Parametrisable Digital Design of a Sphere Decoder with High-Level Synthesis

Benjamin Knoop[1], Liam Schwez, Dagmar Peters-Drolshagen[1], Steffen Paul[1]
[1] Institute of Electrodynamics and Microelectronics (ITEM), University of Bremen, Germany
Email: {knoop, peters, steffen.paul}@me.uni-bremen.de

***Abstract*—A Sphere Decoder is a popular tree search algorithm for the solution of integer least squares minimisation problems. It has gained considerable attention for its application to maximum likelihood detection of digitally modulated signals in MIMO communication systems and can almost universally be applied to a plethora of problems with some modifications to the sphere constraint. This creates the need for a baseline digital hardware design of a configurable Sphere Decoder, which can be adjusted for various applications. This paper presents the implementation of a baseline Sphere Decoder with high-level synthesis (HLS) in connection with a data type-agnostic programming methodology, which makes it even more flexible.**

I. Introduction

A sphere decoder is a popular tree search algorithm for the solution of integer least squares (ILS) minimisation problems. Usually these problems can be written as

$$\hat{\mathbf{x}} = \arg \min_{\mathbf{x} \in \mathbb{Z}^N} \|\mathbf{y} - \mathbf{H}\mathbf{x}\|_2^2 \,, \tag{1}$$

with $\mathbf{y} \in \mathbb{R}^M$, $\mathbf{H} \in \mathbb{R}^{M \times N}$ and $\mathbf{x}$ being constrained to be a vector of integers [1], [2]. The minimising solution $\hat{\mathbf{x}}$ is the closest multi-dimensional lattice point to the vector $\mathbf{y}$ in a Euclidean ℓ_2-norm sense. However, this problem is known to be NP-hard, i.e. of non-polynomial algorithmic complexity or, in other words, infeasible to compute except for a very small problem size N.

Even so, oftentimes the relationship between $\mathbf{y}$ and $\mathbf{H}$ is not arbitrary but known, at least statistically. Most communication applications can be modelled by a linear input-output relationship

$$\mathbf{y} = \mathbf{H}\mathbf{x} + \mathbf{n} \,. \tag{2}$$

In this context, $\mathbf{H}$ denotes the channel matrix, which is generally known at an receiver with channel state information up to some estimation uncertainty. And $\mathbf{y}$ is here the noisy observation vector, perturbed by additive white Gaussian noise (AWGN) $\mathbf{n} \sim \mathcal{N}(0, \sigma_n \mathbf{I})$ with $\mathbf{I}$ being the identity matrix. With known statistical properties like these, Hassibi and Vikalo showed that solutions to the ILS problem (1) can be obtained *on average* in polynomial time by sphere decoding although the worst-case complexity might still be exponential [1]. Nonetheless, this renders sphere decoding tractable for many communication problems.

Therefore, sphere decoding has gained considerable attention for its application to maximum likelihood detection of digitally modulated signals. This fact introduces a further reduction in complexity because the solution space is usually not an infinite N-dimensional lattice but solely a subset $\mathcal{A}^N \subset \mathbb{Z}^N$ of it. $\mathcal{A}$ denotes the finite set of constellation points of a digital modulation scheme, e.g. Binary Phase-Shift Keying (BPSK) with $\mathcal{A}^{\text{BPSK}} = \{-1, 1\}$. The optimisation problem can be reformulated for complex-valued symbols without much effort.

Compared to maximum likelihood (ML) detection, i.e. an exhaustive search in general, SD can still feature the optimality of ML estimation results. It sets therefore a benchmark for other heuristic or approximative methods and it has seen a wide range of applications, e.g. in spatial-multiplexing multiple-input multiple-output (MIMO) communication systems [3], [4] or in sporadically transmitting wireless sensor networks [5], just to mention two of them. Each scenario poses different constraints and various tweaks of the cost function in (1) have been proposed in the literature. Nevertheless, the sphere decoding principle is common to all parametrisations.

