IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received May 20, 2021, accepted June 8, 2021, date of publication June 23, 2021, date of current version July 1, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3091778

Hardware Implementation of a Latency-Reduced Sphere Decoder With SORN Preprocessing

MORITZ BÄRTHEL[1], SIMON KNOBBE[2], JOCHEN RUST[3], AND STEFFEN PAUL[1]

[1]Department of Communication Electronics (ITEM.me), Institute of Electrodynamics and Microelectronics, University of Bremen, 28359 Bremen, Germany
[2]Department of Communications Engineering (ANT), Institute of Telecommunications and High-Frequency Technique, University of Bremen, 28359 Bremen, Germany
[3]DSI Aerospace Technologie GmbH, 28199 Bremen, Germany

Corresponding author: Moritz Bärthel (baerthel@me.uni-bremen.de)

This work was supported by the Research Project "SPEED" through the Zentrale Forschungsfoerderung (ZF), University of Bremen.

ABSTRACT Unum type-II based Sets-Of-Real-Numbers (SORN) arithmetic is a recently proposed, promising number representation providing fast and low complex implementations of arithmetic operations at the expense of low resolution. The format can be applied for constraining large optimization problems by means of preprocessing. In this work SORN arithmetic is applied for reducing the latency of a Sphere Decoder by excluding a number of solutions in advance. In particular, a comprehensive hardware implementation is presented, consisting of an adapted Sphere Decoder, as well as SORN and matrix preprocessing. Logic and physical synthesis evaluations show that the mean number of visited nodes within the Sphere Decoder can be reduced by up to 76%, resulting in an overall latency reduction of up to 20%. This improvement comes with an area and energy increase of up to 58% and 83%, respectively, compared to a standard Schnorr-Euchner Sphere Decoder.

INDEX TERMS Unum, SORN, digital arithmetic, MIMO, sphere decoding.

I. INTRODUCTION AND RELATED WORK

The representation of numbers in digital systems and their manipulation in terms of arithmetic operations is still a paramount challenge for the design of modern high-performance digital circuits and systems [1]. Besides trivial integer formats, fixed point (FxD) and floating point are the two common alternatives for representing real numbers in computer architectures and digital signal processing circuits [2]. In particular, the IEEE-754 standard for floating point arithmetic [3] dominated the market for decades. Beyond that, also logarithmic number formats have turned out to be a suitable choice in some special purpose applications, e.g for solving recursive least-squares problems or computing the roots of a polynomial using the Laguerre algorithm [4].

In the recent decade, several new number formats were proposed [5]. A highly interesting candidate is the universal number format Unum, presented in three different versions, all targeting to overcome the limitations of State-of-the-Art (SotA) formats like traditional floats. Whereas type-I Unums [6] exploit implicit Interval Arithmetic (IA) in order to avoid the propagation of rounding errors, the type-III Unum approach [7] basically extends traditional floats by an extra scale factor and a more flexible interpretation of fraction and exponent field widths, resulting in a higher dynamic range without increasing the wordlength.

The type-II Unum format and the resulting SORN representation [8] denote a coarse resolution format exploiting IA and enabling fast and low complex datapaths. The SORN representation consists of exact values and intervals. Arithmetic operations are mapped to lookup tables (LUTs) which are highly appropriate for hardware implementation. This kind of arithmetic is especially well suited for constraining large optimization problems by means of preprocessing for example when solving linear or nonlinear systems of equations [8].

A possible target application for such kind of arithmetic is the symbol detection at the receiver in a Multiple-Input-Multiple-Output (MIMO) wireless communication system, where a finite-alphabet-constrained least-squares problem has to be solved in order to reconstruct the transmitted data [9]. SORN arithmetic applied to MIMO symbol detection was first introduced in [10], where the basic idea of a SORN preprocessor in such a scenario was evaluated. The presented SORN preprocessing unit reduces the number of possible solutions for the MIMO detection problem in order

The associate editor coordinating the review of this manuscript and approving it for publication was Fang Yang.

to simplify SotA detectors. In [11] the influence of different SORN datatype configurations was studied.

To this end, solely the usability of a SORN preprocessor was investigated in the works mentioned before. This article targets the second step of this approach, the further processing of the remaining solutions after SORN preprocessing. A first simple, software-based examination on this topic has been carried out in [12], where different permutation algorithms for applying the SORN-based reduction to a Sphere Decoder (SD) are evaluated.

In this article the foregoing works are combined and continued by means of a comprehensive hardware implementation of a complete MIMO detector. The main contributions of this article can be summarized as follows:

- Demonstration of the suitability of SORN arithmetic for signal processing architectures, in this work applied to an SD for MIMO symbol detection.
- The first hardware implementation of a MIMO detection algorithm based on SORN preprocessing, combining a SORN preprocessor, a sorting algorithm for constraining the SD, a QR-decomposition (QRD), and the constrained SD into a toplevel SORN-based MIMO decoder.
- A comprehensive evaluation of the proposed approach, based on software- and hardware-related register-transfer-level (RTL) simulations, as well as CMOS 28 nm syntheses, including comparisons with a conventional Schnorr-Euchner SD (SE-SD) implementation and other literature SD and comparable SotA algorithms and implementations.

II. UNUMS AND SORNs

Traditional IA is a technique where computations are carried out on interval operands rather than single values, used for example in scientific computing in order to minimize the effect of rounding errors that come with floating point computations [1]. Processing intervals leads to quite complex datapaths, since not only the bitwidth of the operands is doubled, but also the computational effort of arithmetic operations increases. A multiplication of two interval operands for example requires four single float multiplications and comparison operations [1].

The Unum format is an approach where IA is realized in a different way. With type-I Unums, implicit IA with ULP-wide intervals is introduced whenever maximum precision is exceeded (ULP: Unit of Least Precision [13]). When an IEEE floating point number would be rounded, a Unum value indicates an interval between the represented and the next larger exact value [6]. In this way a doubled operand bitwidth can be avoided, even though arithmetic operations are still more complex than for traditional floats.

A. ORIGINAL TYPE-II UNUMS

Type-II Unums maintain the concept of ULP-wide intervals, but with a rigorous reduced precision, resulting in a very coarse quantization of the real numbers. In [8] a minimal example is given, resulting in the following representation of the reals:

$$\{\pm\infty\ (-\infty, -1)\ -1\ (-1, 0)\ 0\ (0, 1)\ 1\ (1, \infty)\} \tag{1}$$

The idea of SORNs is a binary representation of the given set, where every entry of the set is encoded with a dedicated bit. Union intervals are represented with consecutive bit pattern. A few binary SORN examples for the set from Eq. (1) are given in the following:

$$\begin{aligned} 00010000 &\mathrel{\hat{=}} (-1, 0) \\ 00000110 &\mathrel{\hat{=}} (0, 1] \\ 11111110 &\mathrel{\hat{=}} [-\infty, 1] \end{aligned} \tag{2}$$

With this SORN representation LUTs can be generated, which contain all possible outputs for a certain arithmetic operation with two SORN inputs. The general design-flow for a SORN datapath is depicted in Fig. 1 for a half-open SORN configuration which will be explained in Sec. II-B. From the SORN representation in Fig. 1a LUTs containing the outputs of arithmetic operations for the given datatype are generated. Fig. 1b shows the multiplication LUT for a simplified 3b SORN datatype. The LUTs are mapped to Boolean Logic circuits as shown in Fig. 1c.

This kind of LUT-based computation with comparatively small inputs provides an ultra fast, low complex and very regular way of implementing arithmetic operations. However, due to the low resolution a sequence of SORN-based computations quickly results in large intervals. As a consequence, the application of SORNs for straight-forward signal processing of complex algorithms or tasks usually leads to very poor performance only. Nevertheless, if SORNs are considered for preprocessing recurrently computing simple algorithms, the low-complexity and fast-computing nature can be efficiently utilized, for example for constraining problems with a large solution space.

Further details about the original type-II Unums and SORNs can be found in [8] and [10].

B. ADAPTED SORN ARITHMETIC

The original Unum type-II-based SORN datatypes like Eq. (1) contain exact values and open intervals. A case study evaluating different SORN datatypes within a MIMO preprocessor datapath from [11] shows that the original configuration is not optimal, at least for this kind of application. Due to the fact that the single exact values do not match the application data, mostly consecutive bit pattern occur within the LUT-based SORN datapath. A SORN datatype using half-open intervals performs much better since the redundancy of an exact value next to an adjacent interval is removed [11].

Following this evaluation, in this work SORN datatypes using half-open intervals are considered. The different configurations for different bitwidths are shown in Tab. 1. Solely the *lin*17 configuration contains a non-zero exact value

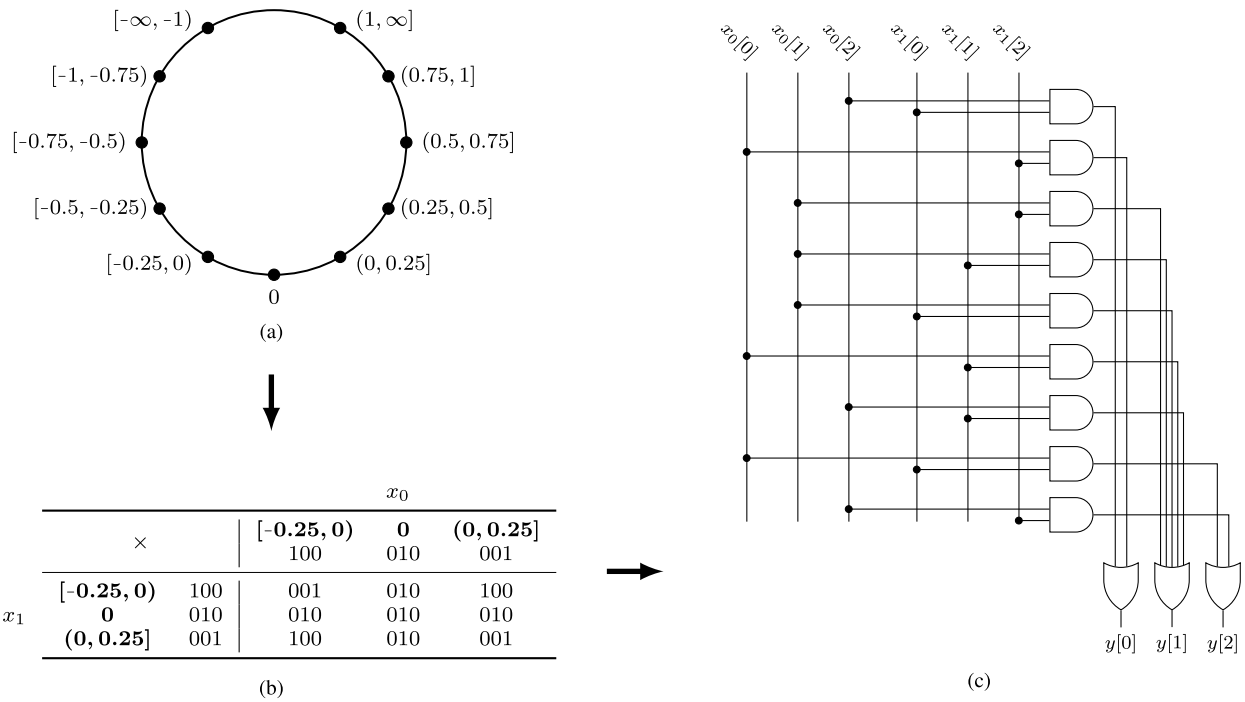

FIGURE 1. Design-flow for a SORN datapath with (a) an adapted SORN representation with 11b using half-open intervals and linear spacing, (b) the LUT structure for a multiplication of two SORN operands using a simplified 3b datatype, and (c) the gate-level structure for the multiplication of two SORN operands using a simplified 3b datatype, evolved from the LUT in (b).

