J Sign Process Syst (2013) 70:125–143
DOI 10.1007/s11265-012-0709-z

VLSI Architecture for MIMO Soft-Input Soft-Output Sphere Detection

Esther P. Adeva · Tobias Seifert · Gerhard Fettweis

Received: 20 April 2012 / Revised: 1 October 2012 / Accepted: 4 October 2012 / Published online: 18 December 2012

Abstract Achieving good detection performance while incurring low complexity is known to be one of the major challenges in multiple-input multiple-output (MIMO) communications based on spatial multiplexing. The tuple search detector (TSD) was recently introduced, improving this trade-off with regard to other tree-search-based algorithms (e.g. single tree search or list sphere detector). Motivated by the tremendous gain achievable through the turbo principle and based on a previously developed soft-output (SO) TSD implementation, this work presents the first soft-input soft-output (SISO) TSD realization, scalable in constellation size and number of antennas and mapped to a highly parallel and pipelined VLSI architecture. The proposed SISO-TSD VLSI realization is instantiated for 4×4 MIMO transmission and 64-QAM constellation in 65-nm CMOS technology. For a given BER↔complexity trade-off, the throughput ranges from 57.3 Mbps (iterative detection-decoding with 3 iterations) to 403.6 Mbps (non-iterative detection-decoding) at a clock frequency of 454 MHz. The BER↔complexity trade-off can be moreover adjusted according to transmission conditions, reaching >1 Gbps in high SNR scenarios. A silicon area of 0.14 mm^2 (97.7 kGEs) is occupied by the SISO-TSD core, reporting low power dissipation (58.2 mW – 73.9 mW) under typical case operating conditions. The proposed detector implementation achieves hence high throughput with reasonable hardware complexity, representing a very competitive strategy with regard to relevant state-of-the-art realizations.

Keywords MIMO · Iterative detection-decoding · Soft-input soft-output (SISO) detection · Sphere detection · Tuple search detector · SIMD · Pipeline-interleaving · VLSI architecture

E. P. Adeva (✉) · T. Seifert · G. Fettweis
Vodafone Chair Mobile Communications Systems,
Technische Universität Dresden, Dresden, Germany
e-mail: esther.perez@ifn.et.tu-dresden.de

T. Seifert
e-mail: tobias.seifert@ifn.et.tu-dresden.de

G. Fettweis
e-mail: gerhard.fettweis@ifn.et.tu-dresden.de

1 Introduction

Future mobile communication systems will make use of multiple-input multiple-output (MIMO) techniques in combination with high constellation orders to enhance spectral efficiency. The turbo principle (i.e. iterative detection-decoding) is in this regard foreseen as a strategy to approach the full potential of MIMO. Additionally, transmission of spatially multiplexed data streams allows increasing data rates as well as diversity. As it is widely known, in such systems the potential search space grows exponentially with increasing number of transmit antennas (N_T) and constellation size (Q). As a consequence, the inherent high detection complexity of most tree-search-based detection strategies represents a limiting factor towards efficient detector implementation. Low-complex detection strategies provide poor bit error rate (BER) performance (e.g. linear detector, successive/parallel interference cancelation—SIC/PIC detector, ...), whereas implementations achieving high BER performance present unusable high complexity (e.g. search approaches like unclipped single tree search sphere

detector—STS-SD). Besides the mentioned difficulties concerning complexity, most of today's communication standards define several transmission modes (e.g. up to 8×8 MIMO and up to 64-QAM in Long Term Evolution Advanced—LTE-A [1, 2]), thus becoming flexibility and scalability of detection approaches an additional challenge to be addressed. The tuple search detector (TSD) introduced in [17] has demonstrated to outperform the complexity↔BER-performance trade-off of comparable tree search detection strategies like STS [25] or list sphere detection (LSD) [31]. Additionally, TSD's search complexity presents a roughly linear trend with N_T and Q [4], hence becoming especially well suited for high-order transmission scenarios ($N_T \geq 4$, $Q > 16$). In contrast to *breadth-first* approaches presenting fixed search complexity, the complexity of the so-called *depth-first* detection algorithms (like STS, LSD and TSD) is variable. This fact represents an additional challenge towards practical realizations, since the inherent irregular and data-dependent control flow typically frustrates efficient algorithm parallelization, thus considerably limitig the achievable throughput enhancement. In this work, the first soft-input soft-output (SISO) TSD realization capable of reaching LTE-A data rates is presented and compared to a previously developed soft-output (SO) TSD implementation [3]. The turbo principle [10] allows significant further improvement of BER performance, but it also generally leads to further complexity increase resulting from the multiple detection-decoding runs per received symbol. Both benefit and cost of processing *a-priori* information from channel decoder will be hence evinced throughout this work.

