Rapid Digital Architecture Design of Orthogonal Matching Pursuit

Benjamin Knoop, Jochen Rust, Sebastian Schmale, Dagmar Peters-Drolshagen, Steffen Paul
Institute of Electrodynamics and Microelectronics (ITEM.me), University of Bremen
Otto-Hahn-Allee 1, 28357 Bremen, Germany
Email: {knoop, rust, schmale, peters, paul}@me.uni-bremen.de

***Abstract*—Orthogonal Matching Pursuit (OMP) is a greedy algorithm well-known for its applications to Compressed Sensing. For this work it serves as a toy problem of a rapid digital design flow based on high-level synthesis (HLS). HLS facilitates extensive design space exploration in connection with a data type-agnostic programming methodology. Nonetheless, some algorithmic transformations are needed to obtain optimised digital architectures. OMP contains a least squares orthogonalisation step, yet its iterative selection strategy makes rank-1 updating possible. We furthermore propose to compute complex mathematical operations, e.g. the needed reciprocal square root operation, with the help of the logarithmic number system to optimise HLS results. Our results can compete with prior works in terms of latency and resource utilisation. Additionally and to the best of our knowledge, we can report on the first complex-valued digital architecture of OMP, which is able to recover a vector of length 128 with 5 non-zero elements in 17.1 µs.**

I. Introduction

The design of faster and 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 in terms of adders, multipliers, etc. for practical implementations. Though the technological evolution scales quite well to accommodate this growth in complexity, the engineer's design capabilities do not. This is often called the productivity design gap. In [1] it was shown that this 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 is, the higher is the design entropy. 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 more difficult to design. To overcome the design productivity 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 [1]. In other words, the design of hardware architectures must be elevated to the algorithmic layer away from traditional hardware description languages (HDL) and the register-transfer level (RTL).

One answer to this is high-level synthesis (HLS) [2], [3]. As input, HLS takes an algorithmic description in a high-level language like structured C/C++ or SystemC and synthesises as output an RTL description in an HDL. Usually, this is accompanied by further design decisions (e.g. the amount of parallelisation or pipelining) and boundary conditions (e.g. resource utilisation). Compared to the otherwise needed design time, HLS accelerates digital architecture design. Another major advantage of HLS is the possibility to explore the design space without much effort, since problem sizes and even data types are configurable parameters.

The contribution of this paper is threefold. First, we will show that it is possible to implement even complex, heterogeneous algorithms, based on the rapid digital architecture design paradigm of HLS using the example of Orthogonal Matching Pursuit (OMP), [4]. OMP is a greedy algorithm best known for its applications to Compressed Sensing (CS). There is also a recent application to sparse multi-user channel estimation in a wireless communications network [5], which can only be facilitated if the output architecture allows for real-time operation. OMP makes frequent use of linear algebra operations and is hence parallelisable to some extent but contains a lot of control structure as well. Therefore, we view OMP as a fitting subject of study in the context of HLS.

As the second contribution, we propose to solve the least squares (LS) step of OMP with rank-1 updates to the modified Gram-Schmidt QR matrix decomposition (QRD). This builds upon ideas from the literature, as we will discuss in the following section, but takes it a step further by calculating the pseudoinverse with rank-1 updates as well. The results of [6] are integrated in our design to efficiently compute the inverse vector norm using the logarithmic number system (LNS).

And, to the best of our knowledge, we propose the first VLSI design of OMP for complex numbers. This is the third contribution of this paper which emphasises the power of HLS if combined with a data type-agnostic design methodology.