Hence, a configurable and parametrisable hardware design of a SD would be favourable ("one size fits most"). We present therefore in this paper a baseline digital architecture, which is configurable with regard to (i) application-specific parameters, that is, e.g., the problem size or the modulation alphabet, (ii) algorithmic variations, e.g. different sphere constraints or candidate enumerations, and (iii) implementation aspects, like norm approximations or used fixed point datatypes, and so on.

This high degree of flexibility can only be achieved because of the employment of a high-level synthesis (HLS) tool. A rapid datatype-agnostic design methodology is applied to create the SD hardware architecture on an elevated design perspective, capturing the essential sphere decoding principle while abstracting certain implementation specifics [6]. This still leads to fair results in terms of hardware performance and resource utilisation, while gaining the enormous flexibility with regard to parametrisation as outlined above.

II. Parameters to Sphere Decoding

Sphere Decoding belongs to the class of tree search algorithms. It performs the search exclusively on a partial set of possible symbol vectors representing a subtree, which induces its efficiency. The problem must first be transformed into a search tree by triangularisation of the ILS problem (1) using matrix QR factorisation [2]. The matrix $\mathbf{H}$ with $M \geq N$ is factorised such that $\mathbf{H} = \mathbf{Q}\mathbf{R}$ with $\mathbf{R}$ being quadratic $N \times N$ and $\mathbf{Q}$ having the dimensions $M \times N$. This is also called the skinny QR decomposition.

ISBN 978-0-9928626-7-1

Then, (1) can be reformulated as

$$\hat{\mathbf{x}} = \arg\min_{\mathbf{x}\in\mathcal{S}^N} \|\tilde{\mathbf{y}} - \mathbf{R}\mathbf{x}\|_2^2 \,, \tag{3}$$

with $\tilde{\mathbf{y}} = \mathbf{Q}^T\mathbf{y}$. Note that the minimisation is now with respect to the set $\mathcal{S}$, which is usually equal to $\mathbb{Z}$, the real-valued integer numbers, but can as well denote complex integers $\mathbb{Z}[i] = \{a + jb \mid a, b \in \mathbb{Z}\}$, also called Gaussian integers. The latter (or any complex) case can easily be reduced to a real-valued problem of size $2N$ by the application of a real-valued decomposition (RVD), either block-wise (BRVD) or element-wise (ERVD) [7].

Additionally, $\mathcal{S}$ can denote any finite alphabet signal constellation as they are used for digital modulation in communications, like Quadrature Amplitude Modulation (QAM) or Phase-Shift Keying (PSK) as long as bits are independently mapped to the real and imaginary parts of the symbols. This problem occurs, e.g., in spatial-multiplexing MIMO communication systems with M receive and N transmit antennas. The SD algorithm complexity naturally depends on the size of the modulation alphabet $\mathcal{S}$ linearly.

The optimisation problem (3) is a maximum likelihood sequence estimation (MLSE) because no further assumptions are made about the statistics of the modulation symbols. Implicitly this means that the transmit symbols of the sequence (x_i) with $i = 1, ..., N$ are independent and identically uniformly distributed. However, for some application scenarios a maximum a posteriori (MAP) detection is favourable when a priori information is available and can be taken into account. In wireless sensor networks with sparse transmission patterns, e.g., inactive sensor nodes can be modelled to transmit a zero symbol with greater probability than a data symbol [5]. In such a case, the ML cost function $f_{\text{ML}}(\mathbf{x}) = \|\tilde{\mathbf{y}} - \mathbf{R}\mathbf{x}\|_2^2$ is augmented by a second MAP regularisation term $f_{\text{MAP}}(\mathbf{x})$ (see Sec. IV-B) such that

$$f(\mathbf{x}) = f_{\text{ML}}(\mathbf{x}) + f_{\text{MAP}}(\mathbf{x}) \,. \tag{4}$$

Now, SD restricts its search to such hypotheses $\mathbf{x}$ which lie within in a (regularised) hypersphere with radius ρ, i.e.

$$f(\mathbf{x}) < \rho \,, \tag{5}$$

which is the sphere constraint (SC). A hypothesis is every possible realisation of $\mathbf{x}$ in $\mathcal{S}^N$, i.e. there are $|\mathcal{S}|^N$ possible solutions or hypotheses. To determine whether or not a specific $\mathbf{x}^*$ fulfills (5) is itself NP-hard [1].

