be measured in terms of what the authors call *design entropy*: the more irregular a hardware architecture, the higher is the design entropy.

On the one hand, memories are highly regular structures and of low entropy. Therefore, huge memories can be designed quickly and are usually the first applications for the latest technology node. On the other hand, general purpose processors, like CPUs, are examples for high-entropy designs, because of their largely irregular structure of the control logic, which consumes most of the chip area. Modern complex signal processing algorithms with irregular data paths and much control overhead, e.g. caused by loops or if-else branches, are of high entropy and, hence, more difficult to design.

In order to overcome this productivity design gap, it was proposed to introduce further tools into the design flow which are capable of automating certain implementation aspects, and especially shift the design perspective to a higher, more abstract level [MS13]. In other words, the coding of hardware architectures must be elevated to the algorithmic layer—away from traditional HDLs and the RTL.

HLS does accelerate digital architecture design by allowing hardware engineers to work on a higher level of abstraction, as desired.

2.2 A Short Historical Review

There has been a shift of meaning regarding the term "High-Level Synthesis". Whereas years ago this meant the possibility to synthesise *behavioural* HDL code [GR94], in recent times it is understood as an automated design compilation process that interprets an algorithmic description in an HLL, like C/C++ or SystemC, and synthesises a synchronous RTL description in an HDL [Cou+09].

HLS tools have been in practice since the 1950s and attracted considerable interest when early commercialisation efforts were made in the last decades [Con+11]. The development and evolution of HLS tools can roughly be divided into three generations [MS09].

- **1980s to early 1990s**. First research was conducted on HLS but with a failed commercialisation of such tools. RTL synthesis with HDLs was upcoming and HLS tools required the users to learn obscure design languages. The quality of results was poor.
- **1990s to early 2000s**. Major companies like Synopsys, Cadence and Mentor Graphics offered their tools and HLS was tried out by quite a number of users. However, this tool generation could not convince out of several reasons. For example, HDLs were used as design languages for HLS and thus competing with existing RTL synthesis tools, obtained results were highly variable and hard to validate, and simulation times were almost as long as with RTL synthesis.
- **2000s to 2010s**. While some tools are still dataflow or DSP domain-specific, most tools adopt C, C++ or SystemC as an HLL design input. The quality of results is improved stemming from compiler-based optimisations. Yet, a major impact made the rise of FPGAs, which changed the measurement criteria for a good design. The hardware architecture simply has to be fast enough and must fit within its capacity when targeting an FPGA. Last but not least, the increased computational complexity of signal processing applications had further raised the need for improved electronic system-level (ESL) tools meanwhile.

Nowadays, the tools of the third generation have matured and are capable not only to synthesise data paths but also control logic. The familiar C programming language and its variants as design inputs accommodate the skills of algorithm designers and embedded software engineers.

2.2.1 Contemporary Developments

There a two significant and game-changing developments taking place currently [HM17]:

- HLS becomes integrated into even more abstract ESL tools. Xilinx, e.g., sells the SDSoC tool which targets software development with

custom hardware acceleration on a System-on-Chip (SoC). System-level profiling identifies computationally intensive functions which can then be implemented with HLS in programmable logic as hardware accelerators. The integration into the SoC is automated to a great extent.

- The Open Computing Language (OpenCL) framework maintained by the Khronos Group finds support with HLS tool vendors [Khr17]. OpenCL is an interface based on C to program diverse parallel computing devices, foremost GPUs. However it is not restricted to GPU computing and targets heterogeneous architectures built with FPGAs as well. Commercially supported is this, e.g., by Intel's FPGA SDK for OpenCL (formerly by Altera) and Xilinx' SDAccel.

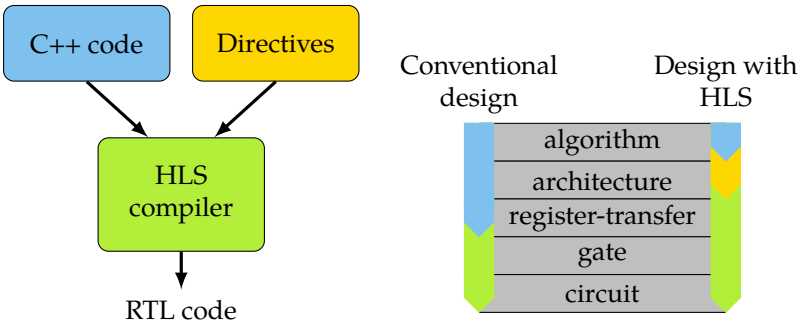
Underlying these trends is the fact that these tools natively support single-precision floating-point numbers compliant with the IEEE Standard 754 [IEEE-754], which will be discussed in detail later on in Sec. 3.2.1.1 for this reason. HLS is used to generate dedicated floating-point architectures which makes a seamless integration into software applications possible.

Intel's Arria and Stratix FPGAs even consist of *hardened* floating-point (!) DSP slices [Alt14; Sin17] while Xilinx Vivado HLS (VHLS) maps such floating-point operations to optimised soft cores consisting of a number of DSP48E1 slices, the exact number thereof depending on the respective operation.

Because of this, FPGAs have already become a major workforce in large-scale data centres for high-performance and cloud computing, e.g. for deep learning or other machine learning applications [HM17].

2.3 The Basic Principles of High-Level Synthesis

The classical design flow of complex digital signal processing applications requires a hardware engineer to devise a detailed architecture that meets system specifications, and then to manually code a RTL design in an HDL, usually VHDL or Verilog. This is an iterative process indeed, and often many design iterations are necessary. A minor re-design of the architecture can lead to substantial changes of the RTL description.



(a) HLS Inputs **(b)** Implementation Layers

Figure 2.3: Architectural HLS compiler directives complement the design abstraction of a high-level algorithmic description.

In contrast to this, the HLS-based digital design flow still needs an algorithmic specification but the architectural design is vastly simplified and the following implementation with an HDL is fully automated. Fig. 2.3 emphasises this fact as it will be explained below.

2.3.1 Inputs to an High-Level Synthesis Design

The HLS compiler commonly needs two inputs (Fig. 2.3a).

- The first input, of course, is the algorithm that is to be synthesised and supposed to be supplied in an HLL. Throughout this work this code is assumed to be a vanilla C++ source code. A considerable number of HLS tools support the family of the C programming languages (see Sec. 2.5). On the one hand, C++ lacks hardware-related features, like the explicit timing of operations or fixed-point data types. But on the other hand, the lack thereof leads to a design abstraction—as intended.
- Some of those drawbacks can very well be mitigated by the second input to the HLS compiler: The HLS synthesis process can be controlled by compiler directives, also called pragmas. These directives mainly influence the synthesised architecture. This endows the

hardware engineer with an abstracted, qualitative control over the architectural HLS output.

Nonetheless, hardware expertise is still necessary to generate efficient digital architectures, as is the adherence to HLS coding guidelines.

Fig. 2.3b shows an excerpt of the Gajski–Kuhn Y chart, namely the functional view only [GK83]. While the conventional design methodology requires manual design steps from the top down to the RTL, the HLS approach solely needs an untimed functional algorithmic specification which of course has to be given in a supported input programming language. The coding of synthesisable RTL code is completely automated and architectural design decisions may or may not be constrained with a small set of directives. The default optimisation of HLS compilers usually is for minimal resource utilisation and sequential operation.

The iterative design process mentioned above is accelerated tremendously because an HLS compiler can "mould" the desired algorithmic functionality into an architecture, and generation of RTL code is a matter of minutes or even seconds. This makes a rapid design space exploration feasible (see Sec. 3.3.2), i.e. the same high-level algorithmic specification can be reused and compiled for different implementation technologies or changed constraints. Problem sizes and even data types become configurable parameters. Additionally, the concept of a self-checking and re-usable test bench as well as fast bit-accurate C-based RTL simulations contribute to a rapid design time. The automated and correct-by-construction synthesis process reduces the occurrence of low-level implementation bugs that are routine in RTL codes, which improves the overall design productivity significantly [Ren14].

The shift of the design perspective to a more abstract implementation layer is often likened to the development of HLLs themselves: Today, nobody would even think of programming a complex software application solely by using assembly language [Cou+09]. The gain in coding efficiency by HLLs outweighs the advantages of Assembler by far, such that assembly code still exists but only in niche applications.

2.3.1.1 Coding Style Guidelines

The adherence to a suitable coding style is mandatory to obtain efficient RTL designs. [FB10] puts it the following way:

> "One of the common misconceptions held by people is that synthesizing hardware from C++ provides users the freedom of expressing their algorithms using any style of C++ coding that they desire. When designing using high-level C++ synthesis, it is important to remember that we are still describing hardware using C++, and a "poor" description can lead to a sub-optimal RTL implementation."

Probably the most severe limitation is that variables can only be allocated on the stack. Due to this restriction, the HLS tool can infer the needed amount of registers to store their values, which is especially important with regard to arrays, the sizes of which need to be fully determined at compile time. The reason why variables cannot be allocated on the heap memory (i.e. with the C/C++ commands `malloc`, `calloc`, `free`, `new` or `delete`) is simple: There is no operating system to control dynamic memory usage on an FPGA or ASIC.

Nonetheless, there are applications which require a variable sizing of arrays or other parameters. A viable solution is to resort to function and class templates. A template is instantiated with a set of constant parameters, a process which is called specialisation. For instance, function recursion is prohibited because the recursion depth is generally undefined at compile time, however tail recursion with a templatised function and additionally a tail specialisation for the final call is possible [Xil-UG902].

Further, data dependencies are potentially problematic and can lead to performance bottlenecks. Hence, sometimes the order of computations needs to be changed to allow for better parallelism. This requires manual intervention to rewrite certain pieces of a source code, yet the HLS test bench checks for—and therefore ensures—functional equivalence.

A significant benefit of HLS is its capability to generate RTL microarchitectures of loop statements (`for`, `while`, `do...while`), but variable loop bounds are to be avoided. It is highly recommendable that the initialisation of the loop iterator is a constant, that the test condition is against a constant, and that the iterator increment is a constant. This allows for an efficient control logic with small bit widths since the value range of the iterator can be determined during synthesis [FB10].

Additional details on other possible design optimisation are given, e.g., in [FB10] and [Xil-UG902]. As a last remark, specialised HLS code libraries are made available by the tool vendors for certain common

problems, e.g. to efficiently compute non-trivial mathematical functions or to solve linear algebra problems.

2.3.1.2 Directives for Architecture Design

Digital architecture design is certainly one of the greatest strengths of HLS with regard to synchronous very-large-scale integration (VLSI) design. There are directives to control basic properties of the generated micro-architectures like area, latency or throughput. A fine-grained application of directives is possible to the top-level function, named sub-functions, variables and interfaces.

Figures of Merit Whereas area can be expressed in terms of Gate Equivalent (GE) for ASIC designs, it is impossible or at least unfair to reuse this metric for FPGA designs, the reason being the largely heterogeneous building blocks of them. Therefore, most works in the literature give figures pertaining the utilisation of certain resources, as also recommended in [Xil-UG902]. *Resource utilisation* is usually stated as a quadruple $(\text{BRAM}, \text{DSP}, \text{FF}, \text{LUT})$, where the abbreviations stand for Block RAM, specialised DSP slices with hardware multipliers (e.g. DSP48E1 by Xilinx), flip-flops and look-up tables, respectively.

Latency is the number of clock cycles from the first input to the first valid output. And *throughput* can be computed as the inverse of the number of cycles per data item. The clock cycles per data item differ from the latency because digital designs are pipelined in general. In HLS parlance, the "clock cycles per data item" is termed *Initiation Interval* (II) and defined as the number of clock cycles before a function can accept new input data and the next loop iteration is started. An $\text{II} = 1$ means a new loop iteration is started every clock cycle [Xil-UG902; FB10].

Architectural Equivalence Transforms Three basic equivalence transforms can universally be applied to an algorithm of combinatorial computations to improve latency and throughput or to exploit parallelism [Kae08]:

- **Decomposition** of a function consisting of a number of calls to an identical smaller sub-function, which can then be executed sequentially with a smaller hardware footprint.

Table 2.1: Selected directives of Xilinx Vivado HLS [Xil-UG902]

Directive	Description
`ALLOCATION`	Limits the number of resources such as DSP slices.
`ARRAY_PARTITION`	Partitions an array into smaller arrays or registers.
`DATA_PACK`	Aggregates multiple variables into a single word.
`DATAFLOW`	Enables concurrent execution of functions or loops.
`EXPRESSION_BALANCE`	Balances logic depth, e.g. to create adder trees.
`INLINE`	Inlines a function and thus improves latency.
`INTERFACE`	Specifies top-level RTL ports and protocols.
`LATENCY`	Sets a latency constraint.
`PIPELINE`	Enables function or loop pipelining.
`RESOURCE`	Constraints a specific HLS library core to be used.
`STREAM`	Declares an array to be implemented as FIFO.
`UNROLL`	Unrolls loops fully or partially (replicates the body).

- **Pipelining** of operations to improve throughput by reducing combinatorial depth and by introducing parallelism.
- **Replication** of a functional entity to process several data elements concurrently.

With regard to HLS, resource sharing is implicitly performed during synthesis (see Sec. 2.3.3) and effectively identical to decomposition. It is safe to assume that design C/C++ code is written maximally compact and a repeatedly executed sub-function would definitely be called from within a loop body with a single line of code. Since HLS keeps loops rolled by default, i.e. a sequential architecture will be synthesised, further decomposition is not possible. Hence, the main attention of HLS architecture design is how to pipeline and replicate certain combinatorial data paths with directives.

Exemplarily, Tab. 2.1 lists an excerpt of all HLS directives supported by VHLS. An optimisation to begin with is pipelining to reduce latency and increase throughput. The `PIPELINING` directive allows for an optional parameter with a target II, and it can be applied to loops and functions. Further, task-level parallelism can be exploited by specifying the `DATAFLOW` directive: Normally, sequential statements are scheduled sequentially, but with data flow optimisation they can be executed in parallel as far as data dependencies permit. Loop unrolling by `UNROLL`

replicates the loop body, and the directive allows for an optional factor to partly unroll loops. This drastically increases throughput (and area). For most designs, only the most inner loops become unrolled for a good area–throughput trade-off. Nonetheless, unrolling often creates data access bottlenecks with BRAM which has a single or dual port for read/write operations. Therefore, this directive goes hand in hand with the partitioning of arrays into several BRAMs or even into registers each element to increase data concurrency. Last but not least input-output (I/O) interface options can be specified to the top-level function. This includes certain handshake protocols or even Advanced Extensible Interface (AXI) buses to integrate the generated Intellectual Property (IP) core into a SoC later on.

Estimated Clock Frequency Additional constraints can be set to configure resets and (target) clocks. Most importantly, VHLS can only estimate the maximal operating frequency, because HLS resorts to a HLS component library (for multiplications, additions, and so on) with expected timing information depending on the configured target devices. The C/C++ code is synthesised into an RTL code and then, again, by RTL synthesis into a gate-level description. A timing margin is reserved for the design steps to come including place and route.

2.3.2 C-Based Test Benches for Verification

The created hardware architecture is correct-by-construction and bit-accurate C-based simulations of the RTL are possible with the C/C++ or SystemC design files. However, verification generally is simulation-based and formal equivalence checking is still a challenge [Ren14].

This leads to the notion of self-checking test benches. Some HLS tools require a C/C++ test bench to be associated with each design which compares the results with given or random inputs to the HLS-designed entity against pre-computed values (“golden device”) or a software model. The choice is with the designer what to check precisely for a successful verification.

Even if later code changes are necessary, e.g. to rearrange the order of computations or rewrite some loop constructs to enable further HLS

compilation optimisations, the test bench ensures functional verification as long as the test passes.

Most of the HLS tools integrate the functionality to generate RTL test benches automatically from C test benches, which makes another manual implementation step obsolete. VHLS, e.g., uses a C-to-RTL transactor and generates besides Verilog and VHDL design files also a bit-accurate and cycle-accurate SystemC RTL design to speed up simulation times [Con+11]. Although formal equivalence checking is not yet solved fully, the much wider simulation coverage of HLS and the possibility for in-system simulations are especially helpful with FPGA hardware (HW) targets.

This powerful test bench concept is utilised by the RDAM, as described later, to ensure functionality even if data types change, e.g. during the transition from floating-point to fixed-point arithmetic.

A modern paradigm of software engineering is test-driven development. This means, that the test bench for functional verification is created first, i.e. before the actual software. The software is then written until the specified target functionality is met. Similarly, the HLS test bench can test for a set of pre-defined verification requirements.

2.3.3 The High-Level Synthesis Compilation Process

Very good introductions into the internals of HLS compilation can be found in [Cou+09; GR94; Ren14]. It is a research topic of its own and can, therefore, only sketched briefly here.

The HLS compilation process consists of three main steps: compilation, optimisation and RTL generation, as shown in Fig. 2.4 with shaded blocks. A more detailed explanation of every step follows below.

1. The synthesis process always starts with the *code compilation* of the design entry sources. This relies to a great extent on the capabilities of the employed compiler. For C/C++ based designs this is the GNU C Compiler of the GNU Compiler Compilation (GCC) or a C/C++ frontend (e.g. Clang) for the Low Level Virtual Machine (LLVM) framework [Zho+16; Ren14]. This step includes code optimisations such as constant folding, loop transformations or simply dead-code elimination.

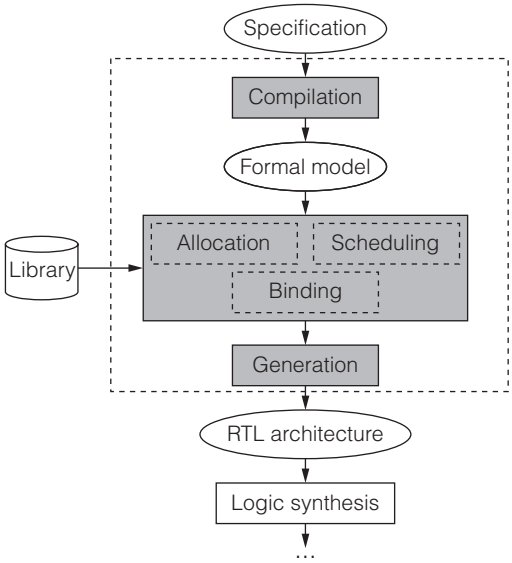


Figure 2.4: The basic HLS compilation process [Cou+09].

To give an example, the modern LLVM framework consists of three stages [LA04; Lat12]. The first stage is the language frontend, which transforms the input source code into an intermediate representation (IR), the LLVM assembly language. This is a static single assignment (SSA) representation that provides type safety, low-level operations and flexibility. The second stage is the LLVM optimisation, i.e. all code optimisations are performed on the IR. And the consecutive third stage, the backend, maps the IR onto the instruction set of the hardware target, e.g. an x86 or Advanced RISC Machine (ARM) architecture.

The LLVM is well-suited for HLS due to this modular design. In particular, the IR is fully bit-accurate and not limited to bit widths of native C types which are multiples of 8 bits in alignment with common micro-architectures. Code optimisations like memory reuse, array partitioning, automatic loop unrolling and function inlining can be exploited directly [Con+11].

2. Data dependencies can be *modelled formally* with a data flow graph (DFG) where nodes represent arithmetic operations and edges are variables. To model control structures like `if` and `switch` statements or loops, the DFG model is extended by a control graph resulting in the control and data flow graph (CDFG). Edges within a CDFG represent the control flow while nodes are either control constructs or so-called basic blocks which contain DFG models. Further optimisations are applied to exploit data parallelism within basic blocks and inter-dependencies between these.

3. Based on the formal CDFG model, *allocation* of resources is performed. "Resources" are functional entities or storage units within this context. For each (arithmetic) operation a component is selected from the RTL component library. The allocation thereby depends on the given design constraints and must satisfy those.

4. Then, operations are *scheduled*. Variables of the DFGs must be loaded and stored and operations take a certain amount of clock cycles. This can either be resource-constrained or time-constrained.

5. Consecutively, *binding* happens. It is not until then that functional resources are mapped to actual hardware units as they are provided by the target hardware platform (FPGA or ASIC). The binding task

assigns the operations and memory accesses within each clock cycle to available units. A hardware unit will be shared and reused if access to a functional resource is mutually exclusive.

6. As a last step, synthesisable *RTL generation* follows, usually with a control and data path. A single or several finite state machines (FSMs) control the inputs and outputs of the data paths within the basic blocks. Also, I/O interface logic is added, e.g. for an AXI bus or simpler handshaking protocols.

All these steps are executed automatically without further intervention by the user apart from the boundary conditions set by the applied HLS compiler directives. The amount of information about the synthesised digital design which the compiler outputs depends on the very HLS tool used, but is in any way critical in identifying design bottlenecks and thus obtaining increased performance with another design iteration.

2.4 Alternative Digital Design Techniques

HLS is not the only solution to cope with increased digital design complexity. Competing alternatives are graphical tools, and extensions to traditional HDLs can also improve design efficiency.

2.4.1 Hardware Description Languages

The two most widespread HDLs are Verilog, specified as IEEE Standard 1364 [IEEE-1364], and VHDL, specified as IEEE Standard 1076 [IEEE-1076]. There is still support to write structural RTL code, but even in the early 1990s behavioural RTL synthesis had already been introduced, as mentioned above in Sec. 2.2. Behavioural RTL synthesis is able to infer logic and can therefore be considered to be an abstraction step of its own. The capabilities depend on the respective synthesis tools but have steadily improved since then.

An advantage of HLS is the simplicity with which algorithmic parameters can be changed, like constant sizes or even data types. This is possible with HDLs as well albeit requiring more time to write. Common to Verilog and VHDL are so-called generics. In Verilog, the line

```
parameter PARAMETER_NAME = VALUE;
```

defines a constant with the help of the `parameter` keyword. This could be used for varying input sizes or fixed-point word lengths, and can later be changed and also passed on to instantiated modules. A parameter of a module can be set by the syntax

```
#(.PARAMETER_NAME(VALUE));
```

after instantiation. Alike, VHDL allows to define generic constants and generic types with the `generic` and `generic map` keywords during entity declaration resp. component instantiation.

Additionally, VHDL supports generic packages. This is at the heart of the predefined standard packages, foremost the `fixed_generic_pkg` for fixed-point arithmetic and `math_complex` for complex-valued mathematical operations [Ash08]. However, some functionalities have only been introduced with the latest revision of the standard in 2008, and its adoption by behavioural synthesis tools has been slow.

Therefore the well-known HDLs can offer basic generic functionality as required by the RDAM, but nonetheless the usual disadvantages apply. On the RTL level, design space exploration, especially regarding the exploration of different degrees of parallelisms, is slow and error-prone due to the needed number of changed lines of code. In every case, the implementation quality depends on the skills of the digital design engineer as it is the case with HLS.

In conclusion, the intentional input abstraction with an HLL speeds up design time for highly flexible digital architectures with HLS. The compiler has to fill in for the missing fine-grained control over the details of hardware design.

2.4.2 Graphical Programming Languages

Another solution being proposed to the hardware design productivity challenge are graphical programming languages. To begin with, National Instruments offers the possibility to program their FPGA products with LabVIEW graphically. Then there is Mathworks Simulink as a standalone product or in conjunction with another tool, which can either be Xilinx' Vivado System Generator for DSP or the Altera DSP Builder Advanced Blockset which Intel markets now under the name DSP Builder for FPGAs.

The Simulink-based tools offer a library of functional blocks which can be utilised to create DSP algorithms. The DSP blocks are modelled

and written in C and have hand-crafted and optimised RTL architectures associated with them. Hence, the quality of results is quite good by design. A major strength of these tools is the convenient support for fixed-point data types with varying bit widths and easy functional design verification.

Nonetheless, common to these tools are the limitations of the graphical design entry. It is mostly data flow oriented and the overview suffers for large designs. When it comes to architectural changes, e.g. exploiting parallelisms, the graphs have to be remodelled substantially, which is time-consuming and tedious. Furthermore and most importantly, the graphical programming languages are non-standardised and subject to change from version to version to come.

These disadvantages are HLS's advantages. Not only are C programming languages well standardised, but also the RTL output in VHDL, Verilog or as a packed IP core in the IP-XACT format is standardised [XilUG902]. The architectural design is vastly simplified by a handful of compiler directives and does not require any code modifications most of the time.

2.5 An Overview of Current High-Level Synthesis Tools

The market for HLS tools is steadily changing. Some tools had originated with academic research and were then bought and advertised commercially, often accompanied by a product name change. Others have been discontinued. [Mee+12; Nan+16; RJ16] comprehensively survey, list and compare known HLS tools.

Selected tools will be mentioned below, which are multi-domain and can synthesise American National Standards Institute (ANSI) C and possibly C++ or SystemC code. Tab. 2.2 gives an overview over these tools and judges their qualification with respect to the RDAM. All tools do support some kind of DTA either founded in the polymorphism and object orientation of C++/SystemC or, as a baseline, the `typedef` keyword of all C dialects. (More on this later in Chap. 4.) However, there are differences regarding the availability of floating-point (floating-point) or fixed-point (fixed-point) data types.

Table 2.2: An overview of several commercial and academic HLS tools and an assessment of their suitability for the RDAM

	Name	Developer	Input	FPGA	ASIC	FLP	FXP	RDAM
Commercial	Catapult HLS	Mentor Graphics	C, C++, SystemC	✔	✔	✘	✔	(✔)
	HLS Compiler	Intel Corporation	C, C++, SystemC	✔	✘	✔	✔	✔
	LegUp 5.0	LegUp Computing	C	✔	✘	✔	✘	(✔)
	Stratus HLS	Cadence	C, C++, SystemC	✔	✔	✔	✔	✔
	Synphony C	Synopsys	C, C++	✔	✔	✘	✔	(✔)
	Vivado HLS	Xilinx	C, C++, SystemC	✔	✘	✔	✔	✔
Academic	Bambu ([PF13])	Politecnico di Milano	C, (C++)	✔	✔	✔	(✘)	(✔)
	GAUT ([Cou+08])	Univ. de Bretagne	C, C++	✔	✘	✘	✔	(✔)
	KiwiC ([SG08])	Univ. of Cambridge	C#	✔	✘	✔	✘	(✔)
	LegUp 4.0 ([Can+11])	Univ. of Toronto	C	✔	✘	✔	✘	(✔)

From the perspective of the RDAM, all HLS tools which support C++ (and thence SystemC) as input language and furthermore floating-point and fixed-point data types are well-suited. The fixed-point conversion of algorithms is optional to the RDAM and can still be applied partly if fixed-point types are missing, as it will be explained in Chap. 3. The floating-point validation and verification step of the C/C++ sources codes can be achieved with any C/C++ compiler although the methodological flow becomes easier if that is performed from within the HLS tool and test bench. Hence, Tab. 2.2 prints the RDAM tool support in brackets if it is with such limitations only.

2.5.1 Commercial Tools

The following paragraphs are in alphabetical order of the HLS tool name.

Mentor Graphics acquired from Calypto Design Systems the HLS tool Catapult-C in 2011, now named *Catapult High-Level Synthesis*. Originally it targeted ASIC hardware development only but has gained support for FPGAs meanwhile as well. It accepts C, C++ and SystemC inputs and also ships with a library for custom bit width fixed-point data types, `ac_fixed` and `ac_int`, called *Algorithmic C* types [FB10].

In 2015, the FPGA manufacturer Altera was bought by the Intel Corporation, which accompanied the takeover of their HLS tools. Intel now offers the *HLS Compiler* and OpenCL SDK targeting their FPGAs, the flagships being the Stratix devices. Some FPGAs are SoCs as well, integrating an ARM hard processor. Furthermore, the Arria 10 has up to about 1.500 hardened single-precision floating-point multipliers/adders besides up to 3.000 18×19 bits hardware multipliers. Hence, the Intel HLS Compiler, which builds upon the GNU C Compiler, targets floating-point development (only). It offers graphical inspection of the CDFGs for analysis [Int17].

The latest addition to the family of commercial HLS tools is *LegUp* 5.0 by LegUp Computing Inc. It is a commercial spin-off from the academic LegUp tool by the University of Toronto and targets FPGAs of every vendor [Can+11].

Cadence bought the Forte Cynthesizer and integrated it with its own C-to-Silicon Compiler into *Stratus High-Level Synthesis*. It supports the design process from SystemC transaction level modeling (TLM) simulations down to the gate level. There is library support for IEEE-754

floating-point data types and their bit-accurate simulation, even with an arbitrary partitioning into mantissa and exponent bits [Cad15].

Synopsys offers the *Synphony C Compiler*. It allows for performance optimisations with loop unrolling or pipelining and supports fixed-point data types, whereas floating-point types are not supported [Nan+16]. Once also Synphony High-Level Synthesis was offered as a product to compile RTL designs from Mathworks Matlab M code [Syn], but it appears that it has been discontinued or partly integrated into other products as it has happened with the very similar Xilinx AccelDSP [Mee+12].

Xilinx *Vivado HLS* has got a pretty long history: It started off as xPilot by the University of California, Los Angeles (UCLA) [Con+06], and had then been licensed as AutoPilot by AutoESL until 2011, when that company was bought by Xilinx. They marketed the product as Vivado ESL until it was re-branded as Vivado HLS and integrated into the Vivado Design Suite. In this text it is abbreviated as VHLS to distinguish between HLS as a methodology and VHLS as a tool. VHLS is now based on the LLVM [Con+11]. And it includes a complete design environment to analyse the architectural optimisations by directives, to perform bit-accurate C-based simulations, RTL co-simulation and IP packaging. The tool ships with several libraries, e.g. for linear algebra and OpenCL support. There is also a ready-to-use C-level CORDIC implementation. SystemC's fixed-point types are re-implemented as "arbitrary precision" types named `ap_fixed` and `ap_int` [Xil-UG902].

Xilinx VHLS is top-ranked in surveys regarding the quality of results and feature set of the tool [Mee+12; Nan+16]. Therefore, VHLS is utilised throughout this dissertation if not stated otherwise.

2.5.2 Academic Tools

Academic HLS tools can compete with commercial tools regarding the quality of results, albeit commercial compilers support more features and are more robust in general [Nan+16]. A recent survey can be found in [RJ16]. With respect to the RDAM, every academic HLS tool exhibits some limitations.

Bambu is part of the PandA project at the Politecnico di Milano [PF13]. It is written in C++ but supports only C as design entry language. Bambu leverages the GCC C Compiler optimisations for HLS and targets FPGAs

of the major vendors but is not limited to FPGA designs. It particular, it supports the dynamic resolution of memory addresses and floating-point arithmetic with the help of the FloPoCo library. Bambu does synthesise C++ code (unofficially) but cannot generate any timing results in that case. If used in conjunction with the `ac_fixed` and `ac_int` data types, fixed-point designs are possible.

GAUT from the Université de Bretagne Occidentale is, in contrast to all other tools being mentioned here, domain-specific for DSP applications. It takes as input a C or C++ description of the algorithm to be synthesised whereby the Algorithmic C class library from Mentor Graphics can be used [Cou+08]. The generated architecture consists of three parts: a processing unit (i.e. the datapath and controlling FSM), a memory unit and a communication and interface unit. It also generates a test bench for validating the equivalence between the behavioural description and the RTL output. GAUT is currently being rewritten and support for advanced data types like `ac_fixed`, `ac_int` is momentarily unavailable because operator clustering is not yet implemented[1].

KiwiC from the University of Cambridge is unique in its design entry HLL which is C# [SG08]. It allows the programmer to use parallel constructs such as events and threads that are closer to hardware concepts than classical software constructs [Nan+16]. Due to the polymorphism of C#, the data type-agnostic programming style of the RDAM is supported although dedicated fixed-point data types are missing.

LegUp is the HLS tool developed at the University of Toronto and available for download in version 4.0 [Can+11]. It is based on the LLVM compiler framework and benefits from its flexibility. LegUp can either compile a digital hardware design or a hybrid processor–accelerator system. The input language is C. A unique feature is its ability to compile parallel software threads into parallel-operating hardware. The quality of results can compete with commercial tools, and last but not least, this may have been the reason why this academic tool was commercialised recently (see above). Arbitrary-precision floating-point data types are supported, but fixed-point types are not.

[1] Personal correspondence between Philippe Coussy and Babar Khan.

Chapter 3

The Rapid Data Type-Agnostic Digital Design Methodology

The Rapid Data Type-Agnostic Digital Design Methodology (RDAM) builds upon the powerful HLS tools available today. This fact alone could already account for the "rapid" development times. Even so, design time is additionally improved because of the data type agnosticism (DTA).

This chapter, therefore, justifies why it is reasonable to abstract data types and which benefits arise. To this end, applicable floating-point and fixed-point binary number formats are surveyed and commonalities highlighted. An important conclusion will be that every binary number encoding, including any floating-point format, has a limited numerical precision and a representable number range, and can, within certain constraints, be exchanged with each other. Further, the scope of application of the RDAM is discussed and limitations juxtaposed with its advantages, especially concerning the resulting capability for enlarged design space exploration. This chapter concludes with a concise step-by-step summary of the methodological design flow in Sec. 3.4.

3.1 A First Observation on Algorithm Development

Algorithms are commonly devised or derived mathematically and can therefore be elegantly described in mathematical language.

Without loss of generality it might be assumed for the time being that a signal processing algorithm operates on the field of real numbers. Then,

many implementation steps (synthesis steps according to Gajski [GK83]) follow until a working gate-level digital circuitry is obtained.

Generally, a first step is a numerical software implementation with single-precision or double-precision floating-point data types. Usually either powerful numerical software packages, e.g. Mathworks' Matlab, Waterloo's Maple or Wolfram's Mathematica, or the C programming language are utilised for this purpose.

For traditional RTL digital designs, the next synthesis step is the fixed-point conversion of the algorithm. A digital architecture has to be devised and described in an HDL. Yet, every effort is spent to model the *mathematical behaviour* of the algorithm *invariantly*, i.e. during the top-down digital design flow data types will change—but not the algorithm itself. Phrased differently: The mathematical definition is per se data type-agnostic.

3.2 Binary Number Systems and Their Deficiencies

Even though the precision of numerical floating-point simulations might be fairly good and sometimes appear to be almost perfect, each and every binary number system for machine computations is *a bad or a worse* approximation of real numbers only. Generally speaking, any number encoded by k bits long sequence can represent 2^k code points, which is far less than infinity—the cardinality of the set of real numbers.

An overview and classification of the data types used or mentioned within this dissertation gives Fig. 3.1. There are various ways to encode numeric values in a bit pattern, which belong either to the class of floating-point or integer/fixed-point numbers.

3.2.1 Floating-Point Data Types

Floating-point data types generally do not represent numerical values exactly, although their current omnipresence in computer science may lead to a certain unawareness of this insight [Gol91]. In fact, every binary floating-point system can only encode rational numbers ($\mathbb{Q}$) and is incapable of expressing irrational numbers, since the denominator is always a power of 2 [Gus15]. Nonetheless, however, they do offer a huge

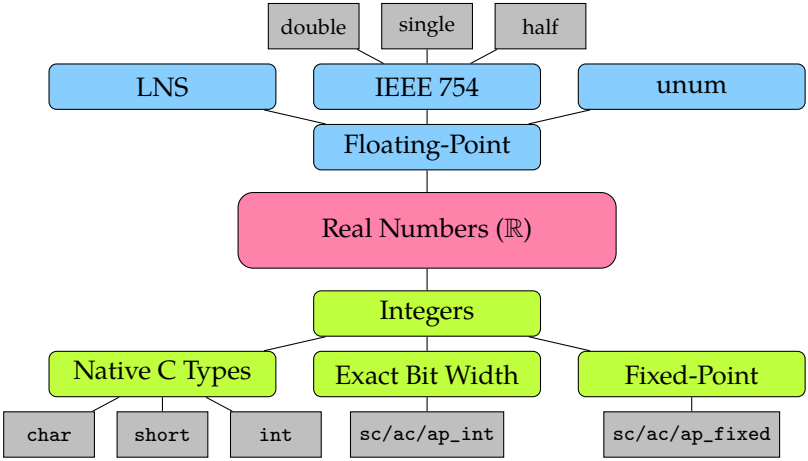


Figure 3.1: Every binary number system is an approximation to the real numbers either belonging to the class of floating-point or fixed-point types, whereby the latter is modelled by integers.

dynamic range, following the ideas of the scientific notation to span the interval between small values on an atomic scale and large numbers on an astronomical scale.

Floating-point systems have a base (or radix) β, which is for binary systems $\beta = 2$, and a precision of p significant digits. A floating-point number z is represented by

$$z = (-1)^s \cdot m \cdot \beta^e \tag{3.1}$$

where $s \in \{0, 1\}$ is the sign bit, m is the significand (sometimes also called mantissa) of p digits given in radix β (for $\beta = 2$ these are bits) and e is the exponent. It is the exponent which allows the radix point to float to the left ($e < 0$) or to the right ($e > 0$). Furthermore there are the parameters e_{min} and e_{max} to floating-point numbers which define the range of the integer e as $e_{\text{min}} \leq e \leq e_{\text{max}}$. In general, every floating-point number system $\mathcal{F} \subset \mathbb{Q} \subset \mathbb{R}$ is well-defined by the tuple $(\beta, p, e_{\text{max}}, e_{\text{min}})$, [Hig02].

It is important to understand that floating-point systems encode numbers in a sign-magnitude format and that the representable values are

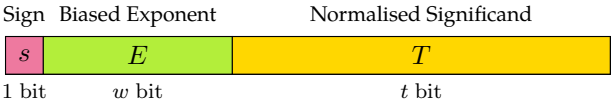


Figure 3.2: The bit string interchange encoding of floating-point numbers as defined by the IEEE Standard 754.

not equally spaced, except for the special case of subnormal numbers (see below).

3.2.1.1 The IEEE Standard 754

Historically, there have been various floating-point number systems resulting in incompatibilities between computing machines. In fact, this had been the very reason for the standardisation process leading up to the current IEEE Standard 754 [IEEE-754]. It does not only define binary (and decimal) floating-point formats of different degrees of precision but also rounding rules and requirements for the computation of certain arithmetic operations to reduce the accumulation and propagation of rounding errors.

The IEEE standard defines $e_{\text{min}} = 1 - e_{\text{max}}$. The exponent e can thus be encoded by $w = \lceil \text{ld}(2e_{\text{max}}) \rceil$ bits. Furthermore, the standard defines bit string encodings as interchange formats and exploits two techniques to store two bits implicitly.

Firstly, e is stored as the (unsigned) biased exponent $E = e + e_{\text{max}}$ and therefore avoids saving the otherwise necessary sign bit to represent negative exponent values.

Secondly, the significand $d_0.d_1d_2 \ldots d_p$ is stored as a normalised number where $d_0 \neq 0$, i.e. for a binary system $1.d_0d_1 \ldots d_t$. Please note that the leading high bit must not be stored and is, therefore, a "hidden bit". The standard differentiates for this reason between "precision bits" p and the size of the "trailing significand field" $t = p - 1$, which actually becomes stored in memory. An added benefit of normalisation is that it leads to a unique mapping between numbers and their floating-point encodings—the downside of it being the impossibility to represent the zero. That is why the all-zeroes and all-ones exponent bit patterns are reserved for special purposes by the IEEE standard.

Table 3.1: Properties of floating-point (floating-point) numbers as defined by the IEEE Standard 754 [Mol17].

Parameter		Floating-Point Format			
		half	**single**	**double**	**quadruple**
IEEE-754 Name		binary16	binary32	binary64	binary128
Size	k/bit	16	32	64	128
Precision	p/bit	11	24	53	113
Max. Exponent	$e_{\max}$	15	127	1023	16383
Significand	t/bit	10	23	52	112
Exponent Bits	q/bit	5	8	11	15
Decimal Digits		3.31	7.22	15.95	34.02
Machine Epsilon	ϵ	9.8×10^{-4}	1.2×10^{-7}	2.2×10^{-16}	1.9×10^{-34}
Overflow Level		65 504	3.4×10^{38}	1.8×10^{308}	1.2×10^{4932}
Underflow Level		6.1×10^{-5}	1.2×10^{-38}	2.2×10^{-308}	3.4×10^{-4932}
Min. Subnormal		6.0×10^{-8}	1.4×10^{-45}	4.9×10^{-324}	6.5×10^{-4966}

Fig. 3.2 shows the exact arrangement of the sign bit, biased exponent bit field and the trailing significand field. The parameters can be taken from Tab. 3.1 which lists the most relevant floating-point formats defined within the standard. The single and double types are available in the C programming languages as `float` and `double`, whereas the other types require the utilisation of specialised libraries. All parameters listed in the table can be derived from p and $e_{\max}$.

Accuracy of Arithmetic Operations The *unit in the last place* (ULP) quantifies the spacing between two consecutive floating-point numbers, i.e. when the least significant digit of the significand changes by one unit; for $\beta = 2$ this is a bit flip of the least significant bit (LSB). This, however, is numerically not a constant since the size of the represented interval scales with the exponent of the number.

Nonetheless, ULP can be upper-bounded by *machine epsilon* ϵ. The largest spacing of floating-point numbers is between 1.0 and β. ϵ is defined as the spacing within this interval,

$$\epsilon = \beta^{1-p} = \beta^{-(p-1)} = \beta^{-t}\,, \tag{3.2}$$

which characterises the attainable precision with that particular floating-point format [Hig02]. In Mathworks Matlab, e.g., this quantity is available to the programmer as `eps` [Mol17].

Closely related to ϵ, and of greater importance for numerical analysis, is the *unit roundoff* u for which

$$u = \frac{1}{2}\epsilon \tag{3.3}$$

holds [Hig02]. Every real number x lying in the range of $\mathcal{F}$ can be approximated by an element of $\mathcal{F}$ with a relative error no larger than u, i.e. the number either becomes rounded up or rounded down to the closest representable code point [1]. That quantisation may be described by the operator $\mathrm{Q}_{\mathcal{N}}\{\cdot\}$ with regard to the number system $\mathcal{N}$ denoted by the index. Then, if $x \in \mathbb{R}$ lies within the range of $\mathcal{F}$ it is [Hig02]

$$\mathrm{Q}_{\mathcal{F}}\{x\} = x(1+\delta), \quad |\delta| < u\,. \tag{3.4}$$

The IEEE standard requires every addition, multiplication, division and even square root operation to be correct within u. This is the standard model of floating-point arithmetic and can mathematically be described as

$$\mathrm{Q}_{\mathcal{F}}\{x \circ y\} = (x \circ y)(1+\delta), \quad |\delta| < u\,, \tag{3.5}$$

with $x, y \in \mathcal{F}$ and $\circ$ being one of the just mentioned arithmetic operations in infix notation [Hig02]. This accuracy is even attainable for a fused multiply-add operation, $x \cdot y \pm z$.

In other words, every result of an arithmetic operation becomes rounded to the nearest floating-point number. Tab. 3.1 states the average number of correct decimal digits, which equates to $p\log(\beta)$ if the floating-point number is converted to a decimal. However, 9 or 17 decimal digits are required to recover a single-precision resp. double-precision floating-point number uniquely [Gol91].

[1] Because of the one-half pre-factor, there are competing definitions of machine epsilon in the literature. [Gol91] defines machine epsilon to be equal with the unit roundoff as $u = \epsilon = \frac{1}{2}\beta^{1-p}$.

Number Range and Subnormal Numbers The range of floating-point numbers is given, in general, by

$$\text{UFL} = \beta^{e_{\min}-1} \leq |z| \leq \beta^{e_{\max}}(1-\beta^{-p}) = \text{OFL} \tag{3.6}$$

for $z \in \mathcal{F}$, [Hig02]. The left bound is also called the underflow level (UFL) and the right bound the overflow level (OFL). Note, that the UFL is well above zero and, hence, the zero itself is not within the representable number range.

For this reason, the IEEE standard reserves two exponent values for special purposes. The all-zero bit pattern of the exponent encodes either $+0$ or -0 if the significand is zero, depending on the sign bit. The specified behaviour for values below UFL is a "flush to zero". Furthermore, the all-ones exponent bit string encodes positive or negative infinity (significand is zero) or not a numbers (NaNs) (significand is non-zero), which are the results of invalid operations, e.g. the division by zero.

An important extension specified by the 2008 IEEE standard 754 are *subnormal numbers* (formerly called denormal numbers). They are encoded by the all-zero exponent pattern and a non-zero significand. However, the implementation thereof is optional for IEEE-compliant floating-point hardware [Gus15].

Subnormal numbers extend the number range and fill the gap between zero and the UFL. They have the minimum exponent and their equidistant spacing is $\beta^{e_{\min}-1}$. They are not normalised (the hidden bit is ignored) and therefore with a digit less of precision. In fact, subnormals are nothing other than fixed-point numbers and enable a "gradual underflow" instead of a flush-to-zero behaviour. The smallest subnormal magnitude is $\beta^{e_{\min}-1} \cdot \epsilon$ (see Tab. 3.1).

3.2.1.2 The Universal Number Format

Although not directly used for this thesis, the Universal Number (unum) format is an interesting extension of the floating-point formats discussed within the previous section. A recommendable reading is the book by Gustafson, who is a major proponent of this number system [Gus15].

Number rounding is absolutely avoided by unums. Instead, a *ubit* u is introduced which determines whether the unum corresponds to an exact number ($u = 0$) or an interval between consecutive exact unums

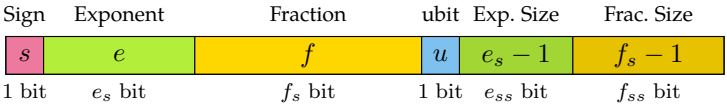


Figure 3.3: The unum bit string encoding [Gus15].

($u = 1$); see Fig. 3.3. In this sense, unums cover the entire real number line without any gaps, including $-\infty$ and $+\infty$ [Gus15]. Computations are performed with a kind of interval arithmetic, providing the guarantee that the resulting interval contains the exact solution.