II. Orthogonal Matching Pursuit

Orthogonal Matching Pursuit (OMP) was introduced by Tropp and Gilbert in 2007 as a greedy algorithm for the recovery of a sparse vector from random linear measurements [4]. It is based on the CS system model

$$\mathbf{b} = \mathbf{A}\mathbf{x}\,, \tag{1}$$

where $\mathbf{A}$ is the $M \times N$ random CS measurement matrix, $\mathbf{b}$ the measurement vector and $\mathbf{x}$ the to be recovered sparse vector. $\mathbf{x}$ 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

978-0-9928-6265-7/16/$31.00 ©2016 IEEE

1: **function** OMP($\mathbf{b}, \mathbf{A}, k$)
2: $\quad \mathbf{r} \leftarrow \mathbf{b}$; $\mathbf{x} \leftarrow \mathbf{0}$; $\Lambda \leftarrow \emptyset$ ▷ initialisation
3: $\quad$ **for** $t = 1, ..., k$ **do**
4: $\quad\quad \lambda \leftarrow \arg\max_\ell |\langle \mathbf{A}_\ell, \mathbf{r} \rangle|$ ▷ correlation
5: $\quad\quad \Lambda \leftarrow \Lambda \cup \{\lambda\}$ ▷ index set
6: $\quad\quad \mathbf{x}_\Lambda \leftarrow \arg\min_{\mathbf{x}_\Lambda} \|\mathbf{b} - \mathbf{A}_\Lambda \mathbf{x}_\Lambda\|_2$ ▷ least squares
7: $\quad\quad \mathbf{r} \leftarrow \mathbf{b} - \mathbf{A}_\Lambda \mathbf{x}_\Lambda$ ▷ residual
8: $\quad$ **end for**
9: $\quad$ **return** $\mathbf{x}$
10: **end function**

Fig. 1: Pseudocode of Orthogonal Matching Pursuit (OMP) according to Tropp and Gilbert [4].

$k \ll M \ll N$. Note, that Eq. (1) can be either real-valued or complex-valued. $\mathbf{A}$ must fulfil certain conditions, but these are with high probability satisfied when its entries are drawn from a random process, e.g. Gaussian. Then, OMP can reliably recover $\mathbf{x}$ with $\mathcal{O}(k \ln N)$ random linear measurements [4].

The algorithm is re-stated as pseudocode in Fig. 1. It recovers a k-sparse vector iteratively in k iterations, with t as the loop counter. The loop body mainly consists of three computational steps or kernels, each dependent on the previous one: (i) correlation and selection, (ii) LS orthogonalisation and (iii) residual update. First, 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 (lines 4 and 5). 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 6 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 6 would be $\mathbf{x}_\Lambda = \mathbf{A}_\Lambda^+ \mathbf{b}$, whereby $(\cdot)^+$ denotes the Moore-Penrose pseudoinverse. The last step within the loop is the update of the residual.

III. Related Work

There has already been published a substantial body of research on the digital hardware design of OMP [7]–[13]. Although there are some ASIC designs, most of the literature focuses on FPGAs as target platform. A comparative overview is given in Tab. I, including some selected results of ours.

Most importantly, a couple of works [7], [8], [10], [12] implement the OMP substantially different compared to the original one published by Tropp and Gilbert [4]. We will refer to this as the two-stage variant of OMP (see Tab. I). 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 $\mathbf{r}$ (Fig. 1, line 7). Septimus and Steinberg proposed in [7] 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 6 and 7, 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 $\mathbf{r}_t = \mathbf{r}_{t-1} - \mathbf{Q}_{\lambda_t} \mathbf{Q}_{\lambda_t}^T \mathbf{r}_{t-1}$. And not until after the loop and before line 9 in Fig. 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 [13]. This is especially true because [7] omits any proof or derivation of their fundamentally different formulation and such has not yet been published by others. However, these are indeed equivalent as we proof in the appendix of this paper.