Yet, the triangularised optimisation problem in (3) allows to compute the optimum iteratively over a hierarchical structure that can be visualised as a search tree with $N+1$ layers, $|\mathcal{S}|$ branches from each non-leaf node and with $|\mathcal{S}|^N$ leaf nodes in total [4]. The squared ℓ_2-vector norm describes the distance between $\mathbf{x}$ and $\tilde{\mathbf{y}}$ and consists therefore of N partial distances (PD) $d_n(x_n, \tilde{y}_n)$,

$$f(\mathbf{x}) = \sum_{n=1}^{N} d_n(x_n, \tilde{y}_n) = \sum_{n=1}^{N} \left| \tilde{y}_n - \sum_{\ell=n}^{N} r_{n\ell} x_\ell \right|^2 \,, \tag{6}$$

and $d_n \geq 0 \; \forall n$ holds. Due to the upper-triangular structure of $\mathbf{R}$, the PDs only depend on symbol hypotheses x_ℓ of higher layers $\ell > n$. Sphere decoding starts at the N-th layer and evaluates the PD of a hypothetical x_N^* (up to $|\mathcal{S}|$ possibilities). If d_N violates the SC (5), all hypotheses $\mathbf{x}$ with $x_N = x_N^*$ will do so as well. These vectors can be excluded from the search tree, which is called tree pruning. Next, the sphere decoder proceeds and descends within the search tree to the level $N-1$ and tests the partial accumulated metric (PAD) $d_{N-1} + d_N$, also called partial Euclidean distance, against the SC. If no hypothesis remains to fulfil the SC, it jumps back up to the last valid one. In general, the PAD is computed as $D_k^N = \sum_{n=k}^{N} d_n$ with $k \leq N$ and because of the non-negativity of the PDs it is $D_N^N \leq D_{N-1}^N \leq \cdots \leq D_1^N = f(\mathbf{x}) < \rho$. That is the core idea of SD.

It is important to highlight that there are two possible search strategies. Since the algorithmic complexity of SD is measured by the number of visited, or processed, tree nodes one or the other enumeration scheme is better suited depending on the specific application setup. SD can perform a "depth first" search following the Fincke-Pohst enumeration of possible candidates, i.e. the hypothesis with the smallest value is evaluated first. Or alternatively and according to the Schnorr-Euchner enumeration, a "best first" search starts with evaluating the PDs of all child nodes and proceeds with the closest candidate with the minimal PD [3]. If $\mathcal{S}$ is a small set of discrete values, an exhaustive search with consecutive sorting is feasible. Then, the algorithm maintains a list with PD values of symbol candidates of a size up to $|\mathcal{S}|$ on each level. The list must be sorted in ascending order, whereby the implemented sorting procedure also affects the computational complexity of the SD algorithm.

Furthermore, the number of processed nodes heavily depends on the SC update strategy. There are basically two alternatives as well. First, the SC can be initialised with a certain value, e.g. with the zero-forcing solution $\hat{\mathbf{x}}_{\text{ZF}} = [\mathbf{H}^+\mathbf{y}]_{\mathcal{S}}$ which is the solution of the relaxed optimisation problem (3) without integer or finite alphabet constraint. $(\cdot)^+$ denotes the Moore-Penrose pseudoinverse and $[\cdot]_{\mathcal{S}}$ quantisation with respect to the discrete set $\mathcal{S}$, i.e. the continuous solution is consecutively quantised to the closest lattice point [1]. Second, the SC can be set to infinity ($\rho \leftarrow \infty$) for best first decoding and updated every time a better leaf node is found. This guarantees that the search sphere contains a valid integer solution and the decoding complexity is thereafter reduced by the repeated shrinking of the SC.