TABLE 1. SORN Datatype Configurations considered in this work, all with linear spacing and half-open intervals. The respective bitwidth is encoded in the label.

*lin*9	$[-\infty,-1)\ [-1,-\frac{2}{3})\ [-\frac{2}{3},-\frac{1}{3})\ [-\frac{1}{3},0)\ 0\ (0,\frac{1}{3}]\ (\frac{1}{3},\frac{2}{3}]\ (\frac{2}{3},1]\ (1,\infty]$
*lin*11	$[-\infty,-1)\ [-1,-\frac{3}{4})\ [-\frac{3}{4},-\frac{1}{2})\ [-\frac{1}{2},-\frac{1}{4})\ [-\frac{1}{4},0)\ 0\ (0,\frac{1}{4}]\ (\frac{1}{4},\frac{1}{2}]\ (\frac{1}{2},\frac{3}{4}]\ (\frac{3}{4},1]\ (1,\infty]$
*lin*13	$[-\infty,-2)\ [-2,-1)\ [-1,-\frac{3}{4})\ [-\frac{3}{4},-\frac{1}{2})\ [-\frac{1}{2},-\frac{1}{4})\ [-\frac{1}{4},0)\ 0\ (0,\frac{1}{4}]\ (\frac{1}{4},\frac{1}{2}]\ (\frac{1}{2},\frac{3}{4}]\ (\frac{3}{4},1]\ (1,2]\ (2,\infty]$
*lin*15	$[-\infty,-2)\ [-2,-\frac{3}{2})\ [-\frac{3}{2},-1)\ [-1,-\frac{3}{4})\ [-\frac{3}{4},-\frac{1}{2})\ [-\frac{1}{2},-\frac{1}{4})\ [-\frac{1}{4},0)\ 0\ (0,\frac{1}{4}]\ (\frac{1}{4},\frac{1}{2}]\ (\frac{1}{2},\frac{3}{4}]\ (\frac{3}{4},1]\ (1,\frac{3}{2}]\ (\frac{3}{2},2)\ (2,\infty]$
*lin*17	$[-\infty,-2)\ [-2,-\frac{3}{2})\ [-\frac{3}{2},-1)\ [-1,-\frac{1}{\sqrt{2}})\ -\frac{1}{\sqrt{2}}\ (-\frac{1}{\sqrt{2}},-\frac{1}{2})\ [-\frac{1}{2},-\frac{1}{4})\ [-\frac{1}{4},0)\ 0\ (0,\frac{1}{4}]\ (\frac{1}{4},\frac{1}{2}]\ (\frac{1}{2},\frac{1}{\sqrt{2}})\ \frac{1}{\sqrt{2}}\ (\frac{1}{\sqrt{2}},1]\ (1,\frac{3}{2}]\ (\frac{3}{2},2]\ (2,\infty]$

which matches the data symbols of the MIMO transmission application.

III. MIMO TRANSMISSION AND SPHERE DECODING

In wireless MIMO transmission systems multi-antenna arrays are used in order to increase the spectral efficiency, i.e. the datarate, compared to single antenna systems [14]. In this work the transmission of N single-antenna clients to a basestation with N antennas is considered, as depicted in Fig. 2. The clients simultaneously transmit digital modulated data over a flat fading Rayleigh distributed channel, described by the channel matrix $\mathbf{H} \in \mathbb{C}^{N\times N}$. The transmit data vector $\mathbf{x} \in \mathcal{S}^N$ is composed of symbols from the finite alphabet $\mathcal{S}$. All symbols have an identical *a priori* probability. The received signal vector $\mathbf{y} \in \mathbb{C}^N$ can be stated as

$$\mathbf{y} = \mathbf{H}\mathbf{x} + \mathbf{n} \tag{3}$$

with the additive, zero-mean white Gaussian noise vector $\mathbf{n} \in \mathbb{C}^N$ [15]. The channel matrix $\mathbf{H}$ is assumed to be known at the receiver due to channel estimation. In order to calculate

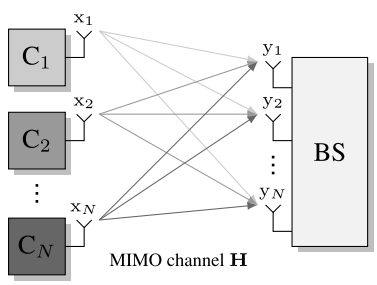

FIGURE 2. Multiple Client upload scenario for a wireless MIMO transmission with N transmit and receive antennas.

the estimate of the transmit vector $\hat{\mathbf{x}} \in \mathcal{S}^N$, the maximum-likelihood-estimation (MLE) problem

$$\hat{\mathbf{x}} = \underset{\mathbf{x}\in\mathcal{S}^N}{\operatorname{argmin}} \|\mathbf{y} - \mathbf{H}\mathbf{x}\|_2 \tag{4}$$

has to be solved at the receiver. Various approaches like the linear detection methods Matched Filter (MF) or Zero Forcing (ZF) can be applied, as well as non-linear detectors like

Soft Interference Cancellation (SIC) or the tree-search-based SD approach [9], [15].

In the following, for all simulations a flat fading Rayleigh distributed channel with additive, zero-mean white Gaussian noise and no coding is assumed. The given signal-to-noise ratio (SNR) values represent the mean SNR over all receive antennas in decibel (dB). A Quadrature Phase-Shift Keying (QPSK) modulation with $\mathcal{S} \in \{\pm\frac{1}{\sqrt{2}} \pm j\frac{1}{\sqrt{2}}\}$ and modulation number $m = 4$ is applied.

A. SPHERE DECODING

The basic principle of Sphere Decoding is to search for the estimate transmit vector $\hat{\mathbf{x}}$ among lattice points $\mathbf{x} \in \mathcal{S}^N$ which lie within a sphere with radius r around the received vector $\mathbf{y}$ [16]. This is done by evaluating the norm of the lattice points:

$$\|\mathbf{y} - \mathbf{H}\mathbf{x}\|_2 \leq r \tag{5}$$

Since the sphere is multi-dimensional, determining those lattice points that lie within the sphere would require a high computational effort. The approach is to reduce the problem to a single dimension and successively calculate the required norm. This transforms the problem into a tree search where every tree level represents one dimension of the sphere.

For applying an element-wise solution, the MLE problem (4) has to be rewritten by using a QRD of matrix $\mathbf{H}$ [9]:

$$\hat{\mathbf{x}} = \underset{\mathbf{x}\in\mathcal{S}^N}{\operatorname{argmin}} \|\underbrace{\mathbf{Q}^{\mathbf{H}}\mathbf{y}}_{\tilde{\mathbf{y}}} - \mathbf{R}\mathbf{x}\|_2 \tag{6}$$

$\mathbf{Q} \in \mathbb{C}^{N\times N}$ is hereby an orthogonal and $\mathbf{R} \in \mathbb{C}^{N\times N}$ an upper triangular matrix [17]. With this transformation, the squared norm can be defined element-wise:

$$\left\|\tilde{\mathbf{y}} - \mathbf{R}\mathbf{x}\right\|_2^2 = \sum_{j=1}^{N} \left| \tilde{\mathrm{y}}_j - \sum_{i=j}^{N} (\mathrm{R}_{ji}\mathrm{x}_i) \right|^2 \tag{7}$$

Since $\mathbf{R}$ is upper triangular, the last element from Eq. (7) $i = j = N$ is computed at the first tree level. Eq. (7) can be defined recursively as the error $e(l)$ of level l with $e(0) := 0$ [18]:

$$e(l) = \left| \tilde{\mathrm{y}}_{N-l+1} - \sum_{i=N-l+1}^{N} \mathrm{R}_{N-l+1,i}\,\mathrm{x}_i \right|^2 + e(l-1) \tag{8}$$

The behavior of the tree search algorithm is illustrated in Fig. 3. The algorithm exploits a depth-width search with adaptive radius r, which is adjusted every time the bottom tree level is reached (also called pruning). The initial radius is set to $r = \infty$. At the root level the error metric of the lower-level nodes $l = 1$ is calculated according to Eq. (8). Following Schnorr-Euchner [19] the path with lowest error metric is evaluated first, which is the left subtree in the given example. When the bottom level is reached, r is adjusted according to the current $e(l)$ and the remaining branches are evaluated with the new radius. Whenever $e(l) > r$ the respective branch is discarded. The symbol vector corresponding to the bottom level node with the lowest determined error metric is the estimated symbol vector $\hat{\mathbf{x}}$.

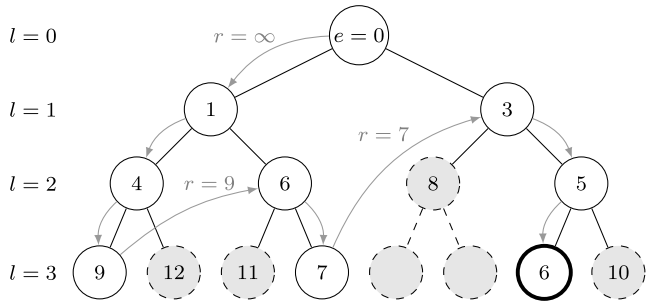

FIGURE 3. Illustration of a SE-SD algorithm with pruning. Each node contains the accumulated error metric $e(l)$ for the respective path. The adaptive radius r is adjusted each time the bottom level is reached.

B. SORN PREPROCESSING

Algorithms like SD are required for MIMO symbol detection because the number of possible solutions for Eq. (4) $|\mathcal{S}|^N$ increases exponentially with the number of transmit antennas N and the modulation. The straight forward approach would be an exhaustive search for all possible solutions which is impractical with standard number formats because of the high computational complexity and/or long computing time. With the fast and low complex SORN arithmetic, however, an exhaustive search becomes feasible.

The SORN preprocessor for MIMO symbol detection processes the squared norm $\|\mathbf{y} - \mathbf{H}\mathbf{x}\|_2^2$ for every possible $\mathbf{x} \in \mathcal{S}^N$ leading to different SORN values, all representing consecutive intervals greater zero (example for *lin*9 datatype):

$$\|\mathbf{y} - \mathbf{H}\mathbf{x}_i\|_2^2 \Big|_{\text{SORN}} = 000001110 \mathrel{\hat{=}} (0, 1] \tag{9}$$

$$\left\|\mathbf{y} - \mathbf{H}\mathbf{x}_j\right\|_2^2 \Big|_{\text{SORN}} = 000000011 \mathrel{\hat{=}} (2/3, \infty] \tag{10}$$

Those vectors $\mathbf{x}$ leading to a norm with an open-zero lower interval bound are marked as valid solution and considered for further processing. In the given example from Eq. (9) $\mathbf{x}_i$ leads to a close-to-zero result and would be considered a valid solution, whereas $\mathbf{x}_j$ in Eq. (10) leads to a larger norm and would be discarded.