The considered communications system model is introduced in Section 2, followed by the description of the complexity-reduced SISO-TSD algorithm (Section 3) representing the basis of our implementation. In order to minimize the required resources and speed up the computations, fixed-point arithmetic is utilized, as described in Section 4. Analysis of the computational complexity and the critical path is presented in Section 5. Considered parallelism approaches for further throughput enhancement are detailed in Section 6. In Section 7 the processor architecture is presented and relevant implementation challenges are introduced. Corresponding VLSI implementation results are presented in Section 8 and discussed in Section 9. As summarized in Section 10, low-complexity, scalability and high throughput (up to 1 Gbps)[1] characterize the proposed SISO-TSD implementation, resulting in a very competitive detection strategy with regard to relevant state-of-the-art realizations.

[1] At 454 MHz, for 4×4 transmission and 64-QAM in high SNR scenarios, non-iterative case.

2 System Model

Throughout this paper, we consider a $N_T \times N_R$ MIMO system based on a bit-interleaved coded modulation (BICM) transmission strategy with N_T transmit and N_R receive antennas, as depicted in Fig. 1. A vector $\mathbf{u}$ of i.i.d. information bits is encoded by the outer channel encoder with rate R. The resulting stream of vectors $\mathbf{c}'$ is bit-interleaved with a random interleaver and portioned into blocks $\mathbf{c}$ of $N_T \cdot L$ bits, where L denotes the number of bits per transmit symbol. For the transmission, the corresponding bits $\mathbf{c} \in \mathscr{C}$, covered in the set of permitted bit vectors, are mapped (e.g. Gray mapping) onto complex constellation symbols $\mathbf{x}(\mathbf{c}) = [x_0, ...x_{N_T-1}]^T = map(\mathbf{c})$ with $x_i \in \mathscr{X}$, being $\mathscr{X}$ the set of valid transmit symbols x_i with cardinality $\#\mathscr{X} = \#\mathscr{C} = 2^L = Q$. The transmit energy is normalized so that $\mathscr{E}\{\mathbf{x}\mathbf{x}^H\} = E_S/N_T\mathbf{I}$, where E_s represents the average transmit energy of the transmitter.

Regarding the transmission, we consider an uncorrelated, flat fading channel and an additive noise vector $\mathbf{n} \in \mathbb{C}^{N_R \times 1}$ at the receiver with complex components of zero mean i.i.d. gaussian random variables of variance $N_0/2$ per real dimension ($\mathscr{E}\{\mathbf{n}\mathbf{n}^H\} = N_0\mathbf{I}$). The considered passive channel is represented by $\mathbf{H} \in \mathbb{C}^{N_R \times N_T}$ and is assumed to be perfectly known at the receiver. The received signal $\mathbf{y}$ is therefore given by

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

and the signal-to-noise-ratio ($SNR = E_s/N_0$) at the receiver applied to the energy of one information bit can be stated as $E_b/N_0 = E_s N_R/N_0 N_T L R$.

At the receiver side, the detection process is carried out by a complex-valued TSD algorithm in conjunction with a PCCC (Parallel Concatenated Convolutional Code) channel decoder, which can operate in iterative detection-decoding fashion. In order to ensure comparability of results, a setup equivalent to the one used in e.g. [11, 32] has been used for our simulations.[2]

[2] Channel decoder is BCJR-based (Bahl, Cocke, Jelinek and Raviv) and uses $(7_R, 5)$ convolutional codes, 8 internal iterations and rate 1/2. Information block size of 9216 bits (including tail bits) and Gray mapping are used.