Burg et al. have shown that SD can also be implemented with non-Euclidean simplified norms [3]. Substituting the ℓ_2 norm in (3) for other ℓ_p vector norms such as ℓ_1 or ℓ_∞ effectively reduces the computational complexity of the algorithm by avoiding the squaring multiplications. Especially it is

$$\|\mathbf{x}\|_1 = \sum_{n=1}^{N} |x_n| \,. \tag{7}$$

ISBN 978-0-9928626-7-1 © EURASIP 2017

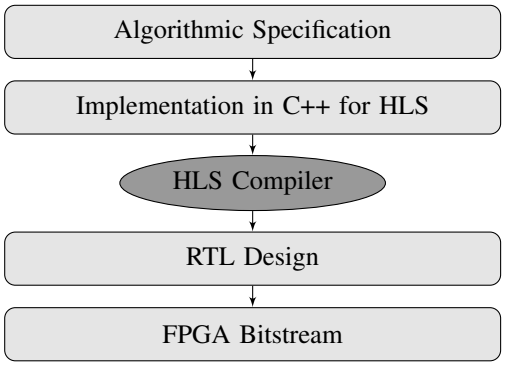

Figure 1. High-level synthesis design flow for FPGAs.

This, of course, trades a simplified hardware for a somewhat decreased detection error rate performance.

III. The Advantages of High-Level Synthesis with Datatype-Agnosticim

High-Level Synthesis (HLS) accelerates digital hardware design because it elevates the hardware engineer's design perspective away from the Register-Transfer Level (RTL). It synthesises specialised hardware from high-level specifications written in a software programming language, commonly C++ or SystemC, which also provide object-oriented and generic programming features. Recent studies show that the C++-to-RTL flow can produce results with a quality close to traditional development methodologies with hardware description languages (HDL) [8]. Fig. 1 highlights the HLS design flow targeting a Field-Programmable Gate Array (FPGA). Though not being restricted to FPGA designs, HLS is very well-suited for FPGAs because they feature fast design cycles for rapid prototyping purposes. The HLS compiler requires two inputs, the source code of the algorithm and a set of compiler directives to influence the architecture of the RTL design, e.g. to exploit parallelisms, and generates an RTL description in an HDL such as VHDL or Verilog.

Since there is no operating system to manage memory resources on the target device, the source code may not contain any dynamic memory allocations and must resort to static memory allocations at compile time only. Nonetheless, HLS designs inherit the potential flexibility the C/C++ language has to offer, foremost the class and template system. This enables the application of the datatype-agnostic programming methodology for HLS [6]. Datatype agnosticism means that the source code only makes use of custom datatype definitions, which can easily be exchanged. The code itself is agnostic to whether it will be compiled for floating point, fixed point or complex-valued datatypes, e.g. based on the C++ standard library `std::complex`. Standard C data types support variables with 8, 16, 32, etc. bit boundaries. SystemC's fixed point class `sc_fixed` moreover allows variables to be defined for any arbitrary precision bitwidth, e.g. 6, 12 or 18 bits, and by this to exercise a better control of FPGA resources. A fixed point variable is characterised by the number of digits before and after the position of the binary point. A common notation is the Q-format written as $\mathrm{Q}(m.n)$, where m is the number of integer bits including the two's complement sign bit and n the number of fractional bits.

Table I
Sphere Decoding Parameters for Two Example Scenarios

Parameter	Symbol	Scenario A	Scenario B
Problem Size	(M, N)	$(4, 4)$	$(16, 16)$, $(32, 20)$
Search Space	$\mathcal{S}$	$\mathcal{A}^{\text{16-QAM}}$	$\mathcal{A}^{\text{BPSK}} \cup \{0\}$
Cost Function	$f(\mathbf{x})$	ML, Eq. (8)	MAP, Eq. (9)
SC Update	$\rho \leftarrow \infty$	yes	yes
Search Strategy	—	best first	best first
Norm Approximation	ℓ_p	ℓ_2	ℓ_1, ℓ_2
Datatype Agnosticism	—	yes, Tab. II	yes, Tab. III
Complexity	—	$\mathbb{C}$, ERVD	$\mathbb{R}$