Due to the limited precision of the SORN datatypes, multiple close-to-zero SORN results appear among the calculated norms for the different symbol vectors. Consequently, the output of the SORN preprocessor is a set of possible solution vectors $\hat{\mathbf{x}} \in \mathcal{R} \subset \mathcal{S}^N$ with $|\mathcal{R}| < |\mathcal{S}|^N$. In the following $\mathcal{R}$ is referred to as *remaining solutions after SORN preprocessing*.

In Fig. 4 the effect of such a SORN preprocessor for MIMO symbol detection is visualized. The remaining solutions after SORN preprocessing for different SORN datatypes and depending on the SNR are given in Fig. 4a for a 4×4 MIMO system and Fig. 4b for 8×8 MIMO. With increasing bitwidth and precision of the SORN datatype, the number of

remaining solutions $|\mathcal{R}|$ decreases. For the *lin*17 datatype and 4×4 MIMO the number of possible solutions can be reduced by more than 94 % on average, for 8×8 and the same datatype the reduction is more than 86 % on average.

Fig. 4c and 4d show the possibility that the maximum-likelihood (ML) solution is discarded by the preprocessing algorithm and does not appear among the remaining solutions. It can be observed that the lower the number of remaining solutions, the higher the probability of excluding the ML result, but only for low SNR values. Further evaluations in Sec. V-A show that this exclusion of the ML result for low SNRs does not affect the uncoded bit error rate (BER) performance of the SORN-reduced SD.

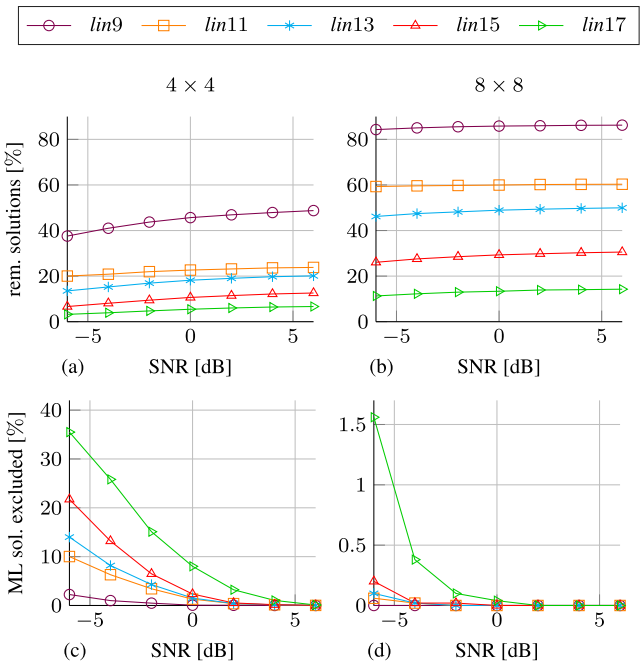

FIGURE 4. (a) & (b) Remaining solutions for the MLE problem after SORN preprocessing and (c) & (d) number of cases where the ML solution is excluded by the SORN preprocessor; each in % for a 4×4 and 8×8 QPSK MIMO system with 5×10^3 simulations.

C. COMPLEXITY-REDUCED SPHERE DECODING

After the SORN preprocessing step, the number of possible vectors $\mathbf{x} \in \mathcal{S}^N$ to solve the MLE problem (4) is reduced. This reduction can now be utilized to reduce the number of nodes in the SD search tree.

Although the standard SD algorithm does not visit every existing node within the tree due to the adaptive radius feature, it still contains every possible vector $\mathbf{x}$ at the bottom tree level. In order to reduce the SD tree complexity, the bottom level nodes corresponding to those symbol vectors $\mathbf{x}$ that are excluded by the preprocessor can be deleted from the search tree. Fig. 5a shows an example for a 3-dimensional Binary Phase-Shift Keying (BPSK) tree where two of the bottom level nodes are deleted.

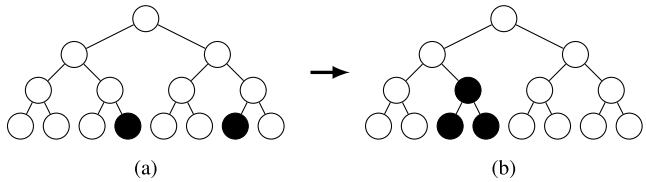

FIGURE 5. Permutation of the Sphere Decoder search tree: (a) original tree with deleted nodes (black) resulting from the SORN preprocessing and (b) permuted tree for an unbalanced node ratio [12].

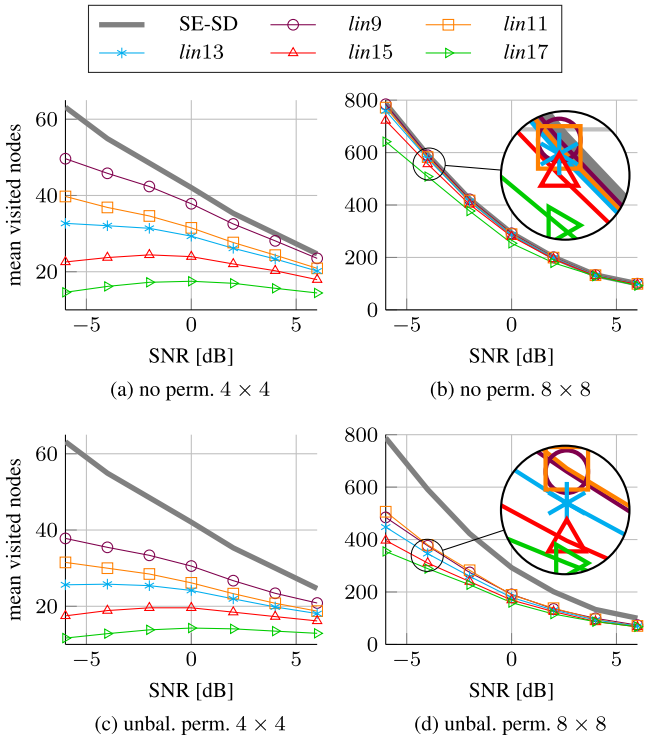

FIGURE 6. Mean visited nodes of a standard SE-SD (), and a SE-SD with deleted nodes after SORN preprocessing without permutation (a) & (b) and with unbalanced permutation (c) & (d); for 4×4 and 8×8 QPSK MIMO with 5×10^3 simulations.

The intended outcome of reducing the number of nodes within the search tree is obviously to reduce the overall number of visited nodes, i.e. the computing time for one detection. Fig. 6a (4×4) and 6b (8×8) show the mean number of visited nodes for one detection over different SNR values. For every detection the SD search tree was reduced according to the result of the respective SORN preprocessing step. The evaluation was carried out for the different SORN configurations and for a standard SE-SD with a full search tree. For a 4×4 system the number of visited nodes is reduced by more than 75 % for low SNRs and a 17b SORN datatype. For the 8×8 case the improvement is much lower.

The reduction of the SD search tree and, consequently, the computing time can be further improved by virtually permuting the order of the transmit antennas. Applied to the SD, this results in a permutation of the search tree after the excluded nodes from the preprocessing are deleted. In [12]

different permutation techniques are evaluated and it is shown that a permutation leading to an unbalanced ratio of the subtree sizes can further decrease the number of visited nodes. The concept is depicted in Fig. 5b: Permuting the tree in such a way that multiple deleted nodes in one subtree can be achieved allows to delete the whole subtree. In the given example from the permutation of two deleted nodes a third node and therefore a complete subtree can be deleted.

With this unbalanced sorting approach the mean number of visited nodes can be further decreased as shown in Fig. 6c (4×4) and 6d (8×8). Especially for the 8×8 case a huge improvement can be observed compared to a reduced tree without permutation, but also the 4×4 case is further improved.

D. STATE-OF-THE-ART COMPLEXITY-REDUCTION APPROACHES

In the history of MIMO detectors, various different approaches for tree-search-based algorithms have been developed in order to optimize the conventional SD. An overview of different algorithms and their hardware implementations can be found in [20]. In the following, some of these approaches are discussed and compared to the proposed SORN-SD approach in terms of computational complexity and the additional preprocessing effort. The complexity of the different algorithms is evaluated in terms of the visited nodes during the tree-search, which are depicted in Fig. 7 for 4×4 and 8×8 MIMO for different SNR values.

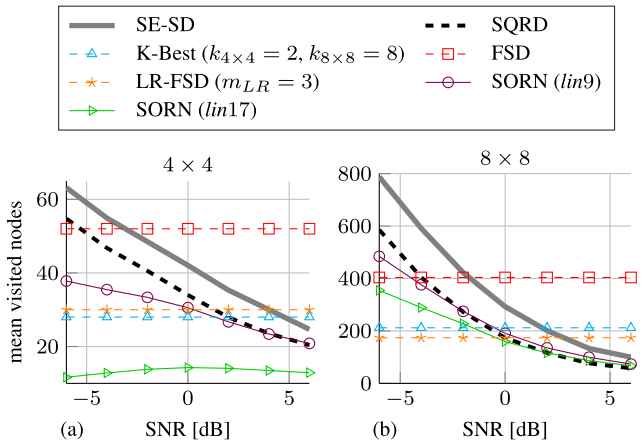

FIGURE 7. Mean visited nodes of a standard SE-SD, with SQRD, K-Best, FSD and LR-FSD, and permuted tree after SORN preprocessing; for (a) 4×4 and (b) 8×8 QPSK MIMO with 5×10^3 simulations. (The K-Best, FSD and LR-FSD results are obtained analytically.).

1) SQRD

One approach for decreasing the number of visited nodes during the detection is given by the SQRD algorithm. This approach, like the proposed SORN-SD, targets the interchange of the symbol order, but based on the properties of the MIMO channel $\mathbf{H}$. The SQRD algorithm is based on the Gram-Schmidt method and reorders the columns of the channel matrix by the norm of the column vectors of $\mathbf{H}$ [21], before the matrix is decomposed. In combination with an SD this sorted QRD leads to an optimized computing time because wrong decisions in the first levels of the search tree are minimized [22].

Considering the visited nodes, the SQRD approach shows an improvement compared to the standard SE-SD, but shows equal or worse results compared to the SORN approach, depending on the SNR and system size. For 4×4 MIMO the 17b SORN approach outperforms the SQRD over all SNRs, for 8×8 the number of visited nodes is lower with the SORN approach for SNRs below 0 dB.

Considering the preprocessing, the QRD is replaced with the SQRD algorithm. Hardware implementations indicate a 90% increased number of FPGA slice LUTs [23], and a CMOS area and latency increase of 40% and 7%, respectively [24]. In comparison, in Sec. V-B and IV-F it is shown that for the hardware implementation of the SORN-SD the extra preprocessing requires at most 0.56 times the area and 0.16 times the latency of the implemented QRD.