The unum system is a variable-width storage format for both the significand (fraction) and exponent, and can therefore be tuned in its numerical accuracy to the demands of the application. The bit width of the exponent, e_s, and the fraction, f_s, are stored in two additional fields, as indicated in Fig. 3.3. The triple (u, e_s, f_s) forms the so-called utag, which describes the "environment" of the unum. A utag of $e_s = 4bit$ and $f_s = 7\,\mathrm{bit}$ would encompass all IEEE floating-point formats, including quadruple-precision.

Being a superset to traditional floating-point formats, unums inherit the advantages of floating-point numbers, like the representable number range, precision and subnormal numbers. [Gus15] argues that most of the computations can be performed with less than 32 bits, as it is the case with single-precision IEEE floating-point numbers. However, varying bit string lengths would make a redesign of traditional memory storage architectures mandatory. Additionally, the resizing of the exponent and fraction bit fields incurs an increased hardware complexity penalty.

As of today, unums solely play a role in academic research, yet their intriguing properties make them an interesting object of study; and since modern HLS tools are able to synthesise floating-point arithmetic, support for unum computing could be induced by properly designed C/C++ libraries. This, in turn, means that the proposed RDAM can support unums at least theoretically.

A first work on digital designs with unums created with HLS has been published recently [Hou+17].

Figure 3.4: The bit string encoding of the LNS with a logarithmic mantissa.

3.2.1.3 The Logarithmic Number System

A number format that will be utilised later on for some computations is the Logarithmic Number System (LNS). The fundamental ideas of the LNS date back to the seminal paper of Mitchell in 1962 [Mit62; Rus14].

The bit string representation of a number in the LNS is quite similar to a floating-point number, whereby the important and sole difference is the logarithmically scaled significand, then called mantissa (see Fig. 3.4). Mostly, the employed logarithm is the logarithmus dualis (or binary logarithm), denoted as $\mathrm{ld}(\cdot)$. However, variations to the bit string encoding and the number format itself exist since the LNS is not standardised [SA75; Has+05].

The value of a real number v, given in the LNS by (s, e, m), can be obtained by

$$v = (-1)^s \cdot 2^m \cdot 2^e \,, \tag{3.7}$$

whereby e is an integer exponent and m the normalised (fractional) mantissa. Because of the exponent, the binary point floats and thus the LNS has a large dynamic range like common floating-point numbers. For number format conversion, the functions

$$\mathrm{LOG}(x) = \mathrm{ld}(x+1)\,, \text{and} \tag{3.8a}$$

$$\mathrm{ALOG}(x) = 2^x - 1 \tag{3.8b}$$

with a bijective mapping $[0,1] \to [0,1]$ are needed. Obviously, $x = \mathrm{ALOG}(\mathrm{LOG}(x)) = x$ holds. The notable contribution of [Mit62] is the approximation $\mathrm{ld}(x+1) \approx x$.

The greatest advantage of logarithmically scaled numbers is that logarithmic identities can be exploited for the simplification of certain arith-

Table 3.2: Properties of different integer and fixed-point number formats.

Type	Format	Minimum	Maximum	Quantisation
Unsigned integer	w bit	0	$2^w - 1$	$(2^0 =)\ 1$
Sign-magnitude	w bit	$-(2^{w-1} - 1)$	$2^{w-1} - 1$	1
Two's complement	$\mathrm{Q}(m, 0)$	-2^m	$2^m - 1$	1
Fixed-point	$\mathrm{Q}(m, n)$	-2^m	$2^m - 2^{-n}$	2^{-n}
Fractional Q	$\mathrm{Q}(0, n)$	-1	$1 - 2^{-n}$	2^{-n}

metic operations. Especially it is

$$\log_\beta(xy) = \log_\beta(x) + \log_\beta(y)\,, \tag{3.9a}$$

$$\log_\beta(x/y) = \log_\beta(x) - \log_\beta(y)\,, \text{ and} \tag{3.9b}$$

$$\log_\beta\left(x^d\right) = d\log_\beta(x)\,, \tag{3.9c}$$

for any real-valued base β and numbers x, y. Hence, the multiplication and division can be reduced with the help of the LNS to a simpler addition and subtraction, resp. The exponentiation becomes a multiplication within the logarithmic domain and for the special case of a square root this boils down to a multiplication with a factor of $1/2$ which can be realised by a simple bit shift in binary arithmetic.

Nonetheless, the application of the LNS requires the evaluation of the conversion functions LOG and ALOG in (3.8). A low-complexity solution and application example will be given in Sec. 4.4.3, page 72, later.

3.2.2 Integer and Fixed-Point Numbers

With regard to digital signal processing, integer or fixed-point arithmetic is of paramount importance since it allows for simpler digital circuits and, hence, more efficient hardware architectures. This derives from the fixed position of the radix (i.e. binary) point, which does not need to be aligned first during computations. Consequently, all integer and fixed-point numbers are equidistantly spaced along the number line.

An introduction to fixed-point number representations can be found, e.g., in [Par10]. The most common way to represent a numeric value with a bit string is the binary weighted number system. Each bit position

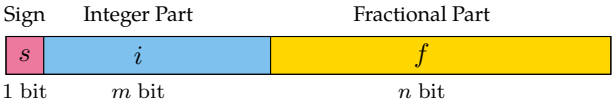


Figure 3.5: Bit string encodings of integer and fixed-point formats.

is assigned a specific weight, which is a power of two. Still, there are several possibilities to encode a numeric value number into a bit string depending on the particular weights assigned to each bit position, as listed in Tab. 3.2. The upper half of the table lists integer formats, whereas the lower half lists fixed-point formats. The Q format will be explained in detail below in Sec. 3.2.2.2. Fig. 3.6 illustrates the integer encodings for $w = 3\,\text{bit}$.

The general bit string encoding of a fixed-point number is illustrated in Fig. 3.5. The total bit width is

$$w = m + n + 1\,, \tag{3.10}$$

whereby m is the number of the integer bits (excluding the sign bit) and n is the number of fractional bits. For integer types, the fractional part vanishes ($n = 0$), whereas for the purely fractional format, it is the integer part which vanishes ($m = 0$). Unsigned numbers do not have a sign bit.

To represent negative values, the most significant bit (MSB) is used as sign bit. This opens up two alternative encoding possibilities: First, the sign bit can be interpreted independently of the remaining bits, giving the sign-magnitude encoding. This leads to a double representation of the zero as $+0$ and -0. Alternatively, a negative weight of -2^m can be attached to the sign bit, resulting in the well-known two's complement representation. The decimal value of a binary fixed-point number in two's complement is

$$v = -d_m 2^m + \sum_{i=0}^{m-1} d_i 2^i + \sum_{f=-n}^{-1} d_f 2^{-f}\,. \tag{3.11}$$

This formula consists of two distinct sums to emphasise the contributions of the integer and fractional parts, although it could be combined in one term.

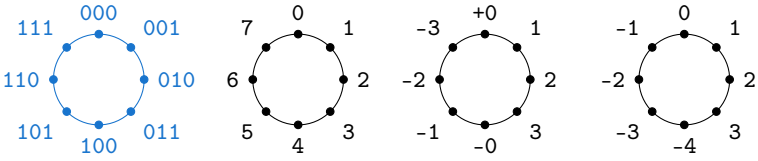


Figure 3.6: Number circles for different integer encodings of the 3 bit words shown in (a): (b) unsigned, (c) sign-magnitude, and (d) two's complement.

3.2.2.1 Properties of Fixed-Point Numbers

Since fixed-point numbers are of finite word length as floating-point numbers are, they share some properties, like limited ranges or number rounding, albeit with distinct differences due to the fixed position of the radix point.

Position of the Binary Point To begin with, the position of the binary point is not encoded within the bit string. For floating-point numbers this is transparently handled by a floating-point ALU. In contrast to this, at every stage of a fixed-point digital architecture there is a specific Q format associated with each numerical bit string, which is not accessible for computations. Therefore, the digital designer must keep track of changes to the radix point position during arithmetic operations.

As it is the case with floating-point numbers, fixed-point numbers represent rational numbers only. Solely the numerator is stored; and the denominator, which must be a power of two in binary systems, is not stored, since the position of the radix point is implicit.

Range and Scaling Contrary to the straight number line of real numbers, fixed-point numbers form a number circle due to the finite word length and the unavailability of a scaling exponent. Fig. 3.6a shows all possible 3 bit words. The results of arithmetic operations can easily overflow and typically wrap around in that case, which leads to totally wrong results. To give an illustration: The addition $111_2 + 1_2 = 1000_2$ gives 000_2 after casting the result to 3 bit again. This wrapping occurs whether

the numeric value is encoded as unsigned natural number (Fig. 3.6b), as a number in sign-magnitude format (Fig. 3.6c) or as two's complement number (Fig. 3.6d). Hence, the representable number range is very limited and the designer must ensure that the expected input and output values fit into the representable number range of the chosen format (see Tab. 3.2).

A common technique is called scaling, which basically is a (repeated) left or right shift of the binary sequence. This corresponds to a division resp. multiplication by two. Sign extension is necessary for numbers given in two's complement, i.e. the former MSB must be copied to the new leading digits if right-shifted.

Overflow and Underflow Handling If an overflow occurs, i.e. the result of an operation is above the maximal or below the minimal representable value, the overflow bit cannot be stored and the number wraps around—a behaviour highly undesired in signal processing. It does not only cause a low signal-to-noise ratio (SNR) or a highly corrupted signal, it can also destabilise recursive digital filters due to the introduced non-linearity [KK06]. Hence, overflows must be avoided at all costs.

This effect can be mitigated by saturation overflow handling, i.e. the value is clipped to the maximal or minimal value, but this requires an additional condition to be tested for and, therefore, increases the digital logic overhead. Solely sufficient spare bits, i.e. a range extension of the fixed-point number format to accommodate the result, can prevent overflows in the first place.

On the other hand, underflows can occur during arithmetic operations. A simple truncation of a number in two's complement rounds to the closest representable number in the direction of negative infinity. Other rounding methods are, among others and in increasing order of complexity: round-to-zero, round-to-nearest and convergent rounding. The latter is statistically the best, since the only unbiased method.

Underflows are not as detrimental as overflows unless very small magnitudes become rounded identically equal to zero, which might lead to unexpected side effects when subsequent conditional statements test for zero.

Accuracy and Quantisation Noise The quantisation step Δ between two consecutive representable numbers, i.e. ULP in floating-point parlance, is given by the weight assigned to the LSB, which is exactly $\Delta = 2^{-n}$. Other than with floating-point formats, Δ is constant because fixed-point numbers are equidistant within the representable range. For instance, Δ is for 16 bit and 32 bit numbers in fractional Q format 3.1×10^{-5} resp. 4.6×10^{-10}, which is equivalent to 4.2 resp. 9.3 decimal digits on average. When compared to machine epsilon in Tab. 3.1 on page 39, same-length fixed-point numbers provide better accuracy.

Within the context of digital signal processing, the average effect of rounding is commonly referred to as quantisation noise. It acts as an intrinsic noise source which degenerates the SNR [KK06]. For a signed integer or fixed-point number of w bits and uniformly distributed input data within the range of the number format, the power of the quantisation error is

$$\sigma_Q^2 = \frac{2^{-(w-1)}}{12}\,, \tag{3.12}$$

which is a well-known result [Kam08]. Note, that the numerator is identical to Δ for a fixed-point number given in fractional Q format. This, of course, makes a couple of idealistic assumptions and can, therefore, only be a first order approximation.

3.2.2.2 The Q Format

The partition of the total word length into an integer part of m bits and a fractional part of n bits can be denoted with the help of the so-called Q format as $\mathrm{Q}(m,n)$. This notation has been introduced by Texas Instruments in their DSP Library Programmer's Reference, e.g. [Tex00].

Because of its importance to signal processing applications, the Q format commonly denotes signed fixed-point numbers in two's complement. Hence, there is always a sign bit which is not counted as part of the integer part m. The total bit width of a $\mathrm{Q}(m,n)$ number is $w = m+n+1$. The important special case of an all-fractional number $\mathrm{Q}(0,n)$ is also known by the shorthand notation $\mathrm{Q}n$. A fixed-point number in $\mathrm{Q}15$ has a word length of 16 bit for example.

Yet, the Q format itself is not standardised and variations of it can be found within the literature. Sometimes, the sign bit is counted as part of m requiring $m \geq 1$; or the triplet (s,m,n) is given, with $s \in \{0,1\}$ to

denote the existence of a sign bit. Alternatively, prefixed letters "s" or "u" are used for signed respectively unsigned numbers, e.g. "u0.15".

The Q format as defined here and used within this document *unambiguously* defines how a bit string has to be interpreted, and the notation makes it easier for the designer to keep track of changes during arithmetic operations.

Sizing of the Integer and Fractional Parts If a bound $M > 0$ is known such that $|a| \leq M$, then $\lceil \mathrm{ld}(M) \rceil + 1$ bits are required for a signed integer encoding, whereby $\lceil \cdot \rceil$ denotes rounding up to the next integer. Additionally, $\lceil \mathrm{ld}(1/p) \rceil$ fractional bits are required for a target quantisation step $\Delta \leq p$, which follows directly from the definition of Δ. For instance, to meet a numerical precision of $p = 1 \times 10^{-3}$, 10 fractional bits need to be provided.

Consequently, the minimally required Q format for a (closed) interval $[-M, M]$ with a target precision p is

$$a \in [-M, M],\ \Delta \leq p \quad \Rightarrow \quad a \in \mathrm{Q}(\lceil \mathrm{ld}(M) \rceil, \lceil \mathrm{ld}(1/p) \rceil)\,. \tag{3.13}$$

Note, if the open interval $(-M, M)$ is sufficient, i.e. $|a| < M$ is strictly less, one integer bit can be spared.

Arithmetic Operations To retain full precision during arithmetic operations, the Q format of the result needs to be sized sufficiently.

Tab. 3.3 lists several operations, the formats of the operands and the formats of the results. These apply to numbers in two's complement as it is implied by the Q notation. Proofs of these laws can be found in App. A.

The significance of the all-fractional Q format $\mathrm{Q}n$ stems from its properties regarding multiplications. $\mathrm{Q}n$ can represent numbers from the interval $[-1, 1)$ and multiplication results are ensured to lie within this interval again, apart from the corner case $-1 \cdot (-1) = 1$. Consequently, the utilisation of this fractional format is recommendable, because intermediate scaling stages can be omitted which eases fixed-point design a lot. Since many computer architectures are aligned with boundaries being multiples of 8 bit (i.e. bytes), the $\mathrm{Q}15$ format is frequently utilised. The multiplication of two such numbers, $\mathrm{Q}15 \cdot \mathrm{Q}15$, maps to $\mathrm{Q}30$ according to Tab. 3.3, which can be stored in a 32 bit wide register. If the result is truncated again to 16 bit, most probably an underflow will occur, i.e. a

Table 3.3: Resulting Q formats of basic arithmetic operations.

Description	Operation	Result
Left Shift	$\mathrm{Q}(m,n) \ll k$	$\mathrm{Q}(m-k, n+k)$
Right Shift	$\mathrm{Q}(m,n) \gg k$	$\mathrm{Q}(m+k, n-k)$
Addition/Subtraction	$\mathrm{Q}(m_1,n_1) \pm \mathrm{Q}(m_2,n_2)$	$\mathrm{Q}(\max(m_1,m_2)+1, \max(n_1,n_2))$
k-fold Add./Sub.	$\sum_{\ell=1}^{k} \mathrm{Q}(m,n)$	$\mathrm{Q}(m+\lceil \mathrm{ld}(k) \rceil, n)$
Multiplication	$\mathrm{Q}(m_1,n_1) \cdot \mathrm{Q}(m_2,n_2)$	$\mathrm{Q}(m_1+m_2, n_1+n_2)$
Reciprocal	$1/\mathrm{Q}(m,n)$	$\mathrm{Q}(n,m)$
Division	$\mathrm{Q}(m_1,n_1)/\mathrm{Q}(m_2,n_2)$	$\mathrm{Q}(m_1+n_2, n_1+m_2)$
Square Root[a]	$\sqrt{\mathrm{Q}(m,n)}$	$\mathrm{Q}(\lceil m/2 \rceil, \lceil \mathrm{ld}(1/p) \rceil)$

[a] The number of fractional bits of the output format can be chosen arbitrarily, but in order to avoid a loss in precision p the choice for at least n fractional bits is recommendable (see App. A.1.6).

precision loss. This, however, is less critical than an overflow, though undesirable as well.

3.2.3 Discussion

The above explanations of floating-point and fixed-point data types highlight common characteristics these number representations share.

Both categories of formats are inexact because of rounding issues due to finite word lengths. The remarkably huge number range which is covered by floating-point types, while offering substantial precision at the same time, makes them the number one choice for algorithm design and analysis. If HLS is employed, this also carries over to digital architecture design as long as there are plentiful hardware resources available. The latest HLS tools do support floating-point data types for synthesis (see Sec. 2.2.1). Keeping floating-point arithmetic even for digital architecture design can minimise implementation loss.

Basically, the pros and cons of fixed versus floating-point designs presented in [IO96] are still valid. Existing design constraints may require a fixed-point conversion step, namely energy consumption, resource utilisation (area) or the speed of computations to meet timing specifications. In that case, fixed-point numbers are the preferred choice, because of

the relatively simple integer arithmetic. But the non-recurring costs of development also need to be factored in, which are higher for fixed-point designs. Because all of the bits of a fixed-point number contribute to the accuracy of the format, there are niche applications where fixed-point arithmetic can yield better accuracy than same-sized floating-point numbers.

Concerning fixed-point design, the most critical issue is the limited number range. Yet, statistics of the input data are often known and binary scaling can be applied to reduce magnitudes such that the data are representable by a fractional Q format. Subsequently, the rules to Q format changes, as set forth in Tab. 3.3 on page 50, can be followed to track the radix point and perform computations with desired precision. This approach could be described as pseudo floating-point arithmetic, since the position of the binary point does change from operation to operation, although it is known and fixed at every stage of the process. Furthermore, the bit width of fixed-point data types is arbitrarily tunable for the requirements of the signal processing application.

To summarise the above: As long as the magnitudes of the data are restricted to a known interval, floating-point and fixed-point arithmetic are quite similar in their behaviour. This can be exploited to simplify the design flow with HLS.

The observation of [Kae08, page 64] is noteworthy:

> "The effort for finding a good compromise between numerical accuracy and hardware efficiency is often underestimated."

A holistic approach to top-down algorithm implementation should, therefore, encompass all these aspects. This is certainly true for the proposed RDAM.

3.3 The Principal Idea of the RDAM

In Sec. 3.1, it has been recognised that conceptual algorithm design and analyses are usually performed with double-precision floating-point data types. An exchange for single-precision or half-precision data types is unavoidable and mandatory to make the design entry C++ code suitable for HLS (double-precision is not supported or incurs heavy resource

utilisation penalty). Apart from that, a fixed-point conversion step must take place if design constraints dictate.

Therefore, the basic principle of the RDAM is to solely work with one algorithmic description which captures the mathematical structure but is agnostic towards data types. Throughout the implementation process the very same generic source code of the algorithm at hand will be used for each and every design step, just with exchanged parametrisations for data types. To express this in more informal language, the RDAM is a design flow "from complex math to fixed-point hardware in no time".

3.3.1 Advantages

The baseline data type-agnostic algorithmic description effectuates several advantages:

- Code reusability is pronounced. A single C++ source code is reused for simulation and implementation. Neither must the algorithm be ported to another programming language, nor are additional validation steps needed due to such re-implementations.
- The DTA augments the design space exploration capabilities with a new dimension "data type" which translates to a tunable numerical accuracy (see Sec. 3.3.2).
- The methodology also applies to complex-valued algorithms (see Sec. 4.2).
- Digital designs operating with fast and resource-efficient fixed-point numbers can be obtained without much overhead (see Sec. 4.3). The fixed-point conversion step is simplified.
- All the advantages of HLS benefit this methodology, namely the abstracted design perspective, the resulting rapid architecture creation and the test benches for speedy functional verification.

Recapitulating the above, all elements in the name *Rapid Data Type-Agnostic Digital Design Methodology* are accounted for. It is "rapid" because the elevated design perspective of HLS accelerates the design time and eliminates the need for RTL coding; the progression to fixed-point arithmetic is simplified. It is "data type-agnostic" since the complete top-down design flow is based on a single source code exhibiting this feature. It is

"digital" because it targets the synthesis of digital logic circuitry. And it is a "design methodology" which can systematically be applied to and followed along with every algorithm, in the sense of the original greek word *methodos*.

3.3.2 Augmented Design Space Exploration

As it has already been mentioned, a major advantage of HLS is its ability for better design space exploration because of the higher level of abstraction [GR94].

Of key interest are the powerful HLS compiler directives to adjust the degree of parallelism of an architecture. VHLS, e.g., supports the exploration of different architectures, called "solutions", out-of-the-box [Xil-UG902]. While all solutions are based on exactly the same source code, each has a unique set of directives associated with it.

Although architecture design and analysis is at the heart of HLS, it is, however, not trivial with VHLS to explore the influence of other algorithmic parameters. Most importantly, many algorithms have a specific problem size, e.g. the size of a Fast Fourier Transform (FFT) or the length of some input vectors to a function. Such parameters are usually defined in preprocessor macros or as constants, but VHLS does not offer the possibility to change hard-coded values or to compare synthesis results for variations of these.

As schematically shown in Fig. 3.7, a tool chain has been developed which links VHLS and Mathworks Matlab [Kno+16c]. Shell scripts are used to set up an environment which accepts certain user inputs, makes automated calls to both tools in background and stores results. The tool chain helps in rapid architecture evaluation and design of algorithms by exploring different dimensions of the design space, namely

- architectural alternatives,
- different problem sizes, and,
- data types,

the latter thanks to the DTA of the RDAM. Possible data types supported by the proposed DTA are floating-point or fixed-point number formats, either real-valued or complex-valued. In addition to that, further HLS

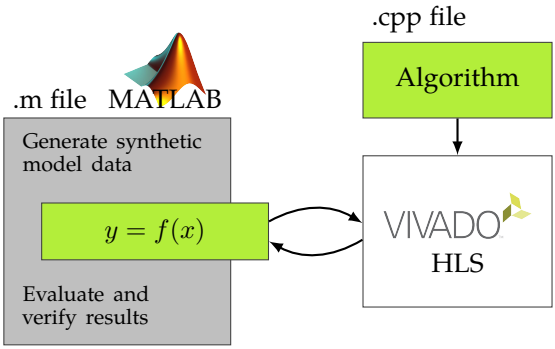


Figure 3.7: A link between Matlab and VHLS enables bit-accurate simulations which can be integrated into a model for verification and validation purposes [Kno+16c].

parameters could be varied, e.g. the target device or target clock frequency.

A data model in Matlab generates a configurable number of inputs and expected outputs to the HLS design at hand, e.g. random vectors, to supply simulation stimuli for verification and test purposes.

The average numerical accuracy of the HLS results, as well as other user-defined metrics, can then be compared, evaluated and averaged over many simulation runs. Hence, the numerical accuracy measured at an output interface can be taken into account during the design process, e.g. expressed by the normalised mean squared error (NMSE) which is defined as

$$\mathrm{NMSE}(\mathbf{x}, \hat{\mathbf{x}}) = \frac{\|\mathbf{x} - \hat{\mathbf{x}}\|^2}{\|\mathbf{x}\|^2}, \tag{3.14}$$

with $\hat{\mathbf{x}}$ being an approximation to $\mathbf{x}$. This is inspired by the ideas of approximate computing, where loosened accuracy constraints open up the possibility for the development of simpler, more hardware-efficient designs (see Sec. 4.4). This endows the designer with another option for trade-off.

Besides classical hardware design metrics—like resource utilisation, energy consumption, latency or data throughput—the numerical accuracy can serve as an additional figure of merit. Fig. 3.8 illustrates this

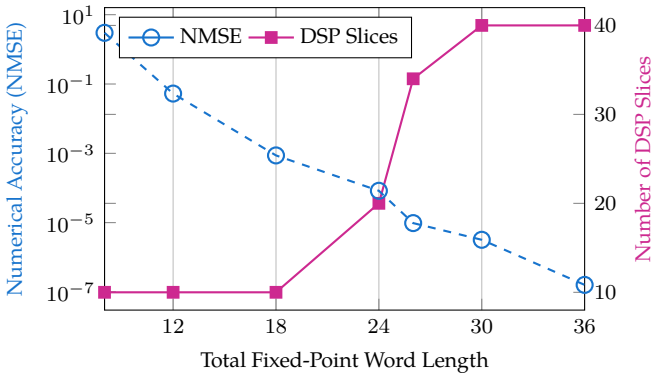


Figure 3.8: Resource utilisation for varying fixed-point word length, which is a configurable parameter with the proposed methodology. This allows for an optimal and hardware-efficient trade-off between resource utilisation (here: DSP slice count) and achieved numerical accuracy (here: NMSE), [Kno+16c].

fact [Kno+16c]. It shows a parametric sweep of the fixed-point word length (abscissa) used as a baseline data type employed for the implementation of the ACGP algorithm [BD08; BDR12], and compares the evaluated numerical accuracy of the output vector with the resource utilisation of the HLS synthesis result. Asking for minimal area, a word length of 18 bits would be the Pareto-optimal choice if an NMSE of 1×10^{-3} is tolerable.

Further details on the specifics of the implementation can be found in [Kno+16c]. Note, however, that a single baseline fixed-point format was used throughout the whole architecture and necessary Q format conversions were handled automatically to a great extent by specialised C++ data types.

3.3.3 Prerequisites and Limitations

Although hardware design gains some kind of a "software feeling" because of HLS, nonetheless deepened hardware expertise is still needed to create efficient architectures. In particular, the concept of concurrency

need to be well understood to identify portions of the sequential design entry code which can be computed in parallel.

The RDAM is not restricted to domain-specific applications. It allows for the implementation of all algorithms in general which are expressed by C++ code. Hence, the realisation of computationally intensive algorithms is possible, consisting of control logic as well as optimised data paths.

Implicit to HLS is the limitation to synchronous logic synthesis. Asynchronous designs are, therefore, out of the scope of the RDAM.

Furthermore, the RDAM mainly targets FPGAs. FPGAs offer rapid design cycles and in-system tests [Con+11]. VLSI designs for ASICs are possible as well, though not covered within this thesis.

3.3.3.1 Prior Algorithmic Transformations

Not every mathematical algorithm is suited for hardware implementation in the first place. Mathematically equivalent algorithmic transformations are often required beforehand to tailor it accordingly and facilitate implementation.

Basically, there a two main points which must be taken into consideration:

- The algorithm might contain complicated mathematical functions which defy a simple and fast mapping to MAC operations.

  Code libraries with explicit support for HLS provided by the tool vendors help a lot. Xilinx, e.g., supplies ready-to-use solutions for trigonometric and transcendental functions and a CORDIC implementation [Xil-UG902].

  Alternatively, polynomial function approximations and look-up tables could be utilised (see Sec. 4.4).

- Sub-algorithms are needed to perform certain computational tasks.

  This is evident in the case of QRD, which can be performed either by Householder reflections, (modified) Gram-Schmidt orthogonalisation, or Given's rotations—each having pros and cons [GV13]. The choice of a procedure for implementation lies with the engineer.

A design example which illustrates both points will be given in Sec. 5.2.

3.3.3.2 The Subset of Synthesisable Code

The RDAM solely supports C++ as design entry programming language. This is inherited from the support of the HLS tools and the mechanisms of the language utilised for data type abstraction (see Sec. 2.5 and 4.1.2). SystemC, which is in fact a class library to C++, can be used partly, foremost with regard to the provided fixed-point data types.

After the mathematical functional specification has been mapped to an algorithmic equivalent ready for implementation, another limitation of HLS has to be considered, namely that HLS tools synthesise a reasonable subset of the C/C++ languages only.

Most importantly, there is no operating system for memory management on FPGAs or ASICs. That makes it mandatory to code the HLS design with static memory allocations to obtain well-defined register or vector sizes at compile time. Although the exact sizes must be known at compile time and cannot change later, i.e. a runtime configuration is not possible, sizes may be configurable (but constant) parameters.

The designer has to adhere to the HLS coding style guide of the HLS tool as outlined in Sec. 2.3.1.1. This could necessitate the rewriting of non-synthesisable existing C++ code. The associated HLS test bench, however, helps to ensure functional equivalence.

3.4 Methodological Design Flow

In conclusion of this chapter, the methodological design flow of the Rapid Data Type-Agnostic Digital Design Methodology (RDAM) can be summarised as follows:

1. Apply equivalent algorithmic transformations to the algorithm at hand to reduce computational complexity or facilitate implementation in the first place.
2. Create a purely data type-agnostic C++ source code for HLS design entry. (This will be explained in detail in Chap. 4.)
3. Verify and validate the C++ code with parametrisation for floating-point accuracy against a possibly existing algorithmic model in, e.g., Mathworks Matlab.

4. (Optional.) Exchange data type parametrisations for fixed-point arithmetic to obtain resource-efficient and fast designs. A self-checking HLS test bench automatically ensures and guarantees functional validity.
5. Evaluate architectural design options and the resulting numerical accuracies using the extensive design space exploration capabilities of HLS with added DTA.
6. Synthesise and package the design as an IP core for implementation with the HLS tool of choice.

Some aspects of this top-down design flow will be described in further detail below. Also, some improvements will be proposed to smoothen this process and overcome certain shortcomings of the HLS tools.

Chapter 4

Data Type Agnosticism for High-Level Synthesis

At the heart of the RDAM are data type-agnostic design sources. This, however, is not a novel concept in itself. In fact, it is a consequent exploitation of the polymorphisms, which an object-oriented programming language, like C++, offers.

The basic idea of DTA for HLS was derived from the Fixed-Point Designer, which can be acquired as an additional toolbox to Mathworks Matlab. It defines the `fi` object to model fixed-point numbers; and existing m-code functions, i.e. functions written in Matlab's own object-oriented programming language, can be called either with standard double-precision floating-point numbers or with `fi` objects as function arguments. Extensive configuration possibilities exist to influence the fixed-point number formats and their arithmetic behaviour. In other words, the data type of the function arguments switches between floating-point or fixed-point operation.

Likewise, [Cro+14] mentions as an incidental remark that floating-point data types (data types) can be used for functional verification of HLS design sources and later be exchanged for fixed-point data types. The RDAM builds upon these ideas and develops them into a mature design methodology to span the complete top-down design flow from the mathematical specification down to the register-transfer level.

Within this chapter, the DTA used by the RDAM is explained and exemplified by a small design example. Afterwards, its significance to complex-valued and fixed-point designs is discussed. The chapter will conclude with a section on function approximations to facilitate

fixed-point arithmetic even for mathematically complex expressions in connection with HLS.

4.1 Data Type-Agnostic Design Sources

The DTA for HLS heavily relies on the capabilities of the C++ programming language, i.e. the different types of polymorphism. This is also supported by a couple of HLS tools, as listed in Tab. 2.2 (on page 31).

The RDAM proposes to keep the algorithmic description of the C++ source code generic with regard to data types. This can be achieved with the help of the `typedef` keyword. In other words, the design entry source code shall *exclusively* make use of custom-declared data types. This offers the advantage that data type definitions can easily be exchanged thereafter.

To allow for a more fine-grained control, several specialised `typedef` declarations can be made according to the use case, e.g. `data_t` for data elements, `io_t` for I/O interfacing, `accum_t` for counters and accumulators, etc.

The following section will provide an illustrative example.

4.1.1 Example Source Codes

Listing 4.1 is a small data type-agnostic code example which implements a sum of squares to illustrate the RDAM. Note, nowhere in this code listing do occur standard C++ data types. Furthermore, data might be complex-valued.

The custom data type declarations have to be specified. These are to be found within the included header file listed in Listing 4.2, where a preprocessor macro definition `DATATYPE` is used to select a specific parametrisation. Parameters to the algorithms are defined as well, e.g. in line 9 of the header file for the length of the input vectors. Thus, all vector sizes are known at compile time and, hence, fit for HLS.

The cpp and h files together are the product of Step 2 of the RDAM (see page 57).

In Sec. 3.1 it has been observed that data types do change during the top-down hardware design of an algorithm, but not the mathematical behaviour of the algorithm itself. Thus, the data type-agnostic code

Code Listing 4.1: The data type-agnostic HLS design file `correlation.cpp` implementing a (squared) scalar vector product.

```
#include "correlation.h" // custom data types, sizes

result_t correlation( data_t a[N], data_t b[N] )
{
    accum_t sum = 0;
    for( counter_t n = 0; n < N; ++n )
    {
        sum += conj( a[n] ) * b[n];
    }
    return sum;
}
```

can now be parametrised for code verification and validation, following Step 3 of the RDAM. To accomplish this, all custom data types are defined as floating-point variables (`double`) or, where appropriate, as signed/unsigned integer variables (`int` or alike).

Subsequently, Step 4 of the RDAM will introduce fixed-point numbers into the design, if desired. These can efficiently be implemented with the help of the SystemC classes `sc_int` and `sc_fixed` or their unsigned variants. In connection with the RDAM, the `sc_fixed` class is of greatest importance and especially well-suited. This will be discussed in further detail in Sec. 4.3.

The example continues with the design space exploration according to Step 5 of the RDAM in Sec. 5.1 (page 75).

4.1.2 Employed C++ Mechanisms

The polymorphisms of the C++ design language bring the DTA to life. There are mainly two key mechanisms at work.

- C++ allows for operator overloading, i.e. operators can be extended in their definition but keep their meaning. To illustrate this with a tiny example, even the simple assignment

  ```
  std::complex< float > z = 3.0;  // 3.0 + 0.0i
  ```

Code Listing 4.2: Header file `correlation.h` defining all custom data types and other algorithmic parameters.

```
#ifndef CORRELATION_H
#define CORRELATION_H

// include data type classes
#include <complex>
#include <ap_fixed.h>

// algorithmic parameters
#define N  32
#define ACCURACY_THRESHOLD  1e-2

// data type definitions and selection
#define DATATYPE 1
typedef unsigned char counter_t;

#if DATATYPE == 0
// float
typedef float data_t;
typedef float result_t;
typedef float accum_t;

#elif DATATYPE == 1
// std::complex<float>
typedef std::complex<float> data_t;
typedef std::complex<float> result_t;
typedef std::complex<float> accum_t;
using std::conj;

#elif DATATYPE == 2
// fixed-point like C's short
typedef ap_fixed<16, 3> data_t;
typedef ap_fixed<16, 3> result_t;
typedef ap_fixed<16, 3> accum_t;

#elif DATATYPE == 3
// std::complex< ap_fixed<W,I,Q,O> >
typedef std::complex< ap_fixed<18, 4, AP_RND,
    AP_SAT> > data_t;
```

Code Listing 4.2 (cont.): Header file `correlation.h`.

```
38 typedef std::complex< ap_fixed<18, 4, AP_RND,
       AP_SAT> > result_t;
39 typedef std::complex< ap_fixed<18, 4, AP_RND,
       AP_SAT> > accum_t;
40 using std::conj;
41
42 #elif DATATYPE == 4
43 ...
44 #else
45 #error "Define data type"
46 #endif // DATATYPE
47
48 // fallback for complex<float> to float
49 template <class T> T conj(T x) { return x; }
50
51 #endif // CORRELATION_H
```

is internally overloaded by the `std::complex` class to initialise the real part as well as the imaginary part of `z`.

- Class and function templates allow for generic data structures not only with constant parameters but also with arbitrary data types, i.e. templates are inherently data type-agnostic. For example,

  ```
  template <typename T>
  T min ( T x, T y ) {
      return x < y ? x : y;
  }
  ```

  defines a templatised function `min` which can be applied to every data type `T` that supports the relational operator "<". This is per default the case for all integer and real number types. The compiler checks the existence of the required overloads.

Additionally, preprocessor macros allow for a conditional compilation of source codes. They are common to C and all derived dialects. In Listing 4.2, the `DATATYPE` macro was used to switch between data type definitions. Nonetheless, such macros circumvent the semantic checks of the compiler and are therefore more prone to coding errors.

The application of the RDAM brings simplifications to at least two domains which traditionally require additional design time and deepened hardware expertise: complex-valued data paths and fixed-point design. These topics will be addressed in the following sections.

4.2 The Application to Complex-Valued Designs

The function `conj()` in Listing 4.1, line 8, implies a complex-valued operation of the code, because the complex conjugate is simply undefined for real numbers. It is imported into the namespace for complex-valued data type parametrisations within the header file, e.g. for the case of the complex-valued single-precision floating-point data type in Listing 4.2, line 27.

Since the field of complex numbers is a superset to real numbers, it is advisable, with regard to the RDAM, to create source code for the most general case, i.e. complex numbers. This might be viewed as a defect to a truly agnostic source code, but, however, at minimal cost compared to the potential gain in design time.

The `complex` class of the C++ Standard Library (namespace `std`) is supported for synthesis, e.g. by VHLS amongst other tools. Data type agnosticism mandates that every called function must be well-defined for the complex-valued case `std::complex<T>` as well as the real-valued case `T`, with `T` denoting any C++ data type. However, if the code is compiled for real-valued operation, the static functions of the `complex` class are undefined and must be made known to the compiler. This can quickly be accomplished by a graceful fallback strategy which redefines the function for every real-valued data type with an empty function body.

For example, the fallback solution for the `std::conj()` function can be implemented with a single line of code, as shown in Listing 4.2, in line 49. Similar functions could be added for other static function members of `std::complex`, if needed. The implemented fallback function immediately returns, skipping conjugacy, and is well-defined for every compilation with a real-valued data type parametrisation of the design code. The compiler will remove such additional function calls during the optimisation stage, and thus no penalty whatsoever is incurred.

The relational operator "<", mentioned in the previous section, could be overloaded and extended in its definition to apply to the magnitude of complex numbers, which normally have no natural ordering. Alike, the `complex` class implements complex addition, multiplication, division, etc., which are well-defined operations.

As a side note: One complex-valued multiplication normally consists of four real-valued multiplications and two real-valued additions/subtractions. A designer can opt, however, to apply Gauss' complex multiplication formula to reduce the number of multiplications to three while increasing the number of additions/subtractions to five [Knu02]. This could be implemented by another function template. Thereby a shift between design resources is possible, e.g. if the utilisation count of DSP slices with dedicated hardware multipliers is a critical design constraint.

Altogether, the proposed RDAM can automate even the otherwise necessary rewriting of complex-valued code with real-valued entities, i.e. it encapsulates the real-value decomposition (RVD) of complex-valued operations. An exemplary application of this feature is presented in Sec. 5.2.

4.3 The Abstraction of Fixed-Point Design

In order to obtain a slim digital hardware with small area footprint or faster execution times, computations should certainly be performed in a fixed-point data format (see the discussion in Sec. 3.2.3). Hence, the transition from floating-point to fixed-point arithmetic is explicitly supported by the RDAM as an optional step.

As it is the case for complex-valued data items, classes for fixed-point numbers encapsulate specialised functionality to a great extent.

The `sc_fixed` class of SystemC, mentioned above in Sec. 4.1.1, is essential for the RDAM. It does not only support arbitrary bit widths of Q format fixed-point numbers (see Sec. 3.2.2.2), but also provides overloaded binary, arithmetic and relational operators as well as conversion methods. The alignment of the fractional parts is automated and fixed-point multiplication intricacies, therefore, transparently handled, cleanly encapsulated by the object-oriented programming paradigm. Furthermore, options concerning number rounding and saturation arithmetic are available.

Basically, a `sc_fixed` variable can be declared from a template with a quintuple (W, I, Q, O, N), where W is the word length in bits, I the number of bits for the integer part including the sign bit if applicable (i.e. $W - I$ bits are for the fractional part), Q is a flag to control quantisation (rounding modes), O is a flag to control overflows (e.g. saturation or wrapping) and N defines the number of saturation bits in overflow wrap modes.

Hence, `sc_fixed` is a most versatile class and it subsumes the functionality of C++ standard integer types and SystemC's `sc_int` as special cases. If parametrised correspondingly, the added functional capacity does not incur any performance penalties to the HLS results.

Some HLS tools introduce their own fixed-point classes derived from `sc_fixed`. These are Xilinx' Vivado HLS (VHLS) (the class prefix being `ap` instead of `sc`) or Mentor Graphics' Catapult-C (class prefix `ac`). These derived classes basically are functionally equivalent, but additionally ensure consistent bit-accurate RTL simulations with C++ and improved synthesis results [Xil-UG902; FB10].

4.3.1 Unavoidable Manual Intervention

With regard to the proposed design methodology, manual intervention by the designer is necessary at least twice.

Firstly, the magnitudes of the data must fit into the representable number interval. A common and simple technique to perform this is the binary scaling of the inputs. As long as the data are properly scaled, which can be tested from within the HLS test bench, the specialised fixed-point classes can, from there on, take care of the intricacies of fixed-point arithmetic, especially the tracking of the Q format changes (see Sec. 3.2.2.2).

Secondly, overflows must be avoided by defining fixed-point data types with large enough bit widths. A single baseline definition will rarely be sufficient for a large design, which motivates the introduction of additional types with more bits for intermediate computations at higher precisions until the final result might be type casted to less bits again.

To utilise a baseline bit width and a larger bit width for intermediate computations is a general design recommendation. The source code examples in Listings 4.1 and 4.2 make use of three custom-declared data types `data_t`, `accum_t` and `result_t` for this reason. (The example is,

however, plain and simple enough not to require this degree of fine tuning. The definitions of these three types are identical and given for didactic purposes only.)

4.3.2 Quick Test-Driven Dimensioning of the Fixed-Point Word Lengths

Test-driven development normally applies to software engineering where verification tests are written first and then implementation follows until all tests are successfully fulfilled. Since HLS provides kind of a software feeling to hardware design, it naturally occurs to reuse this concept to validate the overall functionality of the resulting fixed-point design.

The self-checking nature of the HLS test bench associated with each HLS project inherently supports this concept. It indicates if a chosen fixed-point number format leads to correct results and meets the specified requirements, foremost with regard to overall numerical accuracy. Please note line 10 in Listing 4.2 where a numerical accuracy threshold is defined. It is up to the user to interpret its meaning. In the given example, it is a *minimally* required NMSE of the result for the test. The actual numerical accuracy can be better though.

The last but most important issue related to fixed-point designs is the choice of the word lengths. Here, the efficacy of the augmented design space exploration according to Sec. 3.3.2 can be harnessed by the RDAM. Varying fixed-point bit widths $w = m + n$ and Q formats $\mathrm{Q}(m, n)$ can be explored simply by changing the number of integer and/or fractional bits of the `typedef` declarations within the header file. These changes will be propagated throughout the whole design.

The numerical performance of the algorithm can quickly be assessed with the help of the bit-accurate HLS test bench and it allows for a trade-off between numerical accuracy and hardware resource utilisation. As mentioned earlier, a design choice can be made which is Pareto-optimal (see Fig. 3.8 on page 55).

The design abstraction of the RDAM may not lead to the most efficient design and is therefore rather applicable to rapid prototyping purposes. But still, a fixed-point design can successfully be created

1. with far less time spent on implementation details and
2. with ensured overall functionality and known numerical accuracy.

That the results of the proposed methodology can indeed very well compete with hand-crafted RTL designs will be shown in Chap. 5.

4.4 Function Approximations to Improve HLS Results

Sophisticated signal processing applications often require the computation of mathematically complex operations. Many algorithms can elegantly be expressed with the help of linear algebra and that usually includes vector normalisations, matrix factorisations, singular value decompositions (SVDs), etc. Yet, only additions, subtractions or multiplications are straightforward to implement in a fixed-point digital hardware. Every effort is generally spent to avoid the computation of full divisions. Other non-trivial functions are, to name a few, $\ln(x)$, $\exp(x)$, $1/x$, $\sqrt{x}$ or $1/\sqrt{x}$.

For this reason amongst others, Step 1 of the proposed methodology are algorithmic transformations to evade such difficult operations. Even so, this might not always be possible.

Modern FPGAs are also optimised for floating-point designs, as mentioned above in Sec. 2.2.1. This applies to the HLS tools as well. To give an example, VHLS will fall back to floating-point computations if a mathematical expression cannot be evaluated in conjunction with fixed-point data types.

Specialised HLS libraries are available for a couple of non-trivial mathematical functions. The Unified CORDIC algorithm is very often the method of choice for trigonometric and transcendental functions [And98]. This necessitates a complete reimplementation of the CORDIC for HLS, which is available as a C++ library in VHLS for example. Nonetheless, the iterative digit-by-digit operation of the CORDIC introduces some latency into the design.

If a specialised fixed-point library is not available (or the design sources make no use of it), however, VHLS will insert a fixed-to-floating-point conversion, perform the calculations with single-precision `float`, and then convert the results back to the specified fixed-point data types. The advantage is obvious: The HLS compiler can synthesise a design for

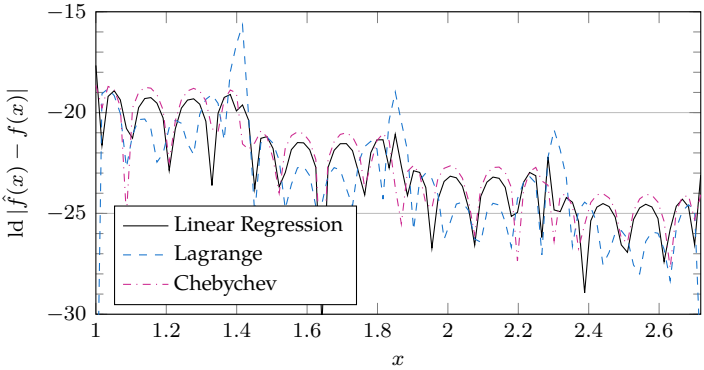


Figure 4.1: Three polynomial approximations of $\ln(x)$ are compared with each other. The ordinate tells the number of correct bits of the approximation.

almost every expression—but the latency introduced hereby is by no means negligible.