Various possibilities were proposed to solve the LS problem. Some works [9], [13] rely on a Cholesky decomposition to avoid the square root operation. In [4], a QRD was used within each iteration. [7] 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

$$\mathbf{x}_\Lambda = \mathbf{A}_\Lambda^+ \mathbf{b} = \left(\mathbf{A}_\Lambda^H \mathbf{A}_\Lambda\right)^{-1} \mathbf{A}_\Lambda^H \mathbf{b} \tag{2}$$

$$= \left(\mathbf{R}^H \mathbf{Q}^H \mathbf{Q} \mathbf{R}\right)^{-1} \mathbf{R}^H \mathbf{Q}^H \mathbf{b} = \mathbf{R}^{-1} \mathbf{Q}^H \mathbf{b}\,. \tag{3}$$

This was realised by Bai et al. [10], who computed an incremental QRD during the first stage and used back substitution for the second stage to obtain $\mathbf{R}^{-1}$ to solve the LS problem.

Most VLSI designs were manually coded in an HDL, probably using generic expressions for different problem sizes, e.g. in [8]. It is noteworthy that other works already acknowledged the need for higher design abstraction. The results in [9], [13] were obtained with Matlab Simulink in conjunction with Xilinx System Generator, which basically is a graphical, model-based hardware description approach. A major strength of these tools is the convenient support for fixed-point data types with varying bit widths and easy design verification, but for huge designs the usability suffers as these are tedious to program and debug. HLS promises to leverage the disadvantages while keeping the benefits. Therefore, a first implementation of OMP with HLS was reported on in [11], where the Xilinx' Vivado High-Level Synthesis (VHLS) tool was used. But unfortunately the authors did not state any details on the problem sizes, synthesised architecture or algorithmic method to compute the LS problem.

IV. Algorithmic Implementation

In this section we will describe two significant modifications to the method of computation of OMP. These are applicable to classical and two-stage OMP alike although we restrict ourselves to the classical variant in the following. First, we propose rank-1 updates to the pseudoinverse as well as to the QRD. And second, we suggest to perform mathematically complex operations, i.e. division and square root, within the logarithmic domain.

A. Least Squares with Rank-1 Updates

As previously discussed, a possible strategy to solve the LS problem of OMP is to apply QR decomposition by modified Gram-Schmidt orthogonalisation. This is the only

TABLE I: Comparison of Related VLSI Designs of Orthogonal Matching Pursuit Targeting FPGAs.

		Problem Size			Time	Freq.	Format		Resource Utilisation				
Ref.	**Variant**	M	N	K	(µs)	(MHz)	$Qm.n$	**Target**	BRAM	DSP	FF	LUT	NMSE
[7]	2-stage	32	128	5	24.0	39	Q10.22	Virtex-5	—	—	—	—	—
[8]	2-stage	64	128	5	10.0	85	Q10.14	Virtex-5	—	—	—	—	—
[8]	2-stage	64	256	8	27.1	85	Q10.14	Virtex-5	—	—	—	—	—
[9]	classical	32	128	5	15.7	107	16 bit	Virtex-5	42	134	2341	4012	47.0 dB (PSNR)
[10]	2-stage	256	1024	36	[c]622.0	100	18 bit	Virtex-6	258	261	—	32010	—
[11]	classical	—	[c]512	—	—	128	float	Zynq-7	19	27	3776	6605	—
[12]	2-stage [a]	64	256	8	7.1	85	Q10.14	Virtex-5	—	—	—	—	7.1×10^{-3}
[13]	classical	256	1024	36	340.0	119	Q9.9	Virtex-6	576	589	—	6208	38.9 dB (PSNR)
This Work	classical	32	128	5	10.7	103	Q4.14	Virtex-7	4	518	22564	18330	1.4×10^{-7}
This Work	classical	32	128	5	16.9	107	Q4.14	Virtex-7	4	130	13781	22345	1.4×10^{-7}
This Work	classical [b]	32	128	5	17.1	114	Q3.15	Virtex-7	1	274	25733	46449	1.0×10^{-3}
This Work	classical [b]	64	256	8	65.1	101	Q3.15	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] First complex-valued OMP and thus not directly comparable either. [b] Parameter extracted from other given results.