2) K-BEST

The main drawback of the conventional, depth-first SD algorithm regarding hardware implementation is the variable number of visited nodes per detection and the resulting non-deterministic runtime. Two approaches targeting this problem are the breadth-first K-Best and the fixed-complexity SD (FSD) algorithms. For the K-Best SD in every level of the search tree the k best child nodes with the lowest error metrics $e(l)$ are determined, all other remaining nodes are discarded [25]. The resulting fixed-number of visited nodes enables parallelization and pipelining, but also leads to a degradation in the BER-performance, depending on the choice of the k parameter [20].

The number of visited nodes for the K-Best decoder can be determined as follows (m is the modulation number, i.e. the number of constellation points):

$$N_{visitedNodes,K\text{-}Best} = \begin{cases} m + mk(N-1) & k \leq m \\ m + m^2 + mk(N-2) & m < k \leq m^2 \end{cases} \quad (11)$$

Fig. 7 shows the number of visited nodes for a K-Best algorithm with $k_{4\times4} = 2$ and $k_{8\times8} = 8$, leading to a worse performance compared to the SORN approach for the 4×4 case and for the 8×8 case with SNR ≥ 0 dB. When choosing the k parameters as $k_{4\times4} = 4$ and $k_{8\times8} = 16$, the number of visited nodes is equivalent to the FSD algorithm.

3) FSD

The FSD algorithm also targets a deterministic runtime, achieved with a fixed number of visited nodes per detection. Here the search tree is divided into a full expansion (FE) stage, where all possible paths are considered, and a single expansion (SE) stage, where only one path per node is followed [27]. In order to achieve a quasi-ML performance, for a 4×4 system the first tree level is implemented as an FE

TABLE 2. Comparison of the preprocessing effort for the SORN-based and FSD algorithms supplementary to a QRD and $\mathbf{Q}^{\mathbf{H}}\mathbf{y}$.

Algorithm	BER	Preprocessing	Operations ($N \times N$ MIMO)	
FSD [26]	quasi-ML	FSD ordering of $\mathbf{H}$	$10N^4 + 8N^3 - 9N - 9$	FLOPs
LR-FSD [26]	suboptimal	Complex LR of $\mathbf{H}$	$59.7N^3$	
		FSD ordering of $\mathbf{H}$	$10N^4 + 8N^3 - 9N - 9$	FLOPs
		ZF estimate	$18N^3 + 6N^2 + N$	
SORN-SD	quasi-ML	SORN exhaustive Search	$m^N(8N^2 + 4N - 1)$	SORN OPs
		Permutation of $\mathbf{H}$	$2Nm + 0.5N^2 - 1.5N$	Integer OPs

stage, where all m paths are evaluated, while the lower levels are considered as SE stage where only one path is followed. For 8×8 the first two levels are considered as FE stages. The values for the FSD algorithm in Fig. 7 are obtained according to [27] and [28]:

$$\begin{aligned} N_{visitedNodes,FSD,4\times4} &= m + (N-1)\,m^2 \\ N_{visitedNodes,FSD,8\times8} &= m + m^2 + (N-2)\,m^3 \end{aligned} \tag{12}$$

Compared to the SORN-based approach, the number of visited nodes is higher for the FSD approach for both the 4×4 and 8×8 case.

For achieving a quasi-ML performance, the FSD algorithm requires a preprocessing step where the channel matrix $\mathbf{H}$ is reordered in order to detect the signals with a high noise amplification in the FE stage and those with a low noise amplification in the SE stage [27]. This reordering introduces an additional preprocessing step before the QRD, which is listed in Tab. 2 in terms of floating point operations (FLOPs) [26] for a quadratic MIMO system $N \times N$. Since the supplementary preprocessing for the SORN-SD requires only SORN and integer operations, which can be implemented much more efficiently, a fair comparison can hardly be made. However, considering a complex $N \times N$ QRD which requires $37.3N^3$ FLOPs [17], [26], it can be shown that the complexity for the FSD ordering of $\mathbf{H}$ is 1.2 times higher than for a QRD ($N = 4$). In contrast, in Sec. V-B and IV-F it is shown that for the implementation of the SORN-SD the extra preprocessing requires at most 0.56 times the area and 0.16 times the latency of the implemented QRD, which indicates a lower preprocessing effort than for the FSD.

Independent of preprocessing and visited nodes, both the K-Best and the FSD allow a deterministic runtime which is an advantage, compared to the conventional and the SORN-SD, when considering hardware implementation. With parallelism and pipelining these approaches can achieve higher throughputs, at the expense of increased hardware resources, which will be shown in Sec. V-C.

4) LR-FSD

Besides the conventional SD, K-Best and FSD, there exist numerous derivatives and intermediate approaches, targeting different improvements or hardware platforms, also apart from ASIC or FPGA [29]. One example is the lattice-reduced FSD (LR-FSD) algorithm [26] where applying a lattice reduction (LR) to the channel matrix $\mathbf{H}$ as a further preprocessing step leads to a reduced search tree in the FE-stage of the FSD algorithm. Comparable to K-Best, the LR-FSD algorithm considers not all m nodes for the FE-stage of the FSD, but a reduced number m_{LR}, which is based on the lattice-reduction step. This approach further reduces the number of visited nodes but also results in a degradation of the BER-performance, depending on the choice of the parameter m_{LR} [26]. The values for the LR-FSD in Fig. 7 are obtained according to Eq. (12) with $m = m_{LR} = 3$ and show a similar performance than the K-Best approach for the simulated scenario. The additional preprocessing complexity is higher than for the FSD algorithm and also listed in Tab. 2.

IV. SORN SPHERE DECODER IMPLEMENTATION

In this section the RTL implementation of a complete SD with reduced complexity based on SORN preprocessing is described for a 4×4 MIMO system. The design can be parameterized for any FxD datapath width, and for all different SORN datatypes from Tab. 1, covering 9b to 17b. The toplevel design combining all required subcomponents is depicted in Fig. 8 for 16b FxD. The design is composed of the five following main submodules:

- *SORN*: A SORN-based preprocessing unit reducing the number of possible solutions for the SD as described in Sec. III-B.
- *SORT*: A sorting unit calculating the permutation for the channel matrix $\mathbf{H}$ based on the preprocessing results as described in Sec. III-C.
- *QRD*: A QRD unit decomposing the permuted channel matrix into an orthogonal matrix $\mathbf{Q}$ and an upper triangular matrix $\mathbf{R}$.
- *MVM*: A unit performing the matrix-vector-multiplication $\tilde{\mathbf{y}} = \mathbf{Q}^{\mathbf{H}}\mathbf{y}$ which is required for the SD according to Eq. (6). $\mathbf{Q}^{\mathbf{H}}$ is the hermitian matrix of $\mathbf{Q}$.
- *SD*: A complexity-reduced SD with deleted nodes depending on the preprocessing result as described in Sec III-C and working on the inputs $\tilde{\mathbf{y}}$ and $\mathbf{R}$ which result from the permuted and decomposed channel matrix.

Additionally, the design contains minor subcomponents for permuting the channel matrix $\mathbf{H}$ and the SORN preprocessing result, as well as applying the inverse permutation to the result of the SD. The submodules QRD, MVM and SD are driven

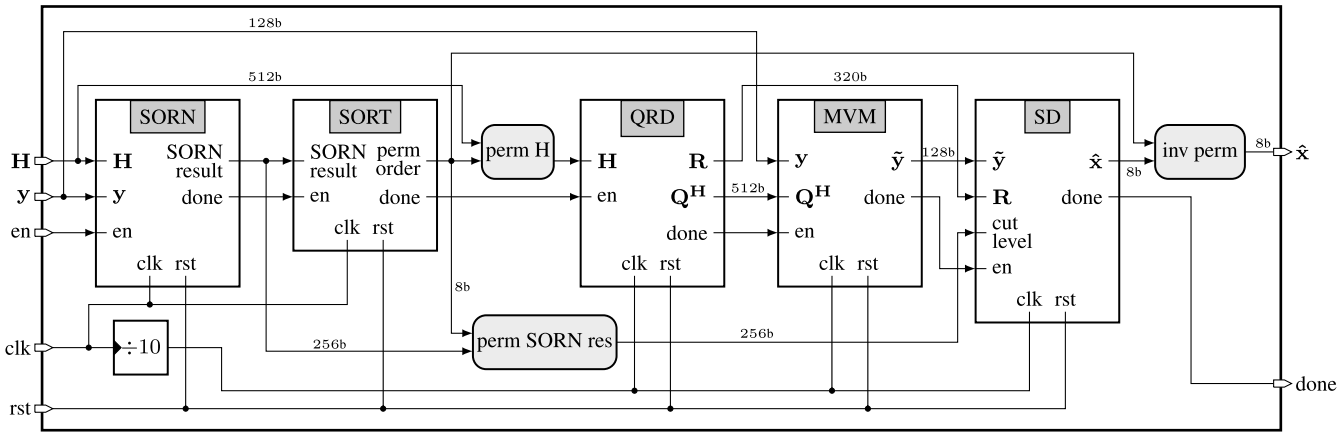

FIGURE 8. Toplevel architecture of the proposed SORN-SD for 4 × 4 MIMO and a 16b FxD datapath: The SORN preprocessor (SORN) and sorting unit (SORT) are running with the fast clock signal, the other modules QRD, matrix-vector-multiplication (MVM) and SD with a 10× lower clock frequency.

by a slower clock signal than SORN and SORT, provided by a frequency divider with a factor of 10.

In order to provide a comparable design, also a traditional SE-SD is implemented, consisting of a QRD, MVM and the actual SD. The QRD and MVM implementations are identical for both designs, the differences of both SD implementations are described in Sec. IV-E. Both designs are evaluated and compared in Sec. V.

A. SORN PREPROCESSOR

The SORN preprocessing unit is responsible for reducing the number of possible solutions for the MLE problem. This is accomplished by means of an exhaustive search such that the squared, complex norm from Eq. (4)

$$\|\mathbf{y} - \mathbf{H}\mathbf{x}\|_2^2 \tag{13}$$

is processed for every possible symbol vector $\mathbf{x} \in \mathcal{S}^N$ in SORN arithmetic. For the given 4 × 4 MIMO scenario using a QPSK modulation the number of possible symbol vectors is

$$|\mathcal{S}|^N = m^N = 4^4 = 256. \tag{14}$$

The architecture of the SORN preprocessor is depicted in Fig. 9. The design is composed of the following subcomponents:

- A conversion unit transforming the FxD inputs $\mathbf{H}$ and $\mathbf{y}$ into the chosen SORN datatype.
- Two parallel SORN solvers processing the squared norm from Eq. (13) in SORN arithmetic, implemented using the SORN datapath generator from [30], both containing 3 pipeline stages.
- A counter unit providing two signals which count the index of the symbol vectors $\mathbf{x}$ that are fed to the SORN solvers, and a unit that selects these vectors according to the counters.
- A register file that stores the calculated norms for every possible solution.