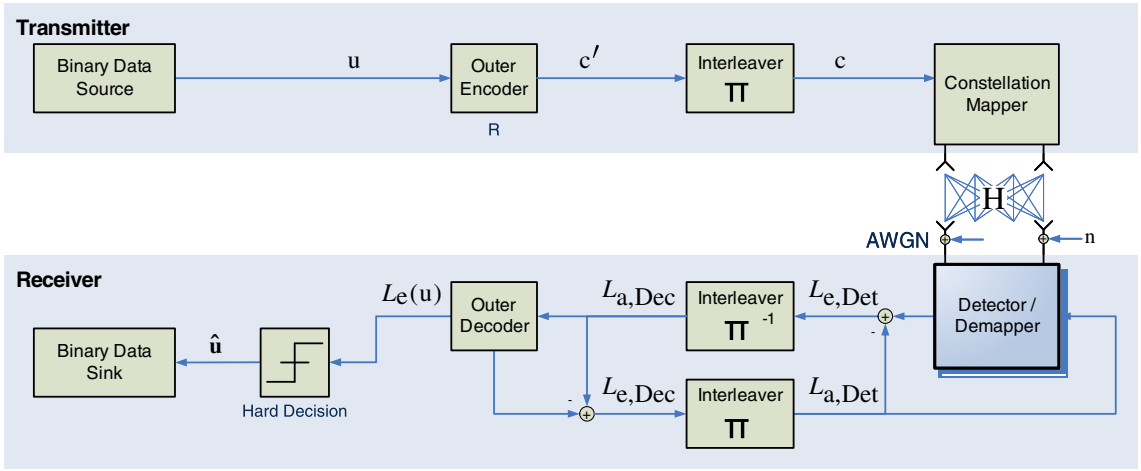


Figure 1 Communications system model with BICM transmitter and iterative receiver.

3 Complexity-Reduced Tuple Search MIMO Detection

3.1 Fundamentals

Task of the focused detector is the determination of the bits c most likely sent as well as of reliability information for these bits. This can be accomplished by calculating log-likelihood ratios (L-values):

$$L\left(c_{m,l}|\mathbf{y}\right) = \ln\left(\frac{P\left(c_{m,l}=+1|\mathbf{y}\right)}{P\left(c_{m,l}=-1|\mathbf{y}\right)}\right) \approx -\frac{1}{N_0}\min_{\mathbf{c}|c_{m,l}=+1}\{\lambda_0\} + \frac{1}{N_0}\min_{\mathbf{c}|c_{m,l}=-1}\{\lambda_0\}, \tag{1}$$

where Eq. 1 results from application of the max-log approximation [23]. The l-th bit of a symbol sent by the m-th antenna is represented by $c_{m,l}$ and

$$\lambda_0\left(\mathbf{y},\mathbf{c},\mathbf{L}_\mathrm{a}\right) = \left\|\mathbf{y}-\mathbf{H}\hat{\mathbf{x}}(\mathbf{c})\right\|^2 - \frac{N_0}{2}\sum_{i=0}^{N_\mathrm{T}-1}\sum_{j=0}^{L-1} c_{i,j}L_\mathrm{a}(c_{i,j}) \tag{2}$$

represents the metric for a set of received symbols $\mathbf{y}$, a given $\mathbf{c}$ and the *a-priori* knowledge $\mathbf{L}_\mathrm{a}$. $\hat{\mathbf{x}}$ corresponds to a possible transmitted symbol. Consequently, besides the most likely sent symbol $\arg\min_{\hat{\mathbf{x}}(\mathbf{c})|\mathbf{c}\in\mathscr{C}}\{\lambda_0\}$ (i.e. the detection hypothesis) and its corresponding metric $\lambda_0(\mathbf{c}^{ML})$, the detector has to determine also the counter-hypotheses $\arg\min_{\hat{\mathbf{x}}(\mathbf{c})|\mathbf{c}\in\mathscr{C},c_{m,l}\neq c_{m,l}^{ML}}\{\lambda_0\}$ and their metrics for each bit.