The efficacy of the datatype-agnostic methodology becomes obvious during the top-down implementation flow of an algorithm, here the SD. Usually algorithms are developed and verified with double floating point accuracy. If complex-valued, an RVD has to be applied at some time, and to obtain a slim and fast hardware design additionally a transition to fixed point operation is necessary. Especially these modifications require deep hardware expertise but that can significantly be alleviated by the datatype-agnostic design methodology.

Further configuration flexibility can be harvested from the fact that the C code can be fully parametrised by custom-defined constants or pre-processor macros. This goes beyond the possibilities HDL generics offer. For the SD, each parameter discussed in the previous section or listed in Tab. I is mapped to such a constant or macro. This makes it extensively parametrisable at compile time. E.g., the problem size, written as a value pair (M, N), can be given as two constants, the optimisation objective function $f(\mathbf{x})$, Eq. (4), can be configured to match the application based upon a macro definition, and so on.

The employed HLS compiler is Xilinx Vivado HLS. The SD algorithm was coded in C++ while oberserving the HLS coding guidelines and verified against a valid Mathworks Matlab model [9]. All SD parameters were defined as C pre-processor macros in a separate header file. The application of directives is not scope of this work. However, this work can be tuned by the later addition of these to meet application-specific timing requirements. The datatype-agnostic methodology was adopted and a macro `DATATYPE` was defined to switch between different C++ `typedef` specifications. This includes single-precision floating point and Vivado's fixed point data type `ap_fixed`, which is derived from and very similiar to SystemC's `sc_fixed`. For all real-valued datatypes `T` there is a corresponding complex-valued definition `std::complex<T>`.

IV. Example Applications and Results

We define two communication application scenarios to which SD has been applied to in the literature. This will demonstrate the flexibility of the proposed parametrisable SD hardware design. Scenario A is a classical MIMO system and

ISBN 978-0-9928626-7-1 © EURASIP 2017

Table II
IMPLEMENTATION RESULTS FOR SCENARIO A

Datatype		Time	Resource Utilisation			
T	$\mathbb{K}$	μs	BRAM	DSP	FF	LUT
float	$\mathbb{C}$	0,634	12	45	5815	11430
ap_fixed	$\mathbb{C}$	0,848	8	29	3438	6961
float	$\mathbb{R}$	0,591	8	8	2346	3815
ap_fixed	$\mathbb{R}$	0,514	5	12	1412	1757

Scenario B is an uplink of a machine-type communications wireless sensor network. The parametrisations of the SD and variations of them are summarised in Tab. I. Details and results will be explained in the following.

Synthesis was carried out for a Xilinx Zynq-7 (xc7z020clg484-1) as hardware target device at $f_{\text{clk}} = 100\,\text{MHz}$ clock frequency, i.e. operations are re-scheduled by HLS until the critical path matches this timing requirement.

A. MIMO Communications

Scenario A is a classical MIMO communications system with multiple anntennas. There are $N = 4$ transmit and $M = 4$ receive antennas. The influence of the wireless transmission channel is modelled as frequency non-selective and non-line-of-sight, i.e. the fading coefficients h_{mn} of the channel matrix $\mathbf{H}$ are Rayleigh-distributed. $\mathbf{H}$ is known at the receiver, i.e. perfect channel estimation is assumed. Data bits are digitally modulated with 16-QAM, $\mathcal{S} = \mathcal{A}^{\text{16-QAM}}$. Hence, optimal ML detection must be according to

$$\hat{\mathbf{x}} = \arg \min_{\mathbf{x} \in (\mathcal{A}^{\text{16-QAM}})^N} \|\tilde{\mathbf{y}} - \mathbf{R}\mathbf{x}\|_2^2 \,, \text{ or} \tag{8a}$$