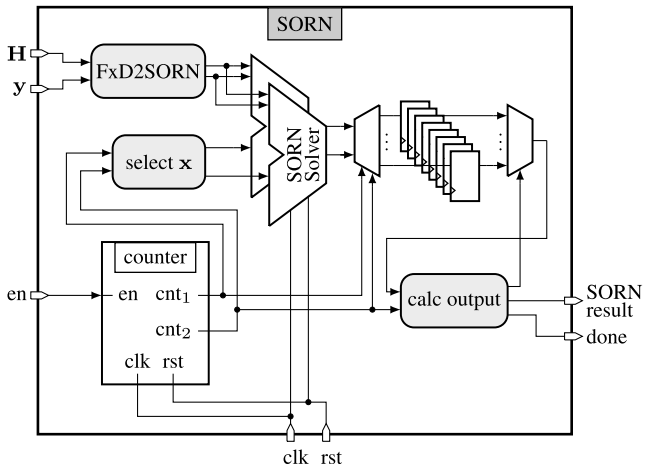

FIGURE 9. SORN preprocessing unit performing an exhaustive search of the MLE problem (4) with two parallel SORN solvers and determining the remaining solutions.

- A unit evaluating the processed results, determining those with a minimal norm as described in Sec. III-B, and setting the output accordingly.

B. SORTING UNIT

The objective of the sorting algorithm is to find a permutation of the transmit antennas such that an unbalanced node ratio in the reduced SD search tree can be achieved, as described in Sec. III-C. In [12] such an algorithm was developed which calculates and maximizes the standard deviation of each subtree size per tree level. Due to the exponentially scaling computational complexity of this algorithm depending on the number of tree levels, an approximate version of the sorting algorithm was developed, which considers only the standard deviations of the first tree level. Additionally, the standard deviation itself is approximated by a squared sum, neglecting some constant terms which are irrelevant for the required comparison. Further details about the sorting algorithm and the performed approximations, as well as a performance evaluation of both algorithm versions can be found in [12].

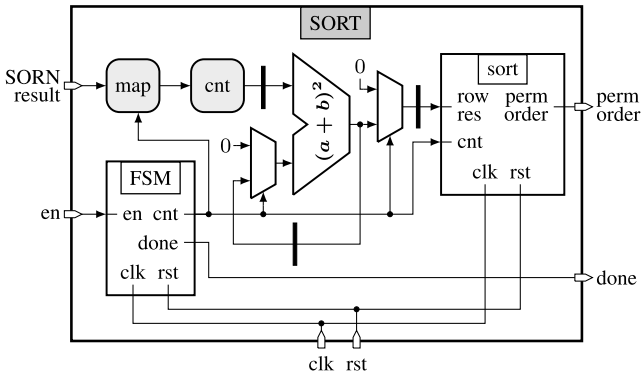

FIGURE 10. Sorting unit calculating the permutation order for the channel matrix and an unbalanced SD search tree based on the SORN preprocessing.

The RTL implementation of the approximate sorting algorithm is given in Fig. 10 and will be explained in the following: The result of the SORN preprocessor is a 256b vector with each bit denoting one of the possible solution vectors $\mathbf{x} \in \mathcal{S}^N$, either included ($\rightarrow$ 1) or excluded ($\rightarrow$ 0) by the preprocessor. The concatenation of all vectors $\mathbf{x}$ can be interpreted as a matrix $\mathbf{X} \in \mathbb{C}^{4\times 256}$, as depicted in Fig. 11. Per row of this matrix, the SORN results for one of the four symbols are mapped to a 64b signal which is passed to the next module. In Fig. 11 this is shown for the symbol $\frac{1}{\sqrt{2}}(1+j)$. This behavior results in 16 different combinations (4 rows, 4 symbols), which are selected according to a counter signal provided by a finite-state-machine (FSM).

The 64b mapped SORN result is passed to a counter which counts all the ones and then adds this value to the result from the previous iteration. The sum is squared and passed to a feedback loop to accumulate with the result from the next iteration. After every fourth iteration, the accumulated result corresponds to the approximated standard deviation of one row and is passed to the module which sorts the results of all rows in a descending order. After all iterations are completed, the final output permutation order can be calculated.

C. QR-DECOMPOSITION

With QRD the complex channel matrix $\mathbf{H} \in \mathbb{C}^{N\times N}$ can be split into an orthogonal matrix $\mathbf{Q} \in \mathbb{C}^{N\times N}$ and an upper triangular matrix $\mathbf{R} \in \mathbb{C}^{N\times N}$, which are required for the SD algorithm. For computing the decomposition orthogonal transformations like Gram-Schmidt, Householder Reflection or Givens Rotation can be applied [17].

In this implementation a complex Givens Rotation is used. The algorithm successively generates zero-elements below the main diagonal of the input matrix by multiplying with a complex rotation matrix:

$$\begin{bmatrix} 1 & \dots & 0 & 0 & \dots & 0 \\ \vdots & & \vdots & \vdots & & \vdots \\ 0 & \dots & c & s & \dots & 0 \\ 0 & \dots & -s^* & c^* & \dots & 0 \\ \vdots & & \vdots & \vdots & & \vdots \\ 0 & \dots & 0 & 0 & \dots & 1 \end{bmatrix} \begin{bmatrix} R_{11} & \dots & R_{1i} & R_{1j} & \dots & R_{1N} \\ \vdots & & \vdots & \vdots & & \vdots \\ 0 & \dots & R_{ii} & R_{ij} & \dots & R_{iN} \\ 0 & \dots & R_{ji} & R_{jj} & \dots & R_{jN} \\ \vdots & & \vdots & \vdots & & \vdots \\ 0 & \dots & R_{Ni} & R_{Nj} & \dots & R_{NN} \end{bmatrix}$$

$$= \begin{bmatrix} R_{11} & \dots & R_{1i} & R_{1j} & \dots & R_{1N} \\ \vdots & & \vdots & \vdots & & \vdots \\ 0 & \dots & R'_{ii} & R'_{ij} & \dots & R'_{iN} \\ 0 & \dots & 0 & R'_{jj} & \dots & R'_{jN} \\ \vdots & & \vdots & \vdots & & \vdots \\ 0 & \dots & R_{Ni} & R_{Nj} & \dots & R_{NN} \end{bmatrix} \tag{15}$$

The R'_{xx} entries are changed by the rotation; c^* and s^* represent the complex conjugates of c and s, respectively. In order to create the zero entry at position $\{j, i\}$, the elements of the rotation matrix are determined as follows:

$$c = \frac{R^*_{ii}}{\sqrt{|R^*_{ii}|^2 + |R^*_{ji}|^2}} \qquad s = \frac{R^*_{ji}}{\sqrt{|R^*_{ii}|^2 + |R^*_{ji}|^2}} \tag{16}$$

After every iteration the matrix $\mathbf{R}$ is updated according to the rotation result and c and s are recalculated for the next rotation until all lower diagonal elements are zero. The $\mathbf{Q}$ matrix is the product of all intermediate rotation matrices [17].

The QRD architecture is composed of two different submodule types:

- The Givens Generate (GG) module calculates the entries of the rotation matrix c and s for the input values R_{ii} and R_{ji} according to Eq. (16). The inverse square root is implemented using the iterative Newton-Raphson (NR) method described in [31] with 3 iterations.
- The Givens Apply (GA) module performs the rotation by calculating the (sub-)matrix multiplication for the current rows $\{i, j\}$ of the matrix $\mathbf{R}$. Two versions of the GA module are implemented to compute either R'_{ik} or R'_{jk}:

$$\left\{ \begin{aligned} R'_{ik} &= c\,R_{ik} + s\,R_{jk} \\ R'_{jk} &= -s^*\,R_{ik} + c^*\,R_{jk} \end{aligned} \right\} \text{ for } k \in \{i, \dots, N\},\ i < j \tag{17}$$

$$\begin{array}{lc} \text{Symbol vectors:} & \frac{1}{\sqrt{2}} \begin{bmatrix} 1+j & -1+j & -1-j & 1-j & 1+j & \cdots & 1-j & 1+j & \cdots & 1-j & 1+j & \cdots & 1-j \\ 1+j & 1+j & 1+j & 1+j & -1+j & \cdots & 1-j & 1+j & \cdots & 1-j & 1+j & \cdots & 1-j \\ 1+j & 1+j & 1+j & 1+j & 1+j & \cdots & 1+j & -1+j & \cdots & 1-j & 1+j & \cdots & 1-j \\ 1+j & 1+j & 1+j & 1+j & 1+j & \cdots & 1+j & 1+j & \cdots & 1+j & -1+j & \cdots & 1-j \end{bmatrix} \\ \hline \text{SORN result:} & \begin{bmatrix} 1 & 0 & 1 & 0 & 0 & \cdots & 0 & 1 & \cdots & 0 & 1 & \cdots & 0 \end{bmatrix} \end{array}$$

FIGURE 11. Representation of the SORN preprocessing result and the corresponding symbol vectors x. Per row, the SORN result bits for the symbol $\frac{1}{\sqrt{2}}(1+j)$ are mapped to the next stage.

The matrix decomposition is performed iteratively. For the implemented 4×4 matrix size 6 global iterations are required, producing one 0 entry in $\mathbf{R}$ per iteration. In every iteration one GG module computes the rotation coefficients before the rotation is applied (GA). The design contains 11 GA modules and 2 complex multipliers to compute all new entries of $\mathbf{Q}$ and $\mathbf{R}$ per iteration in parallel. In addition, registers for storing the current rotation coefficients c and s, and the two matrices $\mathbf{Q}$ and $\mathbf{R}$ are implemented.

D. MATRIX-VECTOR-MULTIPLICATION

The matrix-vector-multiplication (MVM) module is responsible for calculating the SD input $\tilde{\mathbf{y}} = \mathbf{Q}^{\mathbf{H}}\mathbf{y}$, according to Eq. (6). It is composed of 16 complex multiplication and 14 complex addition/subtraction units, implemented as tree structure with one pipeline level, as well as registers for storing the output vector.

E. SPHERE DECODER

The SD algorithm solves the MLE problem element-wise by using a QRD as shown in Eq. (6)-(7) and described in Sec. III-A. While traversing the search tree, at every node the level-specific error $e(l)$ is calculated as described in Eq. (8). Those paths with the lowest error metric are followed first, and at the bottom level a decision about adapting the search radius r is made, before following the next path. In this work a standard SE-SD as well as a SORN-reduced version of the algorithm were implemented. In the following, the design of the standard SD will be described first. The adaptions made for the SORN-based version will be discussed afterwards.

1) SE-SD

The general behavior of the implemented SD architecture is depicted in Fig. 12. At every tree level l, the error metric is calculated for the possible values of $\mathbf{x}$, indexed by the counter value c_{node}. When all m errors are processed, they are sorted and compared with the global radius r in order to determine the next level. If the current level is the bottom one, the final error of the current path is compared to the radius r, and if $e < r$ the search radius is adapted. Before the next level nodes are processed, the counter c_{node} is reset.

The control path of the SE design is composed of an FSM managing the current level and level transitions. In addition, the design contains registers for storing the current errors and those of the previous levels, the counter values, the global radius and the already visited node counts. The datapath consists of the error calculation and comparison operations for sorting the errors and deciding about a radius adaption.