1: **function** LS-UPDATE(${}^{(t-1)}\mathbf{Q}, {}^{(t-1)}\mathbf{R}^{-1}, \mathbf{A}_\lambda, \mathbf{b}$)
2: $\quad\mathbf{r} \leftarrow \mathbf{0}_{t\times 1}$
3: $\quad\mathbf{q} \leftarrow \mathbf{A}_\lambda$
4: $\quad$**for** $j = 1, ..., t-1$ **do** ▷ orthogonalisation
5: $\quad\quad\mathbf{r}_j \leftarrow \langle {}^{(t-1)}\mathbf{Q}_j, \mathbf{q}\rangle$
6: $\quad\quad\mathbf{q} \leftarrow \mathbf{q} - \mathbf{r}_j \cdot {}^{(t-1)}\mathbf{Q}_j$
7: $\quad$**end for**
8: $\quad\zeta \leftarrow 1/\|\mathbf{q}\|$ ▷ normalisation factor
9: $\quad{}^{(t)}\mathbf{Q} \leftarrow \begin{bmatrix}{}^{(t-1)}\mathbf{Q} & \zeta\cdot\mathbf{q}\end{bmatrix}$ ▷ update
10: $\quad{}^{(t)}\mathbf{R}^{-1} \leftarrow \begin{bmatrix}{}^{(t-1)}\mathbf{R}^{-1} & -{}^{(t-1)}\mathbf{R}^{-1}\cdot\mathbf{r}\cdot\zeta \\ \mathbf{0}_{1\times t-1} & \zeta\end{bmatrix}$
11: $\quad\mathbf{x}_\Lambda \leftarrow {}^{(t)}\mathbf{R}^{-1}\cdot{}^{(t)}\mathbf{Q}^H\cdot\mathbf{b}$ ▷ least squares
12: $\quad$**return** $\mathbf{x}_\Lambda, {}^{(t)}\mathbf{Q}, {}^{(t)}\mathbf{R}^{-1}$
13: **end function**

Fig. 2: Least squares estimation based on QR matrix decomposition with rank-1 updating.

method which offers the possibility for iterative updating, is numerically stable enough and can be parallelised nicely for big matrices.

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 (3). 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 $\mathrm{diag}(\mathbf{R})$ is real-valued in either case. Now, block matrix inversion gives us

$$\begin{bmatrix}\mathbf{A} & \mathbf{B}\\ \mathbf{C} & \mathbf{D}\end{bmatrix}^{-1} = \begin{bmatrix}\mathbf{S}^{-1} & -\mathbf{S}^{-1}\mathbf{B}\mathbf{D}^{-1}\\ -\mathbf{D}^{-1}\mathbf{C}\mathbf{S}^{-1} & \mathbf{D}^{-1}+\mathbf{D}^{-1}\mathbf{C}\mathbf{S}^{-1}\mathbf{B}\mathbf{D}^{-1}\end{bmatrix} \quad (4)$$

where $\mathbf{S}$ is the Schur complement of $\mathbf{D}$ being defined as $\mathbf{S} = \mathbf{A} - \mathbf{B}\mathbf{D}^{-1}\mathbf{C}$. Here, $\mathbf{A}$ is the 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 value d, and, due to the triangular structure, $\mathbf{C} = \mathbf{0}_{1\times t-1}$ (row of zeroes). Therefore, Eq. (4) can be simplified to

$$\begin{bmatrix}\mathbf{A} & \mathbf{b}\\ 0 & d\end{bmatrix}^{-1} = \begin{bmatrix}\mathbf{A}^{-1} & -\mathbf{A}^{-1}\mathbf{b}/d\\ 0 & 1/d\end{bmatrix}. \quad (5)$$

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) becomes

$$\begin{bmatrix}{}^{(t-1)}\mathbf{R} & \mathbf{r}\\ 0 & r_{tt}\end{bmatrix}^{-1} = \begin{bmatrix}{}^{(t-1)}\mathbf{R}^{-1} & -{}^{(t-1)}\mathbf{R}^{-1}\cdot\mathbf{r}/r_{tt}\\ \mathbf{0}_{1\times t-1} & 1/r_{tt}\end{bmatrix}, \quad (6)$$

when we insert the $\mathbf{R}$ matrix at hand. 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 then, 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.