3.2 Tree Search Basics

Since brute force (full *Max-Log A Posteriori* Probability—APP) detection of Eq. 1 is known to be of exponentially growing computational complexity with the number of transmit antennas and order of the constellation, several close to optimal detection approaches have been lately proposed, some of the most promising are based on tree search strategies. Transforming the detection problem is permitted by the QR-decomposition (QRD) of $\mathbf{H}=\mathbf{QR}$, where $\mathbf{Q}$ is unitary and $\mathbf{R}$ an upper triangular matrix [30]. With modified received symbols $\mathbf{y}'=\mathbf{Q}^\mathrm{H}\mathbf{y}$, determination of the squared Euclidean distance

$$\left\|\mathbf{y}'-\mathbf{R}\hat{\mathbf{x}}(\mathbf{c})\right\|^2 \tag{3}$$

can be interpreted as a tree search. Resulting from this, λ_0 can be recursively calculated through the layered partial metric

$$\lambda_i = \underbrace{\lambda_{i+1}}_{\substack{\text{metric from}\\ \text{already estimated}\\ \text{symbols}}} + \Big|\underbrace{y_i''}_{\substack{\text{interference reduced}\\ \text{symbol}}} - r_{ii}\hat{x}_i\Big|^2 - \underbrace{\frac{N_0}{2}\sum_{j=0}^{L-1} c_{i,j}L_\mathrm{a}(c_{i,j})}_{\substack{\lambda_\mathrm{a}(\hat{x}_i)\\ (\textit{a-priori}\text{ information})}}, \tag{4}$$

$$y_i'' = y_i' - \sum_{j=i+1}^{N_\mathrm{T}-1} r_{ij}\hat{x}_j. \tag{5}$$

It should be noticed that $\lambda_\mathrm{a}(\hat{x}_i)$ contribution may increase or decrease λ_i, depending on both $c_{i,j}$ and the sign of $L_\mathrm{a}(c_{i,j})$. This may lead to exploration of unfavorable nodes during the tree search as well as to exclusion of favorable ones. In order to avoid this effect,

monotonously increasing λ_i is considered by redefining λ_a [19] as

$$\lambda_a(\hat{x}_i) = -\sum_{j=0}^{L-1}\left(\left|L_a(c_{i,j})\right| - c_{i,j}L_a(c_{i,j})\right) = -2\sum_{\substack{j=0 \\ \text{with } c_{i,j}\neq \text{sign}(L_a(c_{i,j}))}}^{L-1}|L_a(c_{i,j})|. \quad (6)$$

The search is carried out in *depth-first* fashion, successively extending the selected nodes by analyzing their child nodes. As introduced in [20] and described in Section 3.5, a regularized control flow and the parallel calculation of sibling parent nodes as well as of leaf nodes permit a so called one-node-per-cycle implementation [6]. Based on this, the average number of node extensions $\sharp n$ performed in the detection is taken as search complexity measure throughout this paper.

3.3 Tuple Search and Bit-Specific Candidate Determination

Computing the L-values in Eq. 1 requires the determination of a detection hypothesis and all counter-hypotheses as described in Section 3.1. Explicit search for all the needed minimums leads to impractically high $\sharp n$ [12]. Therefore, instead of searching all possible minima, TSD [17] searches a subset of T most likely leaves, similarly to the LSD approach. The metrics λ_0 of these leaves are stored in a search tuple $\mathscr{T} := \{\lambda_0(\mathbf{c}_1), \lambda_0(\mathbf{c}_2), \ldots, \lambda_0(\mathbf{c}_T)\}$, defining the sphere radius as the maximum metric in the tuple:

$$R = \max_{\mathbf{c}_t|\mathbf{c}_t\in\mathscr{T}}\left\{\lambda_{0,t}\right\}. \quad (7)$$

The tuple search is additionally combined with separated bit-specific storage of information for the L-value calculation [17]. The resulting TSD achieves much better BER performance than LSD, at a significantly reduced $\sharp n$ compared to STS detection. Additionally, adjustment of $\sharp n \leftrightarrow$BER-performance trade-off is enabled by varying the size of the tuple T.

3.4 Complexity Reduction Approaches

Strategies like sorted QR decomposition (SQRD) [30], MMSE preprocessing [32] as well as sphere and L-values clipping [9] are widely applied approaches towards $\sharp n$ reduction. Novel approaches like search sequence determination (SSD) [18] and metric estimation (ME) [4] contribute to further reduce the computational complexity, as described in the following.