2) SORN-REDUCED SD

The main additional component for the SORN-based SD calculates the nodes that can be cut out of the search tree. As discussed in Sec. IV-B, the preprocessing result is a bit vector representing the included and excluded symbol vectors $\mathbf{x}$, which also correspond to the bottom level tree nodes of the search tree. Consequently, the discarded bottom nodes can

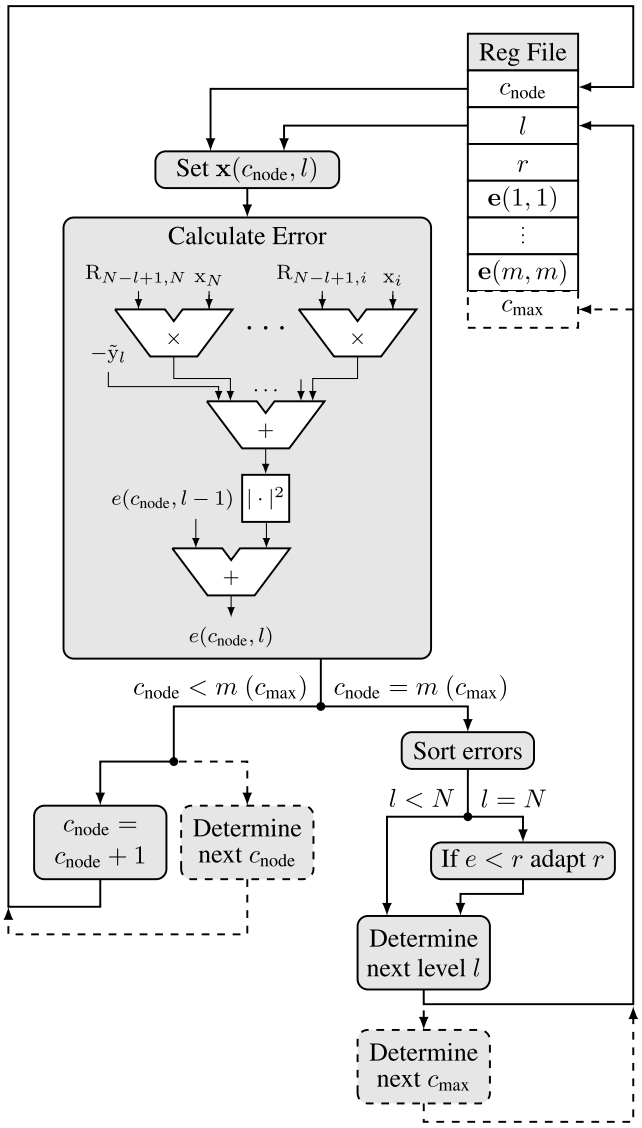

FIGURE 12. Behavior of the standard SE-SD. Additional steps and adaptions made for the SORN-reduced SD are displayed with dashed lines/blocks.

be directly taken from the preprocessing result. In order to delete the nodes of the higher tree levels, the bits from the lower levels are connected by an m dimensional OR-gate. With $\mathcal{N}_l(i)$ representing the i-th node at tree level l and $\mathcal{N}_N(i)$ corresponding to the bottom tree level nodes, the deletion of the higher level nodes can be calculated with

$$\mathcal{N}_l(i) = \bigvee_{j=m\times i}^{m(i+1)-1} \mathcal{N}_{l+1}(j) \tag{18}$$

with $i \in \{0, \ldots, m^l - 1\}$, $l \in \{1, \ldots, N-1\}$, $\bigvee$ as logical OR and the modulation number m.

The standard SE-SD calculates m error values per (sub-)tree level l before taking a decision for the next path. Due to the deletion of nodes through all tree levels, in the SORN-SD the number of error calculations per (sub-)tree

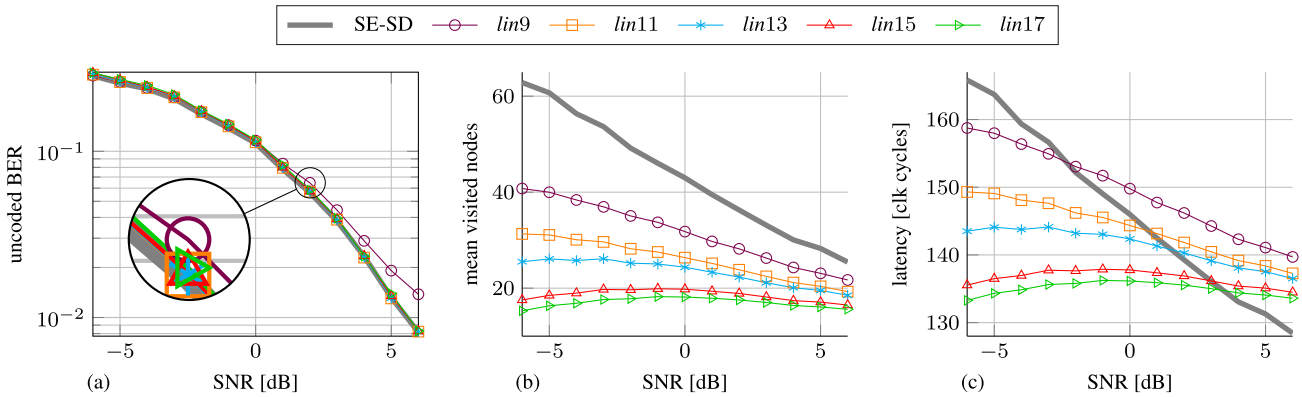

FIGURE 13. RTL simulation results for (a) uncoded BER, (b) mean visited nodes and (c) latency in required clock cycles; all for the hardware implementation of the SORN-based SD with preprocessing and QRD, and the standard SE-SD with QRD for 4 × 4 QPSK MIMO with 5×10^3 simulations and 16b FxD.

level is not fixed anymore. As shown in Fig. 12, after every error calculation the next node count c_{node} has to be determined based on the node deletion described in Eq. (18). Additionally, after the error calculation of one level is completed and the next level is determined, the number of nodes in the next subtree $c_{\max}$ has to be determined.

From the architectural perspective, and compared to the SE-SD design described above, the SORN-based SD implementation contains additional modules for determining the deleted nodes of all tree levels and calculating the next node during the error calculation process, as well as an additional register for the number of nodes $c_{\max}$.

F. TIMING BEHAVIOR

In this section the timing behavior by means of the required clock cycles for both the SE-SD and the SORN-SD is discussed. The standard SE-SD toplevel design is composed of a QRD, MVM and the SD itself, whereas the SORN-based SD additionally contains the SORN preprocessor and the sorting module, as depicted in Fig. 8. The three submodules appearing in both designs, QRD, MVM and SD, are implemented for a frequency of 100 MHz. The SORN preprocessor and the sorting module are driven by a frequency of 1 GHz in order to allow a fast processing of the exhaustive search performed within the preprocessor. The SORN-SD toplevel design contains a frequency divider with a factor of 10 to provide both frequencies.

The number of required clock cycles C for a complete detection for both SD designs at 100 MHz are given in the following:

$$C_{SE-SD} = \underbrace{C_{QRD}}_{=97} + \underbrace{C_{MVM}}_{=5} + C_{SD} = 102 + C_{SD} \tag{19}$$

$$\begin{aligned} C_{SORN-SD} &= \underbrace{C_{SORN}}_{=13.1} + \underbrace{C_{SORT}}_{=1.9} + \underbrace{C_{QRD}}_{=97} + \underbrace{C_{MVM}}_{=5} + C_{SD} \\ &= 117 + C_{SD} \end{aligned} \tag{20}$$

The latency of each submodule is hereby obtained as follows:

- C_{QRD}: The QRD performs 6 global iterations, each requiring 16 clock cycles (14 for the GG and 2 for the GA). Additionally 1 initial clock cycle is required, resulting in 97 cycles in total.
- C_{MVM}: For the MVM 1 of the 4 rows of the input matrix/vector is inserted into the tree structure per clock cycle. As the tree contains a pipeline level, in total 5 cycles are required.
- C_{SD}: Both SD versions require 1 clock cycle per visited node plus 1 initial cycle. The number of visited nodes is non-deterministic.
- C_{SORN}: The SORN module processes 256 possible solutions with two parallel solvers containing three pipeline levels, resulting in a total number of 131 cycles at 1 GHz or 13.1 cycles at 100 MHz.
- C_{SORT}: The sorting module requires 16 clock cycles for traversing the 4×4 matrix, plus one initial and two final cycles, resulting in 19 cycles at 1 GHz or 1.9 cycles at 100 MHz.

V. RESULTS

The implemented Schnorr-Euchner and SORN-based SD designs are evaluated in terms of RTL-simulations and an STMicroelectronis (STM) 28 nm CMOS technology synthesis. The evaluations cover both 16b and 32b FxD datapaths, as well as all different SORN datatypes from 9b to 17b.

A. RTL-SIMULATIONS

In Fig. 13 three different evaluations are given for the implemented detector architectures for a complex 4 × 4 MIMO system using QPSK modulation. When comparing the 16b and 32b FxD implementations no differences are visible.

Fig. 13a shows the uncoded BER after demodulating the detected symbols for the SE-SD and the SORN-based SD using the different SORN datatypes for preprocessing. Even though in Sec. III-B and Fig. 4c it was shown that the SORN preprocessor excludes the correct solution with a certain probability, almost no differences in the BER between the

TABLE 3. Post-Synthesis results for the SORN-based SD with Preprocessing and QRD, and the standard SE-SD with QRD; all for 4 × 4 QPSK MIMO, synthesized for 28 nm CMOS technology.

	Config.	Bitwith		Freq. [MHz]		total Area		partial Area [%]					Energy
		SORN	FxD	SORN	FxD	[μm²]	[kGE][a]	SORN	SORT	QRD	MVM	SD	[μW/MHz]
SORN-SD	C1	9	16	1000	100	154234	315	10.27	9.83	55.18	8.55	7.50	32.77
	C2	11	16	1000	100	159200	325	13.22	9.53	53.36	8.26	7.20	34.15
	C3	13	16	1000	100	165509	338	16.07	9.16	51.90	7.97	6.92	35.50
	C4	15	16	1000	100	170298	348	18.46	8.90	50.46	7.76	6.73	36.60
	C5	17	16	1000	100	172419	352	19.27	8.79	49.93	7.66	6.68	38.23
	C6	9	32	1000	100	424054	866	3.83	3.58	69.76	10.71	8.43	70.63
	C7	11	32	1000	100	430309	879	4.99	3.53	69.12	10.54	8.43	72.52
	C8	13	32	1000	100	432969	884	6.27	3.50	67.79	10.38	8.59	73.38
	C9	15	32	1000	100	442277	903	7.21	3.47	67.62	10.17	8.14	75.26
	C10	17	32	1000	100	441186	901	7.68	3.44	67.29	10.19	8.05	76.25
SE-SD	C11	-	16	-	100	109165	223	-	-	77.12	11.89	9.95	20.93
	C12	-	32	-	100	372897	762	-	-	78.28	11.92	9.06	59.87

[a] Gate Equivalents = Total area divided by the area of a 2-input NAND gate with lowest driver strength (0.4896 μm²)

SE- and the SORN-SD can be observed. Only for the 9b SORN datatype a worse BER performance for high SNR values occurs, caused by the lowest of all implemented resolutions.