The complete LS algorithm with rank-1 updating is given in Fig. 2, which updates $\mathbf{Q}$ and $\mathbf{R}^{-1}$ iteratively. This fits nicely into the OMP algorithm in Fig. 1 as a replacement of line 6, as the for-loop over $t = 1, ..., k$ embraces this update step. Please note that $\mathbf{R}$ must not be stored since only the update vector $\mathbf{r}$ and ζ is of further interest. This proposed algorithmic transformation therefore yields a flatter loop hierarchy and will eventuate optimised implementation results.

B. Application of the Logarithmic Number System (LNS)

The computation of $\zeta = 1/\|\mathbf{q}\| = 1/\sqrt{x}$ with $x = \langle\mathbf{q},\mathbf{q}\rangle$ requires a single reciprocal square root operation per iteration t. However, its impact on computational complexity can be mitigated by performing this operation within the logarithmic number system. Then, the square root and division become a much simpler bit shift and subtraction, respectively, and it is

$$\zeta = \mathrm{ALOG}\left(\mathrm{LOG}(1) - (\mathrm{LOG}(x) \gg 1)\right), \quad (7)$$

with $\mathrm{LOG} = \log_2(x+1)$ and $\mathrm{ALOG} = 2^x - 1$ for number format conversion. We propose to integrate the methodology

TABLE II: Synthesis results for the reciprocal square root operation within the logarithmic domain.

Function	Format	Latency	DSP	FF	LUT	Accuracy
LOG	Q1.15	3	1	182	1619	6.1×10^{-5}
ALOG	Q1.15	4	1	283	2059	6.1×10^{-5}
INVSQRT	Q7.13	11	2	591	4688	7.5×10^{-5}

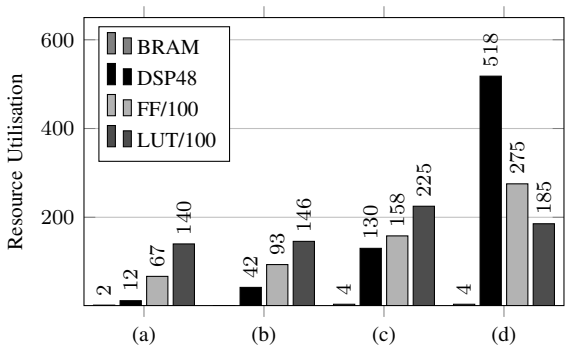

Fig. 3: Resource utilisation of four different architectures from unoptimised (left) to highly parallelised (right).

of [6], i.e. nonuniform piecewise-linear function approximation (NPA), as a very efficient means to implement these functions. The LOG and ALOG functions were approximated by 43 resp. 51 linear segments in the form of $\alpha_0 x + \beta_0$ to achieve a certain target accuracy. Hence, the converter mainly consists of a case differentiation to determine the applicable interval based on the most significant bits of the input number, followed by a multiply-accumulate (MAC) operation. The digital synthesis results are given in Tab. II. The inverse square root according to (7) can be computed in 11 clock cycles, using only two DSP slices and with an average accuracy of 7.5×10^{-5} compared against a double precision software model. The application of the LNS and NPA is needed, because otherwise the HLS tool will infer costly floating point arithmetic, which takes about 70 cycles to compute ζ.

V. High-Level Synthesis Results

We implemented the OMP algorithm with the modifications discussed above with Xilinx Vivado High-Level Synthesis (VHLS) following the coding principles and guidelines set forth in [3]. All synthesis results given in this paper target a Xilinx Virtex-7 FPGA (xc7vx690) at 100 MHz clock frequency. The C++ source code was written with parametrised static memory allocations and verified against a Mathworks Matlab model for floating point data types. Thereby we followed a data type-agnostic design methodology, i.e. we solely utilised self-defined data types with the `typedef` keyword. By this, exactly the same C++ code can serve as a starting point for digital synthesis, simply with exchanged data type definitions or parameter sets, e.g. the OMP problem size (M, N, k). The following results are, if not stated otherwise, for a $(32, 128, 5)$ setup with Q4.14 fixed-point precision (i.e. 18 bits word width with four integer bits and 14 fractional bits) to match the architecture of the DSP slices and Q8.28 for selected MAC operations.