An alternative way to overcome this bottleneck are function approximations because they can efficiently be mapped onto the available resource blocks of an FPGA and also work in conjunction with fixed-point arithmetic.

4.4.1 Polynomial Function Approximation

A simple way to approximate a function $f : \mathcal{X} \to \mathcal{Y}$ where $\mathcal{X}, \mathcal{Y} \subset \mathbb{R}$ is to use a polynomial

$$\hat{f}_T(x) = \sum_{t=0}^{T} a_t x^t \tag{4.1}$$

of a certain degree T. The advantage of polynomials is their easy to compute nature with MAC operations, which are readily available on FPGAs. The quality of the approximation can be adjusted by the degree. Preferable are piecewise smooth functions, but even if there are discontinuities a case differentiation or segmentation can lead to quite acceptable results.

Table 4.1: Synthesis results for the conversion function to and from the LNS with Q15 fixed-point numbers.

Function	Sec.	Cycles	DSP	FF	LUT	Accuracy
LOG	4.4.1	4	4	121	72	5.5×10^{-5}
ALOG	4.4.1	3	3	83	51	3.8×10^{-6}
LOG	4.4.2	3	1	182	1619	6.1×10^{-5}
ALOG	4.4.2	4	1	283	2059	6.1×10^{-5}

There are several possibilities to determine the polynomial coefficients a_t; to name a few [Mul06]:

- LS linear regression,
- Chebychev approximation, or
- Lagrangian interpolation.

All these methods are compared in Fig. 4.1 for $f : [1, \exp(1)] \to [0, 1], x \mapsto \ln(x)$. The domain of the function is split into 4 equispaced segments, each being approximated by a polynomial of order $T = 4$. The difference between the approximation $\hat{f}$ and the original function is expressed by the logarithmus dualis, i.e. the absolute value of ordinate tells the number of correct bits of the approximation. In this case, 16 correct bits and more can be achieved.

The quality of the approximation can be tuned by the order of the polynomials T or the number of segments S. Then, $S(T + 1)$ coefficients have to be computed. This can be carried out *at compile time* of the HLS synthesis process or beforehand by some mathematical numerical software package, like Mathworks Matlab. In either case, the coefficients will be stored in a static array in C++, which is recognised as read-only by HLS and thus implemented as a read-only memory (ROM).

The evaluation for a specific input value x will first look-up the coefficients of the corresponding segment (comparator and multiplexer) and then compute $f(x)$ with MACs. The area-latency trade-off can be steered by HLS directives. For a sequential architecture and bit widths of up to 18 bits a single DSP slice will be utilised.

Tab. 4.1 lists synthesis results for the LNS conversion functions LOG and ALOG, given in (3.8), Sec. 3.2.1.3. The top half shows the results for polynomial approximations of 4-th resp. 3-rd order and no segmentation, as described within this section. The stated accuracy is the mean absolute deviation from the ground truth (computed with double-precision floating-point accuracy) over the domain of the functions. The utilised fixed-point number format is $\mathrm{Q15}$, i.e. 16 bit wide integer arithmetic. The other given synthesis results apply to a Xilinx Zynq-7 (XC7Z020CLG484-1) and 100 MHz target operation frequency. DSP, FF and LUT denote DSP48 slices, flip-flops and look-up tables, respectively. The latency is expressed in clock cycles.

It can be concluded that polynomial approximations are an efficient solution to enhance HLS synthesis. The evaluation of (4.1) is simple to implement with C++ (a loop with a single line loop body), even for fixed-point arithmetic using the specialised classes introduced above.

4.4.2 Piecewise-Linear Function Approximation

Non-Uniform Piecewise-Linear Function Approximation (NPA) is another very efficient means to implement elementary functions, as presented in [RLP13].

The original function f is approximated by a number of consecutive, linear segments in the form of $f_i = \alpha_i x + \beta_i$, where α_i and β_i are the gradient and offset of the i-th segment. To achieve a certain target accuracy, the number of segments is adjustable. The segmentation is non-uniform and whenever a segment fails to meet the required quality of approximation, it is divided into two half-sized segments. The accuracy test is continued recursively until all generated segments fulfil the target accuracy. This creates a binary search tree of varying depth.

The evaluation for a specific input value x is a case differentiation to determine the applicable interval based on the most significant bits of the input (because of the binary search), followed by a single MAC operation. Hence, the generated architecture mainly consists of multiplexers and a ROM to store the α coefficients and β offsets. Again, the segmentation and the determination of the corresponding coefficients of the NPA can be performed at compile time, or included as precomputed constants in a C++ header file.

The bottom half of Tab. 4.1 lists synthesis results for the LNS conversion with the help of NPA. The LOG and ALOG functions were approximated by 43 resp. 51 linear segments. A C++ header file was automatically generated and two functions compute the conversion to and from the LNS. These can be included in any HLS design requiring such a conversion.

In conclusion, the NPA is very resource-efficient concerning the number of utilised DSP slices. The multiplexer structure, implemented as if-clauses in C++, and the somewhat greater number of segments requires a larger ROM to store all coefficients, hence the increased LUT count compared to the previously presented method.

4.4.3 Application of the Logarithmic Number System

A third alternative to implement functions other than MAC operations is to perform these calculations directly within the LNS (see Sec. 3.2.1.3). This exploits the logarithmic identities and reduces multiplication, division, exponentiation and the square root to addition, subtraction, multiplication and a bit shift, respectively.

The LNS is, therefore, especially useful for mathematical expressions, which consist of divisions or exponentiations. The computation takes place in three steps:

1. Conversion of the fixed-point number to the LNS with the LOG function.
2. Perform complexity-reduced arithmetic operations within the LNS.
3. Conversion of the result back to the original Q format by ALOG.

To illustrate this process, the reciprocal square root

$$\mathrm{INVSQRT}(x) = \frac{1}{\sqrt{x}} \tag{4.2}$$

shall serve as example. Due to the application of the LNS, the actual sequence of computations is

$$\mathrm{INVSQRT}(x) = \mathrm{ALOG}\left(\mathrm{LOG}(1) - (\mathrm{LOG}(x) \gg 1)\right) . \tag{4.3}$$

Here, $\gg$ denotes a bit shift to the right in infix notation. $\mathrm{LOG}(1)$ is a constant and evaluates to one, see (3.8a) on page 43. The latency and

Table 4.2: Synthesis results for the reciprocal square root operation.

Type	Eq.	Format	Cycles	DSP	FF	LUT	Accuracy
Normal	(4.2)	`double`	70	0	5815	7167	—
Normal	(4.2)	`float`	37	0	1903	2950	—
LNS	(4.3)	$\mathrm{Q}(6,13)$	11	2	591	4688	7.5×10^{-5}

numerical accuracy of the INVSQRT function is primarily influenced by the LNS conversion functions.

A working HLS implementation of the INVSQRT function was created following the outlined 3-step process. The LNS conversion was performed NPA-based, with the accuracies and FPGA utilisation footprints as stated in Tab. 4.1. The digital synthesis results of the overall design for INVSQRT are given in Tab. 4.2. The inverse square root can be computed with two DSP slices in 11 clock cycles only. The mean accuracy over the domain of definition is 7.5×10^{-5} compared against a floating-point software model. The application of the LNS and NPA is recommendable, because otherwise VHLS will infer costly double-precision floating-point arithmetic for the code expression `1/sqrt(x)`, which takes 6.4 times the cycles to compute. The analysis perspective of the VHLS tool breaks this down into 6 cycles for signed integer to floating-point conversion and 31 cycles each for both, the square root and consecutive division, operations. Nonetheless, even the floating-point variant is synthesisable and a manual rewriting to target single-precision floating-point computations would pose a meet-in-the-middle solution.

Still, the introduction of the LNS, together with the previously discussed conversion functions, is the only way to speed up the design further and to allow for an all-fixed-point operation. To the proposed RDAM, the LNS or one of the presented function approximations are turnkey-ready building blocks which can be applied if design constraints necessitate such measures.

Chapter 5

Design Examples with the Rapid Data Type-Agnostic Digital Design Methodology

Two example algorithms will be implemented along the lines of the RDAM in the following to emphasise some aspects concerning the methodological advantages or implementation-specific issues as they have been discussed theoretically so far.

A first, small-sized computation of a scalar vector product, which basically is a sum of squares, practically explores the—due to the DTA—enlarged design space. A RDAM design recommendation pertaining suitable data types will be substantiated.

Thereupon, Orthogonal Matching Pursuit (OMP) represents the case of a computationally complex algorithm. All steps of the RDAM will carefully be considered, in addition to the proposal of a novel algorithmic transformation with complete rank-1 updating of the nested LS optimisation step. The first design example will constitute an integral part of the OMP architecture. Finally, the obtained IP core will be tested with HIL simulations.

Further applications of the RDAM can be found in Part II of this thesis, namely a Sphere Decoder (Sec. 7.2.1.2) and a Neyman–Pearson frame detector (Sec. 8.3.2).

5.1 Example 1: The Scalar Vector Product

As a first introductory example, the computation of a scalar vector product (also known as inner product or dot product) shall function as a toy prob-

lem. This basically continues the data type-agnostic source code example given in Sec. 4.1.1 and complements its discussion with a complete evaluation of the design space for several data types.

5.1.1 The Algorithm

On the system level of the Gajski–Kuhn chart, the functional view is the mathematical specification given as

$$c = \langle \mathbf{a}, \mathbf{b} \rangle \,, \tag{5.1}$$

with $\mathbf{a}, \mathbf{b}$ being two vectors of the same length N. Note, that this equation is itself agnostic to the fact whether $\mathbf{a}, \mathbf{b}$ are complex-valued or real-valued. If $\mathbf{a}, \mathbf{b} \in \mathbb{C}$, it is $c = \mathbf{a}^H \mathbf{b}$, or, expressed with scalar entities,

$$c = \sum_{n=1}^{N} a_n^* \cdot b_n \,. \tag{5.2}$$

If otherwise real-valued, the Hermitian $(\cdot)^H$ becomes a transposition $(\cdot)^T$, the sole difference being the complex conjugation of the elements of $\mathbf{a}$. The scalar product basically constitutes a sequence of N MACs.

The data type-agnostic HLS design source code implementing (5.2) according to Step 2 of the proposed RDAM (see page 57) is the code in Listings 4.1 and 4.2.

5.1.2 Design Space Exploration with Various Data Types

Moving on, Steps 3 and 4 of the RDAM succeed: floating-point and fixed-point data type parametrisations. The examined data types were the following:

- 32 bit single-precision floating-point `float`,
- 16 bit half-precision floating-point `half`, which ships with Xilinx VHLS as a specialised library,
- 16 bit C++'s integer `short`,
- Xilinx' arbitrary bit width integer types `ap_int`, and

- Xilinx' re-implementation of the `sc_fixed` fixed-point data type, `ap_fixed`.

Additionally, each of these data types was used in conjunction with complex-valued types (see Sec. 4.2) as `T`, namely

- `std::complex<T>`, as it is part of the C++ Standard Library, and
- `hls::x_complex<T>`, which again is a re-implementation by Xilinx.

5.1.2.1 Discussion of the Synthesis Results

Xilinx VHLS (version 2014.2) was employed for synthesis and design space exploration targeting the FPGA fabric of a Zynq-7020. The HLS code was synthesised without any explicit directives given to the compiler, i.e. the default optimisation for minimal resource utilisation has automatically been applied.

Tab. 5.1 compactly lists all important synthesis results and performance metrics generated by the test bench. N was set to 10, which explains that the total latency, measured in cycles at 100 MHz target design frequency, is 10 times the loop latency (plus one cycle to check the loop condition). Resource utilisation is given as the quadruple $(\mathrm{BRAM}, \mathrm{DSP}, \mathrm{FF}, \mathrm{LUT})$, where the column for is omitted since none was instantiated.

Also, some data type parametrisations (rows) are omitted because almost identical results were measured. This is true for all `hls::x_complex` cases compared to `std::complex` as well as for the C++ integer `short` in $\mathrm{Q}(2, 13)$ in comparison to `ap_fixed<16,3>`. The asterisk denotes cases in which saturation overflow handling and number rounding to-the-nearest were enabled.

Numerical Accuracy The achieved numerical accuracy (mean and standard deviation) is averaged over 100 simulation runs and compared against a golden software model with double floating-point precision. Since the input data are randomly generated, the correlation result is subject to variations and data type-specific impairments, e.g. underflows.

Notably, all measured accuracies are close to what is maximally attainable per data type. For instance, machine epsilon for `float` is 1.2×10^{-7} (see Tab. 3.1) which agrees very well with the measured numbers[1]. Sim-

[1] Note, machine epsilon is a worst-case bound of the actual floating-point precision ULP. Therefore the measured accuracy can be even slightly better.

Table 5.1: Resource utilisation and performance figures of a scalar vector product for various data types.

	Datatype			Resource Utilisation			Latency		Performance Metrics	
	$\mathbb{K}$	`T`	$\mathrm{Q}(m,n)$	**DSP**	**FF**	**LUT**	**Loop**	**Total**	**M/s**	**Accuracy**
(1)	$\mathbb{R}$	`float`	—	5	628	806	16	161	1.2	$(5.8\pm4.9)\times10^{-8}$
(2)	$\mathbb{C}$	`float`	—	16	1670	2450	25	251	0.8	$(1.5\pm0.9)\times10^{-7}$
(3)	$\mathbb{R}$	`half`	—	could not be synthesised						$(4.4\pm4.5)\times10^{-4}$
(4)	$\mathbb{C}$	`half`	—	could not be synthesised						$(1.1\pm0.7)\times10^{-3}$
(5)	$\mathbb{R}$	`ap_fixed`	$\mathrm{Q}(2,13)$	1	29	43	4	41	4.9	$(5.8\pm1.9)\times10^{-4}$
(6)	$\mathbb{R}$	`ap_int`	$\mathrm{Q}(2,15)$	1	31	47	4	41	4.9	$(1.5\pm0.6)\times10^{-4}$
(7)	$\mathbb{R}$	`ap_fixed`*	$\mathrm{Q}(2,15)$	1	108	216	10	101	2.0	$(2.9\pm2.4)\times10^{-5}$
(8)	$\mathbb{C}$	`ap_fixed`*	$\mathrm{Q}(3,14)$	4	347	499	13	131	1.5	$(9.8\pm5.1)\times10^{-5}$
(9)	$\mathbb{C}$	`ap_fixed`	$\mathrm{Q}(3,14)$	4	141	102	6	61	3.3	$(4.5\pm1.4)\times10^{-4}$

ilarly, the fixed-point quantisation step is $\Delta = 2^{-15} \approx 3.1 \times 10^{-5}$ for 15 fractional bits, which is, e.g., on par with the measured result for the real-valued `ap_fixed`* case.

Observations on Floating-Point Parametrisations Tab. 5.1 gives resource utilisations for the `float` data type. In fact, VHLS is tuned to efficiently map floating-point arithmetic onto the resource blocks of an FPGA. The reasoning behind this behaviour is simple: Recent FPGAs provide plentiful resources, in particular a large number of DSP slices. Therefore, area might not be the most critical constraint, but design time instead [Con+11]. As mentioned earlier, Intel's Arria10 and Stratix10 FPGAs even offer hardened circuitry for single-precision floating-point operations [Sin17].

Staying with floating-point numbers eliminates the need for a careful fixed-point conversion of the algorithm (compare Sec. 4.3) and thus speeds up the design time. Not every signal processing application, however, requires the accuracy of 32 bit word lengths. Hence, the simplified development process is an advantage dearly bought with a significantly increased area utilisation.

The half-precision floating-point data type `half` poses a compromise between hardware overhead and the just mentioned benefits. As stated in Tab. 5.1, RTL synthesis was defective in VHLS for this data type parametrisation. Nonetheless, C compilation functioned and accuracy measurement could be obtained.

A Note on Integer Data Types The data type-agnostic code in Listing 4.1 does not support integer types (here: `short` or `ap_int`), because these types require code modifications to manually take care of bit shifts according to the Q format.

Just for this examination, additional helper macros had been inserted which were only defined for the above mentioned integer data type cases to facilitate multiplication with fractional bits, foremost the needed bit shift left by 13 bit to align the result to $\mathrm{Q}(2, 13)$ again. This highlights the efficacy of the `sc_fixed` class which automates this fixed-point multiplication transparently to the designer and without any additional coding overhead.

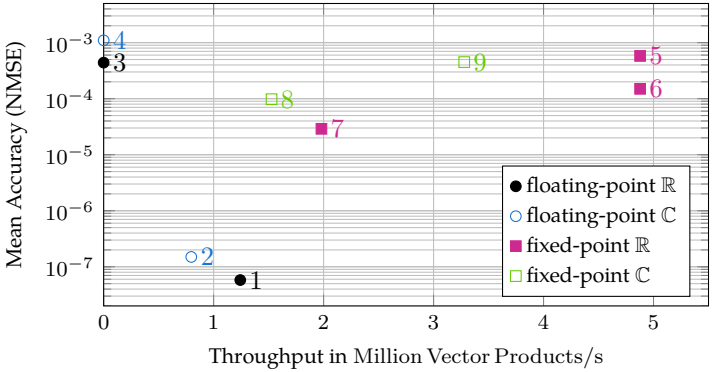


Figure 5.1: Performance evaluation for a data type-agnostic HLS design of a scalar vector product (length 10). The examined data types are listed in Tab. 5.1.

Based on this observation, it is recommendable to exclude support for integer types, but instead create *truly agnostic code*, without any data type-specific special cases whatsoever, for a target subset of data types only: especially, but not limited to, `float`, `ap_fixed`, and `std::complex<T>`.

Throughput Versus Numerical Accuracy The performance metrics of the numbered cases in Tab. 5.1 are compared in Fig. 5.1, which plots the achieved mean numerical accuracy against the data throughput of the HLS designs.

High throughput can be obtained with fixed-point numbers, whereby a longer word length leads to improved numerical accuracy. This again can additionally be improved by enabling saturation overflow handling and number rounding, but only at the cost of significantly increased latencies. Best accuracy is offered by single-precision floating-point computations, of course. The half-precision floating-point numbers perform worse than fixed-point numbers of same size. Even if this reduced-size floating-point format might allow to avoid fixed-point design issues, the ease of use of the RDAM renders fixed-point designs favourable. The same observations equally apply to the complex-valued cases.

5.2 Example 2: Orthogonal Matching Pursuit – A Computationally Complex Algorithm

The next design example shall demonstrate that it is possible to implement even complex, heterogeneous algorithms with the proposed RDAM. The algorithm of choice is OMP, [TG07]. All of the discussed points in the previous chapters will find an application herein.

OMP is a greedy algorithm best known for its applications to Compressed Sensing (CS), which will be explained in Chap. 6 later. There is also a recent application to sparse multi-user channel estimation in a wireless communications network [Kno+16b], which can only be facilitated if the output architecture allows for real-time operation. OMP makes frequent use of linear algebra operations and, hence, is parallelisable to some extent but contains a lot of control structure as well. Therefore, the OMP constitutes a fitting subject of study in the context of HLS.

The OMP algorithm was introduced by Tropp and Gilbert in 2007 as a greedy algorithm for the recovery of a sparse vector from random linear measurements. In a nutshell, the CS system model is

$$\mathbf{b} = \mathbf{A}\mathbf{x}\,, \tag{5.3}$$

where $\mathbf{A}$ is the $M \times N$ random CS measurement matrix, $\mathbf{b}$ the measurement vector and $\mathbf{x}$ the sparse vector to be recovered. $\mathbf{x}$ is said to be k-sparse if it only contains k non-zero entries or, in other words, the ℓ_0-pseudo norm is $\|\mathbf{x}\|_0 = k$. Hence, the problem size can easily be stated as a triplet (M, N, k), where $k \ll M \ll N$. Note, that (5.3) can be either real-valued or complex-valued. $\mathbf{A}$ must fulfil certain conditions, but these are satisfied with high probability when its entries are drawn from a random process, e.g. a Gaussian one. Then, OMP can reliably recover $\mathbf{x}$ with $\mathcal{O}(k \ln N)$ random linear measurements [TG07].

The algorithm is restated as pseudocode in Alg. 5.1. The residual $\mathbf{r}$ is initialised with the measurement vector $\mathbf{b}$. The OMP recovers a k-sparse vector iteratively in k iterations, with t as the loop counter. The loop body mainly consists of three consecutive computational steps (or kernels), each dependent on the previous one:

1. correlation and selection,
2. LS orthogonalisation, and

Algorithm 5.1: Pseudocode of Orthogonal Matching Pursuit (OMP) according to Tropp and Gilbert [TG07].

```
1 function OMP( b, A, k )
2     r ← b; x ← 0; Λ ← ∅                                  ▷ initialisation
3     for t = 1, ..., k do
4         Λ ← Λ ∪ (arg max_ℓ |⟨A_ℓ, r⟩|)                   ▷ correlation
5         x_Λ ← arg min_{x_Λ} ‖b − A_Λ x_Λ‖_2              ▷ least squares
6         r ← b − A_Λ x_Λ                                  ▷ residual
7     return x
```

3. residual update.

Firstly, one column of $\mathbf{A}$ is selected that is most strongly correlated with the residual $\mathbf{r}$, and the set Λ is augmented by the index λ of the chosen column (line 4). Λ is the index set of all indices that have been chosen so far and identifies the non-zero locations (support set), also called atoms, of the sparse result vector $\mathbf{x}$. Next, a LS step is computed in line 5 on a reduced system of equations. $\mathbf{A}_\Lambda$ is a $M \times t$ matrix composed of all selected columns, and $\mathbf{x}_\Lambda$ denotes a vector of length t. An equivalent formulation of line 5 would be

$$\mathbf{x}_\Lambda = \mathbf{A}_\Lambda^+ \mathbf{b}\,, \tag{5.4}$$

whereby $(\cdot)^+$ denotes the Moore–Penrose pseudoinverse. The last step within the loop is the update of the residual.

5.2.1 A Survey of Related Works

A substantial body of research on the digital hardware design of OMP has already been published [SS10; SM12; BRA12; Bai+12; RAA12; SM13; Rab+15; Kno+16a]. Although there are some ASIC designs, most of the literature focuses on FPGAs as target platform. A comparative overview of FPGA designs is given in Tab. 5.2, including some own results, which have been published first in [Kno+16a].

Various possibilities were proposed to solve the LS problem. Some works rely on a Cholesky decomposition in order to avoid the square

Table 5.2: Comparison of related VLSI designs of Orthogonal Matching Pursuit Targeting FPGAs.

				Problem Size			Time	Freq.	Format		Resource Utilisation				
Ref.	**Variant**	$\mathbb{K}$	**Method**	M	N	K	(µs)	(MHz)	$\mathrm{Q}(m, n)$	**Target**	BRAM	DSP	FF	LUT	NMSE
[SS10]	2-stage	$\mathbb{R}$	RTL	32	128	5	24.0	39	$\mathrm{Q}(10, 22)$	Virtex-5	—	—	—	—	—
[SM12]	2-stage	$\mathbb{R}$	RTL	64	128	5	10.0	85	$\mathrm{Q}(10, 14)$	Virtex-5	—	—	—	—	—
[SM12]	2-stage	$\mathbb{R}$	RTL	64	256	8	27.1	85	$\mathrm{Q}(10, 14)$	Virtex-5	—	—	—	—	—
[BRA12]	classical	$\mathbb{R}$	Simulink	32	128	5	15.7	107	16 bit	Virtex-5	42	134	2341	4012	47.0 dB (PSNR)
[Bai+12]	2-stage	$\mathbb{R}$	RTL	256	1024	36	[b] 622.0	100	18 bit	Virtex-6	258	261	—	32010	—
[RAA12]	classical	$\mathbb{R}$	HLS	—	[b] 512	—	—	128	`float`	Zynq-7	19	27	3776	6605	—
[SM13]	2-stage [a]	$\mathbb{R}$	RTL	64	256	8	7.1	85	$\mathrm{Q}(10, 14)$	Virtex-5	—	—	—	—	7.1×10^{-3}
[Rab+15]	classical	$\mathbb{R}$	Simulink	256	1024	36	340.0	119	$\mathrm{Q}(9, 9)$	Virtex-6	576	589	—	6208	38.9 dB (PSNR)
This Work	classical	$\mathbb{R}$	HLS	32	128	5	10.7	103	$\mathrm{Q}(3, 14)$	Virtex-7	4	518	22564	18330	1.4×10^{-7}
This Work [c]	classical	$\mathbb{R}$	HLS	32	128	5	16.9	107	$\mathrm{Q}(3, 14)$	Virtex-7	4	130	13781	22345	1.4×10^{-7}
This Work	classical	$\mathbb{R}$	HLS	32	128	5	23.0	87	$\mathrm{Q}(3, 14)$	Zynq-7	4	130	15804	22486	1.4×10^{-7}
This Work [d]	classical	$\mathbb{C}$	HLS	32	128	5	17.1	114	$\mathrm{Q}(2, 15)$	Virtex-7	1	274	25733	46449	1.0×10^{-3}
This Work	classical	$\mathbb{C}$	HLS	64	256	8	65.1	101	$\mathrm{Q}(2, 15)$	Virtex-7	2	536	67104	129497	5.0×10^{-4}

[a] A thresholding operation was introduced to accelerate the matrix-vector correlation. This is an algorithmic approximation, and therefore the results are not directly comparable. [b] Parameter extracted from other given results. [c] See Fig. 5.3(c). [d] See Fig. 5.4, right "fxp".

root operation [BRA12; Rab+15]. In [TG07], a QRD was used within each iteration. [SS10] proposed an updating modified Gram–Schmidt process for the loop iterations and an alternative Cholesky decomposition for the second stage. However, the computation of a Gram–Schmidt process results in a QRD for free, and the pseudoinverse can then efficiently be computed by

$$\mathbf{x}_\Lambda = \mathbf{A}_\Lambda^+ \mathbf{b} = \left(\mathbf{A}_\Lambda^H \mathbf{A}_\Lambda\right)^{-1} \mathbf{A}_\Lambda^H \mathbf{b} \tag{5.5}$$

$$= \left(\mathbf{R}^H \mathbf{Q}^H \mathbf{Q} \mathbf{R}\right)^{-1} \mathbf{R}^H \mathbf{Q}^H \mathbf{b} = \mathbf{R}^{-1} \mathbf{Q}^H \mathbf{b}\,. \tag{5.6}$$

This was realised by Bai et al., who computed an incremental QRD during the first stage and used back substitution for the second stage to obtain $\mathbf{R}^{-1}$ in order to solve the LS problem [Bai+12].

[SM13] introduces a threshold operation to accelerate the matrix-vector correlation leading to an incomplete and suboptimal but computationally efficient atom selection. Hence, the results cannot be compared directly.

Most notably, a couple of works implement the OMP substantially differently [SS10; SM12; Bai+12; SM13], compared to the original one published by Tropp and Gilbert [TG07], as pointed out in [Rab+15]. The former is referred to as the two-stage variant of OMP within this document (see Tab. 5.2). According to the classical OMP, a LS step is to be solved within each iteration of the embracing loop. This ensures orthogonality between $\mathbf{A}_\Lambda$ and the updated residual $\mathbf{r}$ (Alg. 5.1, line 6). Septimus and Steinberg proposed in [SS10] to partition the OMP into two consecutive stages, of which the first iteratively selects the sparse support and only the second performs a LS solution over that support. Instead of lines 5 and 6, a Gram–Schmidt orthogonalisation process of $\mathbf{A}_\Lambda$ is used to obtain an updated $\mathbf{Q}$ matrix of which only the latest column, $\mathbf{Q}_{\lambda_t}$, is of further interest. Then, the residual is updated recursively by

$$\mathbf{r}_t = \mathbf{r}_{t-1} - \mathbf{Q}_{\lambda_t} \mathbf{Q}_{\lambda_t}^T \mathbf{r}_{t-1}\,. \tag{5.7}$$

Note, that $\mathbf{Q}_t \mathbf{Q}_t^T$ is the outer matrix product (dyadic product). And not until after the loop but before line 7 in Alg. 5.1, a LS solution is computed with $\mathbf{x}_\Lambda = \mathbf{A}_\Lambda^+ \mathbf{b}$, only once.

The question arises, whether these two OMP variants are equivalent or not [Rab+15]. This is especially true because [SS10] omits any proof or

derivation of their fundamentally different formulation. However, it can be shown that these are indeed functionally equivalent, as follows.

Proof of the Algorithmic Equivalence Between Classical and Two-Stage OMP. Let $\mathbf{A}_\Lambda$ be the matrix of selected columns of $\mathbf{A}$. Since the cardinality of Λ increases with each iteration by one, its dimensionality is $M \times t$, with $t = 1, ..., k$. $\mathbf{Q}$ and $\mathbf{R}$ are the products of the Gram–Schmidt orthogonalisation process of $\mathbf{A}_\Lambda$, and as the latter increases in size with each loop iteration, so do $\mathbf{Q}$ and $\mathbf{R}$,

$$\mathbf{QR} = \mathbf{A}_\Lambda \,. \tag{5.8}$$

Further it is $\mathbf{x}_\Lambda = \mathbf{A}_\Lambda^+ \mathbf{b} = \mathbf{R}^{-1}\mathbf{Q}^T\mathbf{b}$ as stated above in (5.6). Now, Tropp's and Gilbert's residual update is given by [TG07]

$$\mathbf{r}_t = \mathbf{b} - \mathbf{A}_\Lambda \mathbf{x}_\Lambda \,, \;\; \text{with} \;\; \mathbf{x}_\Lambda = \mathbf{A}_\Lambda^+ \mathbf{b} \,, \tag{5.9}$$

whereas Septimus' and Seinberg's residual update strategy is [SS10]

$$\mathbf{r}_t = \mathbf{r}_{t-1} - \mathbf{Q}_t \mathbf{Q}_t^T \mathbf{r}_{t-1} = \left(\mathbf{I} - \mathbf{Q}_t \mathbf{Q}_t^T\right) \mathbf{r}_{t-1} \,. \tag{5.10}$$

To show the equivalence of (5.9) and (5.10), Eqs. (5.6) and (5.8) can be inserted into (5.9):

$$\begin{aligned} \mathbf{r}_t &= \mathbf{b} - \mathbf{A}_\Lambda \mathbf{A}_\Lambda^+ \mathbf{b} = \left(\mathbf{I}_{M\times M} - \mathbf{A}_\Lambda \mathbf{A}_\Lambda^+\right) \mathbf{b} \\ &= \left(\mathbf{I} - \mathbf{Q}\mathbf{R}\mathbf{R}^{-1}\mathbf{Q}^T\right)\mathbf{b} = \left(\mathbf{I} - \mathbf{Q}\mathbf{Q}^T\right) \mathbf{b} \,. \end{aligned} \tag{5.11}$$

The first two iterations according to (5.10) are

$$\mathbf{r}_1 = \left(\mathbf{I} - \mathbf{Q}_1 \mathbf{Q}_1^T\right) \mathbf{r}_0 = \left(\mathbf{I} - \mathbf{Q}_1 \mathbf{Q}_1^T\right) \mathbf{b} \,, \text{ and} \tag{5.12}$$

$$\begin{aligned} \mathbf{r}_2 &= \left(\mathbf{I} - \mathbf{Q}_2 \mathbf{Q}_2^T\right) \left(\mathbf{I} - \mathbf{Q}_1 \mathbf{Q}_1^T\right) \mathbf{b} \\ &= \left(\mathbf{I} - \mathbf{Q}_1 \mathbf{Q}_1^T - \mathbf{Q}_2 \mathbf{Q}_2^T + \mathbf{Q}_2 \mathbf{Q}_2^T \mathbf{Q}_1 \mathbf{Q}_1^T\right)\mathbf{b} \\ &= \left(\mathbf{I} - \mathbf{Q}_1 \mathbf{Q}_1^T - \mathbf{Q}_2 \mathbf{Q}_2^T\right)\mathbf{b} \,, \end{aligned} \tag{5.13}$$

considering the orthogonality of the columns of $\mathbf{Q}$, $\langle \mathbf{Q}_i, \mathbf{Q}_j \rangle = 0, \forall i, j, i \neq j$. Hence, after the t-th iteration it is

$$\mathbf{r}_t = \left(\mathbf{I} - \mathbf{Q}_1\mathbf{Q}_1^T - \mathbf{Q}_2\mathbf{Q}_2^T \cdots - \mathbf{Q}_t\mathbf{Q}_t^T\right)\mathbf{b} \tag{5.14}$$

$$= \left(\mathbf{I} - \sum_{\ell=1}^{t} \mathbf{Q}_\ell\mathbf{Q}_\ell^T\right)\mathbf{b} = \left(\mathbf{I} - \mathbf{Q}\mathbf{Q}^T\right)\mathbf{b}\,, \tag{5.15}$$

which is identical to (5.11) for every iteration t.

Therefore, the classical OMP by Tropp and Gilbert and the two-stage OMP by Septimus and Steinberg are indeed formally equivalent. Based on the same residual, they will always select the same support set and return the same solution after the LS step. However, numerical rounding errors accumulate differently. ■

Most of the cited VLSI designs were coded manually in an HDL, probably using generic expressions for different problem sizes, e.g. in [SM12]. It is noteworthy that some works already acknowledged the need for higher design abstraction. The results in [BRA12; Rab+15] were obtained with Matlab Simulink in conjunction with Xilinx System Generator, which basically is a graphical, model-based hardware description approach. A first implementation of OMP with HLS was reported on in [RAA12], where VHLS was used as well; but unfortunately, the authors did not state any details whatsoever on the problem sizes, synthesised architecture or algorithmic method to compute the LS problem.

5.2.2 Algorithmic Transformation with Rank-1 Updating

In this section, two significant modifications to the method of computation of the OMP will be described. These are applicable to classical and two-stage OMP alike, although the following explanations are restricted to the classical variant.

- Firstly, rank-1 updates to the QRD and additionally to the pseudoinverse are proposed. Modified Gram–Schmidt (MGS) is chosen because it is the only QRD algorithm which offers the possibility for iterative updating, is numerically stable enough and can be parallelised nicely for big matrices.

- And secondly, the mathematically non-trivial operations for digital circuitry of MGS, namely division and square root, are to be performed within the logarithmic domain (LNS).

These constitute the algorithmic transformations according to Step 1 of the RDAM (see Sec. 3.4).

Not only can the QRD be updated iteratively; the same applies to the computation of the pseudoinverse as well. The LS solution based on QRD requires the inversion of $\mathbf{R}$ as stated above in (5.6). Since OMP adds another column to $\mathbf{A}_\Lambda$ during each iteration (rank-1 update), $\mathbf{Q}$ and $\mathbf{R}$ grow by a column to the right as well, while the updated $\mathbf{R}$ keeps its upper-triangular structure of course. If the QRD is complex-valued, so will be $\mathbf{Q}$ and $\mathbf{R}$, but the main diagonal of $\mathbf{R}$, $\mathrm{diag}(\mathbf{R})$, is real-valued in either case. Now, block matrix inversion gives us [PP12]

$$\begin{bmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{C} & \mathbf{D} \end{bmatrix}^{-1} = \begin{bmatrix} \mathbf{S}^{-1} & -\mathbf{S}^{-1}\mathbf{B}\mathbf{D}^{-1} \\ -\mathbf{D}^{-1}\mathbf{C}\mathbf{S}^{-1} & \mathbf{D}^{-1} + \mathbf{D}^{-1}\mathbf{C}\mathbf{S}^{-1}\mathbf{B}\mathbf{D}^{-1} \end{bmatrix}, \tag{5.16}$$

where $\mathbf{S}$ is the Schur complement of $\mathbf{D}$ being defined as

$$\mathbf{S} = \mathbf{A} - \mathbf{B}\mathbf{D}^{-1}\mathbf{C}\,. \tag{5.17}$$

Here, $\mathbf{A}$ is the upper triangular matrix of the previous iteration of size $(t-1)\times(t-1)$, which is extended by another column vector of length t, i.e. $\mathbf{B}$ becomes a column vector, $\mathbf{D}$ becomes a real-valued scalar d, and, due to the triangular structure, it is $\mathbf{C} = \mathbf{0}_{1\times(t-1)}$, which is a row of zeroes. Therefore, (5.16) can be simplified to

$$\begin{bmatrix} \mathbf{A} & \mathbf{b} \\ 0 & d \end{bmatrix}^{-1} = \begin{bmatrix} \mathbf{A}^{-1} & -\mathbf{A}^{-1}\mathbf{b}/d \\ 0 & 1/d \end{bmatrix}. \tag{5.18}$$

Only a single real-valued division is necessary per iteration to compute $1/d$, which can then be multiplied $t-1$ times to the vector $-\mathbf{A}^{-1}\mathbf{b}$. This division, though, has already been computed during the updating of the $\mathbf{Q}$ matrix and is still available for the rank-1 update of $\mathbf{R}^{-1}$.

Eq. (5.18) becomes

$$ {}^{(t)}\mathbf{R}^{-1} = \begin{bmatrix} {}^{(t-1)}\mathbf{R} & \mathbf{r} \\ 0 & r_{tt} \end{bmatrix}^{-1} = \begin{bmatrix} {}^{(t-1)}\mathbf{R}^{-1} & -{}^{(t-1)}\mathbf{R}^{-1} \cdot \mathbf{r}/r_{tt} \\ \mathbf{0}_{1\times t-1} & 1/r_{tt} \end{bmatrix}, \quad (5.19) $$

when the $\mathbf{R}$ matrix at hand is inserted. To clarify the notation, the upper left index denotes the iteration index, e.g. ${}^{(t-1)}\mathbf{R}^{-1}$ is the inverse matrix computed during the previous iteration. Obviously, only a further matrix-vector multiplication ${}^{(t-1)}\mathbf{R}^{-1}\mathbf{r}$ of reduced rank $t-1$ scaled by $\zeta := 1/r_{tt}$ is needed to obtain ${}^{(t)}\mathbf{R}^{-1}$. This matrix-vector product is computationally cheap compared to a complete upper triangular matrix inversion with a full back substitution.

Only a single reciprocal square root operation per iteration t is required for ζ, which is computed by the INVSQRT function in (4.3) with the help of the LNS as described in Sec. 4.4.3. Furthermore it is

$$ \zeta = \frac{1}{\|\mathbf{q}\|} = \frac{1}{\sqrt{x}} \quad (5.20) $$

in conjunction with Eq. (5.1) for a vector $\mathbf{q}$ when $x = \langle \mathbf{q}, \mathbf{q} \rangle$ such that $\|\zeta\mathbf{q}\| = 1$. Differently put, the first RDAM design example of Sec. 5.1 is not only a toy problem but it is reused and applied here to the computation of x for the normalisation step of the QRD.

The complete LS algorithm with rank-1 updating is given in Alg. 5.2, which updates $\mathbf{Q}$ *and* $\mathbf{R}^{-1}$ iteratively. This fits nicely into the OMP algorithm in Alg. 5.1 as a replacement of line 5, since the for-loop over $t = 1, ..., k$ embraces this update step. It may not be overlooked, that $\mathbf{R}$ does not need to be stored; solely the update vector $\mathbf{r}$ and ζ are of further interest.

This proposed algorithmic transformation yields a flatter loop hierarchy and eventually leads to improved HLS implementation results.

The consecutive processing steps within each loop iteration are schematically shown in Fig. 5.2. QRD and back substitution both necessitate a loop over all $t = 1, ..., k$, if calculated consecutively, nested within an outer loop over the same index for atom selection. Such nested loop hierarchies increase the HLS design complexity. Therefore, it is good practice to flatten loops either by the application of directives or, even better, by a modified but functionally equivalent design entry source

Algorithm 5.2: Least squares estimation based on QR matrix decomposition with rank-1 updating.

function LS-UPDATE(${}^{(t-1)}\mathbf{Q}, {}^{(t-1)}\mathbf{R}^{-1}, \mathbf{A}_\lambda, \mathbf{b}$)
  $\mathbf{r} \leftarrow \mathbf{0}_{t\times 1}$
  $\mathbf{q} \leftarrow \mathbf{A}_\lambda$
  for $j = 1, ..., t-1$ **do** ▷ orthogonalisation
    $\mathbf{r}_j \leftarrow \langle {}^{(t-1)}\mathbf{Q}_j, \mathbf{q} \rangle$
    $\mathbf{q} \leftarrow \mathbf{q} - \mathbf{r}_j \cdot {}^{(t-1)}\mathbf{Q}_j$
  $\zeta \leftarrow 1/\|\mathbf{q}\|$ ▷ normalisation factor
  ${}^{(t)}\mathbf{Q} \leftarrow \begin{bmatrix} {}^{(t-1)}\mathbf{Q} & \zeta \cdot \mathbf{q} \end{bmatrix}$ ▷ update
  ${}^{(t)}\mathbf{R}^{-1} \leftarrow \begin{bmatrix} {}^{(t-1)}\mathbf{R}^{-1} & -{}^{(t-1)}\mathbf{R}^{-1} \cdot \mathbf{r} \cdot \zeta \\ \mathbf{0}_{1\times t-1} & \zeta \end{bmatrix}$
  $\mathbf{x}_\Lambda \leftarrow {}^{(t)}\mathbf{R}^{-1} \cdot {}^{(t)}\mathbf{Q}^H \cdot \mathbf{b}$ ▷ least squares
  return $\mathbf{x}_\Lambda, {}^{(t)}\mathbf{Q}, {}^{(t)}\mathbf{R}^{-1}$

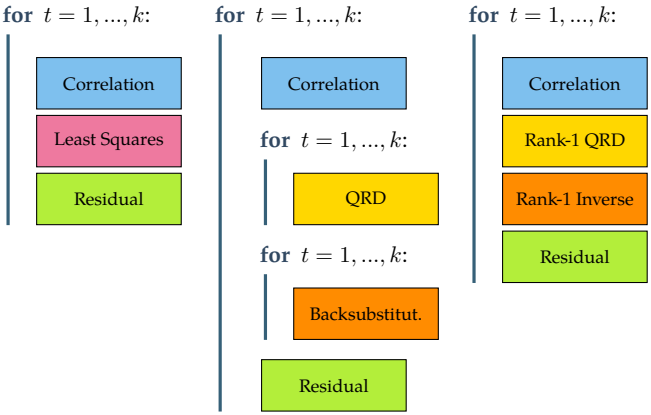


Figure 5.2: The OMP loop body consists of three main steps (left). However, the QRD and back substitution iterate over the same loop index k as the OMP itself to compute the LS solution (middle). Rank-1 updating therefore flattens the hierarchical loop structure (right).

code, i.e. algorithmic modifications. Rank-1 updating to the QRD and the following pseudoinverse, eliminates the need for these nested loops.

5.2.3 High-Level Synthesis Results

The OMP algorithm including the modifications discussed above was implemented with VHLS following the coding principles and guidelines set forth in [Xil-UG902]. All synthesis results given in this section target a Xilinx Virtex-7 FPGA (XC7VX690-2) at 100 MHz clock frequency, if not stated otherwise. The decision in favour of a Virtex device was made with respect to a better comparability to other published results.

The C++ source code was written with parametrised static memory allocations and verified against an existing Mathworks Matlab functional model of OMP for double-precision floating-point data types. This constitutes Step 2 and 3 according to the RDAM. Complex-valued operation was realised as explained in Sec. 4.2. The OMP problem size (M, N, k) is in the following, if not stated otherwise, a $(32, 128, 5)$ setup with $\mathrm{Q}(3, 14)$ baseline fixed-point precision to match the architecture of the DSP slices. To define an 18 bit word length maximally exploits the 18 bit multiplier inputs while at the same time avoiding DSP slice chaining (please compare the conclusions on Fig. 3.8 in Sec. 3.3.2). The double-sized $\mathrm{Q}(7, 28)$ format was employed for selected MACs to improve numerical accuracy. That is Step 4 according to the RDAM.

Fig. 5.3 shows the synthesis results for four different sets of HLS compiler directives:

(a) A design with minimal resource utilisation, which is the default optimisation of VHLS ("unoptimised" with regard to the exploitation of parallelisms).

(b) The correlation and selection step is completely parallelised, i.e. the number of DSP slices is mainly influenced by M and it takes N cycles to compute $\mathbf{A}^H\mathbf{r}$.

(c) The previous design but additionally almost all loops are pipelined. This yields a fair trade-off between resource utilisation and computation time.

(d) A design with all second-level loops unrolled and thus highly parallelised.

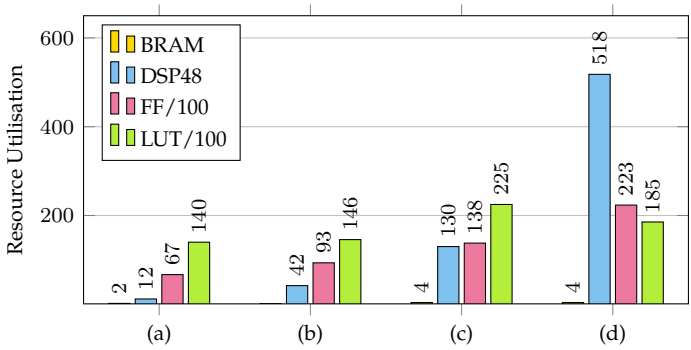


Figure 5.3: Resource utilisation of four different architectures from unoptimised (left) to highly parallelised (right).