The mean visited nodes for the different detector implementations are depicted in Fig. 13b. Compared to the software-based simulation results from Fig. 6c, a similar behavior can be observed: The SORN-based SD requires much less node visits than the SE-SD. The reduction depends on the implemented SORN datatype and the SNR. For negative SNRs a reduction of the visited nodes by up to 76% can be achieved. Another interesting aspect is the fact that the mean number of visited nodes is nearly stable over different SNRs for the higher bit SORN approaches, whereas for the SE-SD it is highly SNR dependent.

Since the SD is only one part of the architecture, in Fig. 13c the latency in terms of clock cycles for the complete design including preprocessing and decomposition is given. As described in Sec. IV-F, the SD is the only component with a variable latency, which is why the overall latency is a shifted version of the mean visited nodes. Even though the speedup of the SORN-based SD is reduced compared to Fig. 13b, an improvement over the SE-SD can still be observed. For negative SNRs the improvement is up to 20% for the 17b SORN-based approach, for 0 dB it is still 7%.

From the presented simulations it can be concluded that the SORN-SD approach is especially well suited for low SNR regions since the number of visited nodes and the latency are lower than for the SE-SD, without any loss of BER-performance. However, at some point towards positive SNR the SE-SD shows a lower latency than the SORN-based approach. Since the implemented SORN-SD is still able to behave like a conventional SD when switching off the SORN preprocessing, the best from both approaches can be combined with an adaptive, SNR-dependent preprocessing disabling.

B. SYNTHESIS

All implemented designs were synthesized for a 28 nm CMOS process from STM. Tab. 3 shows the synthesis results for the SORN-based SD for all combinations of FxD and SORN datapath widths, as well as both SE-SD versions. All designs were synthesized for frequencies of 100 MHz for the FxD and 1 GHz for the SORN components.

The total chip area is given in μm^2 and Gate Equivalents (GE), the latter is a technology independent measure where the total area is normalized by the area of a 2-input NAND gate with lowest driver strength. With increasing SORN bitwidth the total area increases by up to 12% for the 16b FxD designs (up to 4% for 32b FxD). A similar behavior occurs for the energy which is given in [μW/MHz]. Here an increase of up to 17% (8%) can be observed.

The partial area results show that the SORN preprocessor occupies about 10% to 19% (4% to 8%) of the whole design. The area of the sorting module is independent from the SORN bitwidth and requires about 9% (3.5%) of the chip area. The largest submodule is the QRD which utilizes 50% to 55% (67% to 70%). The SD itself is the smallest submodule and requires about 6.5% to 7.5% (8% to 8.5%) of the total chip area.

When comparing the SORN-based SD design with the SE-SD (both with QRD and MVM), the area increase is 41% to 58% for the 16b FxD and 14% to 18% for the 32b FxD design in total. The energy increases by 57 % to 83 % (18 % to 27 %). The main reason for the high energy increase is, besides the larger chip area, the high frequency which is applied for the SORN and sorting components.

C. SotA COMPARISONS

In order to classify the performance of the implemented detectors, Tab. 4 provides a comparison to SotA SDs and comparable architectures. All detectors were implemented for 4 × 4 MIMO systems. To allow a fair comparison the

TABLE 4. Comparison results for the implemented SE-SD and SORN-SD with SORN preprocessing (both without QRD and MVM) and reference architectures.

	[32]	[33]	[34]	[35]	[36]	This work	
Detector	ASE SD	SD ASIP	DF SD	K-Best ($k = 10$)	imbal. FSD	SE-SD	SORN-SD
Dimension	4×4	4×4	4×4	4×4	4×4	4×4	
Modulation	QPSK	16 QAM	16 QAM	64 QAM	64 QAM	QPSK	
Bitwidth	n.a.	22b FP	24b	n.a.	n.a.	16b	16b / 17b
Process / V_{dd}	90 nm / 1.0 V$^{(a)}$	65 nm / n.a.	45 nm / 0.9 V	65 nm / 1.3 V	65 nm / 1.2 V	28 nm / 0.9 V	
Preprocessing	included	-	-	not included	not included	-	included
BER-performance	suboptimal	quasi-ML	quasi-ML	suboptimal	suboptimal	quasi-ML	quasi-ML
Area [kGE]	153.9	30$^{(b)}$	51.2	298	88.2	25	149
Frequency [MHz]	108.7	300	435	833	165	100	100 / 1000
Power [mW]	28.75	8.51	9.56	280	102.7	0.39	7.46
norm. Power [mW]	29.90	8.51	24.55	238.6	102.7	1.61	30.79
Throughput [Mbps]	40	15.59	275.86	2000	1980	18.2	23.5
@SNR [dB]	0	n.a.	12	-	-	0	0

$^{(a)}$Typical V_{dd} for TSMC 90 nm Std Cell Library

$^{(b)}$Given area from [33] (47832 µm^2) divided by the area of a NAND2X1 gate (1.6 µm^2 for TSMC 65 nm)

results for the architectures implemented in this work are given without the QRD and MVM modules since the reference designs do not include these steps either. The SORN preprocessing and sorting, however, are included in the given results. Further, the area of the different designs is given in GE in order to allow a technology-independent comparison. The power consumption is normalized to a 65 nm technology with 1.2 V supply voltage (V_{dd}) [20]:

$$\text{norm. Power} = \text{Power} \times \left(\frac{1.2\,\text{V}}{V_{dd}}\right)^2 \times \left(\frac{65\,\text{nm}}{\text{Tech.}}\right) \tag{21}$$

The throughput is obtained as

$$\text{throughput} = \frac{N \times \log_2(m)}{C} \times f \quad [\text{bit/s}] \tag{22}$$

with the MIMO dimension N, the symbol bitwidth $\log_2(m)$ with modulation number m, the number of required clock cycles C and the clock frequency f [18].

The first observation from the comparison is that the implemented SE-SD achieves a good throughput-area-ratio and low power consumption, compared to the two SD reference designs [32] and [33]. For the SORN-SD the throughput is further increased while power and area are at a moderate level. Although the area and power increase between SE and SORN-SD seem to be quite high for this comparison, it has to be considered that these results do not include the QRD module, which takes the major part of both designs area and power consumption, as discussed in the previous section V-B. When compared to the reference design from [32], the SORN-SD shows similar hardware results with a lower throughput, but achieves a better (quasi-ML) BER-performance. Regarding reference [34], area and power consumption of the implemented SE-SD and SORN-SD are on a comparable level, whereas the throughput is lower by an order of magnitude. However, it has to be considered that for [34] the throughput is given for an SNR of 12 dB whereas the implemented designs are evaluated for lower SNR regions.

TABLE 5. Comparison of the implemented and reference 4 × 4 complex QRD architectures.

	[37]	[38]	[39]	This work
Algorithm	SVD/ QRD	QRD/ SQRD	QRD	QRD
Bitwidth	n.a.	n.a.	13b	16b
Process / V_{dd}	90 nm / 1.0 V$^{(a)}$	90 nm / 1.0 V$^{(a)}$	65 nm / 1.0 V	28 nm / 0.9 V
Area [kGE]	452	375	378	176
Frequency [MHz]	143	220	72	100
Power [mW]	93.54	140	127	1.53
norm. Power [mW]	97.28	145.6	182.9	6.31
Throughput [matr. / s]	35.75 M	44 M	72 M	1.03 M

$^{(a)}$Typical V_{dd} for TSMC 90 nm Std Cell Library

In comparison with the fixed-complexity approaches K-Best [35] and FSD [36] the achieved throughput of the SD and SORN-SD approaches is about two orders of magnitude lower. However, the parallelization and pipelining for these two designs leads to higher area demands and power consumption, also in comparison to the proposed SORN-SD. Additionally, it has to be considered that the results for both implementations [35] and [36] do not include the required preprocessing supplementary to a QRD, as discussed in Sec. III-D, nor do they achieve an optimal BER-performance.

1) QRD

The results of the implemented QRD module are compared to reference implementations in Tab. 5. It can be seen that the QRD architecture implemented in this work provides a low-complexity and low-power approach with a

lower throughput than the reference architectures. This low throughput results from the iterative design approach where the parameter for the Givens Rotation are recalculated using the NR method for every iteration. The reference designs are implemented as systolic array [38], [39] or massively parallel CORDIC processors [37] in order to enhance the throughput, and resulting in a high complexity and power consumption.

If a more complex and higher throughput QRD design is used within the presented SORN-SD approach, the benefit of the SORN preprocessing step would increase accordingly, since the QRD requires at least half of the chip area and the highest amount of required clock cycles. A further increase of the QRD area would minimize the relative complexity overhead that is introduced by the SORN preprocessing. Furthermore, a lower QRD latency would relate to a higher impact of the reduced number of visited nodes for the SORN-SD and a higher overall latency reduction compared to the SE-SD.

VI. CONCLUSION

Sphere Decoding for wireless MIMO communication can be accelerated by introducing SORN-based preprocessing which deletes nodes from the SD search tree and effectively reduces the latency by means of visited nodes. For the presented evaluation the mean number of visited nodes within the adapted SD can be significantly reduced by up to 76% for negative SNRs, compared to a SotA SE-SD. A hardware implementation comprising a SORN preprocessor, a sorting module, a QRD and an adapted SD show an overall, SNR-dependent latency reduction of up to 20%, compared to a standard SE-SD with QRD. Even though the SE-SD performs better for high SNRs, the presented SORN-based design is not restricted to low SNR regions since the detector is capable of behaving like a SE-SD by switching off the preprocessing. With this feature a very flexible design showing best performance in all SNR regions can be achieved.

The area and energy of the SORN-SD increase by up to 58% and 83% for the presented 16b FxD implementation, and by up to 18% and 27% for 32b FxD, compared to the SE-SD. The energy increase is hereby mainly caused by the high frequency which is used for the SORN components. Even though the SORN preprocessing introduces an area and energy overhead compared to the SE design, comparisons to SotA detectors show that this overhead is still on a low level and would not have a high impact in a complex System-on-Chip (SoC) architecture. Additionally, the presented implementation utilizes a comparatively slow and low complex QRD. When a faster and more area- and energy-demanding decomposition implementation is used, the latency improvement of the SORN-SD will be further increased while the relative complexity and energy overhead will decrease.

This work shows how SORN arithmetic can be utilized to effectively reduce the complexity of the SD algorithm. For future work SORN-based preprocessing can be investigated for different MIMO detection approaches as well as for optimization problems in other domains of digital signal processing.

REFERENCES

[1] J.-M. Muller, N. Brunie, F. de Dinechin, C.-P. Jeannerod, M. Joldes, V. Lefèvre, G. Melquiond, N. Revol, and S. Torres, *Handbook of Floating-Point Arithmetic*, 2nd ed. Boston, MA, USA: Birkhäuser, 2018.

[2] D. Goldberg, "Computer arithmetic," in *Computer Architecture: A Quantitative Approach*, D. A. Patterson and J. L. Hennessy, Eds. San Mateo, CA, USA: Morgan Kaufmann, 1990.

[3] *IEEE Standard for Floating-Point Arithmetic*, IEEE Standard 754-2019 (Revision of IEEE 754-2008), 2019, pp. 1–84.