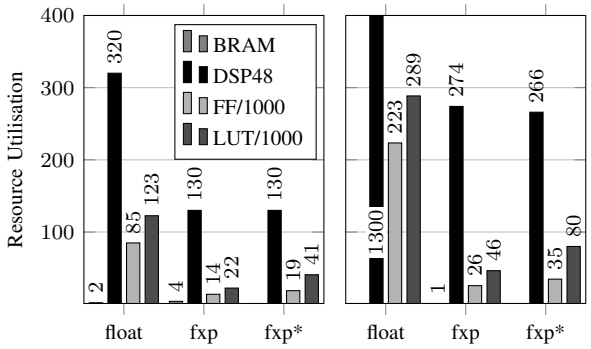

Fig. 4: Resource utilisation with different data types for real-valued (left) and complex-valued (right) computation.

Architecture synthesis by HLS heavily depends on compiler directives (pragma). The two most relevant pragmas supported by VHLS are `HLS UNROLL` for loop-unrolling (parallelisation) and `HLS PIPELINE` for automatic insertion of pipelining stages and reducing loop latencies. Fig. 3 shows the synthesis results for four different sets of directives: (a) is a design with minimal resource utilisation as the default optimisation goal of VHLS; (b) computes the correlation and selection step 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) is the previous design where additionally almost all loops are pipelined; (d) is a design with all second-level loops unrolled and thus highly parallelised. However, (c) constitutes the best trade-off between resource utilisation and computation time.

To facilitate complex-valued operation, we made extensive use of the `complex` class of the C++ Standard Library which is supported by VHLS for synthesis. The data type-agnostic design methodology could be kept even in this case because of further C++ language features, which are (i) class and function templates and (ii) operator overloading. For example, the `std::conj` function was overloaded for all real-valued data types to simply return the value. Complex multiplication was taken care of by the Standard Library since based on the data type the C++ compiler and VHLS can infer the meaning of the multiplication operator.

In Fig. 4 the impact of different data types is compared for architecture (c), as described above. "fxp" denotes the above mentioned Q4.14 format and "fxp*" the same but with saturation arithmetic and number rounding enabled. These are typical DSP features and improve the normalised mean squared error (NMSE), but only a little from 1.4×10^{-7} to 1.8×10^{-8}. VHLS is also able to synthesise a floating point design which yields an NMSE of 1.3×10^{-12} (single precision `float`). However, the resource utilisation is inefficient compared to the gain in accuracy. The right side of Fig. 4 shows results for the complex-valued case. Except for floating point operation, the complex designs consume about twice the resources as it can be expected. Nonetheless, all synthesised architectures do not exceed the capacity of the target FPGA. Floating point designs avoid numerical issues and are faster

to create, but they result in a large resource utilisation.

Details of selected synthesis results are given in Tab. I. Besides the fact that HLS supposedly will never create an RTL design as efficient as an experienced hardware designer would do, the generated digital architectures can compete with existing works. A real-valued OMP of size $(32, 128, 5)$ can be computed within $10.7\,\mu s$ or $16.9\,\mu s$, depending on the degree of parallelisation, which is comparable to the results given in [9]. The same can be said about the resource utilisation, although VHLS obviously favours distributed RAM over BRAM. The complex-valued digital architecture of OMP is with $17.1\,\mu s$ almost as fast as the real-valued one.