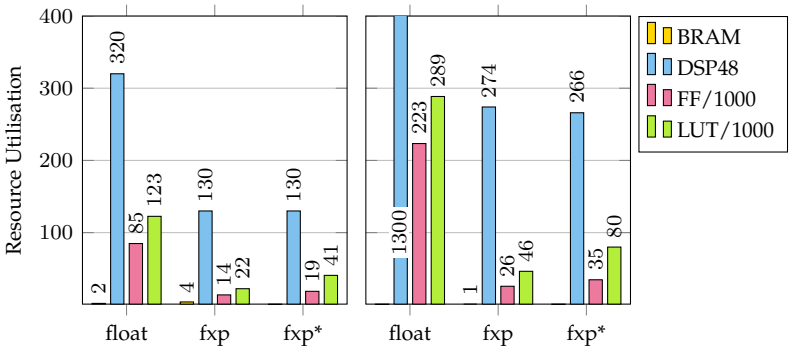


Figure 5.4: Resource utilisation with different data types for real-valued (left) and complex-valued (right) computation.

In Fig. 5.4, the impact of different data types is compared for architecture (c). "fixed-point" denotes the above mentioned $\mathrm{Q}(3, 14)$ format and "fixed-point*" the same but with saturation arithmetic and number rounding enabled. These typical DSP features improve the NMSE a little from 1.4×10^{-7} to 1.8×10^{-8}. The synthesised single-precision floating-point design ("float") yields an NMSE of 1.3×10^{-12}. However, the resource utilisation is inefficient compared to the gain in accuracy.

The right side of Fig. 5.4 shows results for the complex-valued data type parametrisations. Aside from floating-point case, the complex-valued designs consume about twice the resources. This is consistent with the expected outcome as it could be derived from a RVD. Either way, all synthesised architectures do not exceed the capacity of the target FPGA.

Further details of selected synthesis products are given in Tab. 5.2, on page 83. In summary, the synthesised digital architectures can compete with existing works. A real-valued OMP of size $(32, 128, 5)$ can be computed within 10.7 µs or 16.9 µs, depending on the degree of parallelism. This is comparable to the results given in [BRA12]. The same can be said about the resource utilisation, although VHLS obviously favours distributed random access memory (RAM) over BRAM without additional constraints (directives). The complex-valued digital architecture of OMP is with 17.1 µs almost as fast as the real-valued one. This design is, to the best knowledge of the authors, the first complex-valued digital architecture for OMP, and thus can cannot be compared to other works.

This concludes the design space exploration according to Step 5 of the RDAM, and IP core generation and packaging for implementation follow (Step 6).

5.2.4 System-on-Chip Integration and Testing

After successful RTL co-simulation with VHLS, the above presented OMP design was packaged for a specific parametrisation as an IP core in the industry standard IP-XACT format using the export functionality. The HLS design was configured for a $(32, 128, 5)$ problem size with the real-valued $\mathrm{Q}(3, 14)$ fixed-point baseline data type. The targeted FPGA was a Xilinx Zynq-7 type XC7Z020 with speed grade "-1", and the system clock of the programmable logic was 100 MHz. For communication with the processing system, an AXI Lite interface was instantiated which comprises the inputs $\mathbf{x}$, $\mathbf{b}$, $\mathbf{A}$, and the output $\hat{\mathbf{x}}$.

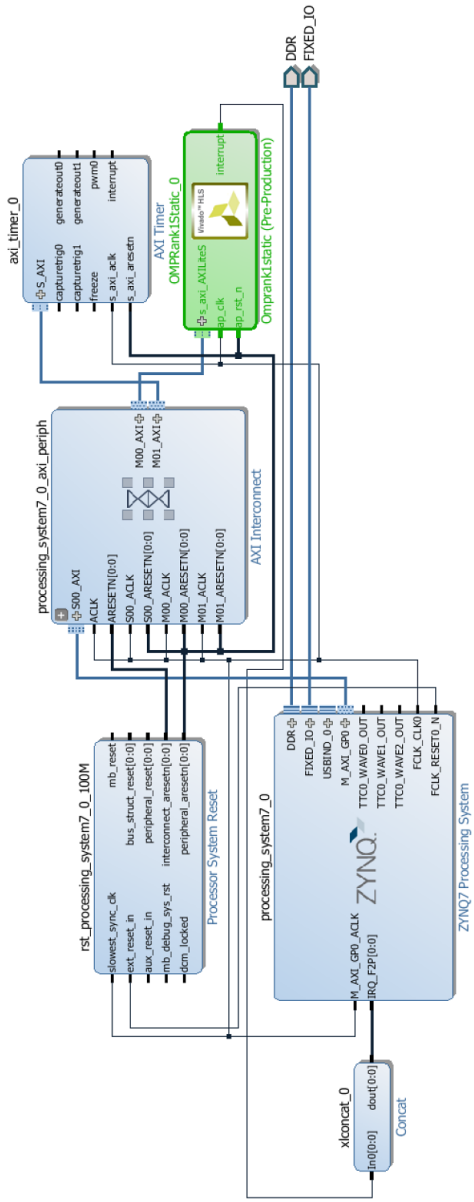


Figure 5.5: The Rank-1 OMP Zynq-7 SoC block design created with Xilinx Vivado IP Integrator.

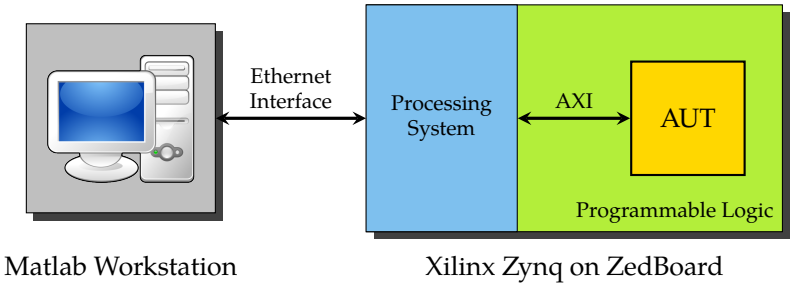


Figure 5.6: Testing an algorithm under test in a hardware-in-the-loop simulation on a AvNet .

A full SoC design was created with Xilinx Vivado, using the IP Integrator, which is a graphical approach to instantiate the processing system of the Zynq and add peripherals with interconnections by AXI buses. It consists, basically, of the Zynq processing system, the OMP IP core named `OMPRank1Static` and a Xilinx AXI Timer to measure the execution time. The remaining blocks were automatically added by IP Integrator; they are purposed to handle a system reset and to connect the OMP and timer as slaves to the AXI Lite bus of the SoC. Fig. 5.5 shows the resulting block design schematic. The following sections will discuss two different deployments of this SoC design.

5.2.4.1 Ethernet-Based Hardware-in-the-Loop Simulations with Matlab

The first application was a HIL simulation with an Avnet ZedBoard, which is a small prototyping platform featuring a Xilinx Zynq-7 with a bunch of peripherals. The general setup is shown in Fig. 5.6. The SoC design with the `OMPRank1Static` IP core as AUT is programmed into the FPGA and controlled by a software written in C on the processing system, which indeed is an ARM Cortex-A9 dual core processor and powerful enough to run an embedded Linux system.

On the other end is a Mathworks Matlab workstation as test vector generator installed. A function called `hil_sim.m` accepts matrices, vectors

Code Listing 5.1: An exemplary HIL simulation with the Matlab script `rank1omp_hilsim.m`.

```
m = 32;  n = 128;  k = 5;
[A, b] = csmodel(m, n, k, 'real');
x = omp(A, b, k);
x_hat = hil_sim('Matrix', b, A);
NMSE = norm(x - x_hat)^2 / norm(x)^2
```

Table 5.3: Resource utilisation before and after RTL synthesis

Description	BRAM	DSP	FF	LUT
HLS estimate (with AXI Lite)	70	130	19900	25968
SoC design (post RTL synthesis)	35	130	11573	13997
PYNQ hardware overlay (Sec. 5.2.4.2)	35	130	5702	13899

or scalar variables as inputs and transfers the data via Ethernet using the unofficial pnet Matlab toolbox [Ryd08] to a small listening server application running on the Linux system. The example Matlab test bench in Listing 5.1 shows how random test data is generated by a Matlab function `csmodel`, which returns a CS system matrix $\mathbf{A}$ and a compressive measurement vector $\mathbf{b}$, and then forwarded to the HIL framework. The numerical accuracy is evaluated by the NMSE as defined in Eq. (3.14).

Test data items can be transferred by `hil_sim.m` one by one for a single execution of the AUT or in batch mode for hundreds of executions in sequence. Data can be any type (complex-valued, real-valued, floating-point or integers) and any size (matrix, vector or scalar). The server application configures the AUT and copies the input data into the memory-mapped registers of the IP core via the AXI Lite interface for a single execution; but direct memory access (DMA) has not been employed to keep the system small and simple. The OMP core emits an interrupt after termination, and in the following, the server application reads the returned output vector from memory-mapped registers, and packs it for Ethernet transfer to Matlab again.

The resource utilisation estimation by VHLS of the `OMPRank1Static` core with the described parametrisation is given in Tab. 5.3. Note, this

design includes in contrast to the one given in Tab. 5.2 on page 83 an AXI Lite interface and meets timing requirements for operation at 100 MHz. After IP core integration, the full Vivado SoC design was synthesised into a bitstream file to program the FPGA and a configuration file to set up the processing system. A comparison between the VHLS and post-RTL synthesis results clearly reveals that VHLS estimates resource utilisation pessimistically. Eventually all given figures, except for the DSP48 slice count, are lower than anticipated by 40 to 50 %. This behaviour of VHLS is obviously intentional to ensure functionality throughout the remaining top-down implementation flow.

The total power dissipation was stated by Vivado to be 2.13 W, which divides into 421 mW (19.8 %) for the `OMPRank1Static` core, 1529 mW (71.8 %) for the processing system and 180 mW (8.4 %) into remaining FPGA logic (AXI Timer, clock tree, etc.).

The HIL simulation with the integrated IP core was designed for functionality, not speed. Hence, one execution of the `OMPRank1Static` core was measured to take 32.0 µs. The performance loss can be explained by the added bus interface and extra time needed to copy data to and from the memory-mapped registers. Generally, data movement within a processor architecture incurs the highest performance penalties. Nonetheless, however, the evaluated numerical accuracy matches the expected NMSE of 1.4×10^{-7}. All measurements were averaged over 1000 HIL simulations.

In conclusion, the HIL test setup can remarkably prove the functional correctness of the HLS design and especially the efficacy of the RDAM with regard to design space exploration, DTA and the simplifications to fixed-point arithmetic with target numerical accuracies.

5.2.4.2 Hardware Overlay for Xilinx PYNQ

The HIL framework of the previous section required a lot of software engineering to create the server application and to integrate a specialised device driver for the OMP IP core into the Linux kernel. A novel approach to circumvent such manual engineering is taken by Xilinx' open-source project PYNQ, which is short for "Python productivity for Zynq" [Xil18]. A first development board is the Xilinx PYNQ-Z1, which comes with the same XC7Z020 device as the ZedBoard.

PYNQ is a Linux-based framework which runs on the processing system and includes a full Python programming environment as well as a web server. A developer can connect with a local Ethernet connection to this server and work with Jupyter notebooks, i.e. small self-documenting Python scripts—edited, controlled and executed from the web frontend [Pro18].

The original innovation, however, is that the programmable logic (FPGA part of the Zynq) can be re-configured at run time by loading so-called hardware overlays. A hardware overlay consists of a bitstream file and a configuration Tcl script. The PYNQ framework transparently handles this reconfiguration and integration of all necessary drivers into the Linux kernel. Firstly, it is possible to build a library of easy-to-reuse hardware designs; and secondly, it is possible use this environment for HIL simulations given the powerful capabilities of the Python HLL and its extensions, e.g. by the NumPy package [Dev18].

The very same SoC design of Fig. 5.5 with the `OMPRank1Static` IP core was employed to create a hardware overlay for PYNQ. From a Jupyter notebook it can be loaded and instantiated with two lines of code:

```
1 from pynq import Overlay
2 hw_overlay = Overlay('OMPRank1Static.bit')
```

The resource utilisation of the hardware overlay is given in Tab. 5.3 and is in accordance with the expected utilisation results of the previous Sec. 5.2.4.1. A Jupyter notebook was created which generated random test data according to the CS model, performed fixed-point conversion from floating-point to $\mathrm{Q}(3, 14)$, and handled data transfer to and from the hardware overlay with the high-level Python application programming interface (API). This concludes the successful verification with PYNQ.

The PYNQ framework, therefore, is a simple and fast way for testing embedded hardware designs created with the RDAM. Not only is re-usability pronounced, but also the deployment to a wider audience is improved, because of the open-source idea of PYNQ and the open-source community around Python in general. Instead of complex Vivado projects containing numerous files, only a few files (bitstream, IP core configuration Tcl file, and Python test script) have to be shared.

Part II

Compressed Sensing Multi-User Wireless Communications

Chapter 6

Principles of Compressed Sensing (CS)

Conventional DSP always assumes signals that are sampled according to the famous Shannon–Nyquist sampling theorem. A band-limited signal must be sampled at a rate f_s which is at least twice the maximum frequency f_c of the signal, $f_s \geq 2f_c$. Aside from the bandwidth limitation, the bandwidth being $B = 2f_c$, the Shannon–Nyquist theorem makes no further assumptions about the signal. This is different for Compressed Sensing (CS), also called Compressive Sampling sometimes.

CS introduces the notion of sparsity as a further constraint, that the signal of interest can be represented by very few non-zero coefficients in some transform basis. The actually sampled time-domain signal need not be sparse itself, but can be sampled at a significantly lower rate, $f_s \ll f_c$. In other words, CS is a sampling framework where a certain signal structure is concealed behind an obvious sampled truth. Only the application of mathematical rigour allows for the recovery of the signal of interest.

In this chapter, the origins and theoretical principles of CS will be outlined and the sampling framework explained. Then, a survey of algorithms for signal recovery will follow. The concluding section will describe the application of CS to the field of digital wireless communications and a communications system model which forms the baseline setup for the chapters to come.

6.1 The Compressed Sensing Framework

The very first ideas of CS date back to the observations of Candès et al. published in 2006. They found that exact signal reconstruction is possible even with highly incomplete frequency information, provided the time-domain signal is sparse [CRT06a]. A more general mathematical treatise on the foundations of CS is given in [Don06]. Furthermore, concise theoretical summaries can be found in [Bar07; CW08; JV11; EK12].

Broadly speaking, the measurement step of CS could be subsumed under the umbrella of methods for analogue-to-information conversions. What is quite similar is Vetterli et al.'s approach who introduce a finite rate of innovation based on a parametric description of the to-be-sampled signal, which is potentially significantly lower than Nyquist rate [VMB02]. Nonetheless, the sampling kernel has to match the signal structure, which is different and, hence, advantageous for CS: the compressive measurement is non-adaptive to the signal of interest—solely transform sparsity is a requirement [Don06].

6.1.1 Sparsity and Transform Coding

Any signal $\mathbf{x} \in \mathbb{R}^N$ can be expressed with the help of an orthonormal basis $\mathbf{\Psi} \in \mathbb{R}^{N \times N}$,

$$\mathbf{\Psi} = \begin{bmatrix} \boldsymbol{\psi}_1 & \dots & \boldsymbol{\psi}_N \end{bmatrix} \tag{6.1}$$

in another domain, such that

$$\mathbf{x} = \sum_{n=1}^{N} \boldsymbol{\psi}_n s_n = \mathbf{\Psi}\mathbf{s}\,. \tag{6.2}$$

Clearly, $\mathbf{x}$ and $\mathbf{s}$ are equivalent representations of the same signal. Yet, $\mathbf{\Psi}$ can very well be a sparsifying basis, i.e. $\mathbf{s}$ is a sparse signal while $\mathbf{x}$ is not. Usually a vector is said to be sparse if it only contains very few non-zero entries. Let the number of non-zeroes be k, then sparsity implies that $k/N \leq 0.5$.

A signal can be exactly or approximately sparse. Often, this is expressed by the ℓ_0-pseudonorm, formally defined as

$$\|\mathbf{s}\|_0 = |\operatorname{supp}(\mathbf{x})| = |\{n \mid s_n \neq 0\}|\,. \tag{6.3}$$

It returns the number of non-zero elements of $\mathbf{s}$, or, expressed differently, the cardinality of the support of $\mathbf{x}$. $\mathrm{supp}(\mathbf{x})$ is the set of positional indices of the non-zero elements. Strictly speaking, $\mathbf{s}$ is now said to be k-sparse if it has up to k non-zero entries, or

$$\|\mathbf{s}\|_0 \leq k\,. \tag{6.4}$$

Alternatively, $\mathbf{s}$ can be approximately sparse if it is compressible, i.e. the magnitudes of coefficients sorted in descending order obey a power law decay.

The ℓ_0-pseudonorm is, in fact, not a norm, because it does not satisfy the triangle inequality. Notwithstanding, it is often falsely seen as the generalised case $p \to 0$ of the ℓ_p vector norm

$$\|\mathbf{x}\|_p = \left(\sum_{n=1}^{N} |x_n|^p\right)^{\frac{1}{p}}, \quad p \geq 1\,. \tag{6.5}$$

The two special cases for $p = 2$, the Euclidean norm, and especially for $p = 1$,

$$\|\mathbf{x}\|_1 = |x_1| + \ldots + |x_N|\,, \tag{6.6}$$

are of utmost importance to CS.

6.1.2 The Measurement Matrix

CS generalises the term "sampling" to a set of linear or compressive measurements. Information about the signal $\mathbf{x}$ is obtained by linear functionals

$$y_m = \langle \boldsymbol{\varphi}_m, \mathbf{x} \rangle\,, \quad m = 1, ..., M\,, \tag{6.7}$$

which essentially constitutes correlations of $\mathbf{x}$ with all $\boldsymbol{\varphi}_m$, [CW08]. Note, $\mathbf{x}$ is a vector of N already discretised values. The measurement vectors $\boldsymbol{\varphi}_m \in \mathbb{R}^N$ can be assembled as rows of the $M \times N$ measurement matrix

$$\boldsymbol{\Phi} = \begin{bmatrix} \boldsymbol{\varphi}_1 & \ldots & \boldsymbol{\varphi}_M \end{bmatrix}^T\,, \tag{6.8}$$

and, hence, it is

$$\mathbf{y} = \boldsymbol{\Phi}\mathbf{x}\,. \tag{6.9}$$

The linear projections onto the M-dimensional subspace must fulfil certain conditions. The measurement matrix $\mathbf{\Phi}$ must be properly designed for a stable embedding of the information content of $\mathbf{x}$ in $\mathbf{y}$. There are several mathematical concepts in the literature to capture this idea.

To begin with, the coherence between $\mathbf{\Phi}$ and the sparsifying transform basis $\mathbf{\Psi}$ measures the largest correlation between the rows of $\mathbf{\Phi}$ and the columns of $\mathbf{\Psi}$, [Bar07]. CS requires incoherence between these matrices. A very important result is that random matrices are largely incoherent with any fixed basis $\mathbf{\Psi}$. Hence, elements of $\mathbf{\Phi}$ can be drawn independent and identically distributed (i.i.d.) from any subgaussian random distribution, e.g. Gaussian or Bernoulli. Without loss of generality, the columns of $\mathbf{\Phi}$ can be normalised.

Another concept is the Null Space Property for noise-free measurements [CDD08], but more versatile is the Restricted Isometry Property (RIP) because it is applicable to noisy measurements [CT05a; CRT06a; Bar+08]. Let $\mathcal{S}$ be a k-sparse support set taken from the integers $[1, N]$. Written as a lower index to a matrix or vector, a set selects only the columns respectively entries indicated by the elements in $\mathcal{S}$, i.e. $\mathbf{s}_\mathcal{S}$ is a vector of length k and $\mathbf{\Phi}_\mathcal{S}$ is an $M \times k$ matrix. A matrix $\mathbf{\Phi}$ satisfies the RIP of order k if there exists a constant $\delta_k \in [0, 1]$ such that

$$(1 - \delta_k) \left\|\mathbf{s}_\mathcal{S}\right\|_2^2 \leq \left\|\mathbf{\Phi}_\mathcal{S} \mathbf{s}_\mathcal{S}\right\|_2^2 \leq (1 + \delta_k) \left\|\mathbf{s}_\mathcal{S}\right\|_2^2 \tag{6.10}$$

holds for all sets $\mathcal{S}$ with $|\mathcal{S}| \leq k$. This property essentially requires that every set of columns with cardinality less than k approximately behaves like an orthonormal system, as [CRT06b] states. [Bar+08] proves that random matrices fulfil the RIP with overwhelmingly high probability. Expressed differently, the RIP ensures that CS measurements are isometric, or length-preserving, in every k-dimensional subspace.

6.1.3 The Compressed Sensing System Model

The CS framework basically consists of two consecutive computational steps: compressive measurement and algorithmic signal recovery.

As [Don06] puts it, the measurement can be described as the application of an information operator $I : \mathbb{R}^N \mapsto \mathbb{R}^M$ that samples M pieces of information about $\mathbf{x}$. Hereby $M < N$ must hold to be a compressive

measurement. Later, an unspecified algorithm $A : \mathbb{R}^M \mapsto \mathbb{R}^N$ offers an approximate reconstruction of $\mathbf{x}$.

The complete (noiseless) CS system model is, therefore,

$$\mathbf{y} = \mathbf{\Phi x} = \mathbf{\Phi \Psi s} = \mathbf{\Theta s}\,, \tag{6.11}$$

with the CS measurement (6.9) and the transform basis in (6.1). $\mathbf{\Theta} = \mathbf{\Phi \Psi}$ is the CS system matrix and the aforementioned information operator I. Furthermore, it is $k < M < N$. The ratio $M/N < 1$ is the subsampling factor and $k/N \ll 0.5$ the sparsity ratio. CS is also robust against noise. [CRT06b] showed that stable signal recovery is possible for noise-corrupted measurements

$$\mathbf{y} = \mathbf{\Theta s} + \mathbf{z}\,, \tag{6.12}$$

where $\mathbf{z}$ is a stochastic or deterministic unknown error term.

The direct recovery of $\mathbf{x}$ from the compressive linear measurements (y_m), $m = 1, ..., M$ in (6.9) is an ill-posed problem, since only M known values are available for N unknowns (underdetermined system of equations). Taking the sparse coding of $\mathbf{x}$ as in (6.2) into account, is, however, game-changing.

If the locations of the k-sparse support set were known a priori, the original signal could be recovered easily, because it would degenerate to a common least squares problem and more measurements than unknowns would be available. Given the knowledge of $\mathcal{S}$, $\mathbf{y} = \mathbf{\Theta}_{\mathcal{S}}\mathbf{s}_{\mathcal{S}}$ is uniquely solvable, but even so, this assumption contradicts the nonadaptivity of CS signal acquisition. Hence, CS recovery demands a combined support and data estimation.

As pointed out in [Bar07], classical compressive signal encoding, e.g. in image compression, requires the computation of the full basis transform after Nyquist-rate sampling, and subsequently identifies the most relevant coefficients while discarding the others.

In contrast to this, the CS measurement itself does not require any knowledge about the transform basis $\mathbf{\Psi}$ which simplifies the process. Furthermore, the $M < N$ compressive measurements can be taken directly at a reduced rate by a factor of M/N.

6.1.4 Signal Recovery

The CS framework is asymmetric compared to traditional compression techniques based on transform coding. Only for signal reconstruction is knowledge about the sparsifying transform basis mandatory.

The ultimate goal would be to recover the sparsest possible representation of a signal, i.e. to solve the optimisation problem [Don06; Bar07; JV11]

$$\min \|\mathbf{s}\|_0 \quad \text{subject to} \quad \boldsymbol{\Theta}\mathbf{s} = \mathbf{y} \tag{6.13}$$

with regard to an ideally sparse signal expressed by the ℓ_0-pseudonorm in (6.3). This, however, is known to be an NP-complete combinatorial problem, since it would require an exhaustive search over all $\binom{N}{k}$ possible support sets for a k-sparse signal. Sometimes this problem is also called *Best Subset Selection*.

CS experienced its breakthrough because it had been discovered that the above problem can be relaxed to a convex optimisation problem called *Basis Pursuit* (BP) [CDS01; CT05a]. The ℓ_0-pseudonorm is replaced by the ℓ_1-norm, and it becomes

$$\min \|\mathbf{s}\|_1 \quad \text{subject to} \quad \boldsymbol{\Theta}\mathbf{s} = \mathbf{y}\,. \tag{6.14}$$

This problem and the problem in (6.13) are equivalent for a k-sparse signal if $\boldsymbol{\Theta}$ fulfils the RIP. Similarly, signal recovery is possible for noisy measurements according to the model in (6.12), [CRT06b]. The solution is obtained by a convex quadratic optimisation problem known as *Basis Pursuit Denoising* (BPDN),

$$\min \|\mathbf{s}\|_1 \quad \text{subject to} \quad \|\mathbf{y} - \boldsymbol{\Theta}\mathbf{s}\|_2 < \varepsilon\,. \tag{6.15}$$

$\varepsilon \in \mathbb{R}^+$ is a parameter and should be chosen such that $\|\mathbf{z}\|_2 \leq \varepsilon$, [JV11]. An alternative formulation of BPDN is the Langrangian form,

$$\hat{\mathbf{s}} = \arg\min_{\mathbf{s}\in\mathbb{R}^N} \|\mathbf{y} - \boldsymbol{\Theta}\mathbf{s}\|_2^2 + \lambda \|\mathbf{s}\|_1\,. \tag{6.16}$$

with the Langrange multiplier λ. The Euclidean norm of $\mathbf{z} = \|\mathbf{y} - \boldsymbol{\Theta}\mathbf{s}\|$ is squared to be differentiable [JV11]. This corresponds to the Least Absolute Shrinkage and Selection Operator (LASSO) by Tibshirani [Tib96].

(If the signal of interest is the non-sparse $\mathbf{x}$, it can be obtained from the estimated $\hat{\mathbf{s}}$ by applying (6.2).)

In conclusion, every sparse signal can be recovered from noiseless or noisy CS measurements if the CS system matrix is RIP-fulfilling. This, in turn, is the case with very high probability for all random measurement matrices, as mentioned earlier.

Successful signal reconstruction is largely dependent on the number of measurements and the sparsity level. Analytically derived guarantees can be found in the literature [CT05a]. With overwhelming probability, a random sensing matrix obeys the RIP provided that $M \sim k \log(N)$, [CW08].

Alternatively, numerical simulations can demonstrate effective recovery for a certain setting (k, M, N, $\mathbf{\Phi}$, $\mathbf{\Psi}$). Donoho–Tanner phase transition diagrams map the two-dimensional parameter space k/M over M/N and divide it into two regions for successful and unsuccessful signal recovery [DT09].

6.2 Algorithms for Compressed Sensing Signal Recovery

As stated above, all sparsely representable signals can be recovered from CS measurements by solving either the BP or BPDN optimisation problem. However, (6.14) resp. (6.15) are optimisation principles, not algorithms.

In order to algorithmically obtain numerical solutions to these problems, hundreds of algorithms have been proposed within the literature so far, and a quite comprehensive list can be found, e.g., in [Car18; EK12]. Basically, these can be grouped into convex optimisation algorithms, greedy algorithms and thresholding algorithms, whereby only the first group returns optimal solutions.

6.2.1 Convex Optimisation Solvers

It lies in the nature of CS that convex optimisation is the method of choice to solve the signal recovery problem, because of the convex relaxation of the objective function (see previous section).

[CDS01] derives that BP can be rewritten as a linear program (LP) and, thus, efficiently be solved by polynomial-time algorithms. The same

is true for BPDN, which can be reformulated as a second-order cone program (SOCP).

Modern convex optimisation solvers use interior-point methods to obtain numerical solutions [BV04]. Common to these is that the intermediate solutions of each iteration follow along a central path within the interior of the multidimensional convex polytope, until the global optimum is found at the boundary of it. The projection of the N-dimensional but k-sparse ℓ_1-norm ball onto an M-dimensional subspace by the CS system matrix $\mathbf{\Theta}$ creates this convex polytope.

There are implementations of algorithms for Mathworks Matlab available, e.g. the ℓ_1-MAGIC toolbox by Candès and Romberg [CR05]. For the LP associated with BP, the primal-dual algorithm is used, taken from [BV04]. The SOCP for BPDN is implemented with the log-barrier method. Both implementations effectively solve the Newton steps within each iteration by the Conjugate Gradient (CG) method.

Another Matlab toolbox worth noting is CVX [GB17]. It is a modelling language based on Matlab for convex optimisation programs. Basically, it is only a frontend to a solver, which is SDPT3 (default) or SeDuMi. It does support non-differentiable objective functions like the ℓ_1-norm. To give an illustrative example, the CVX program for the BPDN problem in (6.15) would look like:

```
cvx_begin
    variable s(N)
    minimize( norm(s, 1) );
    subject to
        norm(y - THETA * s) < epsilon
cvx_end
```

The variables `y`, `THETA` and `epsilon` have to be defined beforehand of course.

Given the CS recovery guarantees and proven equivalence to the infeasible ℓ_0-minimisation, ℓ_1-minimisation returns mathematically optimal and exact solutions. Furthermore, the interior point methods for solving convex optimisation programs feature polynomial time complexity in $\mathcal{O}\left(M^2 N^{3/2}\right)$, albeit with a very large constant [BV04]. Therefore, they are not well-suited for hardware implementations.

At that level of abstraction, algorithms are devised for mathematical feasibility and functionality but without efficient hardware implementations in mind. The presence of computationally complex mathemat-

Algorithm 6.1: Matching Pursuit [MZ93].

```
function MP( y, Θ, k )
    r ← y; ŝ ← 0
    for t = 1, ..., k do
        c ← Θ^T r                          ▷ correlation
        λ ← arg max_ℓ |c_ℓ|                ▷ selection
        ŝ_λ ← ŝ_λ + c_λ                    ▷ update
        r ← r − Θ_λ^T c_λ                  ▷ residual
    return ŝ
```

ical operations like matrix factorisations, matrix inverses, vector norms, etc. are the general case. Even if such problems might be tackled, the large dimensions CS usually deals with pose another challenge to digital hardware design. For these reasons, suboptimal algorithms have been proposed which solve BP and BPDN approximately.

6.2.2 Greedy Algorithms

Greedy algorithms get their name from the way they estimate the support of the sparse vectors: greedily. That is, atom after atom is selected without revising that decision. A review of greedy algorithms can be found in [NTV08; BD08; EK12]. For the sake of brevity, only some algorithms can be mentioned here.

The simplest greedy algorithm is *Matching Pursuit* (MP), [MZ93]. It recovers a signal $\mathbf{x}$ accurately if it can be described fully by elements of a dictionary, i.e. $\mathbf{\Theta}$ for a CS system. The solution vector $\hat{\mathbf{s}}$ is k-sparse. MP iterates k times and selects during each iteration the atom with index λ which has the largest correlation magnitude to the residual, which matches best. Afterwards, the residual is updated. Nonetheless, however, it is possible that MP selects already selected atoms again.

Orthogonal Matching Pursuit (OMP) improves on this aspect by introducing a LS orthogonalisation step. It was first published in [PRK93] and applied to CS in [TG07]. The pseudocode is listed in Alg. 5.1 on page 82 with the minor difference in notation that $\mathbf{x}$ is meant to be the sparse $\mathbf{s}$ in the transform domain. Like MP, OMP selects one atom per iteration but computes the optimal signal approximation with the so far selected atoms

in the sense of minimal residual energy. Due to this orthogonalisation, atoms will never be re-selected and the end result $\mathbf{s}$ will have exactly k non-zero entries. More details on hardware architectures for OMP can be found in Sec. 5.2. Signal recovery with OMP is fast compared to ℓ_1-minimisation and features an algorithmic complexity of $\mathcal{O}(kMN)$. Yet, this recovery guarantee is only proven for fixed signal and measurement matrix constellations [NTV08].

Hence, *Regularised Orthogonal Matching Pursuit* (ROMP) has been developed which is the first algorithm with a sparse recovery guarantee for all sensing matrices satisfying the RIP. It was superseded by *Compressive Sampling Matching Pursuit* (CoSaMP). Both algorithms select multiple atoms at once and augment the already selected support set accordingly; CoSaMP additionally introduces a pruning step to be able to discard wrongly selected indices later on [NTV08].

Gradient Pursuit (GP) algorithms are conceptually slightly different but still very similar to MP-type algorithms [BD08]. Line 6.1 in Alg. 6.1 is replaced for GP by directional updates

$$\hat{\mathbf{s}} \leftarrow \hat{\mathbf{s}} + a\mathbf{d}\,, \tag{6.17}$$

where a is the step size and $\mathbf{d}$ the update direction (vector), which have to be computed beforehand during each iteration. MP and OMP are special cases for certain choices of $a, \mathbf{d}$.

6.2.3 Thresholding Algorithms

Another family of suboptimal algorithms for CS signal recovery are thresholding algorithms [EK12]. Instead of explicitly selecting atoms, these algorithms operate with a threshold which enforces sparsity, i.e. values below a certain threshold are set to zero.

Alg. 6.2 lists *Iterative Hard Thresholding* (IHT), [BD09]. The for-loop iterates until a stopping criterion is met, e.g. a previously specified maximal number of iterations $t_{\max}$. $\mathcal{H}_k(\cdot)$ is the threshold operator which returns a k-sparse vector, retaining only the k elements of largest magnitude. $\mathbf{\Theta}^T\mathbf{r}$ is the update direction with step size μ to control convergence.

Alternatively, a scalar soft threshold operator $\eta_\theta(\cdot)$ can be employed which, if applied to a vector, sets all values below a real parameter θ to

Algorithm 6.2: Iterative Hard Thresholding [BD09].

function IHT($\mathbf{y}, \mathbf{\Theta}, k, \mu$)
  $\mathbf{r} \leftarrow \mathbf{y}; \hat{\mathbf{s}} \leftarrow 0$
  for $t = 1, ..., t_{\max}$ **do**
    $\hat{\mathbf{s}} \leftarrow \mathcal{H}_k \left(\mathbf{x} + \mu \mathbf{\Theta}^T \mathbf{r} \right)$ ▷ hard threshold
    $\mathbf{r} \leftarrow \mathbf{y} - \mathbf{\Theta}\hat{\mathbf{s}}$ ▷ residual
  return $\hat{\mathbf{s}}$

Algorithm 6.3: Approximate Message Passing [DMM09; Mae+12].

function AMP($\mathbf{y}, \mathbf{\Theta}, \lambda$)
  $\mathbf{r} \leftarrow \mathbf{y}; \hat{\mathbf{s}} \leftarrow 0$
  for $t = 1, ..., t_{\max}$ **do**
    $\theta \leftarrow \lambda \left\| \mathbf{r} \right\|_2 / \sqrt{M}$
    $\hat{\mathbf{s}} \leftarrow \eta_\theta(\mathbf{x} + \mathbf{\Theta}^T \mathbf{r})$ ▷ soft threshold
    $\mathbf{r} \leftarrow \mathbf{y} - \mathbf{\Theta}\hat{\mathbf{s}} + \mathbf{r} \left\| \hat{\mathbf{s}} \right\|_0 / M$ ▷ residual
  return $\hat{\mathbf{s}}$

zero and reduces the others in magnitude. It is defined as

$$\eta_\theta(u) = \begin{cases} u - \theta & u \geq \theta \\ u + \theta & u \leq -\theta \\ 0 & \text{else} \end{cases} \tag{6.18}$$

and a replacement of $\mathcal{H}_k$ with η_θ gives *Iterative Soft Thresholding* (IST). This algorithm, though, is slow in convergence.

A significantly improved soft-thresholding algorithm is *Approximate Message Passing* (AMP), which is derived from message passing on graphical models [DMM09]. The pseudocode is given in Alg. 6.3. Within each iteration, the parameter θ is updated. It scales with the residual error, a regularisation parameter λ and M, the number of compressive measurements. This update strategy is taken from [Mae+12], which presents a VLSI implementation of AMP. Critical to the success of AMP is the residual update prescription with the added term $\mathbf{r} \left\| \hat{\mathbf{s}} \right\|_0 / M$. The residual is not only dependent on the current sparse signal estimate $\mathbf{s}$ but also on the

residual information of the previous iteration. This leads to an improved noise suppression, and optimally tuned AMP has a formal mean squared error (MSE) evolving to zero exponentially fast, if recovery is possible in the first place. AMP is the only algorithm discussed here which manages to empirically approach the performance of ℓ_1-minimisation with respect to the Donoho–Tanner phase transition [DMM09].

The greatest advantage of thresholding-type algorithms is their applicability to very large recovery problems where ℓ_1-minimisation might already be infeasible. They require matrix-vector multiplications with $\mathbf{\Theta}$ and $\mathbf{\Theta}^T$ and have an algorithmic complexity of $\mathcal{O}(MN)$ per iteration, if $\mathbf{\Theta}$ is unstructured [BD09].

6.2.4 Discussion

As the previous explanations clearly show, ℓ_1-minimisation constitutes the gold standard for CS signal recovery. Ready-to-use software implementations exist, and they are certainly powerful tools for a mathematical numerical analysis. Notably, only the measurement vector $\mathbf{y}$ and the CS system matrix $\mathbf{\Theta}$ are inputs to BP, which is a very important property of ℓ_1-minimisation. A priori knowledge of the underlying sparsity k/N is not required, nor are other parameters. ℓ_1-minimisation methods, however, do not come off very well with regard to hardware friendliness or total run time.

Therefore, many algorithms have been proposed which solve the recovery problem approximatively. Greedy algorithms feature a deterministic run time (fixed number of iterations) and a correlation-based atom selection. OMP improves on MP, and CoSaMP improves on OMP, yet OMP only requires three inputs $\mathbf{y}, \mathbf{\Theta}, k$ and is thus easy to configure. Furthermore, mathematically proven recovery guarantees exist. These factors explain the huge popularity of OMP.

For very large recovery problems though, e.g. in image processing, IHT and its relatives become preferential because of their simpler structure. They do not need LS computations, but typically require hundreds of loop iterations instead. And depending on the application, a suitable stopping criterion must be found. Additional algorithmic parameters for the soft threshold or the step size are necessary for a successful application.

Every algorithm discussed so far operates on a value continuum. If further constraints are imposed on $\mathbf{x}$, alternatives arise. This will be part of the following sections.

6.3 The Application of Compressed Sensing to Wireless Communications

Although the major milestones in CS research have been reached some years ago, the development of the CS framework raised the awareness for sparse codings in many areas of research, which still persists. One particular area with lots of applications is communications theory.

Examples for communication application are direction of arrival (DoA) estimation of radar waves, spectrum monitoring for cognitive radios or sparse channel estimation in Massive MIMO systems, just to name a few. Another possibility, which is being discussed among other concepts, is Sparse-Coded Multiple Access (SCMA) for the realisation of the so-called IoT in future 5G mobile networks [Nik+14]. A precursor to this idea is the wireless communications scenario outlined in the following.

6.3.1 A Multi-User Uplink in a Wireless Sensor Network

In the remainder of this dissertation, a multi-user uplink scenario within a wireless sensor network (WSN), as depicted in Fig. 6.1, shall serve as a baseline communication application. It resembles the currently discussed ideas for massive MTC in future mobile networks in many ways. In contrast to human-driven network traffic for speech or video signal transmissions, which usually are continuous data streams of comparatively high data rates, MTC assumes a large amount of "users", i.e. machines, which transmit information intermittently or sporadically with small packet sizes [LCL11].

According to the IEEE 801.15.4 standard, such a wireless data link could be achieved by using the Direct Sequence Spread Spectrum (DSSS) technique to transmit over the physical channel with a CSMA with collision avoidance (CSMA-CA) medium access protocol [IEEE-802]. This does, however, not allow for simultaneous transmission of multiple users. Generally, traditional multi-user communication is incapable of differentiating between active and inactive users. Transmission activity must be made known beforehand, requiring additional signalling overhead on upper layers of the OSI/ISO reference model, especially the Medium Access Control (MAC) layer [ITU-X.200].

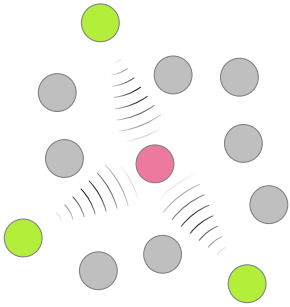


Figure 6.1: A wireless sensor network (WSN) in star topology with a central data aggregation node. The sensor nodes transmit data sporadically, therefore only a subset is simultaneously active.

Given the multi-user uplink setting, the canonical linear input-output model in symbol clock is [Ver98; Küh06]

$$\mathbf{y} = \mathbf{T}\mathbf{x} + \mathbf{n}\,, \tag{6.19}$$

where $\mathbf{x}$ is the multi-user transmit signal, x_n being the n-th user's data symbol ($n = 1, ..., N$), $\mathbf{T} \in \mathbb{C}^{M \times N}$ a matrix comprising all channel effects (multi-user access scheme, slow and fast channel fadings, path losses, etc.), $\mathbf{y}$ the superimposed received signal, and $\mathbf{n}$ an additive noise vector with its elements drawn from a circularly-symmetric complex white Gaussian noise process. Details on these entities will be explained below.

Please note the similarity to the noisy CS system model equation (6.12). This suggests a disruptive, novel approach to communication systems if sparsity as a property can be introduced into the transmit vector $\mathbf{x}$.

It can.

6.3.2 The Introduction of Artificial Sparsity

CS is applicable to systems where the signal of interest is sparse or can be described with an orthogonal basis transform by a sparse coefficient vector (see Eq.(6.2) on page 102). Often, sparsity is a naturally occurring property of signals, such as in images or videos.

Nonetheless, though, it also possible to create artificially sparse signals. With regard to the above described WSN, this can be realised by modelling inactive nodes as active but transmitting a symbol devoid of information [ZG09; TLL09].

Active users in the classical sense transmit information-carrying symbols taken from a digital modulation alphabet $\mathcal{A}$. Usually this is a 2^b-ary Phase-Shift Keying (PSK) or Quarternary Amplitude Modulation (QAM) signal constellation, e.g. for Binary Phase-Shift Keying (BPSK) it is $\mathcal{A}^{\text{BPSK}} = \{-1, 1\}$. The obvious choice for the symbol devoid of information is the zero symbol. Then, the modulation alphabet can be augmented to $\mathcal{A}_0$ by a constellation point at the origin (zero) of the complex plane,

$$\mathcal{A}_0 = \mathcal{A} \cup \{0\} \,. \tag{6.20}$$

The notation is chosen here in analogy to the set of natural numbers including the zero, commonly denoted as $\mathbb{N}_0$. The multi-user transmit vector $\mathbf{x}$ in (6.19) is therefore a vector in $\mathcal{A}_0^N$: if the majority of sensor nodes are inactive, this vector is sparse. The non-zero positions of $\mathbf{x}$, i.e. $\operatorname{supp}(\mathbf{x})$, signify active transmission of the corresponding users. Most noteworthy,

$$\|\mathbf{x}\|_2 = \|\mathbf{x}\|_1 = \|\mathbf{x}\|_0 \tag{6.21}$$

holds for all constant modulus signal constellations, namely for every PSK and furthermore for the Quarternary Phase-Shift Keying (QPSK) alphabet [TLL09]. Constant modulus implies that every symbol has the same magnitude, or $|a_i| = 1 \; \forall a_i \in \mathcal{A}$.

6.3.2.1 User Activity Probabilities

In a real-world scenario, user activities are random. Therefore, let p_a be the activity probability (data transmission probability) of a user. Then, the user's activity implicitly follows a Bernoulli distribution with the discrete probability mass function (PMF)

$$\Pr(|x_n|) = p_X(|x_n|) = \begin{cases} 1 - p_a, & |x| = 0 \\ p_a, & |x| \neq 0 \end{cases} . \tag{6.22}$$

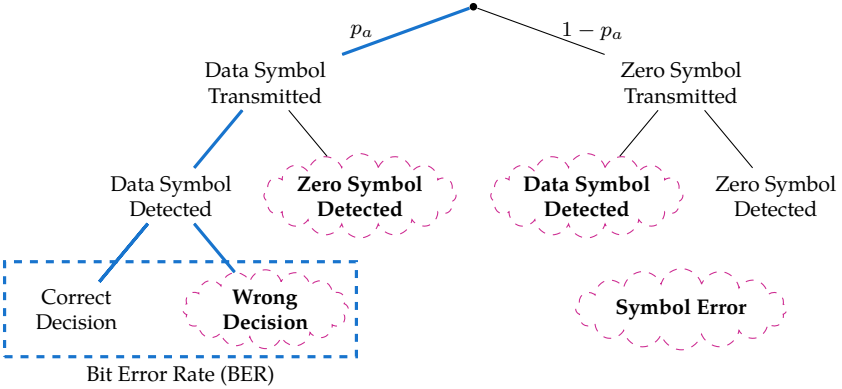


Figure 6.2: The gross symbol error rate (GSER) as a figure of merit accounts for user activities within the detection problem.

With regard to the zero-augmented signal constellation $\mathcal{A}_0$ it is

$$\Pr(x_n) = p_X(x_n) = \begin{cases} 1-p_a, & x_n = 0 \\ p_a/2^b, & x_n \neq 0 \end{cases}, \tag{6.23}$$

where 2^b is the modulation order which equates to the cardinality of the set of constellation points $|\mathcal{A}|$. In the following, a homogeneous activity model is adopted, i.e. all users within the WSN are active with the same probability p_a, although an extension to inhomogeneous user activities is easily possible.

Signal detection, i.e. the estimation of $\hat{\mathbf{x}}$ from the received symbol vector $\mathbf{y}$ such that $\|\hat{\mathbf{x}} - \mathbf{x}\| \to 0$, must be executed with respect to user activities. Given the PMF (6.22) and the fact that $x_n \in \mathcal{A}_0$, there are $p_a N$ active users on average and the multi-user transmission vector $\mathbf{x}$ is $p_a N$-sparse (see (6.4)).

Additionally, it follows from (6.23) that transmit symbols taken from $\mathcal{A}_0$ are not uniformly distributed. Hence, optimal signal detection must incorporate this knowledge of the a priori statistics to maximise the

a posteriori probabilities, i.e. perform maximum a posteriori (MAP) detection.