[4] M. Chugh and B. Parhami, "Logarithmic arithmetic as an alternative to floating-point: A review," in *Proc. Asilomar Conf. Signals, Syst. Comput.*, 2013, pp. 1139–1143.

[5] P. Lindstrom, S. Lloyd, and J. Hittinger, "Universal coding of the reals: Alternatives to IEEE floating point," in *Proc. Conf. Next Gener. Arithmetic (CoNGA)*. New York, NY, USA: ACM, 2018, pp. 1–14.

[6] J. L. Gustafson, *The End of Error: Unum Computing*. Boca Raton, FL, USA: CRC Press, 2015.

[7] J. L. Gustafson and I. T. Yonemoto, "Beating floating point at its own game: Posit arithmetic," *Supercomput. Frontiers Innov.*, vol. 4, no. 2, pp. 71–86, 2017.

[8] J. Gustafson, "A radical approach to computation with real numbers," *Supercomput. Frontiers Innov.*, vol. 3, no. 2, pp. 38–53, 2016.

[9] E. G. Larsson, "MIMO detection methods: How they work [lecture notes]," *IEEE Signal Process. Mag.*, vol. 26, no. 3, pp. 91–95, May 2009.

[10] M. Bärthel, P. Seidel, J. Rust, and S. Paul, "SORN arithmetic for MIMO symbol detection–exploration of the type-2 unum format," in *Proc. 17th IEEE Int. New Circuits Syst. Conf. (NEWCAS)*, Jun. 2019, pp. 1–4.

[11] M. Bärthel, J. Rust, and S. Paul, "Application-specific analysis of different SORN datatypes for Unum type-2-based arithmetic," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, Oct. 2020, pp. 1–5.

[12] S. Knobbe, M. Bärthel, S. Paul, and J. Rust, "Complexity reduction for sphere decoding using Unum-type-II-based SORN-arithmetic," in *Proc. 9th Int. Conf. Modern Circuits Syst. Technol. (MOCAST)*, Sep. 2020, pp. 1–4.

[13] J.-M. Muller, "On the definition of ulp(x)," INRIA, Le Chesnay-Rocquencourt, France, Tech. Rep. RR-5504, LIP RR-2005-09 (inria-00070503), 2005. [Online]. Available: https://hal.inria.fr/inria-00070503/document

[14] F. Rusek, D. Persson, B. K. Lau, E. G. Larsson, T. L. Marzetta, and F. Tufvesson, "Scaling up MIMO: Opportunities and challenges with very large arrays," *IEEE Signal Process. Mag.*, vol. 30, no. 1, pp. 40–60, Jan. 2013.

[15] S. Yang and L. Hanzo, "Fifty years of MIMO detection: The road to large-scale MIMOs," *IEEE Commun. Surveys Tuts.*, vol. 17, no. 4, pp. 1941–1988, Sep. 2015.

[16] B. Hassibi and H. Vikalo, "On the sphere-decoding algorithm I. Expected complexity," *IEEE Trans. Signal Process.*, vol. 53, no. 8, pp. 2806–2818, Aug. 2005.

[17] G. H. Golub and C. F. Van Loan, *Matrix Computations*, 3rd ed. Baltimore, MD, USA: Johns Hopkins Univ. Press, 1996.

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

[19] C. P. Schnorr and M. Euchner, "Lattice basis reduction: Improved practical algorithms and solving subset sum problems," *Math. Program.*, vol. 66, nos. 1–3, pp. 181–199, Aug. 1994.

[20] I. A. Bello, B. Halak, M. El-Hajjar, and M. Zwolinski, "A survey of VLSI implementations of tree search algorithms for MIMO detection," *Circuits, Syst., Signal Process.*, vol. 35, no. 10, pp. 3644–3674, Oct. 2016.

[21] D. Wübben, J. Rinas, R. Boehnke, V. Kuehn, and K. Kammeyer, "Efficient algorithm for detecting layered space-time codes," in *Proc. 4th Int. ITG Conf. Source Channel Coding (SCC)*, Jan. 2002, p. 1.

[22] C. Studer, A. Burg, and H. Bolcskei, "Soft-output sphere decoding: Algorithms and VLSI implementation," *IEEE J. Sel. Areas Commun.*, vol. 26, no. 2, pp. 290–300, Feb. 2008.

[23] A. Chauhan and R. Mehra, "Analysis of QR decomposition for MIMO systems," in *Proc. Int. Conf. Electron. Syst., Signal Process. Comput. Technol.*, Jan. 2014, pp. 69–73.

[24] G. L. Nazar, C. Gimmler, and N. Wehn, "Implementation comparisons of the QR decomposition for MIMO detection," in *Proc. 23rd Symp. Integr. Circuits Syst. Design (SBCCI)*. New York, NY, USA: ACM, 2010, pp. 210–214, doi: 10.1145/1854153.1854204.

[25] K.-W. Wong, C.-Y. Tsui, R. S.-K. Cheng, and W.-H. Mow, "A VLSI architecture of a K-best lattice decoding algorithm for MIMO channels," in *Proc. IEEE Int. Symp. Circuits Syst.*, vol. 3, May 2002, p. 3.
[26] H. Kim, J. Park, H. Lee, and J. Kim, "Near-ML MIMO detection algorithm with LR-aided fixed-complexity tree searching," *IEEE Commun. Lett.*, vol. 18, no. 12, pp. 2221–2224, Dec. 2014.
[27] L. Barbero and J. Thompson, "Fixing the complexity of the sphere decoder for MIMO detection," *IEEE Trans. Wireless Commun.*, vol. 7, no. 6, pp. 2131–2142, Jun. 2008.
[28] J. Fink, S. Roger, A. Gonzalez, V. Almenar, and V. M. Garciay, "Complexity assessment of sphere decoding methods for MIMO detection," in *Proc. IEEE Int. Symp. Signal Process. Inf. Technol. (ISSPIT)*, Dec. 2009, pp. 9–14.
[29] M. Li, B. Bougard, E. E. Lopez, A. Bourdoux, D. Novo, L. Van Der Perre, and F. Catthoor, "Selective spanning with fast enumeration: A near maximum-likelihood MIMO detector designed for parallel programmable baseband architectures," in *Proc. IEEE Int. Conf. Commun.*, 2008, pp. 737–741.
[30] J. Rust, M. Bärthel, P. Seidel, and S. Paul, "A hardware generator for SORN arithmetic," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 39, no. 12, pp. 4842–4853, Dec. 2020.
[31] S. Aslan, E. Oruklu, and J. Saniie, "Realization of area efficient QR factorization using unified division, square root, and inverse square root hardware," in *Proc. IEEE Int. Conf. Electro/Inf. Technol.*, Jun. 2009, pp. 245–250.
[32] K.-J. Yang, S.-H. Tsai, R.-C. Chang, Y.-C. Chen, and G. C.-H. Chuang, "VLSI implementation of a low complexity 4×4 MIMO sphere decoder with table enumeration," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, May 2013, pp. 2167–2170.
[33] J. Rust, C. Osewold, and S. Paul, "Implementation of a low power low complexity ASIP for various sphere decoding algorithms," in *Proc. 17th Eur. Wireless Sustain. Wireless Technol.*, 2011, pp. 1–6.
[34] G. Georgis, K. Nikitopoulos, and K. Jamieson, "Geosphere: An exact depth-first sphere decoder architecture scalable to very dense constellations," *IEEE Access*, vol. 5, pp. 4233–4249, 2017.
[35] D. Patel, V. Smolyakov, M. Shabany, and P. G. Gulak, "VLSI implementation of a WiMAX/LTE compliant low-complexity high-throughput soft-output K-best MIMO detector," in *Proc. IEEE Int. Symp. Circuits Syst.*, May 2010, pp. 593–596.
[36] L. Liu, J. Lofgren, and P. Nilsson, "Area-efficient configurable high-throughput signal detector supporting multiple MIMO modes," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 59, no. 9, pp. 2085–2096, Sep. 2012.
[37] Y.-T. Hwang, K.-T. Chen, and C.-K. Wu, "A high throughput unified SVD/QRD precoder design for MIMO OFDM systems," in *Proc. IEEE Int. Conf. Digit. Signal Process. (DSP)*, Jul. 2015, pp. 1148–1151.
[38] T.-T. Tseng and C.-A. Shen, "The VLSI architecture of a highly efficient configurable pre-processor for MIMO detections," in *Proc. IEEE 36th Int. Perform. Comput. Commun. Conf. (IPCCC)*, Dec. 2017, pp. 1–5.
[39] R. Gangarajaiah, L. Liu, M. Stala, P. Nilsson, and O. Edfors, "A high-speed QR decomposition processor for carrier-aggregated LTE—A downlink systems," in *Proc. Eur. Conf. Circuit Theory Design (ECCTD)*, Sep. 2013, pp. 1–4.

MORITZ BÄRTHEL received the B.Sc. and M.Sc. degrees in electrical engineering and information technology from the University of Bremen, Bremen, Germany, in 2015 and 2018, respectively, where he is currently pursuing the Ph.D. degree with the Communication Electronics Work Group, Institute of Electrodynamics and Microelectronics. Since then, he has been with the Communication Electronics Work Group, Institute of Electrodynamics and Microelectronics, University of Bremen, as a Research Associate. His special research interests include innovative digital number formats, and digital arithmetic and hardware implementations of wireless communication systems.

SIMON KNOBBE received the M.Sc. degree in electrical engineering and information technology from the University of Bremen, Bremen, Germany, in 2020, where he is currently pursuing the Ph.D. degree with the Department of Communications Engineering. Since 2020, he has been with the Department of Communications Engineering, University of Bremen, as a Research Associate. His research interest includes 5G implementations for mobile health care systems with respect to ultrareliable low latency communication.

JOCHEN RUST received the Dipl.-Ing. degree in electrical engineering and computer science from the University of Hanover, Hanover, Germany, in 2007, and the Ph.D. degree from the Communication Electronics Work Group (ITEM), Institute of Electrodynamics and Microelectronics, University of Bremen, Bremen, Germany, in 2014. Since April 2020, he has been working with the Department of Pre-Development, DSI Aerospace Technology GmbH. His research interests include high-performance computing for space applications, radiation-aware, fault-tolerant signal processing and approximate computing, and NTN communication systems. In 2007, he received the IBM Innovation Award for contributions to tree-grammar-based netlist verification.

STEFFEN PAUL received the Dipl.-Ing. degree in electrical engineering from the Technical University Dresden, Dresden, Germany, and the Technical University of Munich, Munich, Germany, in 1989, and the Dr.-Ing. degree in 1993. From 1989 to 1997, he worked with the Institute of Network Theory and Circuit Design, Technical University of Munich. He was a Postdoctoral Fellow with the Department of Electrical Engineering and Computer Sciences, University of California at Berkeley, Berkeley, CA, USA, from 1994 to 1995. From 1997 to 2007, he worked with Infineon Technologies (formerly, Siemens Semiconductor Group), Neubiberg, Germany, in the areas of memory design, xDSL, and UMTS concept engineering. In 2007, he joined the University of Bremen, Bremen, Germany, where he has been a Full Professor for electromagnetic theory and microelectronic systems. His research interests include signal processing for wireless communications, VLSI implementation of signal processing algorithms, and low power digital design.