$$\hat{\mathbf{x}}' = \arg \min_{\mathbf{x}' \in (\mathcal{A}^{\text{4-ASK}})^{2N}} \|\tilde{\mathbf{y}}' - \mathbf{R}'\mathbf{x}'\|_2^2 \tag{8b}$$

for a real-value decomposed equivalent problem. The primed entities are real of double size, e.g. $\mathbf{x}'$ is a vector of length $2N$ with elements taken from the real-valued 4-Amplitude Shift Keying (ASK), $\mathcal{A}^{\text{4-ASK}} = \{-3, -1, 1, 3\}$, whereby $\mathcal{A}^{\text{16-QAM}} = \{a + jb \mid \forall a, b \in \mathcal{A}^{\text{4-ASK}}\}$. Since the QRD is applied after RVD, $\mathbf{R}'$ maintains its upper-triangular structure.

Table II lists the HLS synthesis results of the parametrised SD. The design is optimised for minimal resource utilisation, i.e. without parallelisations. Two variations of the ML MIMO SD are listed: the complex-valued (Eq. 8a) and real-valued with ERVD (Eq. 8b). The times given are the estimated processing time of a node. This is lower for the RVD case, but note that in fact an 8-by-8 system of equations has to be solved which increases the total runtime superlinearly. Hence, it is advantageous to process complex symbols though the resource utilisation is slightly more than twice the utilisation count of the RVD setup. Fig. 2 plots the bit error rate (BER) performance of the SD over the signal-to-noise ratio (SNR) and proves that the problems in (8) are indeed equivalent. This holds true for the implementation as well. The baseline fixed point configuration was $\text{Q}(10.8)$. This choice will be explained in further detail below.

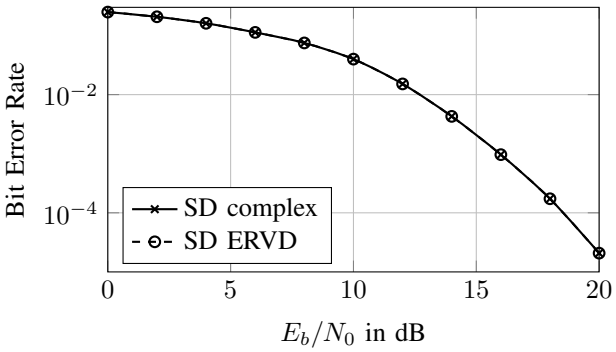

Figure 2. BER vs. SNR (E_b/N_0 in dB) for 4×4 MIMO-system

Table III
IMPLEMENTATION RESULTS FOR SCENARIO B

Datatype	Setup		Time	Resource Utilisation			
$\mathbb{C}$	(M, N)	ℓ_p	μs	BRAM	DSP	FF	LUT
float	(16, 16)	ℓ_2	1.634	6	48	1941	5136
ap_fixed	(16, 16)	ℓ_2	0.754	5	31	1558	3567
float	(16, 16)	ℓ_1	0.982	6	23	1930	1219
ap_fixed	(16, 16)	ℓ_1	0.406	5	17	1497	2214
float	(20, 32)	ℓ_1	0.976	6	23	1948	4063
ap_fixed	(20, 32)	ℓ_1	0.417	5	17	1506	2343

B. Wireless Sensor Network with Sporadic Transmissions

The second examined toy problem follows the system descriptions in [5], [10]. Machine-type communication is often sporadic and of a low data rate. Precisely, we assume a wireless sensor network where N sensor nodes transmit data symbols to a central aggregation node (uplink) with a certain activity probability p_a only. To avoid activity signalling overhead, simultaneous transmissions are allowed and channel access is uncoordinated, however data symbols are spreaded by a node-specific code, similar to a Code-Division Multiple Access (CDMA) system, to allow for user separation at the receiver side. Random Bernoulli sequences (length M) are used as spreading codes and the node-specific channel impulse responses consist of four random Rayleigh fading coefficients.