Fig. 6.2 emphasises the fact that the classical bit error rate (BER) is not suited as detection metric since it neglects user activities; a priori transmission activity and subsequent correct activity detection are implicitly assumed. Hence, the detection quality will be measured in the following by the GSER as a figure of merit. This will be plotted over the SNR whereby the relationship between E_b/N_0 and SNR is given (in linear scale) by [Kno+13]

$$E_b/N_0 = \frac{\mathrm{SNR}}{p_a \,\mathrm{ld}(|\mathcal{A}|)} = \frac{\mathrm{SNR}}{p_a b} \,. \tag{6.24}$$

One could further define error events for false active and false inactive detections as in [Mon+12]. This requires that these error classes be given a weighting of their severity which in turn necessitates additional assumptions about the underlying communication system. For the sake of simplicity, this differentiation is omitted here since the GSER as a neutral figure of merit does encompass all error classes (symbol detection errors, false active detections and false inactive detections). This is depicted as cloudy boxes in Fig. 6.2.

6.3.3 Complex-Valued Compressed Sensing

All explanations on CS assumed real-valued entities so far. The fundamental papers actually do not make any remarks on a complex-valued use case. However, the mathematical formulations of the CS framework as well as of the algorithms for signal recovery naturally extend to complex numbers [Par+17].

Nonetheless, complex-valued arithmetic operations have to be decomposed into real-valued ones. In particular, an equation of the form

$$\mathbf{y} = \mathbf{A}\mathbf{b} \tag{6.25}$$

can be decomposed, as differentiated in [WP12], by two kinds of RVDs: either by the Block-Wise Real-Value Decomposition (BRVD) or by the

Element-Wise Real-Value Decomposition (ERVD). For the following considerations, let

$$\mathrm{RVD}(\mathbf{A}) \rightarrow \begin{bmatrix} \mathrm{Re}\{\mathbf{A}\} & -\mathrm{Im}\{\mathbf{A}\} \\ \mathrm{Im}\{\mathbf{A}\} & \mathrm{Re}\{\mathbf{A}\} \end{bmatrix} \tag{6.26}$$

be the real-value decomposition operator, which can be applied to a matrix $\mathbf{A} \in \mathbb{C}^{M \times N}$. This includes the corner cases of $\mathbf{A}$ being a vector or scalar.

6.3.3.1 Block-Wise Real-Value Decomposition

A first work on complex-valued CS for the DoA detection of radar signals, [End10], proposed to apply the well-known BRVD. The BRVD of (6.25) is

$$\mathrm{RVD}(\mathbf{y}) = \mathrm{RVD}(\mathbf{A})\,\mathrm{RVD}(\mathbf{b}) \tag{6.27}$$

and results in

$$\begin{pmatrix} \mathrm{Re}\{\mathbf{y}\} \\ \mathrm{Im}\{\mathbf{y}\} \end{pmatrix} = \begin{bmatrix} \mathrm{Re}\{\mathbf{A}\} & -\mathrm{Im}\{\mathbf{A}\} \\ \mathrm{Im}\{\mathbf{A}\} & \mathrm{Re}\{\mathbf{A}\} \end{bmatrix} \begin{pmatrix} \mathrm{Re}\{\mathbf{b}\} \\ \mathrm{Im}\{\mathbf{b}\} \end{pmatrix}, \tag{6.28}$$

where the second columns of the decomposed $\mathbf{y}$ and $\mathbf{b}$ have been deleted for being redundant. Note, the dimensions of the vectors and the matrix have doubled. Additionally and of utmost importance, the real parts and their corresponding imaginary parts become decoupled. If these are not independent of each other, further optimisation constraints for successful signal reconstruction are necessary [YKM12].

With regard to the mentioned radar detection, a list of various possibilities to enforce a smooth phase is given in [End10; YKM12]. And with respect to the detection of a zero-augmented digital modulation alphabet $\mathcal{A}_0$ it has to be taken into account that real and imaginary parts have to be detected jointly as zero or non-zero.

The concept of (arbitrary) jointly sparse signals has first been introduced in [Dua+05]. Here, the complex-valued signal could basically be seen as two independent real-valued signals with common locations of the non-zero elements, and, hence, two equivalent real-valued CS problems are obtained [Par+17].

6.3.3.2 Element-Wise Real-Value Decomposition

Alternative to the above block-wise RVD is the element-wise RVD, the latter being inherent to complex numbers. It has already been remarked in Sec. 4.2 that a complex-valued multiplication can be computed by four real-valued multiplications and two real-valued additions/subtractions. Let $z = u \cdot v$ be a complex-valued multiplication. Re-interpreted as a two-dimensional real vector, the first dimension being the real part and the second dimension the imaginary part, this multiplication can be formulated in matrix notation as

$$\begin{pmatrix} \operatorname{Re}\{z\} \\ \operatorname{Im}\{z\} \end{pmatrix} = \begin{bmatrix} \operatorname{Re}\{u\} & -\operatorname{Im}\{u\} \\ \operatorname{Im}\{u\} & \operatorname{Re}\{u\} \end{bmatrix} \begin{pmatrix} \operatorname{Re}\{v\} \\ \operatorname{Im}\{v\} \end{pmatrix} . \tag{6.29}$$

Note, this equation is identical to (6.27) when restricted to complex-valued *scalars* only.

Hence, in [AA09] the RVD operation was applied *element-wise* to all entities. This leads to the ERVD of (6.25) given by

$$\begin{pmatrix} \operatorname{Re}\{y_1\} \\ \operatorname{Im}\{y_1\} \\ \vdots \\ \operatorname{Re}\{y_M\} \\ \operatorname{Im}\{y_M\} \end{pmatrix} = \begin{bmatrix} \operatorname{RVD}(a_{11}) & \cdots & \operatorname{RVD}(a_{1N}) \\ \vdots & & \vdots \\ \operatorname{RVD}(a_{M1}) & \cdots & \operatorname{RVD}(a_{MN}) \end{bmatrix} \begin{pmatrix} \operatorname{Re}\{x_1\} \\ \operatorname{Im}\{x_1\} \\ \vdots \\ \operatorname{Re}\{x_N\} \\ \operatorname{Im}\{x_N\} \end{pmatrix} . \tag{6.30}$$

It is obvious how this ERVD can be applied to the CS model (6.11). In connection with the RDAM, this element-wise decomposition is performed transparently to the designer by the utilisation of specialised complex-valued data types (see Sec. 4.2).

Thus, the complex-valued problems can very well be solved by CS, as they arise, e.g., when PSK or QAM signal constellations are employed with modulation order 2^b, $b > 1$.

Since the ERVD is *implicit*—unlike the BRVD—additional optimisation constraints are not required. For instance, the ℓ_0-pseudonorm (6.3) is well-defined in the complex case; if the magnitude of a complex scalar becomes zero, real and imaginary parts vanish together.

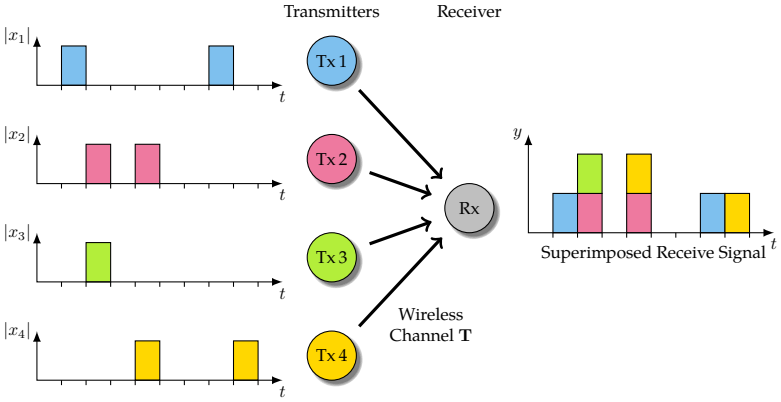


Figure 6.3: Multiple sensor nodes in a time-synchronous DS-CDMA system transmit spread data symbols sporadically and simultaneously. The data aggregation node receives the superposition of the transmit signals in order to perform for multi-user detection with regard to data and activities.

6.3.4 Code-Division Multiple Access for Multi-User Communications

A time-synchronous Direct Sequence CDMA (DS-CDMA) system will be assumed further to enable true multi-user communication. This overcomes the deficit of the IEEE Standard 802.15.4 and DSSS not to support concurrent data transmissions of multiple users, as mentioned in Sec. 6.3.1 above. In fact, the spread spectrum technique is the foundation of every CDMA system as well.

Fig. 6.3 shows such a communication system schematically. Multiple sensor nodes time-synchronously transmit constant-modulus data symbols over a wireless channel. This system model setup follows the ideas set forth in [ZG09; TLL09; ZG11].

Each symbol $x_n \in \mathcal{A}_0$ is spread with a user-specific spreading code $\mathbf{c}_n$ of length N_s, being a complex-valued series in general. The resulting scalar multiplication results in N_s chips $x_n \cdot (c_k)_n$, $k = 1, ..., N_s$, which are transmitted over a wireless channel modelled by a linear time-invariant (LTI) impulse response $\mathbf{h}_n$ of length L_h chips.

A very important requirement for CDMA systems to function is that user-specific code sequences must be orthogonal to each other, or formally

$$\langle \mathbf{c}_i, \mathbf{c}_j \rangle = \mathbf{c}_i^H \mathbf{c}_j = 0 \quad \forall i \neq j \text{ and } i, j = 1, ..., N \,. \tag{6.31}$$

If the user codes, without loss of generality, are normalised to $\|\mathbf{c}_n\|_2 = 1$, $n = 1, ..., N$, and written as columns of a code matrix $\mathbf{C}$ as

$$\mathbf{C} = \begin{bmatrix} \mathbf{c}_1 & \cdots & \mathbf{c}_N \end{bmatrix} \,, \tag{6.32}$$

they form an orthonormal transform basis, and it follows $\mathbf{C}^H\mathbf{C} = \mathbf{I}$. Depending on the application, it might be sufficient to demand quasi-orthogonality, $\langle \mathbf{c}_i, \mathbf{c}_j \rangle \approx 0$. For example: In case of an imperfect synchronisation of the symbol clock, quasi-orthogonal pseudo-noise (PN) sequences perform better than strictly orthogonal (but unsynchronised) Walsh-Hadamard codes with regard to multi-user interference and inter-symbol interference (ISI), because of statistically lower cross-correlations. For this reason, classical CDMA systems are often built based on PN sequences and cannot be fully loaded, i.e. $N < N_s$, to still allow for a successful multi-user detection.

The transmission of the spread user symbols over the wireless fading channels $\mathbf{h}_n$ can formally be described by the so-called signatures $\mathbf{s}_n$ of each user [Küh06]. The signatures are the convolution product

$$\mathbf{s}_n = \mathbf{h}_n * \mathbf{c}_n \,, \tag{6.33}$$

of length $N_s + L_h - 1$, and the operator $*$ denotes convolution of two one-dimensional signals. With regard to the n-th user only, the non-circular convolution of two time-limited discrete signals is given by

$$s[i] = \sum_{k=0}^{L_h - 1} h[k]c[i-k] \quad i = 1, ..., N_s + L_h - 1 \,. \tag{6.34}$$

Alternatively, the convolution operation can be expressed by the discrete-time convolution matrix $\mathbf{H}_n$ of dimensions $N_s + L_h - 1 \times N_s$, which is of Toeplitz structure. Then, $\mathbf{s}_n = \mathbf{H}_n\mathbf{c}_n$.

Analogously to (6.32), the signatures can be assembled as columns of a matrix, the signatures matrix

$$\mathbf{S} = \begin{bmatrix} \mathbf{s}_1 & \mathbf{s}_2 & \cdots & \mathbf{s}_N \end{bmatrix} \in \mathbb{C}^{N_s+L_h-1 \times N} \,. \tag{6.35}$$

Hence, the central data aggregation node of the assumed WSN setup, Sec. 6.3.1, receives under consideration of (6.19) the chip-clock signal

$$\begin{aligned} \mathbf{y} &= \mathbf{S}\mathbf{x} + \mathbf{n} && (6.36) \\ &= \begin{bmatrix} \mathbf{H}_1\mathbf{c}_1 & \cdots & \mathbf{H}_N\mathbf{c}_N \end{bmatrix} \mathbf{x} + \mathbf{n} \,. && (6.37) \end{aligned}$$

Just as a reminder, $\mathbf{x} \in \mathcal{A}_0^N$ is ideally sparse and $\mathbf{n}$ is complex additive white Gaussian noise (AWGN) with variance σ_n^2.

6.3.5 The Compressed Sensing Estimation Problem

The wireless CDMA multi-user uplink described in the previous sections fulfils all requirements to qualify as a CS problem:

- **Sparsity.** The multi-user transmit vector $\mathbf{x} \in \mathcal{A}_0^N$ is sparse.
- **Transform Basis.** The transform basis $\mathbf{\Psi}$ of (6.1), normally causing sparsification, is simply the identity matrix $\mathbf{I}$, because $\mathbf{x}$ is already ideally sparse.
- **Measurement Matrix.** The CS measurement matrix $\mathbf{\Phi}$, Eq. (6.8), corresponds to $\mathbf{T}$ in (6.19) in general. Thus, the CS system matrix is $\mathbf{\Theta} = \mathbf{\Phi}\mathbf{\Psi} = \mathbf{S}$ for the above application.
- **Restricted Isometry Property.** $\mathbf{C}$ is constructed by the user spreading codes, which can be drawn from any subgaussian random distribution. Therefore, the RIP is fulfilled with high probability (w.h.p.), see Sec. 6.1.2.

  This carries over to $\mathbf{S}$ due to its construction rule in (6.35). Each $\mathbf{s}_n$ is a convolution product (6.34), i.e. each sample of it is in fact a linear combination of random variables. The distribution of the s_i's is certainly nontrivial to analyse. However, let the elements of $\mathbf{c}_n$ be i.i.d. normally distributed with zero mean and unit variance, and assume the user-specific channel coefficients to be constants. Then, every s_i is a weighted sum of L_h normal distributions, which in

turn results in a normal distribution with zero mean and variance $\sigma^2 = |h_1|^2 + ... + |h_{L_h}|^2$, apart from edge effects due to the non-circular nature of the convolution. Since each user has its own random channel realisation and signatures are thus i.i.d. as well, $\mathbf{S}$ will fulfill the RIP w.h.p., too. For more complicated random distributions the central limit theorem is applicable, although in general the CS detection performance can also be evaluated with numerical simulations for a specific setup.

For the symbol detection problem, two cases have to be differentiated:

- **The underdetermined system of equations ($M < N$)**

  This truly constitutes a CS system, since $\mathbf{\Theta}$ performs only M compressive measurements of N transmitted user symbols. Here, the knowledge about the sparse multi-user transmit vector is critical to a successful detection, hence, it could be called an optimisation problem with *sparsity exploitation*.

  The communication system is overloaded in the classical sense, which renders it resource-efficient. Signal recovery must be performed with ℓ_1-minimisation or alternative CS algorithms.

- **The overdetermined system of equations ($M \geq N$)**

  Classical CDMA systems are overdetermined to facilitate multi-user symbol detection or to employ strictly orthogonal codes, like Walsh–Hadamard sequences. Hence, $M = N_s + L_h - 1 \geq N$, i.e. more measurements than unknowns are available.

  This is by no means "compressive" in nature, nonetheless the signal features the sparsity property. Hence, symbol detection can be performed with *sparsity awareness*. Consequently, the multi-user communication system can be overloaded with a factor $1/p_a > 1$ such that effectively up to N/p_a users can potentially be supported. Symbol detection becomes a regression problem solvable by ℓ_2-minimisation (least squares) regularised by the sparse coding. Since $M \geq N$ and $\mathbf{S}$ being (approximately) orthogonal or even unitary with normalised columns, the Gram matrix $\mathbf{S}^H\mathbf{S}$ has full rank and the inverse problem is well-conditioned.

6.3.5.1 The Finite-Alphabet Constraint

Note, the symbols of multi-user transmit vector, $\mathbf{x} \in \mathcal{A}_0$, are drawn randomly from a finite set of PSK constellation points with a PMF according to (6.23). The cardinality of this set is $|\mathcal{A}_0|$, but more importantly $\mathcal{A}_0$ is not convex because it is discrete set. Therefore, CS recovery with BPDN is not directly applicable, yet there are two possibilities to overcome this issue.

Until now, all CS signal recovery algorithms discussed in Sec. 6.2 operate on real numbers, a value continuum. Thus, the non-convex $\mathcal{A}_0$ could be relaxed to a convex set. The convex hull $\mathrm{conv}(\mathcal{A}_0^{\mathrm{QPSK}})$ to the zero-augmented QPSK constellation, e.g., is a square with $\sqrt{2}$ side length. This technique is therefore called *box relaxation*. ℓ_1-minimisation can be applied, and subsequently the recovered signal $\tilde{\mathbf{x}} \in \mathbb{C}^N$ can be quantised with respect to $\mathcal{A}_0$, $\hat{\mathbf{x}} = \mathrm{Q}_{\mathcal{A}_0}\{\tilde{\mathbf{x}}\}$.

Alternatively, a different class of algorithms from integer optimisation theory can directly account for the finite-alphabet constraint, as will be shown later. These solve the ℓ_0-regularised LS recovery problem

$$\hat{\mathbf{x}} = \arg \min_{\mathbf{x} \in \mathcal{A}_0^N} \|\mathbf{y} - \mathbf{T}\mathbf{x}\|_2^2 + \tilde{\lambda} \|\mathbf{x}\|_0 \,. \tag{6.38}$$

The authors of [ZG11] call this Sparsity MAP (S-MAP) detection. The factor $\tilde{\lambda}(\mathcal{A}_0, p_a, \sigma_n)$ is determined by the prior knowledge about the source data and is a function of the signal constellation, user activity probability and noise power. A precise derivation will be given in Sec. 7.1.

The constant-modulus assumption, i.e. all symbols of the modulation alphabet have the same magnitude, allows to substitute $\|\mathbf{x}\|_0 = \|\mathbf{x}\|_2^2$ according to (6.21) in Sec. 6.3.2, [TLL09]. Then, the regularisation term can be included into the LS metric [Kno+14]. The hereby reformulated minimisation problem becomes

$$\hat{\mathbf{x}} = \arg \min_{\mathbf{x} \in \mathcal{A}_0^N} \|\mathbf{y} - \mathbf{T}\mathbf{x}\|_2^2 + \|\sqrt{\lambda} \cdot \mathbf{x}\|_2^2 \tag{6.39}$$

$$= \arg \min_{\mathbf{x} \in \mathcal{A}_0^N} \left\| \begin{bmatrix} \mathbf{y} \\ \mathbf{0}_N \end{bmatrix} - \begin{bmatrix} \mathbf{T} \\ \sqrt{\lambda} \cdot \mathbf{I}_N \end{bmatrix} \mathbf{x} \right\|_2^2 \tag{6.40}$$

$$= \arg \min_{\mathbf{x} \in \mathcal{A}_0^N} \|\tilde{\mathbf{y}} - \tilde{\mathbf{T}}\mathbf{x}\|_2^2 \,. \tag{6.41}$$

Naturally, this increases the dimensions of $\tilde{\mathbf{y}}$ and $\tilde{\mathbf{T}}$ to a $(M+N)$ vector and a $(M+N)\times N$ regularised matrix, respectively. $\tilde{\mathbf{T}}$ contains activity information as well as the wireless communication channel $\mathbf{T}$. If $\lambda > 0$ holds, $\tilde{\mathbf{T}}$ has full rank. However, solving (6.41) may lead to numerical difficulties when λ approaches zero.

Since the model application is a low-rate WSN, the assumption of constant modulus signal constellations is sensible. This includes the 4-QAM and all PSK modulations. The consideration of higher order QAM modulations is not within the scope of this thesis. A simple approximative approach would be to normalise λ by the expected value, i.e. mean power, of the employed finite alphabet symbols. For more developed theories the reader is referred, e.g., to [LL16], where a multistage (hierarchical) binary encoding based on 4-QAM is proposed in connection with a channel encoder, or to [Ais+15], where the 2^b constellation symbols are mapped onto one-hot coded binary vectors of length 2^b. [LML17] follows the approach mentioned above to apply a BRVD and a relaxed ℓ_1-minimisation with consecutive quantisation to the finite alphabet.

In conclusion, Eq. (6.38) and (6.41) constitute a joint multi-user symbol and activity detection on the physical layer (PHY) of the ISO/OSI model, when translated to communications parlance. This will be the subject of the following chapter.

Chapter 7

Sparsity-Aware Symbol Detection with Tree Search Algorithms

Based upon the previously described multi-user uplink scenario in a WSN with sporadically active sensor nodes, this chapter discusses adaptations of tree search algorithms to take advantage of the system-inherent sparsity with faithfulness to the finite-alphabet constraint.

There are, at least, two remarkable points to observe: Firstly, a joint data and activity detection renders additional activity signalling overhead on higher protocol layers beforehand obsolete (see Sec. 6.3.1). This is an efficient cross-layer approach, which spans the PHY and MAC layers. Secondly, the CDMA system can be overloaded such that the average number of active users, $p_a N$, equals the spreading factor N_s—the number of total users, i.e. active and inactive, is larger than N_s. This increased support for more sensor nodes is desirable with regard to the IoT.

The general ideas on this sparse communications scenario have been laid out and discussed in [ZG09; TLL09]. Following from the a priori distribution of transmit symbols, the optimal receiver will maximise the a posteriori probability. This necessitates the solving of the joint data and activity detection problem (6.38). Below, three closely related tree search algorithms will be adapted to the problem at hand, namely SD in Sec. 7.2.1, K-Best in Sec. 7.2.2 and SIC in Sec. 7.2.3. Publications related to these are [KWP12], [Kno+14] and [Kno+13], respectively.

7.1 Joint Data and Activity Detection

A MAP decoding criterion can be derived from the a priori knowledge of the stochastic distribution of transmit symbols (6.23) and the system model (6.36) [KWP12]. Here, the derivation is given for PSK symbol alphabets of any modulation order, and not restricted to real-valued BPSK.

Generally speaking, every MAP detector finds the event with maximal probability given a certain observation,

$$\hat{\mathbf{x}} = \arg \max_{\mathbf{x} \in \mathcal{T}_0^N} \Pr(\mathbf{x}|\mathbf{y}) \,, \tag{7.1}$$

i.e. evaluating the conditional probability $\Pr(\mathbf{x}|\mathbf{y})$ for each possible transmit vector $\mathbf{x} \in \mathcal{A}_0^N$ and selecting the most probable. With Bayes' rule this can be rewritten as

$$\hat{\mathbf{x}} = \arg \max_{\mathbf{x} \in \mathcal{A}_0^N} p_{\mathbf{Y}|\mathbf{x}}(\mathbf{y}) \cdot \Pr(\mathbf{x}) \,, \tag{7.2}$$

where $\Pr(\mathbf{x})$ is the a priori probability of the transmit hypothesis $\mathbf{x}$. For uniformly distributed elements of $\mathbf{x}$, this prior would become a constant and fall out of the maximisation problem, thus leading to maximum likelihood (ML) detection. For statistically i.i.d. transmit symbols x_n, the prior can be calculated as

$$\Pr(\mathbf{x}) = \prod_{n=1}^{N} \Pr(x_n) = (1 - p_a)^{N - \|\mathbf{x}\|_0} \cdot (p_a / |\mathcal{A}|)^{\|\mathbf{x}\|_0} \tag{7.3}$$

according to the PMF in (6.23).

Note, that $p_{\mathbf{Y}|\mathbf{x}}(\mathbf{y})$ in (7.2) is the conditional probability on a specific $\mathbf{x}$. Normally, this term can only be evaluated with an exhaustive search over all $|\mathcal{A}_0|^N$ possibilities for $\mathbf{x}$, yet SD is an efficient solution to this problem. Further, the measurements in $\mathbf{y}$ are realisations of a random process $\mathbf{Y}$. However, the system model (6.36) itself is deterministic (CSI assumed) and the only randomness originates in the additive thermal noise, being modelled as AWGN. The probability density function (PDF) of a complex-valued circularly-symmetric and zero-mean white Gaussian

random variable with power σ_n^2 is

$$p_{\mathbf{N}}(\mathbf{n}) = \frac{1}{(\pi\sigma_n^2)^N} \exp\left(-\left\|\mathbf{n}\right\|_2^2 / \sigma_n^2\right) . \tag{7.4}$$

Next, $\mathbf{n}$ can be substituted with the re-arranged system model in chip-clock $\mathbf{y} - \mathbf{T}\mathbf{x}$, Eq. (6.36), to describe $p_{\mathbf{Y}|\mathbf{x}}(\mathbf{y})$. Inserting this into Eq. (7.2) leads to

$$\hat{\mathbf{x}} = \arg\max_{\mathbf{x}\in\mathcal{A}_0^N} \frac{1}{(\pi\sigma_n^2)^N} \exp\left(-\frac{\left\|\mathbf{y} - \mathbf{T}\mathbf{x}\right\|_2^2}{\sigma_n^2} + \ln(\Pr(\mathbf{x}))\right) , \tag{7.5}$$

where the a priori probability $\Pr(\mathbf{x})$ was lifted into the exponent. This expression can be greatly simplified by omitting constant pre-factors and considering the strictly monotonic behaviour of the exponential function, such that

$$\hat{\mathbf{x}} = \arg\max_{\mathbf{x}\in\mathcal{A}_0^N} -\left\|\mathbf{y} - \mathbf{T}\mathbf{x}\right\|_2^2 + \sigma_n^2 \ln(\Pr(\mathbf{x})) . \tag{7.6}$$

The natural logarithm of the prior is

$$\begin{aligned} \ln(\Pr(\mathbf{x})) &= (N - \left\|\mathbf{x}\right\|_0) \ln(1 - p_a) + \left\|\mathbf{x}\right\|_0 \ln(p_a/|\mathcal{A}|) \\ &= -\left\|\mathbf{x}\right\|_0 \left(\ln(1 - p_a) - \ln(p_a/|\mathcal{A}|)\right) + N \ln(1 - p_a) \\ &= -\lambda \left\|\mathbf{x}\right\|_0 + N \ln(1 - p_a) \end{aligned} \tag{7.7}$$

with

$$\lambda = \ln\left(\frac{1 - p_a}{p_a/|\mathcal{A}|}\right) . \tag{7.8}$$

This parameter is the ratio of the two transmit hypotheses for inactivity and activity, that any symbol from $\mathcal{A}$ is transmitted. Note, solely the first term in (7.7) is a function of $\mathbf{x}$.

Putting the pieces together and exchanging the maximisation by a minimisation to remove the minus signs results in the S-MAP detector

$$\hat{\mathbf{x}} = \arg\min_{\mathbf{x}\in\mathcal{A}_0^N} \left\|\mathbf{y} - \mathbf{T}\mathbf{x}\right\|_2^2 + \sigma_n^2 \lambda \left\|\mathbf{x}\right\|_0 . \tag{7.9}$$

The argument of the $\arg\min$-operation constitutes a regularised distance metric $D_{\text{S-MAP}}(\mathbf{y}, \mathbf{x})$. In respect thereof, $\lambda > 0$ must hold, which is the

case as long as $p_a \leq |\mathcal{A}|/(|\mathcal{A}|+1)$, e.g. for BPSK $p_a \leq 2/3$. This, however, essentially means that every sparse signal fulfils this condition. Otherwise, the tuning parameter λ is usually chosen with cross-validation techniques as a function of the sizes M and N; here, it is directly coupled with the user activity factor p_a, [ZG11]. The cost function due to regularisation in (7.9) penalises non-sparse transmit hypotheses. Its impact is moderated by the noise power.

7.1.1 Correlation-Based Detection in Symbol-Clock

The optimal receiver of a multi-user CDMA signal in symbol-clock is a correlator, as it can be derived from the (quasi)-orthogonality constraint (6.31), [Küh06]. This equates to a left-multiplication of the chip-clock model (6.36) with $\mathbf{S}^H$ to take the user-specific channel realisations into account. Of course, this step requires CSI at the receiver, which is assumed in the following if not stated otherwise.

The system symbol-clock I/O relation is, therefore,

$$\mathbf{r} = \mathbf{S}^H\mathbf{y} = \mathbf{S}^H\mathbf{S}\mathbf{x} + \mathbf{S}^H\mathbf{n}\,. \tag{7.10}$$

$$= \mathbf{A}\mathbf{x} + \tilde{\mathbf{n}}\,, \tag{7.11}$$

with $\mathbf{A}$ being the Gram matrix $\mathbf{S}^H\mathbf{S} \in \mathbb{C}^{N\times N}$, and of full rank if $M > N$, as it is the case for classical multi-user systems. Since the receiver computes the correlation, $\mathbf{A}$ is a quadratic positively semi-definite correlation matrix. Afterwards, symbol detection will be executed with regard to $\mathbf{r}$.

In fact, the noise term $\tilde{\mathbf{n}}$ is correlated, leading to coloured noise with covariance matrix $\mathbf{\Phi_{nn}} = \mathrm{E}\{\mathbf{S}^H\mathbf{n}\mathbf{n}^H\mathbf{S}\} = \sigma_n^2\mathbf{S}^H\mathbf{S}$. This violates the white noise assumption (7.4), and for an improved symbol detection a pre-whitening (de-correlation) filter $\mathbf{P}$ is needed [Ver98]. With filtering, Eq. (7.10) becomes

$$\mathbf{r}_w = \mathbf{P}\mathbf{r} = \mathbf{P}\mathbf{S}^H\mathbf{S}\mathbf{x} + \mathbf{n}_w\,, \tag{7.12}$$

with $\mathbf{n}_w = \mathbf{P}\mathbf{S}^H\mathbf{n}$ being additive white noise again, indicated by the index "w". The derivation of $\mathbf{P}$ will be subject of Sec. 7.2.3.1.

Given the underdetermined case of $M < N$, it will be $\mathrm{rank}(\mathbf{A}) = M$. Hence, further regularisation of the system of equations, like in (6.41), is needed.

7.2 Sparsity-Regularised Tree Search Algorithms

Generally, integer least squares (ILS) problems are defined on the set of integer numbers $\mathbb{Z}$. The search space is the N-dimensional integer lattice $\mathbb{Z}^N$ or a finite subset $\mathcal{D} \subset \mathbb{Z}^N$. SD is known to solve this class of optimisation problems [DEC03; HV05]. Hence, SD can be adapted to solve the S-MAP detection problem (7.9) as well ($\mathcal{D} = \mathcal{A}_0^N$).

SD is the obvious choice of an algorithm. It is guaranteed to find the optimal solution on average in polynomial time, though it is also theoretically upper-bounded to terminate with exponential time complexity only [HV05]. Suboptimal, complexity-reduced variants like K-Best [Won+02], or relatively simple SIC, can provide feasible alternatives [Bur06; Wüb06]. A common prerequisite for these tree search algorithms is a QRD of the system matrix to transform the linear system of equations into a triangular structure. The search metric has to be regularised according to the sparsity constraint.

SIC and K-Best detection are closely related to SD. SD performs a depth-first search. The search tree has N levels, and after processing exactly N nodes, a first solution is found. In case the search radius has not been restricted previously, this intermediate solution equates to the SIC detection result. However, SD can revise its solution until all other possibilities are excluded. For this reason, SD finds the global optimum. K-Best performs a breadth-first search while progressing unidirectionally within the search tree. At each level, the K best hypotheses are retained and all others dismissed. K is a tuning parameter: for $K \to \infty$, K-Best detection equates to optimal SD, whereas for $K = 1$, it degenerates to SIC.

7.2.1 Sphere Decoding

The most prominent application of SD in communications theory so far has been symbol detection in multi-antenna (MIMO) systems [Bur06; Wüb06; Wie12]. Several configuration alternatives of the algorithm are discussed in the cited works. A brief overview is also given in [Kno+17].

There are applications of SD to underdetermined systems of equations, foremost to mention the Generalised Sphere Decoder (GSD) [DAB00;

CT05b]. For a sparsity-constrained setup, the underdetermined case was examined in [Mon17]. Given any $M \times N$ system matrix with $M < N$, SD will still find the optimal solution, albeit it will resort to an exhaustive search of the $(N - M)$ upper layers and looses as such its polynomial complexity and becomes infeasible.

Hence, practical applications of SD are only found with respect to an overdetermined setting. For instance, the regularisation according to Eq. (6.41) results in an overdetermined system of equations again. Sparsity-aware detection significantly reduces the run time complexity of SD [BV14], as it will also be shown in the following.

7.2.1.1 Mode of Operation

To enable SD, the discretised search space has to be transformed into a search tree beforehand, which is accomplished by triangularisation of the system of equations. Starting with the S-MAP metric in (7.9),

$$D_{\text{S-MAP}}(\mathbf{r}, \mathbf{x}) = \|\mathbf{r} - \mathbf{A}\mathbf{x}\|_2^2 + \sigma_n^2 \lambda \|\mathbf{x}\|_0 \ , \tag{7.13}$$

the matrix $\mathbf{A}$ of size $M \times N$ with $M \geq N$ is QR factorised such that $\mathbf{A} = \mathbf{Q}\mathbf{R}$. It is sufficient to consider the skinny QRD here, i.e. $\mathbf{Q}$ is a $M \times N$ unitary matrix of rank N, and $\mathbf{R}$ is a $N \times N$ upper-triangular matrix with real-valued entries on the main diagonal [GV13]. Left-multiplying $\mathbf{Q}^H$ to the ML part in (7.13) does not change the norm, $\left\|\mathbf{Q}^H \mathbf{a}\right\|_2 = \|\mathbf{a}\|_2$, hence

$$D_{\text{S-MAP}}(\tilde{\mathbf{r}}, \mathbf{x}) = \|\tilde{\mathbf{r}} - \mathbf{R}\mathbf{x}\|_2^2 + \sigma_n^2 \lambda \|\mathbf{x}\|_0 \ , \tag{7.14}$$

with $\tilde{\mathbf{r}} = \mathbf{Q}^H \mathbf{r}$ of length N.

In scalar notation the S-MAP metric reads

$$D_{\text{S-MAP}} = \sum_{n=1}^{N} d_n(\tilde{\mathbf{r}}^{(n)}, \mathbf{x}^{(n)}) \, , \tag{7.15}$$

with the partial metric

$$d_n(\tilde{\mathbf{r}}^{(n)}, \mathbf{x}^{(n)}) = \left| \tilde{r}_n - \sum_{l=n}^{N} r_{nl} x_l \right|^2 + \sigma_n^2 \lambda |x_n| \, . \tag{7.16}$$

The notation $\mathbf{x}^{(t)}$ denotes the partial vector $\begin{pmatrix} x_t & \dots & x_N \end{pmatrix}^T$ composed out of the N-dimensional $\mathbf{x}$, i.e. the first $n-1$ elements are removed. $d_n(\tilde{\mathbf{r}}^{(k)}, \mathbf{x}^{(k)})$ shall be named the n-th partial distance (PD) and the sum $D_n^N = \sum_{\nu=n}^{N} d_\nu$ partial accumulated distance (PAD). Therefore, the PDs and PADs are functions of partial vectors, and the PADs can be calculated recursively by

$$D_\nu^N = D_{\nu+1}^N + d_\nu \,, \tag{7.17}$$

given the starting condition $D_{N+1}^N = 0$.

The search tree consists of N levels, the root node being at the top. The SD algorithm begins its search at the root node on level N and then descends within the search tree. Each node has $|\mathcal{A}_0|$ child nodes or symbol transmit hypotheses. Associated with each hypothesis is a cost metric, the PD. "Processing a node" means evaluating the $|\mathcal{A}_0|$ PDs $d_\nu(x_\nu)$, $\forall x_\nu \in \mathcal{A}_0$. Subsequently, the PDs are accumulated as in (7.17) to the metric associated with the parent node, which is the PAD $D_{\nu+1}^N$.

The fundamental idea of SD, and the reason for its efficacy, is tree pruning to discard large subtrees on the basis of the sphere constraint (SC), which is a real-valued parameter $\varrho > 0$. Only such transmit hypotheses are considered which are within the ℓ_0-regularised hypersphere formed by the S-MAP metric and the diameter ϱ. All valid solution candidates $\check{\mathbf{x}}$ have to fulfil $D_{\text{S-MAP}} \leq \varrho$. This works, because the inequality $D_\nu^N \geq D_{\nu+1}^N$ holds for all ν, which in turn holds because $\lambda > 0$ and (7.13) being, therefore, a convex metric. Consequently, there is a monotonicity between the PADs,

$$D_N^N \leq D_{N-1}^N \leq \dots \leq D_1^N = D_{\text{S-MAP}} \leq \varrho \,. \tag{7.18}$$

If a node at level ν already violates the SC, so will all child nodes of the complete subtree: it can be excluded safely from the search.

The SC can be initialised with $\varrho \leftarrow \infty$. Then, SD will start processing the root node and selecting the child with minimal PAD. Then again, move on to process the chosen node, and so on, until a leaf node is reached. The followed path from the root to the first leaf node being reached corresponds exactly to the SIC detection result, as already mentioned. The candidate vector $\check{\mathbf{x}}$ with least S-MAP metric is chosen as intermediate solution and the SC is updated, $\varrho \leftarrow D_{\text{S-MAP}}(\tilde{\mathbf{r}}, \check{\mathbf{x}})$. Until now, no tree pruning happened due to the initialisation of ϱ with infinity. Thus, SD ascends again within the search tree comparing all so far computed PADs

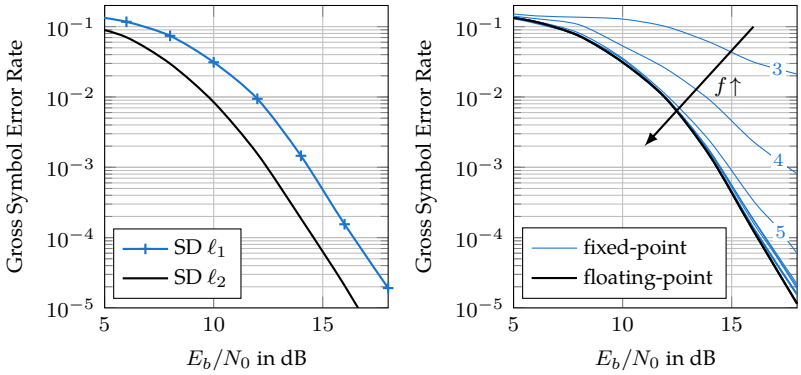
against the shrunk ϱ. As soon as a still valid node is found, the search direction will switch again, and the node will be processed and followed along until either another leaf node with a better solution candidate will be found or the SC becomes violated. Hence, SD is non-deterministic in its behaviour and tree traversal. The search is continued until the whole tree is pruned and a single solution, the global optimum, is left as final detection result $\hat{\mathbf{x}}$.

The above explanations assume a "best-first" search, also called Schnorr–Euchner candidate enumeration, i.e. the hypothesis with the least PAD is followed first [SE94]. This implies some kind of an intelligent choice of the best hypothesis or, if feasible, an exhaustive search over all $|\mathcal{A}_0|$ possible symbol hypotheses per node with subsequent (partial) sorting. Although this incurs additional computational complexity, it reduces the run time of SD considerably.

7.2.1.2 An RDAM-Based Sphere Detector

There is quite a number of options to adapt SD to a given application. A list of eight points was compiled in [Kno+17]:

1. The problem size which is given by the dimensions of the system matrix on which QRD is applied.
2. The search space which is given by a set of discrete values, e.g. a lattice formed by integers $\mathbb{Z}$, Gaussian integers $\mathbb{Z}[i]$ (i.e. complex integer numbers) or a finite alphabet, like a digital modulation constellation.
3. The cost function depends on the application. For instance, the MAP detection metric (7.13) includes the ℓ_0-pseudonorm regularisation, whereas ML detection, e.g. in classical non-sparse multi-antenna MIMO systems, would only be with regard to $\|\mathbf{r} - \mathbf{A}\mathbf{x}\|_2^2$.
4. The SC update strategy, can be non-updating, i.e. a fixed initialised value, or updating, i.e. the radius shrinks every time a better intermediate solution is found.
5. The search strategy can be depth-first (a.k.a. Fincke–Pohst enumeration) or best-first (a.k.a. Schnorr-Euchner enumeration).
6. Norm approximations can be applied to reduce computational complexity, e.g. the ℓ_2 norm could be replaced by the simpler ℓ_1



(a) Norm Approximation **(b)** Fixed-Point Format

Figure 7.1: Detection performance of a sparsity-aware SD created with the RDAM [Kno+14]. See the main text for details.

norm, deliberately trading a degradation in detection performance with improved data throughput.

7. The data type used for implementation.
8. The mathematical complexity of the data, being either complex-valued or real-valued.

With all these options in mind, a baseline functional SD design was created following the RDAM as outlined in Chap. 3. Special attention was paid to three issues: fixed-point design, RVD of complex-valued operations, and norm approximation. The highly parametrisable HLS design was then adapted to two application scenarios, one being a classical ML detector in a 4×4 multi-antenna MIMO system, the other being the above S-MAP detector. The following will discuss results for the latter case briefly.

Zero-augmented BPSK symbols taken from $\mathcal{A}_0^{\text{BPSK}}$ of N users were spread with random Bernoulli sequences of length M, and the user channel impulse responses were modelled by four i.i.d. Rayleigh fading coefficients. The activity probability was set to $p_a = 0.2$. Fig. 7.1a compares the GSER performance of the HLS-based SD in single-precision floating-point accuracy. The influence of the ℓ_1 norm was investigated,

Table 7.1: HLS results of the SD built with the RDAM [Kno+14].

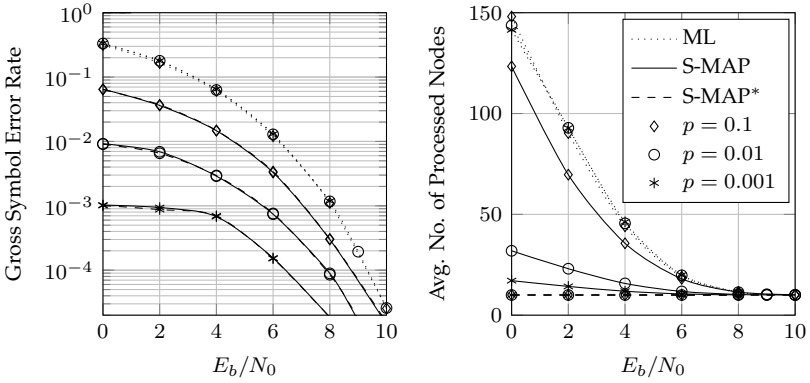
Data Type	Setup		Time	Resource Utilisation			
$\mathbb{C}$	(M, N)	ℓ_p	µs	BRAM	DSP	FF	LUT
`double`	(16, 16)	ℓ_2	1.787	20	67	2296	7059
`float`	(16, 16)	ℓ_2	1.634	6	48	1941	5136
`ap_fixed`	(16, 16)	ℓ_2	0.754	5	31	1558	3567
`double`	(16, 16)	ℓ_1	1.547	20	67	2228	7017
`float`	(16, 16)	ℓ_1	0.982	6	23	1930	1219
`ap_fixed`	(16, 16)	ℓ_1	0.406	5	17	1497	2214
`double`	(32, 20)	ℓ_1	1.557	20	67	2245	6977
`float`	(32, 20)	ℓ_1	0.976	6	23	1948	4063
`ap_fixed`	(32, 20)	ℓ_1	0.417	5	17	1506	2343

inspired by [Bur06]; the exchange of ℓ_2 for ℓ_1 is motivated by a significantly simpler computational complexity. A moderate loss in GSER performance of about 2 dB is observable. The communication is uncoded, and with further channel coding this SNR loss becomes less pronounced. On the other hand, the reduction in complexity yields a 54 % faster and smaller design, in terms of the DSP slices count (see Tab. 7.1).

Concerning the fixed-point conversion step, the optimal partitioning of the $w = 18$ bits word lengths into integer and fractional parts was found by a parametric sweep. The SD design was tested for Q formats $\mathrm{Q}(w - 1 - f, f)$, with f being the number of fractional bits. Fig. 7.1b shows the GSER detection performance for an increasing f. Small fractional length ($f \leq 7$) exhibit insufficient numerical precision and underflows degrade the detection performance. With increasing f, the GSER approaches the floating-point detector and nearly achieves it for $f = 8, ..., 13$ bits. Increasing f further leads to a severe performance collapse due to overflows (not shown).

Hence, the $\mathrm{Q}(9, 8)$ was chosen as a baseline fixed-point format for all computations throughout the design, and only selected accumulation operations are executed with higher precision and later reduced again to this word length.

Table 7.1 lists synthesis results for three configurations. Firstly, a system with $N = M = 16$ was investigated, i.e. the CDMA system is



(a) Detection Performance **(b)** Run Time Complexity

Figure 7.2: The ℓ_0-regularised sparsity-aware Sphere Decoding performs significantly better than ML detection [KWP12].

critically loaded. The metric is computed as in (7.13). The fixed-point design considerably reduces the number of utilised DSP slices without a deterioration in detection performance. Double-precision offers no advantages over single-precision floating-point, but can also successfully be synthesised. Secondly, $(M, N) = (32, 20)$ was chosen for an overdetermined system, with and without ℓ_1 norm approximation. Note, the computational complexity is mostly with the per-node processing (PD evaluation and sorting, SC testing). The SD iterates until termination and processes a single node per iteration. Since its run time is variable, Tab. 7.1 states the per-node computation time. However, an increased problem size only increases the run time complexity of the SD algorithm but not the per-node computational complexity. Therefore the resource utilisation count is about the same for both cases, $(16, 16)$ and $(32, 20)$.