VI. Conclusion

The results show that the rapid digital architecture design flow based on HLS used in this paper is a superior alternative to the traditional RTL design methodology. HLS plays an important role to enlarge the design space and facilitates fast design space exploration: architectures for various problem sizes, data types (real and complex) and different degrees of parallelisation can all be synthesised based on the very same C++ source code and set of HLS directives. And the results can compete with prior works. The key enabler for this is the applied data type-agnostic programming style. This allows us to report on the first digital architecture for complex-valued sparse signal recovery with OMP.

VII. Appendix

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 $\mathbf{R}$,

$$\mathbf{QR} = \mathbf{A}_\Lambda\,. \tag{8}$$

Further it is $\mathbf{x}_\Lambda = \mathbf{A}_\Lambda^+\mathbf{b} = \mathbf{R}^{-1}\mathbf{Q}^T\mathbf{b}$ as stated above in eq. (3). Now, Tropp's and Gilbert's residual update is given by [4]

$$\mathbf{r}_t = \mathbf{b} - \mathbf{A}_\Lambda\mathbf{x}_\Lambda\,, \quad \text{with} \quad \mathbf{x}_\Lambda = \mathbf{A}_\Lambda^+\mathbf{b}\,, \tag{9}$$

whereas Septimus' and Seinberg's residual update strategy is [7]

$$\mathbf{r}_t = \mathbf{r}_{t-1} - \mathbf{Q}_t\mathbf{Q}_t^T\mathbf{r}_{t-1} = \left(\mathbf{I} - \mathbf{Q}_t\mathbf{Q}_t^T\right)\mathbf{r}_{t-1}\,. \tag{10}$$

To show the equivalence between (9) and (10), we will first insert (8) and (3) into (9):

$$\begin{aligned}\mathbf{r}_t &= \mathbf{b} - \mathbf{A}_\Lambda\mathbf{A}_\Lambda^+\mathbf{b} = \left(\mathbf{I}_{M\times M} - \mathbf{A}_\Lambda\mathbf{A}_\Lambda^+\right)\mathbf{b} \\ &= \left(\mathbf{I} - \mathbf{QRR}^{-1}\mathbf{Q}^T\right)\mathbf{b} = \left(\mathbf{I} - \mathbf{QQ}^T\right)\mathbf{b}\,.\end{aligned} \tag{11}$$

The first two iterations according to (10) are

$$\mathbf{r}_1 = \left(\mathbf{I} - \mathbf{Q}_1\mathbf{Q}_1^T\right)\mathbf{r}_0 = \left(\mathbf{I} - \mathbf{Q}_1\mathbf{Q}_1^T\right)\mathbf{b}\,, \text{ and} \tag{12}$$

$$\begin{aligned}\mathbf{r}_2 &= \left(\mathbf{I} - \mathbf{Q}_2\mathbf{Q}_2^T\right)\left(\mathbf{I} - \mathbf{Q}_1\mathbf{Q}_1^T\right)\mathbf{b} \\ &= \left(\mathbf{I} - \mathbf{Q}_1\mathbf{Q}_1^T - \mathbf{Q}_2\mathbf{Q}_2^T + \mathbf{Q}_2\mathbf{Q}_2^T\mathbf{Q}_1\mathbf{Q}_1^T\right)\mathbf{b} \\ &= \left(\mathbf{I} - \mathbf{Q}_1\mathbf{Q}_1^T - \mathbf{Q}_2\mathbf{Q}_2^T\right)\mathbf{b}\,,\end{aligned} \tag{13}$$

using 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

$$\mathbf{r}_t = \left(\mathbf{I} - \mathbf{Q}_1\mathbf{Q}_1^T - \mathbf{Q}_2\mathbf{Q}_2^T \cdots - \mathbf{Q}_t\mathbf{Q}_t^T\right)\mathbf{b} \tag{14}$$

$$= \left(\mathbf{I} - \sum_{\ell=1}^{t}\mathbf{Q}_\ell\mathbf{Q}_\ell^T\right)\mathbf{b} = \left(\mathbf{I} - \mathbf{QQ}^T\right)\mathbf{b}\,, \tag{15}$$

which is identical to (11) for every iteration t. Therefore, the classical OMP by Tropp and Gilbert and the two-stage OMP by Septimus and Steinberg are indeed equivalent. Based on the same residual they will always select the same support set and return the same solution after the LS step. ■

Acknowledgment

This work was funded by the German Research Foundation (DFG) under grant PA 438/8-1.