The receiver has to jointly detect the activities and data symbols of all users. The digital modulation alphabet is augmented with a zero symbol to model inactivity: $\mathcal{A}_0 = \mathcal{A} \cup \{0\}$, i.e. here $\mathcal{A}_0^{\text{BPSK}} = \{-1, 0, +1\}$. Since activity occurs with $p_a = 0.2$, a priori information about the statistics of $\mathbf{x}$ is available and a MAP optimisation problem can be derived as

$$\hat{\mathbf{x}} = \arg \min_{\mathbf{x} \in \mathcal{A}_0^N} \|\mathbf{y} - \mathbf{H}\mathbf{x}\|_2^2 + 2\lambda\sigma_n^2 \|\mathbf{x}\|_0 \,, \tag{9}$$

with $\lambda = \ln(\frac{1-p_a}{p_a/|\mathcal{A}|})$ being the ratio of the a priori probabilities for inactivity and a certain symbol from the modulation alphabet. σ_n^2 is the channel noise power and most importantly $\|\mathbf{x}\|_0$ is the number of non-zero elements of $\mathbf{x}$, often said to be the ℓ_0-pseudo norm.

Table III lists synthesis results for three variations of this parametrisation for SD. First, a system with $N = M = 16$ was

ISBN 978-0-9928626-7-1

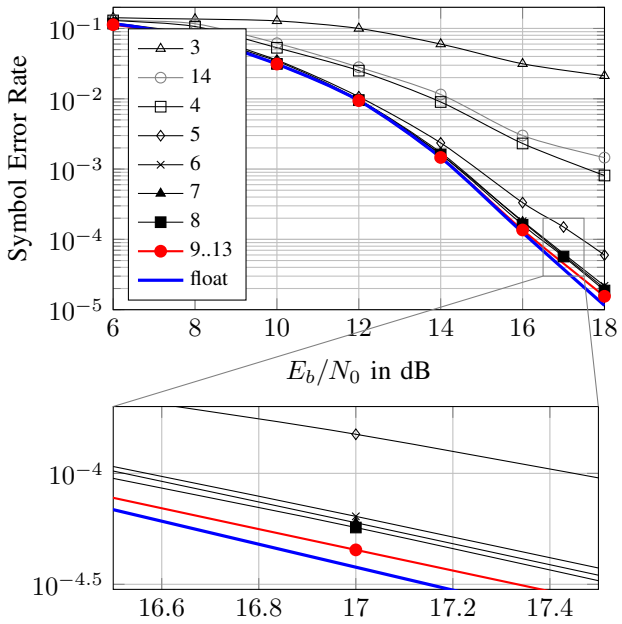

Figure 3. Detection performance for a fixed-point wordlength of $w = 18$ bits but varying number of fractional bits f.

investigated, i.e. the CDMA system is critically loaded. Due to the sparse nature of the channel access a joint activity and data detection is still possible. Second, $(M, N) = (32, 20)$ was chosen for an overdetermined system. Note, that the fixed point design considerably reduces the number of utilised DSP slices (dedicated mulipliers on the FPGA) while still maintaining floating point detection performance. 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 basically equal for both cases.

The optimal Q fixed point format of $w = 18$ bits wordlength was found by a parametric sweep. This wordlength best matches the input wordlength of the DSP slices and leads to an optimal utilisation-accuracy trade-off. The SD design was tested for Q formats $\mathrm{Q}(w - f.f)$ with f being the number of fractional bits. Fig. 3 shows the detection performance in terms of symbol error rate over the SNR for a range f. As it can be seen, for small fractional lengths ($f = 3, ..., 8$) the detection performance is degraded due to insufficient numerical precision (underflows). For $f = 9, ..., 13$ detection performance nearly achieves floating point accuracy. But if there are too few integer bits, overflows occur and detection is severely deteriorated. Hence, $\mathrm{Q}(10.8)$ was chosen for the overall design.