7.2.1.3 Sparsity-MAP Run Time Complexity

In [KWP12] is has been shown, that the penalisation due to the sparsity constraint in (7.13) has a strong impact.

Fig. 7.2 compares the ML SD with the S-MAP SD and a variant with non-convex distance metric (S-MAP*). The latter leads to a virtually reduced

search sphere but is only approximative in nature, since it might happen that the global optimum becomes pruned. Nonetheless, the Fig. 7.2a shows the improved detection performance because of ℓ_0 regularisation. Fig. 7.2b shows the measured average run time complexity of the three examined algorithmic variants expressed by the number of processed nodes. This is reciprocal to the data throughput of SD.

Monte Carlo simulations were performed for a $N = 10$ users, synchronous CDMA system over a two-tap Rayleigh fading channels with AWGN, as described earlier. Binary PN codes of length $N_s = 127$ were used as spreading sequences, and the modulation alphabet was zero-augmented QPSK.

It is noteworthy that for low SNRs the GSER of the S-MAP SD approaches p_a asymptotically. Hence, the S-MAP decoding scheme itself constitutes a rather conservative estimator the larger the noise power gets: all data symbols then tend to be estimated as zeroes. Furthermore, no significant quality degradation from optimal to approximative S-MAP SD can be observed. This is a surprising result, since Fig. 7.2b clearly shows that non-convex SD almost always terminates after processing the minimial number of N nodes. Usually, this is the well-known property of SIC detection. Hence, by intentionally violating the ideal check of the SC, many improbable transmit hypotheses can be dismissed early, thus reducing decoding complexity but without notable loss of quality. Optimal S-MAP SD has a somewhat reduced run time complexity compared to ML detection, which becomes less the less active users are. This can be explained with its preference for the zero symbol.

A similar conclusion is drawn in [BV14]. Sparsity-aware decoding allows for aggressive tree pruning although the per-node computational complexity is slightly increased. This is beneficial overall. The authors also propose to initialise the SC by a sparse solution obtained by a relaxed ℓ_1 problem with by OMP to reduce the number of processed nodes further.

The significantly improved detection and run time performances of sparsity-aware SD bring up the question whether further complexity-reduced SIC or K-Best estimation can facilitate sparsity-constrained multi-user detection. This is the subject of the following sections.

7.2.2 K-Best Detection

K-Best detection can be interpreted as a trade-off between SD and SIC with an adjustable parameter K which determines performance and complexity of the detector.

The algorithm sacrifices optimality but for large values of the parameter K it is able to achieve near optimal performance. It performs a breadth-first search, and only retains those K hypotheses, or search paths, with the smallest associated PD per iteration. Thus, the algorithm operates on the search tree unidirectionally, which leads to a deterministic run time and constant throughput. Hence, K-Best is better suited for hardware implementations and can be pipelined nicely [Won+02].

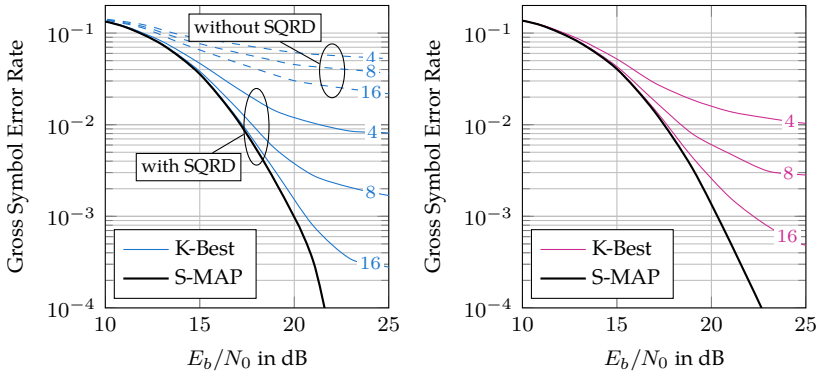
In [Kno+14], K-Best detection is modified to approximately solve the finite-alphabet CS problem (6.38). Note, in contrast to the previous section and SD, the following results are obtained in an underdetermined setting without a correlation receiver, i.e. multi-user symbol detection is performed directly on the measured chip-clock receive signal. $N = 20$ users transmit zero-augmented BPSK with activity probability $p_a = 0.2$ over an AWGN channel to a central data aggregation node. Spreading is applied as well, although with shorter sequences of length $M = 10$. This time, the random spreading sequences are i.i.d. drawn from a zero-mean Gaussian distribution. This system is clearly underdetermined, as the ratio of observations to nodes is $1/2$.

7.2.2.1 Pre-Processing Steps

To improve K-Best performance, two additional preprocessing steps are introduced.

Firstly, the $M \times N$ system matrix $\mathbf{T}$ is regularised as in (6.41). This incorporates the prior information into the ML detection metric while transforming the system of equations into an $(M + N) \times N$ system described by $\tilde{\mathbf{T}}$. Hence, the optimisation problem differs from known K-Best implementations only in the zero augmentation of the modulation alphabet. A third PD has to be evaluated for $x_\nu = 0$ in addition to the two hypotheses for $x_\nu = -1$ or $x_\nu = +1$. This naturally affects the sorting afterwards as well.

Secondly, instead of a regular QRD, a SQRD is applied [Wüb+01]. SQRD permutes the detection order heuristically such that data of nodes



(a) Detection Performance **(b)** With Activity Mismatch

Figure 7.3: K-Best estimation is able to detect a sparse multi-user signal from underdetermined measurements, even if user activities are not exactly known [Kno+14]. The parameter K is given for each plot line.

with a high post-detection SNR are estimated first. $\tilde{\mathbf{T}}$ is factorised such that

$$\tilde{\mathbf{T}} = \mathbf{QR\Pi}\,, \tag{7.19}$$

where $\mathbf{\Pi}$ is a permutation matrix, which determines the ordering of the subsequent detection. Eq. (6.41) becomes with SQRD

$$\hat{\mathbf{x}} = \arg \min_{\mathbf{x} \in (\mathcal{A}_0^{\text{BPSK}})^N} \|\check{\mathbf{y}} - \mathbf{R\Pi x}\|_2^2\,, \tag{7.20}$$

where $\check{\mathbf{y}} = \mathbf{Q}^T\tilde{\mathbf{y}}$. $\mathbf{R}$ is an upper-triangular matrix of size $N \times N$ and has full rank as long as $\lambda > 0$ holds.

7.2.2.2 Numerical Simulation Results

Fig. 7.3a shows the GSER over the zero-augmented BPSK symbol alphabet $\mathcal{A}_0^{\text{BPSK}} = \{0, -1, 1\}$. Sparse K-Best detection without SQRD as pre-processing (dashed lines) cannot achieve the S-MAP performance of S-MAP SD, which is used as a benchmark. It can be seen that K-Best detection with very low K results in huge performance losses. Increas-

ing K to higher values reduces the loss, and with $K = 16$ the gap to optimal performance is small in the mid-SNR range. At higher SNR values, K-Best detection shows an error floor behaviour which is due to the limited coverage of the search within the search tree. In contrast to this, the S-MAP SD detector can achieve better performance but only with non-deterministic throughput and latency.

The sparsity-aware symbol detection relies on the correct knowledge of the user activity p_a at the receiver, which can be derived from long term statistics, for example. Even if there is a mismatch between true and assumed activity probability, K-Best is able to correctly estimate the multi-user detection problem. This is shown in Fig. 7.3b. The mismatch Δ_p is modelled to be uniformly random $\Delta_p \sim \mathcal{U}(-\Delta_{p,\max}, \Delta_{p,\max})$, i.e. the node activity probability in the system is $p = p_a + \Delta_p$, where p_a is the activity probability assumed at the detector and p is the true activity probability of the transmit nodes. The plot shows the performance of K-Best detection averaged over $10\,000$ model realisations. The results are similar to the previous ones without activity mismatch, but a performance loss is observable (for S-MAP as well). However, even for $K = 8$ it is still possible to achieve a GSER $< 10^{-2}$. Therefore, it can be concluded that K-Best detection is, to some extent, robust against parameter mismatch.

7.2.2.3 Complexity Analysis

Setting $K = 16$ results in a computational complexity of $N \cdot K \cdot |\mathcal{A}_0| = 960$ processed hypotheses for K-Best decoding, which is still feasible from a practical perspective. In comparison, the computational complexity of S-MAP detection is not fixed with a worst case upper bounded of $|\mathcal{A}_0|^N \approx 3.5 \cdot 10^9$, which is prohibitively high.

The algorithmic complexity of the K-Best algorithm is

$$C_{\text{K-Best}} = \mathcal{O}\{N(K|\mathcal{A}_0|C_{\text{pm}} + C_{\text{sort}})\}\,, \tag{7.21}$$

with C_{pm} modelling the necessary operations for the partial metric computations and C_{sort} the complexity of the sorting algorithm [Kno+14]. Further, these complexities can be assessed as

$$C_{\text{pm}} \sim N^2\,, \tag{7.22}$$

and, if Odd-Even Mergesort is employed,

$$C_{\text{sort}} \sim K|\mathcal{A}_0| \log^2(K|\mathcal{A}_0|)\,. \tag{7.23}$$

Consequently, K-Best detection is of polynomial complexity in the number of users ($\mathcal{O}(N^3)$), and scales merely log-linearly with the number of search paths K. Since only the smallest K partial path metrics are of interest in subsequent processing, a completely sorted list of all metrics is not required. Hence, the sorting could be simplified algorithmically if a digital modulation of higher modulation order is employed.

Summarising the above, K-Best detection works with a reasonably low K even for underdetermined systems: it can achieve good symbol error rate performance with a complexity far less than S-MAP detection and is robust against unpredictable changes of user activities in the system.

7.2.3 Successive Interference Cancellation

The astonishingly low run time complexity of the sparsity-aware MAP detection above raised the question whether low-complexity alternatives to SD could also be utilised. It is well-known that SIC with proper pre-processing, such as sorting, is a viable candidate for a suboptimal approach. As it turns out, a pre-whitening filter is, nonetheless, critical to successful detection [Kno+13].

SIC features the lowest computational complexity of the three tree search algorithms discussed here. Its run time and throughput is constant, and, hence, a potential hardware implementation could be pipelined.

Algorithm 7.1 lists the pseudocode of Sparsity-Aware Successive Interference Cancellation (SA-SIC). It requires knowledge of the pre-whitened received vector $\mathbf{r}_w$, the CSI (expressed by $\mathbf{R}$ of the QRD and σ_n^2), and the system-specific global activity parameter p_a.

7.2.3.1 Efficiently Combined Sorted QR Decomposition and Pre-Whitening

Given the overdetermined system model and the correlation receiver as described in Sec. 7.1.1, the CDMA system model according to (7.10) in symbol clock is

$$\mathbf{r} = \mathbf{S}^H\mathbf{S}\mathbf{x} + \mathbf{S}^H\mathbf{n}\,, \tag{7.24}$$

Algorithm 7.1: Sparsity-Aware Successive Interference Cancellation

function SA-SIC($\mathbf{r}_w, \mathbf{R}, \sigma_n, p_a$)
  $\hat{\mathbf{x}} \leftarrow 0$
  $\lambda \leftarrow 2\sigma_n^2 \log((1-p_a)/(p_a/|\mathcal{A}|))$
  for $n \leftarrow N, ..., 1$ **do**
    $z \leftarrow \sum_{j=n+1}^{N} r_{nj}\hat{x}_j$ ▷ residual
    for all $x \in \mathcal{A}_0$ **do** ▷ symbol hypotheses
      $d_x \leftarrow (r_{w,n} - r_{nn}x - z)^2 + \lambda|x|$
    $\hat{x}_n \leftarrow \arg\min_{\forall x \in \mathcal{A}_0} d_x$ ▷ decision
  return $\hat{\mathbf{x}}$

where the signatures of the users are the columns of $\mathbf{S}$. Therefore, the noise is coloured.

To achieve improved detection performance on a pre-whitened system model, only the inverse of a triangular matrix $\mathbf{R}$ is needed additionally, and SIC can subsequently be executed on the permuted symbol vector $\mathbf{\Pi x}$. $\mathbf{S}$ is factorised by SQRD such that $\mathbf{S} = \mathbf{QR\Pi}$, with $\mathbf{\Pi}$ as permutation matrix for sorting. By the way, $\mathbf{\Pi}^H\mathbf{\Pi} = \mathbf{\Pi\Pi}^H = \mathbf{I}$ holds for any permutation matrix. Then, the optimal pre-whitening filter $\mathbf{P}$ is given by [Kno+13]

$$\mathbf{P} = \mathbf{R}^{-H}\mathbf{\Pi}\,. \tag{7.25}$$

To clarify notation, $(\cdot)^{-H}$ is a shorthand for $\left((\cdot)^{-1}\right)^H = \left((\cdot)^H\right)^{-1}$. The task of this pre-whitening filter $\mathbf{P}$ is twofold. Firstly, the filter de-correlates the noise prior to the application of a detector. Secondly, filtering with $\mathbf{P}$ leads to a sorted triangular system description which allows for optimised successive detection.

The pre-whitening filter diagonalises the symbol clock noise covariance matrix at the output of the filter as

$$\begin{aligned}
\mathbf{\Phi}_{\mathbf{n}_w\mathbf{n}_w} &= \sigma_n^2\mathbf{P}\mathbf{S}^H\mathbf{S}\mathbf{P}^H && \text{(7.26a)}\\
&= \sigma_n^2\mathbf{R}^{-H}\mathbf{\Pi}\left(\mathbf{QR\Pi}\right)^H\mathbf{QR\Pi}\left(\mathbf{R}^{-H}\mathbf{\Pi}\right)^H && \text{(7.26b)}\\
&= \sigma_n^2\mathbf{R}^{-H}\mathbf{\Pi\Pi}^H\mathbf{R}^H\mathbf{Q}^H\mathbf{QR\Pi\Pi}^H\mathbf{R}^{-1} && \text{(7.26c)}\\
&= \sigma_n^2\mathbf{I}\,. && \text{(7.26d)}
\end{aligned}$$

Moreover, the application of (7.25) to the system description in (7.12) automatically leads to a triangular system of equations, as it is

$$\begin{aligned}\mathbf{r}_w &= \mathbf{P}\mathbf{S}^H\mathbf{S}\mathbf{x} + \mathbf{n}_w && \text{(7.27a)}\\ &= \mathbf{R}^{-H}\mathbf{\Pi}\left(\mathbf{Q}\mathbf{R}\mathbf{\Pi}\right)^H\mathbf{Q}\mathbf{R}\mathbf{\Pi}\mathbf{x} + \mathbf{n}_w && \text{(7.27b)}\\ &= \mathbf{R}^{-H}\mathbf{\Pi}\mathbf{\Pi}^H\mathbf{R}^H\mathbf{Q}^H\mathbf{Q}\mathbf{R}\mathbf{\Pi}\mathbf{x} + \mathbf{n}_w && \text{(7.27c)}\\ &= \mathbf{R}\mathbf{\Pi}\mathbf{x} + \mathbf{n}_w\,. && \text{(7.27d)}\end{aligned}$$

As just proven, $\mathbf{n}_w$ is a vector containing uncorrelated white Gaussian noise samples with variance σ_n^2, and $\mathbf{R}$ is the sorted upper-triangular system matrix ensuring optimised successive detection of the *permuted* source vector $\mathbf{\Pi}\mathbf{x}$.

Additionally, the matrix inversion in (7.25) is computationally cheap, because an upper-triangular matrix can be easily inverted by back substitution. Another advantage is that the SQRD only has to be updated when the underlying channel matrix $\mathbf{S}$ changes (time variance), which is quite suitable for a hardware realization. QRD and SQRD share the same algorithmic complexity of $\mathcal{O}\left(N^3\right)$, although the constant is a bit larger for the latter since it requires heuristic sorting [Wüb06]. Therefore, the additionally necessary computational effort for SQRD is small and the gain in performance comes almost for free.

7.2.3.2 Detection Performance

The described system model was simulated for two scenarios. The first setup is an overdetermined system with $N = 20$ transmitting nodes, or users, and with a CDMA spreading sequence length of $N_s = 32$ (random Bernoulli). The second setup is a fully-loaded CDMA system with $N = N_s = 16$. The wireless channels are modelled by 4-tap Rayleigh fading impulse responses. The GSER is evaluated over the zero-augmented BPSK symbol alphabet $\mathcal{A}_0^{\text{BPSK}}$, and the per-node activity probability is homogeneously $p_a = 0.2$.

Fig. 7.4 plots the GSER over the channel quality expressed by E_b/N_0. Optimal performance is achieved by a S-MAP SD with pre-whitening filter (SD$^{\text{w}}$). A performance with nearly constant degradation is achieved by the SA-SIC without sorting but with pre-whitening, denoted as SA-SIC$^{\text{w}}$. SA-SIC with pre-whitening and sorting, SA-SIC$^{\text{w,s}}$, converges with

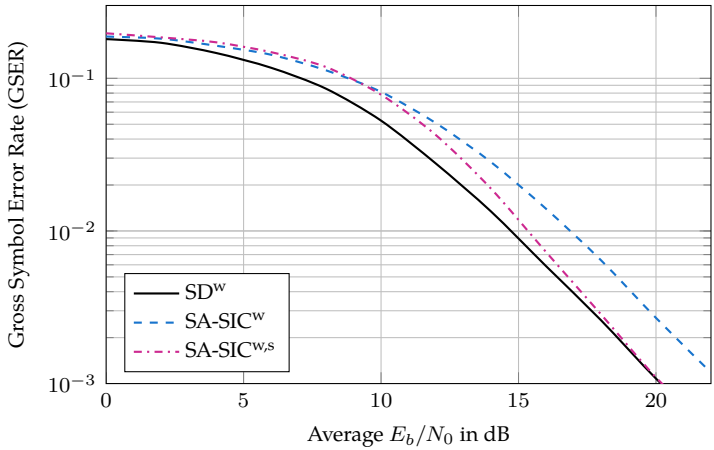


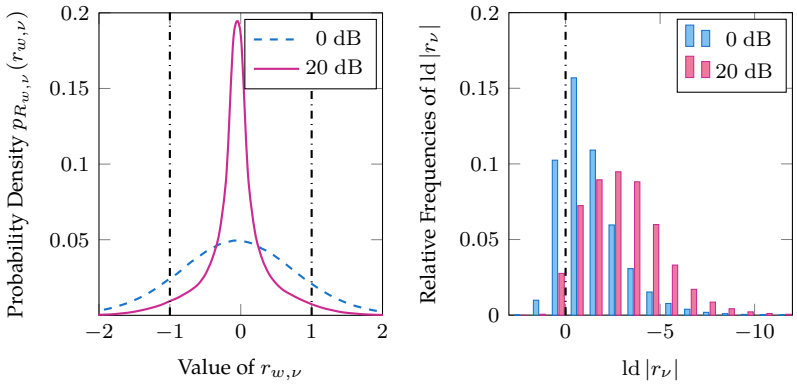
Figure 7.4: SA-SIC detection in an overdetermined setup with floating-point accuracy. Pre-whitening and sorting improves detection [Kno+13].

the performance of S-MAP detection at high SNR. Note, the application of sorting as pre-processing makes no difference for the GSER result of SD, since it will always find the globally optimal S-MAP solution.

Fig. 7.5 shows the measured input statistics of the pre-whitened $r_{w,\nu}$, $\nu = 1, ..., N$. Unsurprisingly, the distribution resembles a Gauss curve, which is in fact a superposition of several normal distributions, according to the PDF of the received samples $r_{w,\nu}$,

$$p_{R_{w,\nu}}(r_{w,\nu}) = \sum_{\forall x \in \mathcal{A}_0} \Pr(x)\, \mathcal{N}(\mu = x, \sigma_n)\,, \tag{7.28}$$

when we assume perfect CSI. Fig. 7.5a highlights an interval between dash-dotted lines, which signifies the number range of the fractional Q format. The lesser the channel quality, the more probable higher magnitudes are, which leads to overflows. To fit $r_{w,\nu}$ with high probability into a Qn fixed-point representation, the data has to be scaled at least with $f = 1/2$ or better $f = 1/4$, which is algorithmically nothing else than an efficient bit shift to the right by one respectively two bit locations.



(a) Linear **(b)** Logarithmic

Figure 7.5: Measured input statistics of the elements of the pre-whitened vector $\mathbf{r}_w$ in linear scale (a) and in logarithmic scale with respect to the binary fixed-point representation (b), [Kno+13].

Fig. 7.5b shows a histogram of the same distribution but scaled by the logarithmus dualis. Each bin of the histogram corresponds to one bit position of a binary number. On the left are the MSBs and on the right the LSBs. Values to the left of the dash-dotted line are outside the number range of any fractional Q format. For higher SNRs, the dynamic range becomes larger, which, in turn, makes greater bit widths necessary. After a sufficient binary scaling, i.e. bit shifts to the right by at least one or two bit positions, the dynamic range of the input data fits w.h.p. into 16 bits wordlength, which enables almost lossless fixed-point arithmetic.

This is verified in Fig. 7.6 which plots the GSER over SNR for SIC after fixed-point conversion. For high SNR values the FXP SA-SIC runs into an error floor. Notably, the GSER does not really improve for larger bit widths than Q9 at the same scaling factor $1/2$. SA-SIC$^{\mathrm{w,s}}$ Q11^{+} denotes an additional simulation with a scaling factor of $1/4$ and clearly improved GSER, which emphasises the necessity of a sufficiently dimensioned scaling as long as the dynamic range of the data allows for it. The plot also shows that 8 bit quantisation is a feasible low-complexity choice,

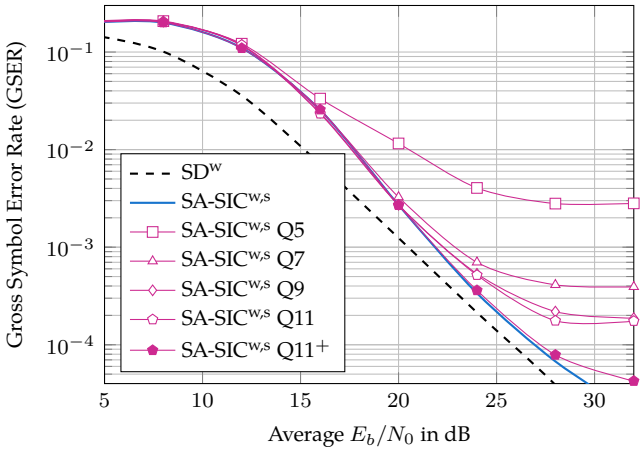


Figure 7.6: Performance of fixed-point SA-SIC detection for a fully-loaded CDMA system of 16 users [Kno+13].

with GSER better than 10^{-3} for high SNRs, which could be improved further with channel coding techniques.

7.3 Discussion

Tree search algorithms solve the joint estimation problem of user activities and data with a finite-alphabet constraint. Hence, the ideas of CS could profitably be applied to a wireless communication system with a sparse communication behaviour of the sensor nodes.

SD represents the optimal detection benchmark, because it is guaranteed to find the globally optimal solution vector to the S-MAP criterion. With regard to its run time complexity, however, SD is by no means the first choice, although it has been shown that the sparsity constraint reduces it considerably.

Alternatively, ℓ_0-regularised K-Best and SIC detections have been discussed. These are able to achieve acceptable GSERs, if some preprocessing steps are performed. Common to both algorithms was the application of the SQRD, which heuristically permutes the detection order of the

users. This reduces decision errors and their propagation which otherwise severely deteriorates detection. A pre-whitening filter further improves GSER performance, if a correlation receiver is part of the setup. Most notably, K-Best is also able to recover a multi-user transmit vector from compressive measurements.

After all, a joint multi-user activity and data estimation is possible. The algorithms throughout this chapter, however, require the knowledge of certain parameters, foremost CSI. But how can CSI be obtained at the receiver if users transmit intermittently without prior activity signalling? The following chapter will provide an answer.

Chapter 8

Joint Multi-User Activity and Channel Estimation

The previous chapter adapted finite-alphabet tree search algorithms to a novel joint activity and data detection at the multi-user receiver, or data aggregation node, on the PHY layer. All the time, perfect CSI at the receiver has been assumed, but for practical implementations, CSI has to be estimated to enable channel equalisation and phase-coherent reception of data symbols.

It is practically impossible to reliably estimate user-specific channels, activity and data symbols simultaneously [SBD13]. Hence, the partitioning of this problem into a joint multi-user *activity and channel estimation* followed by a classical non-sparse data detection is a viable solution [Kno+16b].

In this chapter, the ideas and results from [Kno+16b] are presented. User-specific code sequences will be utilised to form a multi-user pilot signal to facilitate a pilot-based channel estimation. Additionally to random Gaussian sequences, ZCSs will be investigated, which promise good performance because of their favourable autocorrelation properties [ZF05].

User activities and channel impulse responses can mathematically be written as a vector, which is either sparse or block-sparse, depending on whether there are frequency-flat or frequency-selective fading channels, respectively. However, detection algorithms may not be arbitrarily complex with respect to a potential implementation. For this reason, the detection performance of relatively simple OMP [TG07], and its variant for block-sparse signals, Block Orthogonal Matching Pursuit (BOMP) [EKB10], will be investigated.

8.1 System Model

Basically, all explanations given so far about the wirless uplink in a WSN with sporadically transmitting users stay valid (see Sec. 6.3.1 and following). The detector has probabilistic, but not instantaneous, knowledge about the user activities, i.e. p_a is known by the receiver.

Other than before, a data frame is assumed, which consists of a preamble, containing multi-user pilots, and payload, containing data. In the following, no further assumptions are made about the payload. Yet, it is assumed that all active users begin transmission synchronously at the same time instance.

Each user is assigned a specific pilot code sequence of length N_p. The user-specific pilot sequences become superimposed due to the simultaneous channel accesses, similar to a DS-CDMA system. The mathematical system description in the following is with respect to the pilot preamble only.

The system model in symbol clock can summarised by

$$\mathbf{y} = \mathbf{C}_p\mathbf{H}\mathbf{a} + \mathbf{n} = \mathbf{C}_p\mathbf{h} + \mathbf{n}\,, \tag{8.1}$$

where $\mathbf{y}$ is the received signal vector and $\mathbf{n}$ denotes AWGN with zero mean and variance σ_n^2.

The multi-user vector $\mathbf{a} \in \{0,1\}^N$ defines the activity pattern of the N sensor nodes during a frame and its n-th entry, a_n, corresponds to the activity of node n. A transmitter is modelled as inactive if $a_n = 0$, or active if $a_n = 1$, i.e. the user-specific pilot code sequence is transmitted. If p_a is sufficiently small, $\mathbf{a}$ is a sparse vector containing a considerable number of zero symbols.

Each user transmits its pilot sequence $\mathbf{c}_n \in \mathbb{C}^{N_p}$ over a unique wireless channel impulse response $\mathbf{h}_n$, a column vector of length L. This corresponds to a convolution, and for the product

$$\mathbf{C}_p\mathbf{H} = \begin{bmatrix}\mathbf{c}_1 * \mathbf{h}_1 & \cdots & \mathbf{c}_N * \mathbf{h}_N\end{bmatrix} \tag{8.2}$$

holds. With the help of the convolution matrix, $\mathbf{c}_n * \mathbf{h}_n = \mathbf{C}_n \mathbf{h}_n$, Eq. (8.2) can alternatively be written as

$$\mathbf{C}_p \mathbf{H} = \begin{bmatrix} \mathbf{C}_1 & \cdots & \mathbf{C}_N \end{bmatrix} \begin{bmatrix} \mathbf{h}_1 & & 0 \\ & \ddots & \\ 0 & & \mathbf{h}_N \end{bmatrix}, \tag{8.3}$$

where $\mathbf{C}_p$ is the horizontal, or row-wise, concatenation of the convolution matrices $\mathbf{C}_n$, and $\mathbf{H}$ is a block-diagonal matrix of the channels $\mathbf{h}_n$, with $n = 1, ..., N$. $\mathbf{C}_p$ is $N_p + L - 1 \times LN$ and $\mathbf{H}$ is $LN \times N$.

In high data rate CDMA systems, the channel impulse responses are sparse such that multipath propagation delays are distributed over a couple of echoes, usually appearing in clusters [Kam08]. However, MTC is assumed to be of low data rate. Hence, it can be justified to model frequency-selective channels with only a few, non-sparse channel coefficients, all i.i.d. Rayleigh. If the transmission is of very low data rate, occupying only a narrow bandwidth, the fading becomes frequency-flat, i.e. $L = 1$.

Of special interest is the product

$$\mathbf{h} = \mathbf{H}\mathbf{a} = \begin{bmatrix} a_1 \mathbf{h}_1 \\ \vdots \\ a_N \mathbf{h}_N \end{bmatrix}, \tag{8.4}$$

with $\mathbf{h} \in \mathbb{C}^{LN}$ being the stacked vector of all user channel impulse responses multiplied with the activity of the corresponding users. $\mathbf{h}$ is like $\mathbf{a}$ a sparse vector, yet it is further *block-sparse* with $N_a = p_a N$ blocks of non-zero elements of length L.

Generally, a vector $\mathbf{x}$ of N equally sized blocks can be defined as

$$\mathbf{x} = \big[\underbrace{x_1 \cdots x_d}_{\mathbf{x}^T[1]} \; \underbrace{x_{d+1} \cdots x_{2d}}_{\mathbf{x}^T[2]} \; \cdots \; \underbrace{x_{(N-1)d} \cdots x_{Nd}}_{\mathbf{x}^T[N]} \big]^T, \tag{8.5}$$

whereby $\mathbf{x}[\ell]$ denotes or selects the ℓ-th block of length d. If only k blocks contain non-zero elements, this vector is said to be k-block-sparse, or

mathematically

$$\sum_{\ell=1}^{N} I(\|\mathbf{x}[\ell]\|_2) = \|\mathbf{x}\|_{2,0} \leq k\,, \tag{8.6}$$

with $I(\cdot)$ being the indicator function [EKB10]. The index notation of the norm $\|\cdot\|_{2,0}$ denotes ideal block sparsity in an ℓ_0-pseudonorm sense, whereby for a particular block the Euclidean norm is utilised to judge if all elements belonging to it are zero. The employed indicator function can be defined as

$$I(x) = \begin{cases} 1 & x \neq 0 \\ 0 & x = 0 \end{cases}. \tag{8.7}$$

In other words, $I(\|\mathbf{x}[\ell]\|_2)$ returns one if the ℓ-th block contains a non-zero value. A deepened study of block-sparse signal recovery via convex optimisation and with some generalisations to the block lengths can be found in [EV12].

8.1.1 Zadoff–Chu Sequences as Pilots

For a good multi-user interference cancellation, the chosen user-specific code sequences should preferably be quasi-orthogonal, i.e. $\langle \mathbf{c}_i, \mathbf{c}_j \rangle \approx 0$, $i \neq j$; in other words, should have good cross-correlation properties. Furthermore, their length N_p should be small to save bandwidth for payload data. Random zero-mean complex Gaussian sequences are employed as pilot codes because they have been shown to facilitate a better user separation compared to traditional PN sequences [AMR09].

Additionally, ZCSs promise a good performance as well, because they are so-called CAZAC sequences, which means "constant amplitude and zero autocorrelation" [ZF05; Chu72]. Good autocorrelation properties means that the autocorrelation function (ACF) of the periodic sequence is zero except at multiples of the period, where the ACF has a single maximum which equates to the energy of the sequence.

Mathematically, the periodic ACF of a ZCS $z(k)$ of length N_p and of the same period is given by

$$\phi_{zz}(\kappa) = \sum_{k=0}^{N_p-1} z^*(k)\, z\big((k+\kappa) \bmod N_p\big) = \begin{cases} \sigma_z^2 & \kappa \bmod N_p = 0 \\ 0 & \kappa \bmod N_p \neq 0 \end{cases}. \tag{8.8}$$

ZCSs exist for an arbitrary integer length N_p and can be constructed with the help of the N_p-th complex primitive root of unity

$$W_{N_p} = \exp\left(-j2\pi a/N_p\right) , \tag{8.9}$$

where a is any integer relatively prime to N_p, i.e. $\gcd(a, N_p) = 1$. Since $|W_{N_p}| = 1$, ZCSs are polyphase sequences of constant magnitude, defined as [Pop92]

$$z(k) = \begin{cases} W_{N_p}^{k^2/2+qk}, & N_p \text{ even}, \\ W_{N_p}^{k(k+1)/2+qk}, & N_p \text{ odd}, \end{cases} \tag{8.10}$$

for $k = 0, ..., N_p - 1$ and q being any integer. q highlights that certain linear phase shifts will not affect the correlation, and w.l.o.g. can be set to zero. Further, trivial variations such as cyclic shifts, addition of a constant or conjugating the entire code will not affect the ACF as well [Chu72]. Besides the ideal ACF properties, there is also a bound on the cross-correlation for N_p being prime.

The ideal cyclic ACF is especially useful to achieve a good multi-user separation. This property can be exploited by generating N pilot sequences from a single ZCS $\mathbf{z}_r$ of length $N_p = LN$, called root sequence in the following [Kno+16b]. Each user $n = 1, ..., N$ is then assigned a circularly shifted version of the root sequence, shifted by $(n-1)L$ elements:

$$\mathbf{c}_n(k) = \mathbf{z}_r\big((k + (n-1)L) \bmod N_p\big) . \tag{8.11}$$

Consequently, all user pilot sequences are strictly orthogonal to each other, which guarantees perfect user separation. Assembled as columns of $\mathbf{C}$, it is $\mathbf{C}^H\mathbf{C} = \sigma_z^2\mathbf{I}$. Even in case of frequency-selective fading this is essentially still true due to the shift by at least L elements, which accommodates for the (non-cyclic) convolution with the channel impulse responses according to (8.2). However, $N_p \geq LN$ must hold in order to be able to estimate LN unknown channel coefficients.

8.2 Solving the Estimation Problem with Block Orthogonal Matching Pursuit

The task is to estimate the multi-user activity and channel coefficent vector $\mathbf{h}$, Eq. (8.4), given the measurements $\mathbf{y}$ according to (8.1). The support of $\mathbf{h}$, i.e. the positions of the non-zero entries, represents the activity pattern of the transmit nodes, and the values of the non-zero entries correspond to the channel impulse response of the active users.

Hence, the estimation of $\mathbf{h}$ equates to a joint activity and channel estimation. Subsequently, channel equalisation and non-sparse symbol detection could be performed.

As mentioned previously, there can be frequency-selective fading ($L > 1$) or frequency-flat fading ($L = 1$). The recovery problems and applicable algorithms differ depending on this parameter.

8.2.1 Frequency-Flat Fading Channels

If $L = 1$, each user channel impulse response $\mathbf{h}_n$ simply becomes a single Rayleigh-distributed complex-valued channel coefficient h_n. Consequently, $\mathbf{C}_p = \begin{bmatrix} \mathbf{c}_1 & \cdots & \mathbf{c}_N \end{bmatrix}$ contains column-wise every user-specific pilot sequence and $\mathbf{H} = \operatorname{diag}\begin{bmatrix} h_1 & \cdots & h_N \end{bmatrix}$ is a diagonal matrix of the user's channel coefficients.

The S-MAP optimisation problem associated with (8.1) and $L = 1$ is

$$\hat{\mathbf{h}} = \arg \min_{\mathbf{h} \in \mathbb{C}^N} \|\mathbf{y} - \mathbf{C}_p \mathbf{h}\|_2^2 + \lambda \|\mathbf{h}\|_0 \ , \tag{8.12}$$

with

$$\lambda = 2\sigma_n^2 \ln\left(\frac{1 - p_a}{p_a}\right) \ , \tag{8.13}$$

which reflects the a priori statistics of the activity vector $\mathbf{a}$ and scales with the noise power σ_n^2. This is a classical CS problem which could be solved by ℓ_1 minimisation. Given the additional constraint of a low-complexity implementation in modem hardware, it is evident to opt for alternative algorithms.

In [Kno+16b], OMP was proposed to solve this, and the algorithm has already been discussed in detail in Sec. 5.2. The presented digital architecture created with the RDAM could be applied to this problem.

Algorithm 8.1: Block Orthogonal Matching Pursuit (BOMP), [EKB10]

1 **function** BOMP($\mathbf{y}, \mathbf{S}, k, L$)
2 $\quad \mathbf{r} \leftarrow \mathbf{y}$; $\mathbf{h} \leftarrow \mathbf{0}$; $\mathcal{I} \leftarrow \emptyset$
3 $\quad$ **for** $1, ..., k$ **do**
4 $\quad\quad \mathbf{c} \leftarrow \mathbf{S}^H \mathbf{r}$
5 $\quad\quad \ell^* \leftarrow \arg\max_\ell \left\| \mathbf{c}_{[\ell]} \right\|_2$ ▷ block selection
6 $\quad\quad \mathcal{I} \leftarrow \mathcal{I} \cup \ell^*$
7 $\quad\quad \mathbf{h}_{[\mathcal{I}]} \leftarrow \arg\min_{\mathbf{h}_{[\mathcal{I}]}} \left\| \mathbf{y} - \mathbf{S}_{[\mathcal{I}]} \mathbf{h}_{[\mathcal{I}]} \right\|_2$ ▷ least squares
8 $\quad\quad \mathbf{r} \leftarrow \mathbf{y} - \mathbf{S}\mathbf{h}$
9 $\quad$ **return** $\mathbf{h}$

OMP solves eq. (8.12) approximately in a greedy fashion while it exploits the fact that $\mathbf{h}$ is maximally k-sparse. Since p_a is known, k can be chosen such that $k = \lceil p_a N \rceil$. When (8.12) is solved by OMP, the result vector is denoted as $\hat{\mathbf{h}}_{\text{OMP}}$.

8.2.2 Frequency-Selective Fading Channels

Frequency-selective fading necessitates the estimation of a block-sparse vector $\mathbf{h}$ of size LN, as stated in (8.4). The MAP optimisation problem for block-sparse joint activity and channel estimation is

$$\hat{\mathbf{h}} = \arg \min_{\mathbf{h} \in \mathbb{C}^{LN}} \|\mathbf{y} - \mathbf{C}_p \mathbf{h}\|_2^2 + \lambda \|\mathbf{h}\|_{2,0} \ . \tag{8.14}$$

Then, the cost function promotes block-sparsity with $\|\mathbf{h}\|_{2,0}$ according to (8.6) and block length $d = L$, i.e. the lengths of the channel impulse responses. Note, there are two only minor differences to the specialisation of flat-fading in (8.12).

Hence, a generalised OMP, with an extension for the recovery of block-sparse signals, should approximately solve it. This algorithm is called Block Orthogonal Matching Pursuit (BOMP) and was introduced in [EKB10]. When (8.14) is solved by BOMP, the result vector is denoted as $\hat{\mathbf{h}}_{\text{BOMP}}$. BOMP expects alls blocks to have the same length.

The pseudocode of BOMP is given in Fig. 8.1. The major difference to OMP is to be found in line 5, where not a single atom is selected, but instead the ℓ^*-th *block* of atoms of block size L. The selection is based

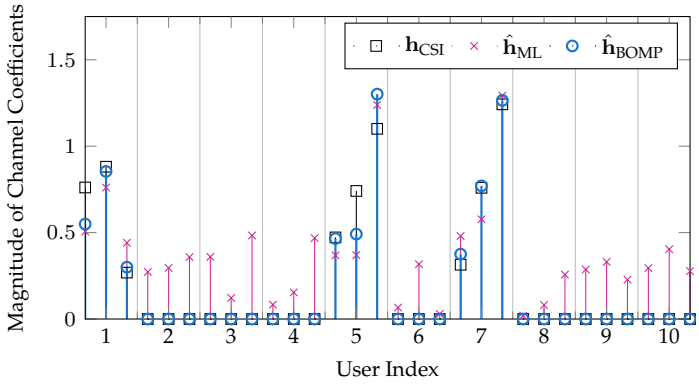


Figure 8.1: Channel impulse responses estimated by BOMP compared to the ground truth and ML estimation.

on the maximal Euclidean norm of the correlation values of each block. Therefore, $\mathcal{I}$ here denotes the set of selected block indices.

In order to clarify notational issues, the index notation $\mathbf{c}_{[\ell]}$ returns a smaller vector $\mathbf{c}[\ell]$ of length L, which is the ℓ-th block in $\mathbf{c}$. This is analogous to how single elements of a vector are addressed with an index, $c_i = \mathbf{c}(i)$. When a set is given as index, e.g. $\mathbf{h}_{[\mathcal{I}]}$ in line 7, all elements of the blocks in $\mathcal{I}$ are returned as a vector. Thus, the LS step of the same line computes $L \cdot |\mathcal{I}|$ values. The result of BOMP is, therefore, a block-sparse vector with k blocks of non-zero coefficients.

For CS with block sparsity constraint beyond fixed block sizes, the reader is referred to [Zia+10; EV12]. The variable block sizes nonetheless must be known beforehand.

8.2.3 Numerical Simulation Results

The system performance of the above explained joint activity and channel estimation was investigated with numerical Monte-Carlo simulations. For illustration purposes, a single random channel realisation of a multi-user channel impulse response vector $\mathbf{h}_{\text{CSI}}$ according to (8.4) is shown in Fig. 8.1. The system consists of $N = 10$ users, whereby $N_a = 3$ users are active (no. 1, 5 and 7). The channel impulse response length

Table 8.1: Simulation parameters for the joint user activity and channel estimation.

Parameter	Symbol	Value
No. of Users	N	30
Activity Probability	p_a	$\{0.1, 0.2, 0.3\}$
No. of Active Users	N_a	$p_a N$
Channel Imp. Resp. Length	L	$\{1, 3\}$
Pilot Sequence Length	N_p	$\{LN, LN/2\}$
Pilot Sequence Type	—	Zadoff-Chu, Random Gaussian

is $L = 3$, thus $\mathbf{h}_{\text{CSI}}$ is of length $LN = 30$ and N_a-block-sparse with $LN_a = 9$ non-zero elements. The system of equations is fully determined since $N_p = LN$ was chosen. BOMP correctly detects all active users and returns a truly sparse estimation $\hat{\mathbf{h}}_{\text{BOMP}}$. In contrast to that, simple ML estimation distributes the measured noise over all coefficients; an arbitrary threshold to recover the support would obviously lead to an erroneous activity detection.

For the following results, a standard set of simulation parameters was used, given in Table 8.1. Each data point was averaged over at least 20 000 random system model realisations.

Figure 8.2 shows the average detection activity error rate (AER) over the SNR. AER is defined to be N_f/N, i.e. the average number of false detections (false-active and false-inactive both accumulated) over the number of users. In this case, the system is fully determined with $N_p = LN = 90$. Hence, only measurement noise degrades the detection performance. It becomes obvious that frequency-selective fading enhances the activity and channel detection process. This is mainly because of the lesser outage probability of Rayleigh fading channels with $L > 1$. ZCSs can benefit more from this effect, though, especially when many users are active (better user separation). The plotted curves are for different levels of user activity ($p_a = 0.2$ not shown for the sake of clarity), where $p_a = 0.3$ corresponds to 9 active users, which is quite a lot compared to results given in other works [SBD13]. In all cases, the specially constructed multi-user ZCSs exhibit a better performance than random Gaussian sequences.

In Fig. 8.3, the same setup is investigated with increased bandwidth efficiency and with shorter pilot sequences of length $N_p = LN/2$, i.e.

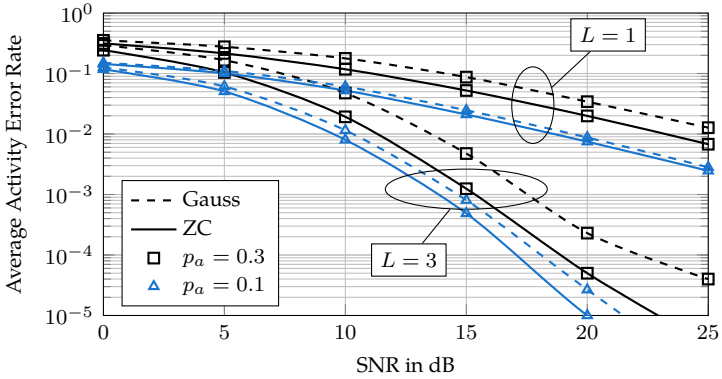


Figure 8.2: Average activity error rate for random Gaussian (dashed) and Zadoff–Chu (solid) sequences. The system is fully determined.

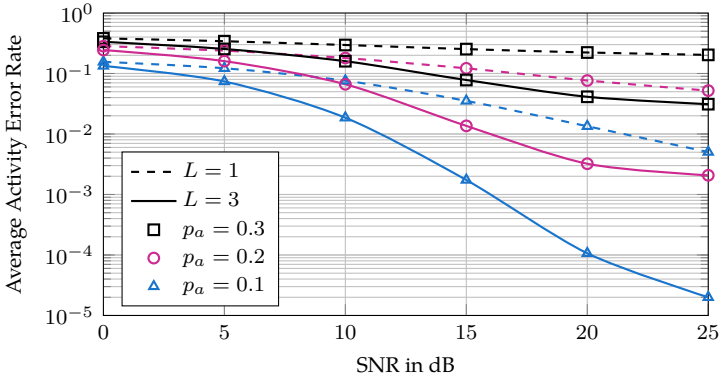


Figure 8.3: Average activity error rate for flat (dashed) and frequency-selective (solid line) fading channels and random Gaussian sequences. The system is underdetermined with the factor $1/2$.

Figure 8.4: Active users transmit user-specific pilots and additionally a common root sequence for frame synchronisation purposes.

an underdetermined system with factor $1/2$. Since ZCSs according to the construction rule given above do not exist for this case, one can only resort to random Gaussian sequences. These, however, perform fairly well and only little loss can be observed for $p_a = 0.1$ compared to the fully determined case. Detection is again better if diversity due to frequency-selective channels is available. Nonetheless, the spread of the curves is wider, which means the underdetermined system is less tolerant to support many active users. For $p_a > 0.3$, any reliable estimation is not possible.