3.4.1 Search Sequence Determination (SSD)

The SSD strategy introduced in [18] replaces the costly metric calculations in Eq. 4 and sorting operations required by SE enumeration by few inexpensive basic operations. The SSD approach is based on geometrical position analysis relative to reference nodes x^r, determined by

$$x^r = \lfloor y_i''' \rceil = \lfloor \frac{y_i''}{r_{ii}} \rceil \quad (8)$$

(with $\lfloor\cdot\rceil$ representing a rounding operation to the closest constellation symbol). Resulting enumeration is defined by fixed sequences which are mapped to constellation symbols during the detection, based on the relative position between y_i''' and x^r. The computational complexity of this strategy can be further decreased by reducing the amount and length of the considered sequences. As proposed in [18], in this work only $m = 2$ sequences of $n = 14$ partially combined elements have been used.

It should be noticed that the SSD strategy represents a very good approximation of the ideal SE enumeration under the assumption that λ_i depends exclusively on the Euclidean distance (as it occurs in SO detection). As introduced in [18] and further investigated in [24], when *a-priori* information is present (i.e. in SISO detection), the node enumeration resulting from the SSD approach may differ significantly from the ideal SE enumeration. This effect leads to significant BER performance degradation, especially if *specific leaf sequences* (Fig. 2) are used. In [18] the *Min-Search* approach is proposed to mitigate this effect. By means of this strategy the detection performance is significantly enhanced, but the computational complexity increases due to the additional metric computations and sorting operations required. In this work an *adaptive hypothesis* strategy is applied. In contrast to *Min-Search*, this approach makes use of the already analyzed symbols instead of searching for additional ones. The set of nodes $\mathscr{B}$ considered by the *specific leaf sequence* comprises seven elements (one possible hypothesis x^r and six candidate counter-hypotheses in real and imaginary directions of x^r) [18]. In case that $\mathscr{B}$ contains a symbol $x^{r'}$ with lower λ_0 than the initially selected x^r, the proposed approach rectifies the potential leaf hypothesis as $x^{r'}$, whereas remaining symbols in $\mathscr{B}$ (including the originally selected hypothesis x^r) are treated as potential counter-hypotheses, as illustrated in Fig. 2.

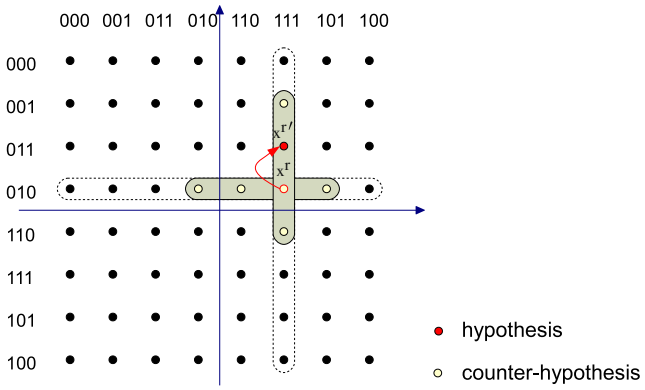


Figure 2 Candidate symbols after application of the *adaptive hypothesis* approach (64-QAM constellation).

Even though the *adaptive hypothesis* approach presents a slightly worse BER↔ ♯n trade-off than *Min-Search* [24], it does not require the costly additional metric computation and sorting operations incurred by the latter. *Adaptive hypothesis* represents therefore a very attractive approach towards efficient hardware implementation, as shown in Section 7.2.

3.4.2 Metric Estimation

The computationally expensive operations required for the metric calculation (Eq. 4) can be considerably simplified by an estimation based on the SSD's geometrical approach [4]. Euclidean distances in Eq. 4 are replaced by predefined geometrical distances $d_{m,n}$ corresponding to each of the fixed enumeration sequences:

$$r_{ii}^2 \left\| y_i''' - \hat{x}_i \right\|^2 \approx r_{ii}^2 d_{m,n}^2. \tag{9}$$

It is additionally possible to precalculate r_{ii}^2 as well as $r_{ii}^2 d_{m,n}^2$, hence reducing the complex products in Eq. 4 to a single real multiplication or even completely dissolving it. This considerable reduction in computational complexity makes this technique especially interesting for realizable hardware implementations.