Furthermore, the influence of a complexity-reduced norm was investigated. Fig. 4 compares the detection performance of the SD metric given in (9) with a metric where the ℓ_2-norm of the ML part is substituted for the simpler ℓ_1 norm in (7). A moderate loss in SER performance of about 2 dB can yield a 54 % faster and smaller (DSP slices) design as a trade-off (see Tab. III).

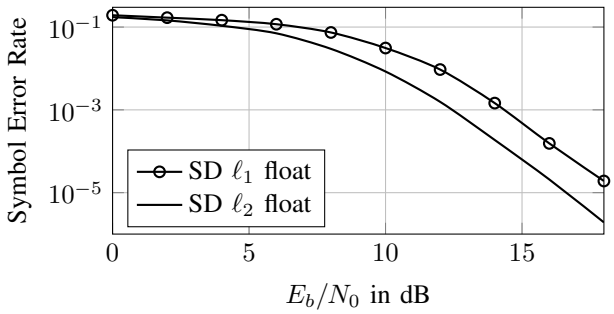

Figure 4. Influence of the norm computation with reduced-complexity on the detection performance of an uncoded system.

V. Conclusion

A parametrisable hardware design of a Sphere Decoder was presented. Fine-granular settings are possible for the problem size, modulation alphabet, norm approximation, sphere radius and search metric, datatypes etc. With further HLS directives added to the design it could easily be adapted to meet stricter timing constraints. The datatype-agnostic design methodology allowed for the fast development of a fixed point design: the fixed point Q format could be correctly scaled and verified with an application-driven test solely based on a single baseline definition. The presented results show that the proposed parametrisable Sphere Decoder based on HLS is a versatile design for a variety of applications.

Acknowledgment

This work was partially funded by the German Research Foundation (DFG) under grant PA 438/8-1 and the German Federal Ministry of Education and Research within the project “HiFlecs” (Reference number: 16KIS0271).

References

[1] B. Hassibi and H. Vikalo, “On the sphere-decoding algorithm I. Expected complexity,” *IEEE Transactions on Signal Processing*, vol. 53, no. 8, pp. 2806–2818, Aug. 2005.

[2] S. Qiao, “Integer least squares: Sphere decoding and the LLL algorithm,” in *Proceedings of the 2008 C3S2E Conference*, 2008, pp. 23–28.

[3] A. Burg, M. Borgmann, M. Wenk, M. Zellweger, W. Fichtner, and H. Bolcskei, “VLSI implementation of MIMO detection using the sphere decoding algorithm,” *IEEE Journal of Solid-State Circuits*, vol. 40, no. 7, pp. 1566–1577, Jul. 2005.

[4] E. G. Larsson, “Mimo detection methods: How they work,” *IEEE signal processing magazine*, vol. 26, no. 3, pp. 91–95, May 2009.

[5] H. Zhu and G. B. Giannakis, “Exploiting sparse user activity in multiuser detection,” *IEEE Transactions on Communications*, vol. 59, no. 2, pp. 454–465, Feb. 2011.

[6] B. Knoop, J. Rust, S. Schmale, D. Peters-Drolshagen, and S. Paul, “Rapid digital architecture design of orthogonal matching pursuit,” in *Proceedings of the 24th European Signal Processing Conference (EUSIPCO)*, Aug. 2016, pp. 1857–1861.

[7] T. Wiegand and S. Paul, “Reduced complexity computation unit for a sphere decoding algorithm,” *European Wireless 2012, VDE VERLAG GMBH*, Jun. 2012.

[8] H. Ren, “A brief introduction on contemporary high-level synthesis,” *IC Design & Technology (ICICDT), 2014 IEEE International Conference on*, May 2014.

[9] I. Xilinx, *Vivado Design Suite User Guide. High-Level Synthesis. UG902 (v2016.4)*, Nov. 2016.

[10] F. Monsees, C. Bockelmann, D. Wubben, and A. Dekorsy, “Sparsity aware multiuser detection for machine to machine communication,” in *2012 IEEE Globecom Workshops*, Dec. 2012, pp. 1706–1711.

ISBN 978-0-9928626-7-1 © EURASIP 2017