8.3 Extension to Multi-User Frame Synchronisation

The above demonstrated that a joint multi-user activity and channel estimation in a WSN based on user-specific pilot code sequences can be facilitated. Diversity due to frequency-selective fading environments is helpful for detection. Specifically designed ZCSs, derived from a longer root sequence, outperform random Gaussian codes in detection performance. However, only the latter are able to support sequences shorter than the number of users, but only when there are few active users. No assumptions whatsoever were made about the user's data payload. Therefore, the obtained results are generally valid and applicable to any possible frame structure.

Nonetheless, the above setup still assumed a perfect frame synchronisation to obtain vectorised samples in $\mathbf{y}$. The sparse nature of the user's communication patterns poses a challenge to traditional frame synchronisation structures. Therefore, the first task of a receiver is to detect the beginning of multi-user frames and collect all pilot and payload into chunks of data for consecutive processing.

Building upon the constructed orthogonal basis of user pilot sequences derived from a single root ZCS $\mathbf{z}_r$, an extension to frame detection has been proposed in [Zed+17]. Principally, the root sequence itself is reserved for frame synchronisation in time. Each user transmits, if active, the assigned user pilot code for activity and channel estimation and additionally the root sequence. This is depicted in Fig. 8.4 schematically. The superposition of the root sequence $\mathbf{z}_r$ and user-specific pilot code $\mathbf{c}_n$, Eq. (8.11), has to be formed to obtain the preamble of the n-th device $\tilde{\mathbf{c}}_n$ such that

$$\tilde{\mathbf{c}}_n = \frac{1}{\sqrt{2}} \left(\mathbf{z}_r + \mathbf{c}_n \right) , \tag{8.15}$$

where $n = 2, ..., N$, i.e. the user index $n = 1$ is reserved for the root ZCS. Again, no further assumptions are made about the payload.

The receiver does not have to detect and synchronise each active user separately, but instead can look for the superimposed multi-user frame. In other words, the root sequence will be present in the preamble if at least one user is active. Afterwards, channel equalisation and user identification can be performed as suggested in [Kno+16b] and Sec. 8.1.

8.3.1 Neyman–Pearson Detection

The Neyman–Pearson (NP) lemma can be applied for simple and robust transmission detection because the root ZCS is known at the receiver and is transmitted by all active users as part of the preamble [HI14]. In the following, flat-fading channels are assumed, which is a reasonable assumption for low data rate MTC. Moreover, a couple of simplifications are made to reduce the amount of utilised FPGA resources.

The NP detector is a binary hypothesis testing algorithm that compares two hypotheses $\mathcal{H}_0$ and $\mathcal{H}_1$,

$$\mathcal{H}_0 : r(k) = n(k) \tag{8.16a}$$

$$\mathcal{H}_1 : r(k) = h z_r(k) + c(k) + n(k) , \tag{8.16b}$$

with $n(k)$ being complex-valued AWGN. The null hypothesis $\mathcal{H}_0$ indicates that the received sample $r(k)$ contains only noise, while $\mathcal{H}_1$ indicates the presence of noise as well of the superimposed multi-user preamble. At any time instance, on average $p_a N$ active users synchronously transmit their specific pilot preamble (8.15) over unique channels.

The low-rate assumption allows to model each user's wireless channel by a single coefficient. On the one hand, and with regard to the commonly transmitted root sequence, the channel coefficient h in (8.16) is the sum of all active user channels. This multi-user interference constructively leads to a channel hardening effect and makes detection robust, because the outage probability decreases the more users are active. On the other hand, the superposition of the user-specific pilot sequences can be viewed as an interfering random noise process sampled as $c(k)$, because instantaneous user activities are not known.

While $\mathcal{H}_0$ is a zero-mean random process, $\mathcal{H}_1$ is not. This statistical difference between the two hypotheses, can be exploited to establish a threshold which will provide a tunable detection performance. In [Zed+17], the NP test statistics Λ was derived to be the cross-correlation between the root sequence $z_r(k)$ and the received samples $r(k)$,

$$\Lambda(k) = \frac{1}{N_p} \left| \sum_{k=0}^{N_p-1} z_r^*(k) r(k) \right| \tag{8.17}$$

which is interpreted to fulfil $\mathcal{H}_1$ or $\mathcal{H}_0$ whether Λ is above or below a certain threshold λ, respectively. However, estimated signal and noise powers are necessary in order to obtain this analytical threshold. Further, please note that ideal multi-user interference suppression would require a cyclic computation with period N_p. This cross-correlation is acyclic though.

Frame synchronisation in time is achieved when the correlation peaks $N_p - 1$ samples after receiving the first transmitted preamble sample. Therefore, a buffer is required in order to record the N_p preamble samples that are necessary for further synchronisation, equalisation and user identification.

Functional verification of the system model was performed by numerical simulation in Mathworks Matlab for a system with $N = 31$ users and $N_p = 97$ over a range of SNR values. The frame detection rates p_{md}, for missed detection, and p_{fa}, for false-active detection, are plotted in Fig. 8.5. The threshold λ was chosen such that p_{fa} is a constant of low value while p_{md} improves with increasing SNR, which is a property of NP hypothesis testing. The length of the multi-user preambles was set to accommodate for frequency-selective fading channels up to $L = 3$,

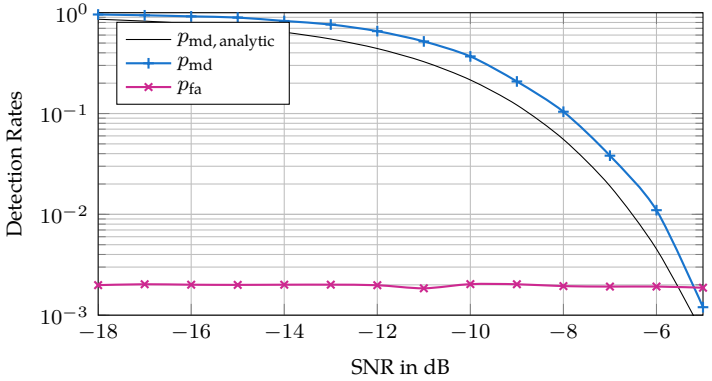


Figure 8.5: Simulated missed frame detection rate and false-active detection rate versus SNR [Zed+17].

however single-tap channels were simulated. Dummy payload data were randomly generated and spread by random codes drawn from a uniform distribution on the unit circle.

There is, nonetheless, a gap between the theoretical rate of missed detections and the simulated curve. The already mentioned aperiodic computation of the cross-correlation leads to a performance degradation which is, obviously, tolerable due to the length of the preamble sequences. Also, the power of the multi-user interference $c(k)$ is imperfectly known to the receiver. Anyway, Fig. 8.5 proves that the cross-correlation based frame detection is capable of successfully detecting superimposed multi-user preambles of sporadically active users.

8.3.2 RDAM-Based Implementation

A hardware architecture for Neyman–Pearson frame detector was created following the paradigm of the RDAM, Chap. 3. To begin with, three algorithmic transformations or implementation techniques were applied to simplify the design.

Firstly, given that the root ZCS is even in time, the number of multiplications of the correlation can be halved by adding up the mirrored received samples before multiplying them with the reference sequence.

Table 8.2: FPGA Resource Utilisation and Timing Results for the Correlation-Based Frame Detection

Name	Clock	Latency	II	DSP48E	FF	LUT
Sequential	9.77 ns	54	50	6	6790	6219
Parallel	9.68 ns	9	4	50	7461	6666
		Available Resources:		220	106400	53200

This is a symmetry property of all odd-length ZCSs as proven in [Zed+17]. Then, Eq. (8.17) becomes

$$\Lambda = \frac{1}{N_p} \left| \sum_{k=0}^{\frac{N_p-1}{2}} z_r^*(k) \big[r(k) + r(N_p - k - 1) \big] \right| . \tag{8.18}$$

This is basically analogous to how a symmetric finite impulse response (FIR) filter can be computed. Solely the complex-conjugated root sequence is needed, thus conjugation can be omitted in hardware if $z_r^*(k)$ is pre-computed and stored.

Secondly, for the first and last received samples $r(k = 0)$ and $r(k = N_p - 1)$ complex-valued multiplication can be skipped as they are always multiplied by one, $z_r(0) = 1$. This will save four real-valued multiplication operations.

Thirdly, the Maximum Alpha Minimum Beta (MAMB) algorithm can be used to compute the absolute value of the correlation in (8.18), [Lyo04]. MAMB approximates the absolute value $v = \sqrt{a^2 + b^2}$ of a complex number $a + jb$ by

$$v = \alpha \max(a, b) + \beta \min(a, b) . \tag{8.19}$$

With $\alpha = 0.9486$ and $\beta = 0.3929$ this will result in a mean squared approximation error of 0.5773×10^{-3} for the evaluation of the absolute values of complex numbers that have unit magnitude. Doing so trades a square root operation for a much simpler if-statement.

As a next step, the detection algorithm has been ported to C++ and synthesised with VHLS for a Zynq-7020 FPGA, as it is the processing core

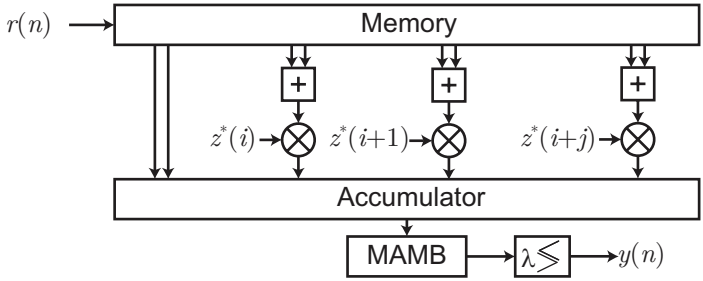


Figure 8.6: Synthesised micro-architecture of the Neyman–Pearson frame detector.

of a Nutaq ZeptoSDR, for a target operating clock frequency of 100 MHz. For reference, Tab. 8.2 states the number of available hardware resource blocks. Restricting HLS compilation by a pipeline directive with an II of 4 clock cycles yields a parallel architecture which can process a new input sample every 4 clock cycles with a latency of 9 clock cycles. Other necessary directives are related to the partitioning of the data arrays to enable multiple concurrent data read and write accesses to the memory. Tab. 8.2 lists all figures of merit for both architectures. Additionally, a sequential architecture was defined by a more relaxed timing constraint of $\mathrm{II} = 50$, which drastically reduces the number of utilised hardware multipliers (DSP48E slices). Both architectures comfortably fit into the available resources of the FPGA. Input and output samples are 12 bit wide fixed-point data types to match the design of the incoming in-phase & quadrature-phase (IQ) samples from the analogue-digital converter (ADC).

Since the generated architecture will have a a clock period of less than 10 ns duration, even the sequential implementation can throughput more than 2 MS/s. Therefore, the algorithm meets the timing constraint of a potential low-rate MTC application and is capable of real-time operation. The implementation was successfully tested in hardware using Xilinx ChipScope with debug data streams.

Fig. 8.6 shows the HLS hardware architecture of the proposed NP activity detection algorithm. Input samples are stored within a memory consisting of registers which allow parallel and random read accesses.

The k-th input sample (with regard to the root ZCS of length N_p) is delayed by $N_p - (2n + 1)$ clock cycles to exploit the central symmetry of odd-length ZCSs. There is a number of parallel complex-valued additions followed by complex-valued multiplications, depending on the degree of inferred parallelism by HLS. In case of the parallel architecture, there are $J = 12$ such parallel data paths. The complex conjugated coefficients of the root sequence are read from a ROM. Subsequently, an accumulator sums up the iteratively computed partial correlation products (i.e. 4 iterations for $N_p = 97$). The approximated magnitude of the final scalar result is determined by the MAMB algorithm and then compared against a threshold for hypothesis testing. The threshold λ can be implemented as a constant read from a LUT as a function of the coarsely estimated signal and noise powers.

8.4 Discussion

A joint multi-user activity and channel estimation in a WSN based on user-specific pilot code sequences can successfully be facilitated.

Diversity due to frequency-selective fading environments is helpful for detection. Specifically designed ZCSs, derived from a longer root sequence, outperform random Gaussian codes in detection performance. However, only the latter are able to support sequences shorter than the number of users, but only when there are few active users.

As a last innovation, a frame detection for sporadically active users was presented to obtain vectorised received samples in a multi-user measurement vector $\mathbf{y}$ ready for discrete CS processing. Results demonstrate that a frame detection based on cross-correlations and on a shared root ZCS is possible. Neither for frame detection nor for the joint user activity and channel estimation any assumptions were made about the data payload. Therefore, the obtained results are generally valid and applicable to any possible frame structure.

After frame detection, the joint user activity and channel estimation problem can be solved by OMP or a generalisation thereof for block-sparse multi-user channel vectors, BOMP. These algorithms can supply the baseband receiver with needed user activity and CSI.

Chapter 9

Conclusion and Open Research Issues

Computationally complex algorithms challenge traditional digital VLSI architecture design techniques on the behavioural register-transfer (RT) layer, and the productivity design gap requires digital design to be elevated to a more abstract level. The proposed and presented Rapid Data Type-Agnostic Digital Design Methodology (RDAM) based on HLS is an answer to this challenge.

9.1 The Rapid Data Type-Agnostic Digital Design Methodology

Designing with HLS delegates fine-grained control over hardware details to the HLS compiler. Actually, such control is not always necessary, and with respect to algorithmic architecture design, configurability and the exploitation of parallelisms are of far greater importance. The compiler steps in for explicitly unspecified aspects, like timings, resets and so on. Hence, designers can focus on the functional implementation.

The RDAM inherits all the advantages of HLS and further benefits from synergy effects obtained by the consequent exploitation of the added DTA. The data type-agnostic design sources bring simplifications regarding the fixed-point conversion of algorithms and fixed-point arithmetic with a verified overall functionality and precision by the test-driven development with the HLS test bench. Having a target numerical precision leans on ideas of approximate computing and augments the design space with an additional dimension for trade-offs. It is similar with the

configurable data type parametrisations. Informed choices of data types, bit widths, etc. can be made in a Pareto-optimal sense by scripts which link Mathworks Matlab and Xilinx Vivado HLS for semi-automated parametric sweeps. Often, a single baseline fixed-point format can be applied throughout the whole design.

The survey on various floating-point and fixed-point encodings of binary numbers in Sec. 3.2 revealed that these two classes share many characteristics, foremost a limited precision but also limited ranges such that overflows and underflows can occur. As long as fixed-point values are properly scaled and confined to the representable interval, they perform very much like floating-point data types indeed. The utilisation of the Q format notation further simplifies fixed-point design, and specialised data types based on the SystemC class `sc_fixed` encapsulate the necessary fixed-point intricacies, adhering to the rules for Q format changes due to arithmetic operations. Hence, a pseudo floating-point behaviour is attained and an exchange of floating-point data types in favour of the former comes with no degradation of numerical accuracy (given the same bit widths).

Mathematically non-trivial functions beyond the four basic arithmetic operations can be approximated by polynomials or by computations in the logarithmic domain. A design example for an (inverse) square root has been presented. In connection with the RDAM, such reusable code libraries can accelerate, or enable in the first place, the fast deployment of fixed-point implementations.

Another area where the RDAM benefits from the DTA are complex-valued designs with an implicit conversion to real-valued operations. The creation of the first complex-valued digital architecture for OMP required no code modifications but solely a different complex-valued parametrisation of the custom data type definitions.

For testing and functional verification besides the bit-accurate C simulations and RTL co-simulations offered by Vivado HLS, a HIL simulation framework has been created which provides a communication link for arbitrary data between Matlab and the I/O interface of an AUT implemented on the programmable logic part of a Xilinx Zynq SoC. For instance, the generated IP core of the OMP design was successfully verified in this way. A comparison of the estimated resource utilisation of Vivado HLS (VHLS) and the eventually utilised resources lead to the significant observation that only the DSP slice count is invariant between HLS

synthesis and post-RTL synthesis. Therefore, the number of used DSP slices is the best-suited figure of merit to grasp the area complexity of an FPGA design and corresponds to what the GE is for VLSI designs targeting ASICs.

9.2 Sparse-Coded Multi-User Communications

The efficacy of the RDAM has been demonstrated in particular by its application to OMP in Sec. 5.2 [Kno+16a], yet additional "products" of the RDAM are the first digital architecture of ACGP [Kno+16c], the highly configurable SD [Kno+17] and the data frame detector based on Neyman–Pearson hypothesis testing [Zed+17], mentioned in Part II.

These algorithms altogether play a role for the detection of sparse-coded wireless multi-user communications, relying on ideas of CS. Given the finite-alphabet constraint of digital communication systems, a new class of algorithms, namely tree search algorithms, could successfully be adapted to solve the CS recovery problem. The performance of a joint multi-user activity and symbol detection was measured in terms of the GSER and run time, i.e. the number of processed nodes within the search tree.

A first examination of sparsity-aware SD showed, that the constrained search is able to find the optimal discrete multi-user solution vector with a tremendously reduced run time. This stimulated the study of complexity-reduced alternatives, namely sparsity-aware K-Best and SIC. The unique approach put forward to combine these two algorithms with an SQRD for preprocessing within this CS context facilitates successful detection with only minor GSER performance degradation compared to optimal SD. Furthermore, the sorting simultaneously leads to a pre-whitening filter for improved SIC detection in symbol clock after a correlation-based reception of multi-user CDMA signals. With regard to K-Best, the SQRD also enables detection from compressive measurements for reasonable values of the parameter K if the system of equations is regularised by the sparsity constraint accordingly.

In order to solve the CSI estimation problem in a setup with sporadic and instantaneously unknown user activities, a novel multi-user orthogonal code system based on ZCSs and derived from a root sequence is proposed. The estimation problem can be recast as the CS recovery of a

block-sparse vector containing the impulse responses of all users. BOMP and, given flat-fading channels, OMP could fruitfully be applied to a combined multi-user activity and channel estimation.

Last but not least, the time-synchronously superimposed data frames of the sporadically active users have to be detected by the receiver within the incoming IQ data stream from the antenna. To this end, a Neyman–Pearson hypothesis test was examined and implemented with the RDAM.

9.3 Open Issues for Future Research

Part II identified necessary algorithms for the reception of sparse-coded multi-user wireless transmissions. However, each function, i.e. frame detection, CSI and activity estimation and symbol detection, was designed and examined individually. A further proof of concept would ultimately be possible now by incorporating all these pieces into a technology demonstrator for sparse-coded multi-user reception.

Although the number of users assumed in the WSN is quite substantial, given the length of the user-specific code sequences, it is still relatively small compared to the postulated MTC scenarios for 5G communications. Further study should examine how the proposed solutions scale for larger networks. Alternatively, the concept of Massive MIMO communications could be included in the system model to improve detection performance with a spatial dimension. Until now, a multi-user synchronisation in time has been assumed and non-idealities of RF front-ends, like carrier frequency offsets, have been neglected. These must be further taken into account. The question is open if truly uncoordinated access of users with a priori unknown activities is practically feasible.

A potential technology demonstrator would require the creation of a custom baseband signal processing chain. The efficacy of the RDAM in relation to the digital design of algorithms operating as a packaged IP core has been demonstrated. Yet, several of these blocks need to be chained up and integrated into a complete system design. Future research should investigate if the RDAM can be extended to the creation of signal processing chains.

The augmented design space exploration capabilities, due to the DTA utilised by the RDAM, certainly help the designer to make sound design decisions. Nonetheless, however, this process is only automated partly.

Additional research effort could lead to an intelligent and entirely automated process in order to exploit algorithmic parallelisms (pipelining and loop unrolling) or to size bit widths sufficiently, given appropriate constraints, e.g. maximal resource utilisation figures (number of HW multipliers, adders, etc.) and a target numerical accuracy, respectively.

The HLS synthesis process is dependent on the knowledge of known constant parameters at compile time. This allows for vast configuration options, e.g. demonstrated by the RDAM-based SD in Sec. 7.2.1.2, but not for a re-configurable system at run time. Algorithmic parameters concerning sizes, e.g. the (M, N, k) tuple of OMP, are fixed once the compilation finishes. Even though, a dynamic change of such parameters is desirable to obtain more versatile digital architectures. This open issue aims directly at a core limitation of VHLS C synthesis not (yet) supporting dynamic memory allocations and asks for a solution bypassing it.

Further, the DTA of the RDAM builds upon the notion that different number encoding altogether model real numbers. This works very well with floating-point formats and fixed-point numbers with the help of the Q format. Sec. 3.2.1.2 brushed the idea of unums; the examination of these configurable floating-point numbers for their application by the RDAM would certainly be valuable. The same applies to the ideas of stochastic computing, where numbers are represented by weighted bit streams [AH13].

Appendix A

Fixed-Point Arithmetic with the Q Format

Basically, every fixed-point number is an integer value. Only its interpretation is weighted or scaled by a binary power. The so-called Q format, introduced in Sec. 3.2.2.2, conveniently describes this circumstance and moreover helps to keep track of format changes due to arithmetic operations. This facilitates pseudo floating-point computations.

A.1 Format Changes Due to Arithmetic Operations

Tab. 3.3 on page 50 lists how arithmetic operation with fixed-point numbers affect the Q format of the resulting bit string. These laws pertaining Q format computations will be proven in the following, but before that some preliminary definitions shall be introduced.

The format $\mathrm{Q}(m,n)$ does not only partition the bit string of length $w = m+n+1$ into m integer bits and n fractional bits plus a sign bit, but also represents the *set* of real-valued rational numbers which can exactly be represented by this encoding.

Let a, b (inputs) and r (result) be binary fixed-point numbers where

$$a \in \mathrm{Q}(m_a, n_a)\,, \qquad b \in \mathrm{Q}(m_b, n_b)\,, \text{ and} \qquad r \in \mathrm{Q}(m_r, n_r)\,, \tag{A.1}$$

the parameters m_μ, n_ν for all $\mu, \nu \in \{a, b, r\}$ being positive integers. The number range, here exemplarily for a, is bounded by

$$a \in [a_{\min}, a_{\max}] = [-2^{m_a}, 2^{m_a} - \Delta_{n_a}] \tag{A.2}$$

$a+b$	$b_{\min}$	$b_{\max}$
$a_{\min}$	-2^{m+1}	$-\Delta$
$a_{\max}$	$-\Delta$	$2^{m+1}-2\Delta$

(a) Addition

$a-b$	$b_{\min}$	$b_{\max}$
$a_{\min}$	0	$-2^{m+1}+\Delta$
$a_{\max}$	$2^{m+1}-\Delta$	0

(b) Subtraction

Figure A.1: Resultant maximal and minimal values of addition and subtraction ($m = \max(m_a, m_b)$ and $\Delta = 2^{-\max(n_a,n_b)}$).

with $\Delta_{n_a} = 2^{-n_a}$ being the quantisation step. Then, an arithmetic operation $r = a \circ b$ of two variables (binary operator) can be described by the relation

$$a \circ b : \mathrm{Q}(m_a, n_a) \times \mathrm{Q}(m_b, n_b) \to \mathrm{Q}(m_r, n_r)\,. \tag{A.3}$$

The resulting Q format of any arithmetic operation can be evaluated by an analysis of the corner cases.

A.1.1 Addition and Subtraction

For additions $r = a + b$ and subtractions $r = a - b$ the position of the radix point of the operands a, b needs to be aligned, which is expressed by n_a and n_b. In order to avoid underflows, $n_r = \max(n_a, n_b)$ must hold.

Assume w.l.o.g. that $m_a \geq m_b$. The maximal absolute value of a is $\max_{a\in\mathrm{Q}(m_a,n_a)} |a| = |-2^{m_a}| = 2^{m_a}$. Hence, the largest possible absolute value for the sum r will be $2^{m_a} + 2^{m_a} = 2 \cdot 2^{m_a} = 2^{m_a+1}$, which implies equality $m_a = m_b$. The same holds analogously for the difference $r = a+(-b)$; the sign inversion of b does not change this bound. The resulting number range can thus be bounded by $r \in [-2^{m_a+1}, 2^{m_a+1})$. Note, that this is a right-open interval, because for additions the maximal positive result $a_{\max} + a_{\max} = (2^{m_a} - \Delta) + (2^{m_a} - \Delta) = 2^{m_a+1} - 2\Delta$ falls short of the upper bound by two times the quantisation step $\Delta = 2^{-n_r}$. For subtractions, it is similar as it can be seen from Fig. A.1, which gives all eight corner cases. Thus, $r \in [r_{\min}, r_{\max}]$ with

$$r_{\min} = a_{\min} + b_{\min} = -2^{m_a+1}\,, \text{ and} \tag{A.4}$$

$$r_{\max} = a_{\max} - b_{\min} = 2^{m_a+1} - \Delta_{n_r} \tag{A.5}$$

holds for both operations combined. In conclusion, the resulting Q format of an addition or subtraction is

$$a \pm b : \mathrm{Q}(m_a, n_a) \times \mathrm{Q}(m_b, n_b) \rightarrow \mathrm{Q}(\max(m_a, m_b)+1, \max(n_a, n_b)) \,. \quad \text{(A.6)}$$

The additional accumulation bit is also required if one of the operands is very small, e.g. $b = \Delta$: the computations $a_{\max} + \Delta$ or $a_{\min} - \Delta$ will lead to overflows if the integer width m_a is not extended by one bit.

A.1.2 Repeated Accumulation

Of special interest is the k-fold accumulation of fixed-point numbers of same format,

$$r = \sum_{\ell=1}^{k} a_\ell \,, \qquad \forall\, a_\ell \in \mathrm{Q}(m_a, n_a) \,. \quad \text{(A.7)}$$

The derivation of the resulting Q format follows the proof above analogously, with the exception that the maximal and minimal values are bounded by $\pm k 2^{m_a}$. Raising the parameter k into the exponent leads to $2^{m_a + \lceil \mathrm{ld}(k) \rceil}$. Hence,

$$a_1 \pm ... \pm a_k : \mathrm{Q}(m_a, n_a) \times ... \times \mathrm{Q}(m_a, n_a) \rightarrow \mathrm{Q}(m_a + \lceil \mathrm{ld}(k) \rceil, n_a) \,. \quad \text{(A.8)}$$

If the number of accumulations is known beforehand, this rule allows to efficiently reserve sparse bits because $m_a + \lceil \mathrm{ld}(k) \rceil < m_a + k$. Note, the precision n is unaffected, since the radix points are already aligned.

A.1.3 Multiplication

Let

$$A \in \mathrm{Q}(m_a + n_a, 0), \quad B \in \mathrm{Q}(m_b + n_b, 0), \quad R \in \mathrm{Q}(m_r + n_r, 0) \quad \text{(A.9)}$$

as corresponding integer values to a, b, r and $A, B, R \in \mathbb{Z}$, since the fractional part vanishes. Yet, these integers are range-limited $\mathrm{Q}(t = m_\mu + n_\nu, 0) = \{-2^t, ..., -1, 0, 1, ..., 2^t - 1\} \subset \mathbb{Z}$ because of their finite word lengths. It is

$$a = A \cdot 2^{-n_a} \qquad b = B \cdot 2^{-n_b} \qquad c = R \cdot 2^{-n_r} \,. \quad \text{(A.10)}$$

The multiplication of two fixed-point multiplicands can then be expressed with integers as

$$r = a \cdot b = A\,2^{-n_a} \cdot B\,2^{-n_b} = AB\,2^{-(n_a+n_b)} = R\,2^{-n_r}\,. \tag{A.11}$$

Obviously, $n_r = n_a + n_b$ fractional bits are needed for full-precision multiplications. Practically, however, this growth becomes problematic for chained multiplications. Word length restrictions need to be applied, therefore, and underflows are unavoidable, but a certain target precision can still be ensured. The resulting maximal absolute value can be bounded by $|R| \leq \max|AB| = \max|A| \cdot \max|B| = 2^{m_a+n_a} \cdot 2^{m_b+n_b}$. From Eq. (3.13) (on page 49) follows directly that $R \in \mathrm{Q}(m_a + n_a + m_b + n_b, 0)$. Taking the binary scaling with $2^{-(n_a+n_b)}$ into account, it follows that

$$a \cdot b : \mathrm{Q}(m_a, n_a) \times \mathrm{Q}(m_b, n_b) \to \mathrm{Q}(m_a + m_b, n_a + n_b)\,. \tag{A.12}$$

The special case of all-fractional fixed-point numbers $\mathrm{Q}(0, n) = \mathrm{Q}n$ is very convenient in handling for the reason that the integer part never changing its size (which is zero). In other words, multiplications of $\mathrm{Q}n$ numbers will absolutely never cause overflows for multiplicands *strictly* less than one in magnitude.

A.1.4 Reciprocal

The reciprocal value of a, $a \neq 0$ and

$$a \in \{-2^{m_a}, ..., -2^{-n_a}, 2^{-n_a}, ..., 2^{m_a} - 2^{-n_a}\}\,, \tag{A.13}$$

is given by

$$\frac{1}{a} \in \{-2^{-m_a}, ..., -2^{n_a}, 2^{n_a}, ..., \frac{1}{2^{m_a} - 2^{-n_a}}\}\,. \tag{A.14}$$

Whereas $|a| \in [2^{-n_a}, 2^{m_a}]$ holds, the absolute values of the reciprocal lie within the interval $|1/a| \in [2^{-m_a}, 2^{n_a}]$. The needed quantisation step is 2^{-m_a} and the range is limited by $|1/a| \leq 2^{n_a}$. According to Eq. (3.13) it follows

$$1/a : \mathrm{Q}(m_a, n_a) \to \mathrm{Q}(n_a, m_a)\,. \tag{A.15}$$

The sizes of the fractional and integer parts are exchanged. Since the reciprocal function $f(a) = 1/a$ is its own inverse (involution), $f(f(a)) = a$, a re-application will change fractional and integer parts again to the original state.

A.1.5 Division

The division $r = a/b$ can be re-written as a multiplication using the just derived relation (A.15) for the reciprocal, therefore also known as multiplicative inverse. Then, $r = a \cdot \frac{1}{b}$ with $1/b \in \mathrm{Q}(n_b, m_a)$ results in

$$a/b : \mathrm{Q}(m_a, n_a) \times \mathrm{Q}(m_b, n_b) \to \mathrm{Q}(m_a + n_b, n_a + m_b) \tag{A.16}$$

by application of (A.12).

A.1.6 Square Root

The resulting Q format of the square root $r = \sqrt{a}$ with the radicand $a \geq 0$ can be derived using the bound $r \leq \sqrt{a_{\max}} = \sqrt{2^{m_a} - 2^{-n_a}} < \sqrt{2^{m_a}} = 2^{m_a/2}$. The square root maps real values to real values, i.e. irrational numbers, $\mathbb{R}_0^+ \to \mathbb{R}_0^+$. Even if the domain is restricted to code points $\mathrm{Q}(m_a, n_a) \subset \mathbb{Q} \subset \mathbb{R}_0^+$, function values will still be within the co-domain $\mathbb{R}_0^+$ in general. Underflows are thus unavoidable to encode the result with a quantised Q format, albeit the desired level of precision can be chosen arbitrarily. With Eq. (3.13),

$$\sqrt{a} : \mathrm{Q}(m_a, n_a) \to \mathrm{Q}(\lceil m_a/2 \rceil, \lceil \mathrm{ld}(1/p) \rceil)\,. \tag{A.17}$$

is obtained with p denoting the target precision of the operation. The choice of $n_c \geq n_a$ for the output comes up naturally to be as precise as the input value. Note, the sign bit is included by the Q format notation and could be economised further due to the restriction to positive real values.

List of Figures

List of Tables

List of Code Listings

List of Algorithms

Acronyms and Abbreviations

5G	5th Generation Mobile Network
ACF	Autocorrelation Function
ACGP	Approximate Conjugate Gradient Pursuit
ADC	Analogue-Digital Converter
AER	Activity Error Rate
a.k.a.	Also Known As
ALU	Arithmetic Logic Unit
AMP	Approximate Message Passing
ANSI	American National Standards Institute
API	Application Programming Interface
ARM	Advanced Risc Machine
ASIC	Application-Specific Integrated Circuit
AUT	Algorithm Under Test
AWGN	Additive White Gaussian Noise
AXI	Advanced Extensible Interface Bus
BER	Bit Error Rate
BOMP	Block Orthogonal Matching Pursuit
BP	Basis Pursuit
BPDN	Basis Pursuit Denoising
BPSK	Binary Phase-Shift Keying
BRAM	Block RAM
BRVD	Block-Wise Real-Value Decomposition
CAZAC	Constant Amplitude and Zero Autocorrelation

CDFG	Control and Data Flow Graph
CDMA	Code-Division Multiple Access
CG	Conjugate Gradient
CORDIC	Coordinate Rotation Digital Computer
CoSaMP	Compressive Sampling Matching Pursuit
CPU	Central Processing Unit
CS	Compressed Sensing (or Compressive Sampling)
CSI	Channel State Information
CSMA	Carrier-Sense Multiple Access
CSMA-CA	CSMA With Collision Avoidance
CSMA-CD	CSMA With Collision Detection
DFG	Data Flow Graph
DMA	Direct Memory Access
DoA	Direction of Arrival
DS-CDMA	Direct Sequence CDMA
DSP	Digital Signal Processor or Digital Signal Processing
DSSS	Direct Sequence Spread Spectrum
DTA	Data Type Agnosticism
ERVD	Element-Wise Real-Value Decomposition
ESL	Electronic System-Level
FF	Flip-Flop
FFT	Fast Fourier Transform
FIFO	First-In First-Out Register
FIR	Finite Impulse Response
FLOP	Floating-Point Operation
FLP	Floating-Point
FPGA	Field-Programmable Gate Array
FSM	Finite State Machine
FXP	Fixed-Point

GCC	GNU Compiler Compilation
GE	Gate Equivalent
GP	Gradient Pursuit
GPU	Graphics Processing Unit
GSD	Generalised Sphere Decoder
GSER	Gross Symbol Error Rate
HDL	Hardware Description Language
HIL	Hardware-in-the-Loop
HLL	High-Level Language
HLS	High-Level Synthesis
HW	Hardware
IEEE	Institute of Electrical and Electronics Engineers
IHT	Iterative Hard Thresholding
II	Initiation Interval
i.i.d.	Independent and Identically Distributed
ILS	Integer Least Squares
IMT-2020	International Mobile Telecommunication System 2020
I/O	Input-Output
IoT	Internet of Things
IP	Intellectual Property
IQ	In-Phase & Quadrature-Phase
IR	Intermediate Representation
ISI	Inter-Symbol Interference
ISO	International Organization For Standardization
IST	Iterative Soft Thresholding
ITU	International Telecommunication Union
LASSO	Least Absolute Shrinkage and Selection Operator
LLVM	Low Level Virtual Machine
LNS	Logarithmic Number System

LP	Linear Program
LS	Least Squares
LSB	Least Significant Bit
LTE	Long Term Evolution
LTI	Linear Time-Invariant
LUT	Look-Up Table
MAC	Multiply-and-Accumulate Operation
MAC	Medium Access Control
MAMB	Maximum Alpha Minimum Beta
MAP	Maximum A Posteriori
MGS	Modified Gram–Schmidt Orthogonalisation
MIMO	Multiple-Input Multiple-Output
ML	Maximum Likelihood
MP	Matching Pursuit
MSB	Most Significant Bit
MSE	Mean Squared Error
MTC	Machine-Type Communications
NaN	Not a Number
NMSE	Normalised Mean Squared Error
NP	Neyman–Pearson
NPA	Non-Uniform Piecewise-Linear Function Approximation
OFL	Overflow Level
OMP	Orthogonal Matching Pursuit
OpenCL	Open Computing Language
OSI	Open Systems Interconnection
PAD	Partial Accumulated Distance
PD	Partial Distance
PDF	Probability Density Function
PHY	Physical Layer

PMF	Probability Mass Function
PN	Pseudo-Noise
PSK	Phase-Shift Keying
QAM	Quarternary Amplitude Modulation
QPSK	Quarternary Phase-Shift Keying
QRD	QR Matrix Decomposition
RAM	Random Access Memory
R&D	Research and Devolopment
RDAM	Rapid Data Type-Agnostic Digital Design Methodology
RF	Radio Frequency
RIP	Restricted Isometry Property
RISC	Reduced Instruction Set Computer
ROM	Read-Only Memory
ROMP	Regularised Orthogonal Matching Pursuit
RTL	Register-Transfer Level
RVD	Real-Value Decomposition
SA	Sparsity-Aware
SA-SIC	Sparsity-Aware Successive Interference Cancellation
SC	Sphere Constraint
SCMA	Sparse-Coded Multiple Access
SD	Sphere Decoding (or Sphere Decoder)
SDR	Software-Defined Radio
SER	Symbol Error Rate
SIC	Successive Interference Cancellation
S-MAP	Sparsity MAP
SNR	Signal-to-Noise Ratio
SoC	System-on-Chip
SOCP	Second-Order Cone Program
SotA	State of the Art

SQRD	Sorted QR Decomposition
SSA	Static Single Assignment
SVD	Singular Value Decomposition
TLM	Transaction Level Modeling
UFL	Underflow Level
ULP	Unit in the Last Place
unum	Universal Number
VHDL	Very High Speed Integrated Circuit Hardware Description Language
VHLS	Xilinx Vivado HLS
VLSI	Very-Large-Scale Integration
w.h.p.	With High Probability
w.l.o.g.	Without Loss of Generality
WSN	Wireless Sensor Network
ZCS	Zadoff–Chu Sequence

Mathematical Notations and Symbols

Latin Symbols

Symbol	Description
a	Operand or Input Value
$\mathbf{a}$	User Activity Vector
$\mathbf{A}$	System Matrix
$\mathcal{A}$	Finite Modulation Alphabet
b	Operand or Input Value
b	Bits
$\mathbf{b}$	Measurements
B	Bandwidth in Hertz
c	Correlation
$\mathbf{c}$	(Spreading) Code
$\mathbf{C}$	Code Matrix
$\mathbf{C}_p$	Matrix of Pilot Code Sequences
d	(Binary) Digit
d_ν	Partial Distance
D_ν	Partial Accumulated Distance
e	Exponent
E	Floating-Point Biased Exponent
E_b	Energy per Bit
f	Fraction Value (unum)
f	Function

Symbol	Description
$\hat{f}$	Approximated Function
f_{clk}	Clock Frequency in Hertz
f_c	Corner Frequency
f_s	Sampling Frequency
$\mathcal{F}$	Set of Representable Floating-Point Numbers
$\mathbf{h}$	Channel Impulse Response
$\mathbf{H}$	Channel Matrix
$\mathcal{H}_{0/1}$	Hypotheses
I	Integer Bits (`sc_fixed`)
II	Initiation Interval
k	Integer Parameter
k	Sparsity Level of a Vector
K	K-Best Parameter
ℓ	Integer Parameter
ℓ_p	Vector Norm
L_h	Channel Length
m	Mantissa or Significand
m	Integer Part of Fixed-Point Numbers in Bits
M	(Upper) Bound
M	Size (e.g. of a Matrix)
n	Integer Parameter
n	Fractional Part of Fixed-Point Numbers in Bits
$\mathbf{n}$	Additive Noise
N	Size (e.g. of a Vector)
N	Number of Users
N	Saturation Bits (`sc_fixed`)
N_0	Noise Density
N_a	Number of Active Users
N_p	Length of Pilot Sequences

Symbol	Description
N_s	Spreading Factor
O	Overflow Mode (`sc_fixed`)
p	Floating-Point Precision bits
p	Target Precision (linear)
p_a	Activity Probability
$\mathbf{P}$	Pre-Whitening Filter
q	Exponent Size in Bits
Q	Fractional Bits (`sc_fixed`)
$\mathbf{Q}$	Orthogonal or Unitary Matrix
r	Operation Result or Received Value
$\mathbf{r}$	Residual
$\mathbf{R}$	Upper-Triangular Matrix
s	Sign Bit
$\mathbf{s}$	User's CDMA Signature
$\mathbf{s}$	k-Sparse Vector
$\hat{\mathbf{s}}$	Recovered k-Sparse Vector
S	NPA Segmentation
$\mathbf{S}$	Signatures Matrix
t	Integer Parameter (Iterator)
t	Floating-Point Trailing Significand
t	Time Duration
$\mathbf{T}$	Wireless Communications Channel
T	Polynomial Degree
`T`	Data Type `T`
u	Unit Roundoff
u	u Bit (unums)
v	A Value
w	Word Length in Bits
W	Word Length in Bits (`sc_fixed`)

Symbol	Description
W_N	N-th Root of Unity
x	Input Value
$\mathbf{x}$	Input Vector
$\hat{\mathbf{x}}$	Recovered Input Vector
y	Input or Output Value
$\mathbf{y}$	Result Vector
$\mathbf{z}$	Zadoff–Chu Sequence
$\mathbf{z}_r$	Root ZCS

Greek Symbols

Symbol	Description
α, β	Parameters
β	Base or Radix
δ	Small Difference
δ_k	RIP Constant
Δ	Quantisation Step
ϵ	Machine Epsilon
ε	Optimisation Parameter
ζ	Inverse of a Vector Norm
$\mathbf{\Theta}$	CS System Matrix
λ	Lagrange Parameter
Λ	Index Set
Λ	Neyman–Pearson Test Statistics
μ	Mean
$\mathbf{\Pi}$	Permutation (Matrix)
ϱ	Sphere Constraint
σ	Standard Deviation
σ^2	Variance

Symbol	Description
σ_n^2	Noise Power
σ_Q^2	Quantisation Noise Power
$\mathbf{\Phi}$	CS Measurement Matrix
$\mathbf{\Phi_{nn}}$	Covariance Matrix
ϕ_{XX}	Autocorrelation Function
ϕ_{XY}	Cross-Correlation Function
$\mathbf{\Psi}$	CS Transform Basis

Mathematical Notations

Notation	Description
$[a, b], (a, b)$	Closed and Open Interval
$(\cdot, ..., \cdot)$	Tuple
$\{\cdot, ..., \cdot\}$	Set
$\mathcal{A}$	Set
$\mathcal{A}_0$	A Set $\mathcal{A}$ Including the Zero
$\lvert\mathcal{A}\rvert$	Cardinality of the Set $\mathcal{A}$
A	Scalar (Integer)
a	Scalar
$\lvert a\rvert$	Absolute Value of a
$\mathbf{a}$	Column Vector
(a_i)	Vector $\mathbf{a}$
a_i	i-th Element of $\mathbf{a}$
$a_{[\ell]}$	ℓ-th Block of a Vector
$\mathbf{a}^{(k)}$	Partial Vector $(a_k), k \geq i$
$\mathbf{a}_\Lambda$	Vector $\mathbf{a}$ Only with the Elements Indexed by Λ
$\mathbf{a}_w$	Pre-Whitened Vector
$\lVert\mathbf{a}\rVert_0$	ℓ_0 Pseudonorm
$\lVert\mathbf{a}\rVert_p$	ℓ_p Vector Norm ($1 \leq p < \infty$)

Notation	Description
$\mathbf{A}$	Matrix
$\mathbf{A}_\Lambda$	Matrix $\mathbf{A}$ Only with the Columns Indexed by Λ
$[a_{ij}]$	Matrix $\mathbf{A}$
a_{ij}	Element of $\mathbf{A}$ at the i-th Row, j-th Column
$\mathbf{I}$	Identity Matrix
$(\cdot)^T$	Transposition
$(\cdot)^*$	Complex Conjugate
$(\cdot)^H$	Complex-Conjugated Transposition (Hermitian)
$\mathbf{A}^{-1}$	Inverse Matrix, if Existent
$\mathbf{A}^+$	Moore–Penrose Pseudoinverse of $\mathbf{A}$
$x(t)$	Time-Continuous Signal
$x[k]$	Time-Discrete Signal
$\hat{\cdot}$	Recovery or Detection Result
$\tilde{\cdot}$	Pre-Processed Entity
$\bar{\cdot}$	Averaged Value
$\circ$	General Operator (Placeholder)
$d(\cdot,\cdot)$	Distance Metric
$\langle\cdot,\cdot\rangle$	Scalar Product
$\ll, \gg$	Left and Right Bit Shift
$\mathcal{E}(\cdot)$	Expected Value
$\gcd(\cdot,\cdot)$	Greatest Common Divisor
$\mathcal{H}_k(\cdot)$	Hard Threshold Operator (k-sparse)
$\eta_\theta(\cdot)$	Parametrised Soft Threshold Operator
$I(\cdot)$	Set Indicator Function
$\mathrm{Im}\{\cdot\}$	Imaginary Part
$\mathrm{ld}(\cdot)$	Logarithmus Dualis $\log_2(\cdot)$ (binary logarithm)
mod	Modulo Operation
$\mathcal{N}(\mu,\sigma)$	Gaussian Normal Distribution
$\mathcal{O}(\cdot)$	Big O Notation (Algorithmic Complexity)

Notation	Description
$\Pr(\cdot)$	Probability Operator
$\mathrm{Q}_{\mathcal{A}}\{\cdot\}$	Quantisation with Respect to the Set $\mathcal{A}$
$\mathrm{Q}(m,n)$	Fixed-Point Q Format
$\mathrm{Q}n$	All-Fractional Fixed-Point Q Format
$\mathrm{Re}\{\cdot\}$	Real Part
$\lceil\cdot\rceil$	Round to the Next Integer Towards $+\infty$ (Ceiling)
$\lfloor\cdot\rfloor$	Round to the Next Integer Towards $-\infty$ (Flooring)
$\mathrm{RVD}(\cdot)$	Block-Wise or Element-Wise Real Value Decomposition
$\mathcal{U}(a,b)$	Uniform Distribution in Interval $[a,b]$

Supervised Student Work

The following student projects and theses have been supervised and contributed to the creation of this dissertation.