References

[1] B. Menhorn and F. Slomka, "Confirming the Design Gap," in *Computational Science, Engineering and Information Technology, Proceedings of the Third International Conference on*, ser. Advances in Intelligent Systems and Computing, vol. 225. Heidelberg: Springer International Publishing, Jul. 2013, pp. 281–292.

[2] R. Nane, V.-M. Sima, C. Pilato, J. Choi, B. Fort, A. Canis, Y. Chen, H. Hsiao, S. Brown, F. Ferrandi, J. Anderson, and K. Bertels, "A Survey and Evaluation of FPGA High-Level Synthesis Tools," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, Dec. 2015.

[3] Xilinx Inc., *Vivado Design Suite User Guide High-Level Synthesis*, Nov. 2015, UG902 (v2015.4).

[4] J. Tropp and A. Gilbert, "Signal Recovery From Random Measurements Via Orthogonal Matching Pursuit," *Information Theory, IEEE Transactions on*, vol. 53, no. 12, pp. 4655–4666, Dec. 2007.

[5] B. Knoop, S. Schmale, D. Peters-Drolshagen, and S. Paul, "Activity and Channel Estimation in Multi-User Wireless Sensor Networks," in *20th International ITG Workshop on Smart Antennas (WSA)*, ser. ITG-Fachbericht, no. 261, Information Technology Society in the VDE (ITG). VDE Verlag, Mar. 2016, pp. 289–293.

[6] J. Rust, F. Ludwig, and S. Paul, "Low-Complexity QR Decomposition Architecture using the Logarithmic Number System," in *Design, Automation Test in Europe Conference Exhibition (DATE)*, Mar. 2013, pp. 97–102.

[7] A. Septimus and R. Steinberg, "Compressive Sampling Hardware Reconstruction," in *Circuits and Systems (ISCAS), Proceedings of 2010 IEEE International Symposium on*, May 2010, pp. 3316–3319.

[8] J. Stanislaus and T. Mohsenin, "High Performance Compressive Sensing Reconstruction Hardware with QRD Process," in *Circuits and Systems (ISCAS), IEEE International Symposium on*, May 2012, pp. 29–32.

[9] P. Blache, H. Rabah, and A. Amira, "High-Level Prototyping and FPGA Implementation of the Orthogonal Matching Pursuit Algorithm," in *Information Science, Signal Processing and their Applications (ISSPA), 2012 11th International Conference on*, Jul. 2012, pp. 1336–1340.

[10] L. Bai, P. Maechler, M. Muehlberghuber, and H. Kaeslin, "High-Speed Compressed Sensing Reconstruction on FPGA using OMP and AMP," in *Electronics, Circuits and Systems (ICECS), 19th IEEE International Conference on*, Dec. 2012, pp. 53–56.

[11] H. Rabah, A. Amira, and A. Ahmad, "Design and Implementaiton of a Fall Detection System using Compressive Sensing and Shimmer Technology," in *Microelectronics (ICM), 2012 24th International Conference on*, Dec. 2012, pp. 1–4.

[12] J. Stanislaus and T. Mohsenin, "Low-Complexity FPGA Implementation of Compressive Sensing Reconstruction," in *Computing, Networking and Communications (ICNC), International Conference on*, Jan. 2013, pp. 671–675.

[13] H. Rabah, A. Amira, B. Mohanty, S. Almaadeed, and P. Meher, "FPGA Implementation of Orthogonal Matching Pursuit for Compressive Sensing Reconstruction," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 23, no. 10, pp. 2209–2220, Oct. 2015.