[1] Jakob Döring. 'Programmierung eines Sphere Decoders in C mit Schnittstelle zu Matlab'. German. Bachelor's Project. Apr. 2013.

[2] Dominic Oehlert. 'Detektion eines eigenen Paketformats mit Hilfe von Software Defined Radios'. German. Bachelor's Project. Apr. 2013.

[3] Sven Päsler. 'Compressed Sensing Orthogonal Matching Pursuit'. German. Bachelor's Project. Apr. 2013.

[4] Sebastian Schmale. 'Compressed Sensing für neurologische Signale'. German. Diploma Thesis. May 2013.

[5] Xiuhua Song. 'Simulation nachrichtentechnischer Algorithmen mit Festkomma-Arithmetik'. German. Diploma Project. Jan. 2013.

[6] Modeste Teugue. 'Orthogonal Matching Pursuit auf einem digitalen Signalprozessor'. German. Diploma Project. Dec. 2013.

[7] Jakob Döring. 'Entwurf einer Hardwarearchitektur zur K-Best-Detektion mit konfigurierbarer Komplexität'. German. Bachelor's Thesis. Aug. 2014.

[8] Sebastian Eller and Andreas Beering. 'High-Level Synthesis digitaler Hardwarestrukturen zur linearen Algebra'. German. Bachelor's Project. Nov. 2014.

[9] Guy Michel Prince Penka. 'Orthogonal Matching Pursuit mit effizienter Hardwarebeschleunigung'. German. Diploma Thesis. Apr. 2014.

[10] André Stuke. 'Entwicklung einer automatisierten Harware-in-the-Loop Simulationsstrecke'. German. Diploma Thesis. Apr. 2014.

[11] Sadreddin Kücük. 'Kanalschätzung für nichtlineare Compressive Sensing Mehrnutzerdetektion'. German. Diploma Thesis. Nov. 2015.

[12] Junpeng Liu. 'Compressive Sensing Based on the Approximate Message Passing'. Master's Project. Mar. 2015.

[13] Sven Päsler. 'Optimierte VLSI-Architektur zur Successive Interference Cancellation dünn besetzter Signale'. German. Bachelor's Thesis. Apr. 2015.

[14] Modeste Teugue. 'Simulink-Modell eines Empfängers für dünnbesetzte Mehrnutzersignale'. German. Diploma Senior Project. Mar. 2015.

[15] Yao Zhu. 'Experimentelle Generierung dünn besetzter Mehrnutzersignale'. German. Bachelor's Project. Jan. 2015.

[16] Yao Zhu. 'Messstrecke für dünn besetzte Mehrnutzersignale mit GNU Radio'. German. Bachelor's Thesis. Oct. 2015.

[17] Mujahid Abbas. 'Interfacing a Custom Digital Hardware Design on Xilinx Zynq'. Master's Thesis. Sept. 2016.

[18] Karthik Vinod. 'Rapid Architecture Design and Evaluation with Vivado HLS and Matlab'. Master's Thesis. Apr. 2016.

[19] Ayham Zedan. 'Multi-User Frame Synchronization in Wireless Networks with Sporadic User Activity'. Master's Thesis. Nov. 2016.

[20] Mattis Jaksch. 'Messreihenaufnahme dünn besetzter Mehrnutzersignale in einem drahtlosen Sensornetzwerk mit Software-Defined Radios'. German. Bachelor's Project. Feb. 2017.

[21] Liam Artjom Schwez. 'Parametrierbares Digital-Design eines Sphere Decoders mit High-Level Synthesis'. German. Master's Thesis. Jan. 2017.

[22] Taimur Wajad. 'Implementation of a Low-Latency Viterbi Decoder for Industrial Radios'. Master's Thesis. Oct. 2017.

[23] Lenard Wiedekamp. 'Polynomielle Funktionsapproximation für High-Level Synthesis'. German. Bachelor's Project. June 2017.

[24] Babar Khan. 'Data Type Agnosticism with Various High-Level Synthesis Tools'. Master's Project. Mar. 2018.

[25] Jianming Li. 'A Design Flow with Synopsys System Studio and High-Level Synthesis for the Implementation of Signal Processing Chains'. Master's Thesis. Mar. 2018.

[26] Sitaratnam Ponnaganti. 'Tracking of Signals with the RT820T Software-Defined Radio'. Master's Project. Mar. 2018.

[27] Shuvo Sarkar. 'Live Video Inpainting System-on-Chip'. (work in progress). Master's Thesis. 2018.

[28] Bhavanithya Thiraviaraja. 'A Simulink Implementation of the HiFlecs Radio with a Software-Defined Radio Interface'. Master's Project. Mar. 2018.

[29] Tai Yang. ‘Custom Overlay Design with PYNQ’. Master’s Project. Mar. 2018.

Publication List

The peer-reviewed scientific publications listed below have been co-authored or been composed as the lead author.

[KWP12] Benjamin Knoop, Till Wiegand and Steffen Paul. 'Low-complexity and approximative sphere decoding of sparse signals'. In: *46th Asilomar Conference on Signals, Systems and Computers*. IEEE, Nov. 2012, pp. 1241–1244.

[Kno+13] Benjamin Knoop, Fabian Monsees, Carsten Bockelmann, Dirk Wübben, Steffen Paul and Armin Dekorsy. 'Sparsity-Aware Successive Interference Cancellation with Practical Constraints'. In: *17th International ITG Workshop on Smart Antennas (WSA)*. Mar. 2013, pp. 1–8.

[Sch+13] Sebastian Schmale, Benjamin Knoop, Janpeter Höffmann, Dagmar Peters-Drolshagen and Steffen Paul. 'Joint Compression of Neural Action Potentials and Local Field Potentials'. In: *47th Asilomar Conference on Signals, Systems and Computers*. IEEE, Sept. 2013, pp. 1823–1827.

[Kno+14] Benjamin Knoop, Fabian Monsees, Carsten Bockelmann, Steffen Paul and Armin Dekorsy. 'Compressed Sensing K-Best Detection for Sparse Multi-User Communications'. In: *22nd European Signal Processing Conference (EUSIPCO)*. Sept. 2014, pp. 1726–1730.

[Sch+14] Sebastian Schmale, Janpeter Hoeffmann, Benjamin Knoop, Gernot Kreiselmeyer, Hajo Hamer, Dagmar Peters-Drolshagen and Steffen Paul. 'Exploiting Correlation in Neural Signals for Data Compression'. In: *22nd European Signal Processing Conference (EUSIPCO)*. Sept. 2014, pp. 2080–2084.

[Lan+15] Heiner Lange, Sebastian Schmale, Benjamin Knoop, Dagmar Peters-Drolshagen and Steffen Paul. 'ADC Topology Based on Compressed Sensing for Low-Power Brain Monitoring'. In: *Eurosensors 2015*. Vol. 120. Elsevier BV, 2015, pp. 315–319.

[Sch+15] Sebastian Schmale, Benjamin Knoop, Dagmar Peters-Drolshagen and Steffen Paul. 'Structure Reconstruction of Correlated Neural Signals Based on Inpainting for Brain Monitoring'. In: *IEEE Biomedical Circuits and Systems Conference (BioCAS)*. IEEE, Oct. 2015, pp. 1–4.

[Kno+16a] Benjamin Knoop, Jochen Rust, Sebastian Schmale, Dagmar Peters-Drolshagen and Steffen Paul. 'Rapid Digital Architecture Design of Orthogonal Matching Pursuit'. In: *24th European Signal Processing Conference (EUSIPCO)*. IEEE, Aug. 2016, pp. 1857–1861.

[Kno+16b] Benjamin Knoop, Sebastian Schmale, Dagmar Peters-Drolshagen and Steffen Paul. 'Activity and Channel Estimation in Multi-User Wireless Sensor Networks'. In: *20th International ITG Workshop on Smart Antennas (WSA)*. Mar. 2016, pp. 1–5.

[Kno+16c] Benjamin Knoop, Karthik Vinod, Sebastian Schmale, Dagmar Peters-Drolshagen and Steffen Paul. 'Fast Digital Design Space Exploration with High-Level Synthesis: A Case Study with Approximate Conjugate Gradient Pursuit'. In: *50th Asilomar Conference on Signals, Systems and Computers*. IEEE, Nov. 2016, pp. 412–416.

[Sch+16a] Sebastian Schmale, Hendra Kesuma, Heiner Lange, Jochen Rust, Benjamin Knoop, Dagmar Peters-Drolshagen and Steffen Paul. 'Hardware-Accelerated Reconstruction of Compressed Neural Signals Based on Inpainting'. In: *23rd International Conference Mixed Design of Integrated Circuits and Systems (MIXDES)*. IEEE, June 2016, pp. 399–404.

[Sch+16b] Sebastian Schmale, Heiner Lange, Benjamin Knoop, Dagmar Peters-Drolshagen and Steffen Paul. 'Compression and Reconstruction Methodology for Neural Signals Based on Patch Ordering Inpainting for Brain Monitoring'. In: *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, Mar. 2016, pp. 859–863.

[Sch+16c] Sebastian Schmale, Jochen Rust, Nils Hülsmeier, Heiner Lange, Benjamin Knoop and Steffen Paul. 'High throughput architecture for inpainting-based recovery of correlated neural signals'. In: *24th European Signal Processing Conference (EUSIPCO)*. IEEE, Aug. 2016, pp. 1728–1732.

[Sch+16d] Sebastian Schmale, Pascal Seidel, Heiner Lange, Benjamin Knoop, Dagmar Peters-Drolshagen and Steffen Paul. 'Efficient and Fast SOP-Based Inpainting for Neurological Signals in Resource Lim-

ited Systems'. In: *IEEE International Conference on Electronics, Circuits and Systems (ICECS)*. IEEE, Dec. 2016.

[Tsc+16] Konstantin Tscherkaschin, Benjamin Knoop, Jochen Rust and Steffen Paul. 'Design of a Multi-Core Hardware Architecture for Consensus-Based MIMO Detection Algorithms'. In: *50th Asilomar Conference on Signals, Systems and Computers*. IEEE, Nov. 2016, pp. 1–5.

[Kno+17] Benjamin Knoop, Liam Schwez, Dagmar Peters-Drolshagen and Steffen Paul. 'Parametrisable digital design of a sphere decoder with high-level synthesis'. In: *25th European Signal Processing Conference (EUSIPCO)*. IEEE, Aug. 2017, pp. 1389–1393.

[Rus+17] Jochen Rust, Pascal Seidel, Benjamin Knoop and Steffen Paul. 'Hardware-Efficient QR-Decomposition Using Bivariate Numeric Function Approximation'. In: *IEEE Transactions on Circuits and Systems I: Regular Papers* 64.12 (Dec. 2017), pp. 3150–3159.

[Zed+17] Ayham Zedan, Benjamin Knoop, Dagmar Peters-Drolshagen and Steffen Paul. 'Multi-User Frame Synchronization in Wireless Networks with Sporadic User Activity'. In: *21st International ITG Workshop on Smart Antennas (WSA)*. Mar. 2017, pp. 1–5.

[KRP18] Benjamin Knoop, Jochen Rust and Steffen Paul. 'Complex-Valued Digital Design of Orthogonal Matching Pursuit by Data Type-Agnostic High-Level Synthesis'. Manuscript in preparation. 2018.

Bibliography

[3GPP18] 3rd Generation Partnership Project (3GPP). *Submission of initial 5G description for IMT-2020*. Jan. 2018. URL: http://www.3gpp.org/news-events/3gpp-news/1937-5g_description (visited on 11/04/2018).

[AA09] Luay Azzam and Ender Ayanoglu. 'Reduced complexity sphere decoding via a reordered lattice representation'. In: *IEEE Transactions on Communications* 57.9 (Sept. 2009), pp. 2564–2569.

[AH13] Armin Alaghi and John P. Hayes. 'Survey of Stochastic Computing'. In: *ACM Transactions on Embedded Computing Systems* 12.2s (May 2013), pp. 1–19.

[Ais+15] Abdeldjalil Aissa-El-Bey, Dominique Pastor, Si Mohamed Aziz Sbai and Yasser Fadlallah. 'Sparsity-Based Recovery of Finite Alphabet Solutions to Underdetermined Linear Systems'. In: *IEEE Transactions on Information Theory* 61.4 (Apr. 2015), pp. 2008–2018.

[AlF+15] Ala Al-Fuqaha, Mohsen Guizani, Mehdi Mohammadi, Mohammed Aledhari and Moussa Ayyash. 'Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications'. In: *IEEE Communications Surveys & Tutorials* 17.4 (2015), pp. 2347–2376.

[Alt14] Altera Corporation. *The industry's first floating-point FPGA*. Tech. rep. 2014.

[AMR09] M. Salman Asif, William Mantzel and Justin Romberg. 'Channel protection: Random coding meets sparse channels'. In: *IEEE Information Theory Workshop*. IEEE, Oct. 2009, pp. 34–352.

[And+14] Jeffrey G. Andrews, Stefano Buzzi, Wan Choi, Stephen V. Hanly, Angel Lozano, Anthony C. K. Soong and Jianzhong Charlie Zhang. 'What Will 5G Be?' In: *IEEE Journal on Selected Areas in Communications* 32.6 (June 2014), pp. 1065–1082.

[And98] Ray Andraka. 'A survey of CORDIC algorithms for FPGA based computers'. In: *Proceedings of the 1998 ACM/SIGDA Sixth International Symposium on Field Programmable Gate Arrays*. Monterey, California, USA: ACM Press, 1998, pp. 191–200.

[Ash08] Peter J. Ashenden. *The Designer's Guide to VHDL*. 3rd ed. Morgan Kaufmann, 2008.

[Bai+12] Lin Bai, Patrick Maechler, Michael Muehlberghuber and Hubert Kaeslin. 'High-speed compressed sensing reconstruction on FPGA using OMP and AMP'. In: *19th IEEE International Conference on Electronics, Circuits and Systems (ICECS)*. IEEE, Dec. 2012, pp. 53–56.

[Bar+08] Richard Baraniuk, Mark Davenport, Ronald DeVore and Michael Wakin. 'A Simple Proof of the Restricted Isometry Property for Random Matrices'. In: *Constructive Approximation* 28.3 (Jan. 2008), pp. 253–263.

[Bar07] Richard G. Baraniuk. 'Compressive Sensing [Lecture Notes]'. In: *IEEE Signal Processing Magazine* 24.4 (July 2007), pp. 118–121.

[BB17] Stephen M. Blust and Sergio Buonomo. 'Forging paths to IMT-2020 (5G)'. In: *ITU News Magazine* (2 2017). Ed. by Matthew Clark, pp. 14–18.

[BD08] Thomas Blumensath and Mike E. Davies. 'Gradient Pursuits'. In: *IEEE Transactions on Signal Processing* 56.6 (June 2008), pp. 2370–2382.

[BD09] Thomas Blumensath and Mike E. Davies. 'Iterative hard thresholding for compressed sensing'. In: *Applied and Computational Harmonic Analysis* 27.3 (Nov. 2009), pp. 265–274.

[BDR12] Thomas Blumensath, Michael E. Davies and Gabriel Rilling. 'Greedy algorithms for compressed sensing'. In: *Compressed Sensing*. Ed. by Yonina C. Eldar and Gitta Kutyniok. Cambridge University Press, 2012, pp. 348–393.

[Boc+14] Federico Boccardi, Robert Heath, Angel Lozano, Thomas Marzetta and Petar Popovski. 'Five disruptive technology directions for 5G'. In: *IEEE Communications Magazine* 52.2 (Feb. 2014), pp. 74–80.

[BRA12] Pierre Blache, Hassan Rabah and Abbes Amira. 'High level prototyping and FPGA implementation of the orthogonal matching pursuit algorithm'. In: *11th International Conference on Information Science, Signal Processing and their Applications (ISSPA)*. IEEE, July 2012, pp. 1336–1340.

[Bur06] Andreas Burg. 'VLSI circuits for MIMO communication systems'. PhD thesis. Eidgenössische Technische Hochschule ETH Zürich, 2006.

[BV04] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

[BV14] Somsubhra Barik and Haris Vikalo. 'Sparsity-Aware Sphere Decoding: Algorithms and Complexity Analysis'. In: *IEEE Transactions on Signal Processing* 62.9 (May 2014), pp. 2212–2225.

[Cad15] Cadence Design Systems Inc. *Stratus High-Level Synthesis*. 2015. URL: `https://www.cadence.com/content/dam/cadence-www/global/en_US/documents/tools/digital-design-signoff/stratus-ds.pdf` (visited on 15/12/2017).

[Can+11] Andrew Canis, Jongsok Choi, Mark Aldham, Victor Zhang, Ahmed Kammoona, Jason H. Anderson, Stephen Brown and Tomasz Czajkowski. 'LegUp: High-level Synthesis for FPGA-based Processor/Accelerator Systems'. In: *Proceedings of the 19th ACM/SIGDA International Symposium on Field Programmable Gate Arrays*. FPGA '11. New York, NY, USA: ACM Press, 2011, pp. 33–36.

[Car18] Igor Carron. *Compressive Sensing: The Big Picture*. Online blog. 2018. URL: `https://sites.google.com/site/igorcarron2/cs` (visited on 12/02/2018).

[CDD08] Albert Cohen, Wolfgang Dahmen and Ronald DeVore. 'Compressed sensing and best k-term approximation'. In: *Journal of the American Mathematical Society* 22.1 (July 2008), pp. 211–231.

[CDS01] Scott Shaobing Chen, David L. Donoho and Michael A. Saunders. 'Atomic Decomposition by Basis Pursuit'. In: *SIAM Review* 43.1 (Jan. 2001), pp. 129–159.

[Chu72] D. Chu. 'Polyphase codes with good periodic correlation properties (Corresp.)' In: *IEEE Transactions on Information Theory* 18.4 (July 1972), pp. 531–532.

[CLC09] Thomas H. Cormen, Charles Eric Leiserson and Ronald Linn Rivest and Clifford Stein. *Introduction to Algorithms*. 3rd ed. MIT Press, 2009.

[Con+06] Jason Cong, Yiping Fan, Guoling Han, Wei Jiang and Zhiru Zhang. 'Platform-Based Behavior-Level and System-Level Synthesis'. In: *IEEE International SOC Conference*. IEEE, Sept. 2006, pp. 199–202.

[Con+11] Jason Cong, Bin Liu, Stephen Neuendorffer, Juanjo Noguera, Kees Vissers and Zhiru Zhang. 'High-Level Synthesis for FPGAs: From Prototyping to Deployment'. In: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 30.4 (Apr. 2011), pp. 473–491.

[Cou+08] Philippe Coussy, Cyrille Chavet, Pierre Bomel, Dominique Heller, Eric Senn and Eric Martin. 'GAUT: A High-Level Synthesis Tool for DSP Applications'. In: *High-Level Synthesis: From Algorithm to Digital Circuit*. Ed. by Philippe Coussy and Adam Morawiec. Dordrecht: Springer Netherlands, 2008. Chap. 9, pp. 147–169.

[Cou+09] Philippe Coussy, Daniel D. Gajski, Michael Meredith and Andres Takach. 'An Introduction to High-Level Synthesis'. In: *IEEE Design & Test of Computers* 26.4 (July 2009), pp. 8–17.

[CR05] Emmanuel Candès and Justin Romberg. *l1-MAGIC: Recovery of Sparse Signals via Convex Programming*. online. Oct. 2005.

[Cro+14] Louise H. Crockett, Ross A. Elliot, Martin A. Enderwitz and Robert W. Stewart. 'The Zynq Book: Embedded Processing with the ARM Cortex-A9 on the Xilinx Zynq-7000 All Programmable SoC'. In: *Strathclyde Academic Media* (2014).

[CRT06a] Emmanuel J. Candès, Justin K. Romberg and Terence Tao. 'Robust uncertainty principles: exact signal reconstruction from highly incomplete frequency information'. In: *IEEE Transactions on Information Theory* 52.2 (Feb. 2006), pp. 489–509.

[CRT06b] Emmanuel J. Candès, Justin K. Romberg and Terence Tao. 'Stable signal recovery from incomplete and inaccurate measurements'. In: *Communications on Pure and Applied Mathematics* 59.8 (2006), pp. 1207–1223.

[CT05a] Emmanuel J. Candès and Terence Tao. 'Decoding by linear programming'. In: *IEEE Transactions on Information Theory* 51.12 (Dec. 2005), pp. 4203–4215.

[CT05b] Tao Cui and C. Tellambura. 'An efficient generalized sphere decoder for rank-deficient MIMO systems'. In: *IEEE Communications Letters* 9.5 (May 2005), pp. 423–425.

[CW08] Emmanuel J. Candès and M. B. Wakin. 'An Introduction To Compressive Sampling'. In: *IEEE Signal Processing Magazine* 25.2 (Mar. 2008), pp. 21–30.

[DAB00] M. Oussama Damen, K. Abed-Meraim and J.-C. Belfiore. 'Generalised sphere decoder for asymmetrical space-time communication architecture'. In: *Electronics Letters* 36.2 (Jan. 2000), p. 166.

[DEC03] Mohamed Oussama Damen, Hesham El Gamal and Giuseppe Caire. 'On maximum-likelihood detection and the search for the closest lattice point'. In: *IEEE Transactions on Information Theory* 49.10 (Oct. 2003), pp. 2389–2402.

[Dev18] NumPy Developers. *NumPy*. 2018. URL: http://www.numpy.org/ (visited on 27/04/2018).

[DH14] Rainer Drath and Alexander Horch. 'Industrie 4.0: Hit or Hype? [Industry Forum]'. In: *IEEE Industrial Electronics Magazine* 8.2 (June 2014), pp. 56–58.

[DMM09] David L. Donoho, Arian Maleki and Andrea Montanari. 'Message-passing algorithms for compressed sensing'. In: *Proceedings of the National Academy of Sciences* 106.45 (Oct. 2009), pp. 18914–18919.

[Don06] David L. Donoho. 'Compressed sensing'. In: *IEEE Transactions on Information Theory* 52.4 (Apr. 2006), pp. 1289–1306.

[DT09] David Donoho and Jared Tanner. 'Observed universality of phase transitions in high-dimensional geometry, with implications for modern data analysis and signal processing'. In: *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences* 367.1906 (Oct. 2009), pp. 4273–4293.

[Dua+05] Marco F. Duarte, Shriram Sarvotham, Dror Baron, Micahel B. Wakin and Richard G. Baraniuk. 'Distributed Compressed Sensing of Jointly Sparse Signals'. In: *Conference Record of the Thirty-Ninth Asilomar Conference on Signals, Systems and Computers*. IEEE, Nov. 2005.

[EK12] Yonina C. Eldar and Gitta Kutyniok. *Compressed Sensing: Theory and Applications*. Ed. by Yonina C. Eldar and Gitta Kutyniok. Cambridge: Cambridge University Press, 2012.

[EKB10] Yonina C. Eldar, Patrick Kuppinger and Helmut Bolcskei. 'Block-Sparse Signals: Uncertainty Relations and Efficient Recovery'. In: *IEEE Transactions on Signal Processing* 58.6 (June 2010), pp. 3042–3054.

[EMD09] Wolfgang Ecker, Wolfgang Müller and Rainer Dömer. 'Hardware-Dependent Software'. In: *Hardware-Dependent Software*. Ed. by Wolfgang Ecker, Wolfgang Müller and Rainer Dömer. Springer, 1st Jan. 2009. Chap. 1, pp. 1–13.

[End10] Joachim H. G. Ender. 'On compressive sensing applied to radar'. In: *Signal Processing* 90.5 (May 2010), pp. 1402–1414.

[EV12] Ehsan Elhamifar and René Vidal. 'Block-Sparse Recovery via Convex Optimization'. In: *IEEE Transactions on Signal Processing* 60.8 (Aug. 2012), pp. 4094–4107.

[FB10] Michael Fingeroff and Thomas Bollaert. *High-Level Synthesis: Blue Book*. Xlibris Corporation, 2010.

[GB17] Michael C. Grant and Stephen P. Boyd. *CVX User's Guide*. CVX Research, Inc., Dec. 2017.

[GK83] Daniel D. Gajski and Robert H. Kuhn. 'Guest Editors' Introduction: New VLSI Tools'. In: *Computer* 16.12 (Dec. 1983), pp. 11–14.

[Gol91] David Goldberg. 'What every computer scientist should know about floating-point arithmetic'. In: *ACM Computing Surveys* 23.1 (Mar. 1991), pp. 5–48.

[GR94] Daniel D. Gajski and Loganath Ramachandran. 'Introduction to high-level synthesis'. In: *IEEE Design & Test of Computers* 11.4 (1994), pp. 44–54.

[Gus15] John L. Gustafson. *The End of Error: Unum Computing*. 2nd ed. Chapman and Hall/CRC, 23rd Jan. 2015. 416 pp.

[GV13] Gene H. Golub and Charles F. Van Loan. *Matrix Computations*. 4th ed. Johns Hopkins University Press, 2013.

[Has+05] Michael Haselman, Michael Beauchamp, Aaron Wood, Scott Hauck, Keith Underwood and K. Scott Hemmert. 'A Comparison of Floating Point and Logarithmic Number Systems for FPGAs'. In: *13th Annual IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM)*. IEEE, Apr. 2005.

[HI14] Ghaith Hattab and Mohammed Ibnkahla. 'Enhanced pilot-based spectrum sensing algorithm'. In: *2014 27th Biennial Symposium on Communications (QBSC)*. IEEE. IEEE, June 2014, pp. 57–60.

[Hig02] Nicholas J. Higham. *Accuracy and Stability of Numerical Algorithms*. 2nd ed. Philadelphia: Society for Industrial and Applied Mathematics (SIAM), 2002.

[HJ09] Stefan Heinen and Michael Joost. 'Hardware-Dependent Software'. In: *Hardware-Dependent Software*. Ed. by Wolfgang Ecker, Wolfgang Müller and Rainer Dömer. Springer, 1st Jan. 2009. Chap. 6, pp. 151–171.

[HM17] Nicole Hemsoth and Timothy Prickett Morgan. *FPGA Frontiers: New Applications in Reconfigurable Computing*. Next Platform Press, 2017.

[Hou+17] Junjie Hou, Yongxin Zhu, Yulan Shen, Mengjun Li, Qian Wu and Han Wu. 'Enhancing Precision and Bandwidth in Cloud Computing: Implementation of a Novel Floating-Point Format on FPGA'. In: *4th International Conference on Cyber Security and Cloud Computing (CSCloud)*. IEEE, June 2017.

[HV05] Babak Hassibi and Haris Vikalo. 'On the sphere-decoding algorithm I. Expected complexity'. In: *IEEE Transactions on Signal Processing* 53.8 (Aug. 2005), pp. 2806–2818.

[IEEE-1076] Institute of Electrical and Electronics Engineers. *IEEE Standard VHDL Language Reference Manual*. IEEE Std. 1076-2008. IEEE Computer Society.

[IEEE-1364] Institute of Electrical and Electronics Engineers. *IEEE Standard for Verilog Hardware Description Language*. IEEE Std. 1364-2005. IEEE Computer Society.

[IEEE-754] Institute of Electrical and Electronics Engineers. *IEEE Standard for Floating-Point Arithmetic*. IEEE Std. 754-2008. IEEE Computer Society.

[IEEE-802] Institute of Electrical and Electronics Engineers. *IEEE Standard for Local and metropolitan area networks–Part 15.4: Low-Rate Wireless Personal Area Networks (LR-WPANs)*. IEEE Std. 802.15.4-2011. IEEE.

[Int17] Intel Corporation. *Intel High Level Synthesis Compiler User Guide*. UG-20037. Nov. 2017.

[IO96] Cristopher Inacio and Denise Ombres. 'The DSP decision: fixed point or floating?' In: *IEEE Spectrum* 33.9 (Sept. 1996), pp. 72–74.

[ITU-X.200] International Telecommunication Union (ITU-T). *X.200 : Information Technology - Open Systems Interconnection - Basic Reference Model: The Basic Model*. also ISO/IEC 7498-1. Telecommunication Standardization Sector, July 1994.

[JV11] Laurent Jaques and Pierre Vandergheynst. 'Compressed Sensing: "When Sparsity Meets Sampling"'. In: *Optical and Digital Image Processing: Fundamentals and Applications*. Ed. by Gabriel Cristóbal, Peter Schelkens and Hugo Thienpont. Weinheim: Wiley-VCH, 2011. Chap. Compressed Sensing: "When Sparsity Meets Sampling", pp. 507–528.

[Kae08] Hubert Kaeslin. *Digital integrated circuit design: from VLSI architectures to CMOS fabrication*. Cambridge University Press, 2008.

[Kam08] Karl-Dirk Kammeyer. *Nachrichtenübertragung*. German. 4th ed. Stuttgart, Germany: B.G. Teubner, Mar. 2008. 845 pp.

[Khr17] Khronos Group. *The OpenCL Specification Version 2.2*. Rev. 2.2-3. OpenCL Working Group, May 2017.

[KK06] Karl-Dirk Kammeyer and Kristian Kroschel. *Digitale Signalverarbeitung. Filterung und Spektralanalyse mit MATLAB-Übungen*. German. 6th ed. Wiesbaden: Teubner, 2006.

[Knu02] Donald Ervin Knuth. *The art of computer programming Vol. 2: Seminumerical algorithms*. 3rd ed. Addison-Wesley, 2002.

[Küh06] Volker Kühn. *Wireless Communications over MIMO Channels: Applications to CDMA and Multiple Antenna Systems*. Wiley, 2006.

[LA04] Chris Lattner and Vikram Adve. 'LLVM: A compilation framework for lifelong program analysis & transformation'. In: *International Symposium on Code Generation and Optimization, 2004. CGO 2004.* CGO '04. Palo Alto, California: IEEE, 2004, p. 75.

[Lat12] Chris Lattner. 'The architecture of open source applications: LLVM'. In: *The architecture of open source applications*. Ed. by Amy Brown and Greg Wilson. lulu.com, 2012. Chap. 11, pp. 151–166.

[LCL11] Shao-Yu Lien, Kwang-Cheng Chen and Yonghua Lin. 'Toward ubiquitous massive accesses in 3GPP machine-to-machine communications'. In: *IEEE Communications Magazine* 49.4 (Apr. 2011), pp. 66–74.

[LL16] Jianping Li and Wan Li. 'A novel sphere detection algorithm for MQAM MIMO systems'. In: *2016 IEEE International Conference on Electronic Information and Communication Technology (ICEICT)*. IEEE, Aug. 2016.

[LML17] Lin Li, Weixiao Meng and Cheng Li. 'Compressed sensing based semidefinite relaxation detection algorithm for overloaded uplink multiuser massive MIMO system'. In: *IEEE International Conference on Communications (ICC)*. IEEE, May 2017.

[Lyo04] Richard G. Lyons. *Understanding Digital Signal Processing*. 2nd ed. Upper Saddle River, NJ, USA: Prentice Hall PTR, 2004.

[Mae+12] Patrick Maechler, Christoph Studer, David E. Bellasi, Arian Maleki, Andreas Burg, Norbert Felber, Hubert Kaeslin and Richard G. Baraniuk. 'VLSI Design of Approximate Message Passing for Signal Restoration and Compressive Sensing'. In: *IEEE Journal on Emerging and Selected Topics in Circuits and Systems* 2.3 (Sept. 2012), pp. 579–590.

[Mee+12] Wim Meeus, Kristof Van Beeck, Toon Goedemé, Jan Meel and Dirk Stroobandt. 'An overview of today's high-level synthesis tools'. In: *Design Automation for Embedded Systems* 16.3 (Aug. 2012), pp. 31–51.

[Mit62] John N. Mitchell. 'Computer Multiplication and Division Using Binary Logarithms'. In: *IEEE Transactions on Electronic Computers* EC-11.4 (Aug. 1962), pp. 512–517.

[Mol17] Cleve Moler. *Quadruple Precision, 128-bit Floating Point Arithmetic*. May 2017. URL: https://blogs.mathworks.com/cleve/2017/05/22/quadruple-precision-128-bit-floating-point-arithmetic/ (visited on 20/12/2017).

[Mon+12] Fabian Monsees, Carsten Bockelmann, Dirk Wübben and Armin Dekorsy. 'Sparsity Aware Multiuser detection for Machine to Machine communication'. In: *IEEE Globecom Workshops*. Anaheim, USA: IEEE, Dec. 2012.

[Mon17] Fabian Monsees. 'Signal Processing for Compressed Sensing Multiuser Detection'. PhD thesis. Bremen: Institut für Telekommunikation und Hochfrequenztechnik, Universität Bremen, 2017.

[Moo65] Gordon Earle Moore. 'Cramming More Components Onto Integrated Circuits'. In: *Electronics* 38.8 (Apr. 1965). Reprint, pp. 111–114.

[Moo98] Gordon Earle Moore. 'Cramming More Components Onto Integrated Circuits'. In: *Proceedings of the IEEE* 86.1 (Jan. 1998), pp. 82–85.

[MS09] Grant Martin and Gary Smith. 'High-Level Synthesis: Past, Present, and Future'. In: *IEEE Design & Test of Computers* 26.4 (July 2009), pp. 18–25.

[MS13] Benjamin Menhorn and Frank Slomka. 'Confirming the Design Gap'. In: *Proceedings of the Third International Conference on Computional Science, Engineering and Information Technology*. Vol. 225. Advances in Intelligent Systems and Computing. Heidelberg: Springer International Publishing, July 2013, pp. 281–292.

[Mul06] Jean-Michel Muller. *Elementary Functions Algorithms and Implementation*. 2nd ed. Boston: Birkhäuser, 2006.

[MZ93] Stéphane G. Mallat and Zhifeng Zhang. 'Matching pursuits with time-frequency dictionaries'. In: *IEEE Transactions on Signal Processing* 41.12 (1993), pp. 3397–3415.

[Nan+16] Razvan Nane, Vlad-Mihai Sima, Christian Pilato, Jongsok Choi, Blair Fort, Andrew Canis, Yu Ting Chen, Hsuan Hsiao, Stephen Brown, Fabrizio Ferrandi, Jason Anderson and Koen Bertels. 'A Survey and Evaluation of FPGA High-Level Synthesis Tools'. In: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 35.10 (Oct. 2016), pp. 1591–1604.

[Nik+14] Hosein Nikopour, Eric Yi, Alireza Bayesteh, Kelvin Au, Mark Hawryluck, Hadi Baligh and Jianglei Ma. 'SCMA for downlink multiple access of 5G wireless networks'. In: *IEEE Global Communications Conference*. IEEE, Dec. 2014, pp. 3940–3945.

[NTV08] Deanna Needell, Joel Tropp and Roman Vershynin. 'Greedy signal recovery review'. In: *42nd Asilomar Conference on Signals, Systems and Computers*. IEEE, Oct. 2008, pp. 1048–1050.

[Par+17] Hosung Park, Kee-Hoon Kim, Jong-Seon No and Dae-Woon Lim. 'Reconstruction of Complex Sparse Signals in Compressed Sensing with Real Sensing Matrices'. In: *Wireless Personal Communications* 97.4 (Aug. 2017), pp. 5719–5731.

[Par10] Behrooz Parhami. *Computer Arithmetic: Algorithms and Hardware Designs*. 2nd ed. New York: Oxford University Press, 2010. 641 pp.

[PF13] Christian Pilato and Fabrizio Ferrandi. 'Bambu: A modular framework for the high level synthesis of memory-intensive applications'. In: *23rd International Conference on Field programmable Logic and Applications*. IEEE, Sept. 2013, pp. 1–4.

[Pop92] Branislav M. Popovic. 'Generalized chirp-like polyphase sequences with optimum correlation properties'. In: *IEEE Transactions on Information Theory* 38.4 (July 1992), pp. 1406–1409.

[PP12] Kaare Brandt Petersen and Michael Syskind Pedersen. *The Matrix Cookbook*. Nov. 2012.

[Pra17] Hemanth Prabhu. 'Hardware Implementation of Baseband Processing for Massive MIMO'. PhD thesis. Department of Electrical and Information Technology, Lund University, Mar. 2017.

[PRK93] Yagyensh Chandra Pati, Ramin Rezaiifar and Perinkulam Sambamurthy Krishnaprasad. 'Orthogonal matching pursuit: Recursive function approximation with applications to wavelet decomposition'. In: *27th Asilomar Conference on Signals, Systems and Computers*. IEEE. 1993, pp. 40–44.

[Pro18] Project Jupyter. *Project Jupyter*. 2018. URL: `https://jupyter.org/` (visited on 27/04/2018).

[RAA12] Hassan Rabah, Abbes Amira and Afandi Ahmad. 'Design and implementaiton of a fall detection system using compressive sensing and shimmer technology'. In: *24th International Conference on Microelectronics (ICM)*. IEEE, Dec. 2012, pp. 1–4.

[Rab+15] Hassan Rabah, Abbes Amira, Basant Kumar Mohanty, Somaya Almaadeed and Pramod Kumar Meher. 'FPGA Implementation of Orthogonal Matching Pursuit for Compressive Sensing Reconstruction'. In: *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 23.10 (Oct. 2015), pp. 2209–2220.

[Ren14] Haoxing Ren. 'A brief introduction on contemporary High-Level Synthesis'. In: *IEEE International Conference on IC Design & Technology*. IEEE, May 2014.

[RJ16] S. Ravi and M. Joseph. 'Open source HLS tools: A stepping stone for modern electronic CAD'. In: *2016 IEEE International Conference on Computational Intelligence and Computing Research (ICCIC)*. IEEE, Dec. 2016, pp. 1–8.

[RLP13] Jochen Rust, Frank Ludwig and Steffen Paul. 'Low-Complexity QR Decomposition Architecture using the Logarithmic Number System'. In: *Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE Conference Publications, Mar. 2013, pp. 97–102.

[Rus14] Jochen Rust. 'Effiziente Entwurfsverfahren zur hardwarebasierten Signalverarbeitung elementarer Funktionen für die drahtlose Kommunikation'. PhD thesis. Bremen: Institut für Theoretische Elektrotechnik und Mikroelektronik, Universität Bremen, Mar. 2014, p. 151.

[Ryd08] Peter Rydesäter. *TCP/UDP/IP Toolbox 2.0.6. TCP/IP connections or UDP packets in MATLAB*. 2008. URL: `https://de.mathworks.com/matlabcentral/fileexchange/345-tcp-udp-ip-toolbox-2-0-6` (visited on 27/04/2018).

[SA75] Earl E. Swartzlander and Aristides G. Alexopoulos. 'The Sign/-Logarithm Number System'. In: *IEEE Transactions on Computers* C-24.12 (Dec. 1975), pp. 1238–1242.

[SBD13] Henning Schepker, Carsten Bockelmann and Armin Dekorsy. 'Exploiting Sparsity in Channel and Data Estimation for Sporadic Multi-User Communication'. In: *10th International Symposium on Wireless Communication Systems (ISWCS 13)*. Ilmenau, Germany, Aug. 2013.

[SE94] Claus-Peter Schnorr and M. Euchner. 'Lattice basis reduction: Improved practical algorithms and solving subset sum problems'. In: *Mathematical Programming* 66 (1-3 Aug. 1994), pp. 181–199.

[SG08] Satnam Singh and David J. Greaves. 'Kiwi: Synthesis of FPGA Circuits from Parallel Programs'. In: *16th International Symposium on Field-Programmable Custom Computing Machines*. IEEE, Apr. 2008, pp. 3–12.

[Sin17] Udayan Sinha. *Enabling Impactful DSP Designs on FPGAs with Hardened Floating-Point Implementation*. Tech. rep. WP-01227-1.1. Intel Corporation, 2017.

[SM12] Jerome L. V. M. Stanislaus and Tinoosh Mohsenin. 'High performance compressive sensing reconstruction hardware with QRD process'. In: *IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE, May 2012, pp. 29–32.

[SM13] Jerome L. V. M. Stanislaus and Tinoosh Mohsenin. 'Low-complexity FPGA implementation of compressive sensing reconstruction'. In: *International Conference on Computing, Networking and Communications (ICNC)*. IEEE, Jan. 2013, pp. 671–675.

[SS10] Avi Septimus and Raphael Steinberg. 'Compressive sampling hardware reconstruction'. In: *Proceedings of 2010 IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE, May 2010, pp. 3316–3319.

[Syn] Synopsys Inc. *Synopsys Introduces Synphony High Level Synthesis*. URL: `https://news.synopsys.com/index.php?item=123096` (visited on 24/01/2018).

[Tex00] Texas Instruments, Inc. *TMS320C55x DSP Library Programmer's Reference*. SPRU657C. SPRU422J, revised 2013. Texas Instruments Inc. Dallas, TX, May 2000.

[TG07] Joel A. Tropp and Anna C. Gilbert. 'Signal Recovery from Random Measurements Via Orthogonal Matching Pursuit'. In: *IEEE Transactions on Information Theory* 53.12 (Dec. 2007), pp. 4655–4666.

[Tib96] Robert Tibshirani. 'Regression Shrinkage and Selection via the Lasso'. In: *Journal of the Royal Statistical Society. Series B (Methodological)* 58.1 (1996), pp. 267–288.

[TLL09] Zhi Tian, Geert Leus and Vincenzo Lottici. 'Detection of sparse signals under finite-alphabet constraints'. In: *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, Apr. 2009, pp. 2349–2352.

[TW17] Thomas N. Theis and H.-S. Philip Wong. 'The End of Moore's Law: A New Beginning for Information Technology'. In: *Computing in Science & Engineering* 19.2 (Mar. 2017), pp. 41–50.

[Ver98] Sergio Verdú. *Multiuser Detection*. Cambridge University Press, 1998.

[VMB02] Martin Vetterli, Pina Marziliano and Thierry Blu. 'Sampling signals with finite rate of innovation'. In: *IEEE Transactions on Signal Processing* 50.6 (June 2002), pp. 1417–1428.

[Vol59] Jack E. Volder. 'The CORDIC Trigonometric Computing Technique'. In: *IRE Transactions on Electronic Computers* EC-8.3 (Sept. 1959), pp. 330–334.

[Wal71] John S. Walther. 'A unified algorithm for elementary functions'. In: *Proceedings of the Spring Joint Computer Conference*. ACM. ACM Press, 1971, pp. 379–385.

[Wie12] Till Wiegand. 'Hardwareoptimierte Entwicklung nachrichtentechnischer Algorithmen zum Prototypenentwurf von Mobilfunkempfängern'. German. PhD thesis. Bremen: Institut für Theoretische Elektrotechnik und Mikroelektronik, Universität Bremen, Sept. 2012.

[Won+02] Kwan-wai Wong, Chi-ying Tsui, R.S.-K. Cheng and Wai-ho Mow. 'A VLSI architecture of a K-best lattice decoding algorithm for MIMO channels'. In: *Proceeding of the IEEE International Symposium on Circuits and Systems (ISCAS)*. Vol. 3. IEEE, 2002, pp. 273–276.

[WP12] Till Wiegand and Steffen Paul. 'Reduced complexity computation unit for a sphere decoding algorithm'. In: *18th European Wireless Conference*. Apr. 2012, pp. 1–6.

[Wüb+01] Dirk Wübben, Ronald Böhnke, Jürgen Rinas, Volker Kühn and Karl-Dirk Kammeyer. 'Efficient algorithm for decoding layered space-time codes'. In: *Electronics Letters* 37.22 (Nov. 2001), pp. 1348–1350.

[Wüb06] Dirk Wübben. 'Effiziente Detektionsverfahren für Multilayer-MIMO-Systeme'. PhD thesis. Bremen: Institut für Telekommunikation und Hochfrequenztechnik, Universität Bremen, 2006.

[Xil-UG479] Xilinx Inc. *7 Series FPGAs DSP48E1 Slice User Guide*. Version UG479. UG479 (v1.9). Xilinx Inc. Sept. 2016.

[Xil-UG871] Xilinx Inc. *Vivado Design Suite Tutorial: High-Level Synthesis*. Version UG871. UG871 (v2017.4). Xilinx Inc. Dec. 2017.

[Xil-UG902] Xilinx Inc. *Vivado Design Suite User Guide: High-Level Synthesis*. Version UG902. UG902 (v2017.2). Xilinx Inc. June 2017.

[Xil18] Xilinx Inc. *Pynq: Python Productivity for Zynq*. 2018. URL: `http://www.pynq.io/` (visited on 27/04/2018).

[YKM12] Siwei Yu, A. Shaharyar Khwaja and Jianwei Ma. 'Compressed sensing of complex-valued data'. In: *Signal Processing* 92.2 (Feb. 2012), pp. 357–362.

[ZF05] Hans-Jürgen Zepernick and Adolf Finger. *Pseudo Random Signal Processing: Theory and Application*. Chichester: Wiley, 2005.

[ZG09] Hao Zhu and Georgios B. Giannakis. 'Sparsity-embracing multiuser detection for CDMA systems with low activity factors'. In: *IEEE International Symposium on Information Theory (ISIT)*. IEEE, June 2009, pp. 164–168.

[ZG11] Hao Zhu and Georgios B. Giannakis. 'Exploiting Sparse User Activity in Multiuser Detection'. In: *IEEE Transactions on Communications* 59.2 (Feb. 2011), pp. 454–465.

[Zho+16] Guanwen Zhong, Alok Prakash, Yun Liang, Tulika Mitra and Smail Niar. 'Lin-analyzer: A High-level Performance Analysis Tool for FPGA-based Accelerators'. In: *Proceedings of the 53rd Annual Design Automation Conference (DAC)*. DAC '16. Austin, Texas: ACM Press, 2016.

[Zia+10] Ali Ziaei, Ali Pezeshki, Saeid Bahmanpour and Mahmood R. Azimi-Sadjadi. 'Compressed sensing of different size block-sparse signals: Efficient recovery'. In: *Conference Record of the Forty Fourth Asilomar Conference on Signals, Systems and Computers*. IEEE, Nov. 2010.