3.5 Algorithm Partitioning and Regularization

The TS-based detection process can be decomposed, as proposed in [20], in an arbitrary number of regularized loops. The operations performed within each loop have been partitioned into the task blocks §a ... §n, as illustrated in Fig. 3. Selection of first node to be extended (Eq. 8) is performed in §a. SSD's geometric position analysis is subsequently carried out in §b, determining the corresponding search sequence and the second node to be extended. Metric (Eq. 9) and interference (Eq. 4) associated to these nodes are determined by §c, § f, §g and §h. Subsequently, the radius tuple is updated (Eq. 7) by §d. Final task within the loop is the selection of the tree layer to be processed during next loop (§e). As mentioned in Section 3.2, a so called one-node-per-cycle(-loop) realization is desired. Consequently, sibling parent nodes and leaf nodes are assumed to be processed in parallel [18]. For this purpose, it is necessary to define separated blocks which process first (§a, §c, § f), second (§b , §g, §h) and subsequent nodes (§i, § j, §k) individually. Likewise, definition of task blocks processing candidate symbols (§l, §m) for later L-values calculation (Eq. 1) in §n is required. In order to extend the SO-TSD functionality to process soft-information from the channel decoder (SISO-TSD), the task blocks computing metric distances (§c, §g, § j and §l, as highlighted in Fig. 3) have been modified to include *a-priori* information according to Eq. 6. Additionally, the task block §m has been modified to include the *adaptive hypothesis* strategy described in Section 3.4.1.

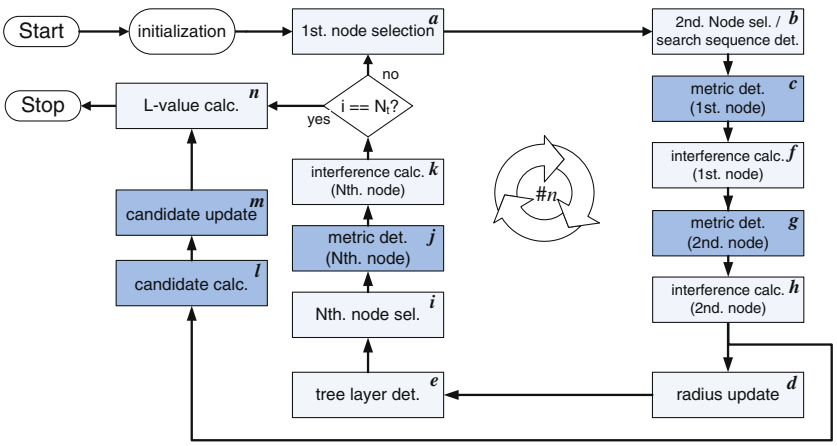


Figure 3 TSD regularized flow diagram.

4 Fixed-Point Representation

Fixed-point arithmetic is commonly applied in order to limit the latency and hardware resources required in a practical implementation. However, overflow and quantization errors incurred due to the inherent limited precision typically lead to BER performance degradation in the context of MIMO detection. It is therefore necessary to define the bit-resolution appropriately, according to the trade-off between BER performance and hardware complexity. In [3] a maximum word width of 8 bits[3] was proposed, whereas ~ 0.3 dB SNR loss has to be accepted. In this work, suitable fixed-point representation of $L_a(c_{i,j})$ has been additionally investigated and required considerations taken into account, as described throughout the following sections.

4.1 Normalization

As investigated in [3], parameters involved in Eq. 1 and hence in Eq. 4 (y'_i, $\mathbf{R} = \{r_{ii}, r_{ij}\}_{i \neq j}$ and $L_a(c_{i,j})$, with $_{i,\, j = \{1,\ldots,(N_T-1)\}}$) possess a dynamic range fluctuating with the received energy E_r. In order to define fixed ranges, independent of transmission conditions, elimination of E_r dependency is required. For this purpose, a scaling factor $s_f = f\sqrt{E_r}$ (with f representing a constant factor) is utilized to normalize parameters $\tilde{r}_{ii} = r_{ii}/s_f$ and $\tilde{L}_a(c_{i,j}) = L_a(c_{i,j})/s_f^2$, leading to $\tilde{\lambda}_i = \lambda_i/s_f^2$ and Eq. 1 is modified as $\tilde{L}\left(c_{m,l}|\mathbf{y}\right) = L\left(c_{m,l}|\mathbf{y}\right)/s_f^2$. Concerning y'_i and r_{ij}, normalization with s_f is not necessary since application of the SSD strategy (Eq. 8) eliminates the energy dependency. A certain scaling factor f' is nevertheless required in order to accommodate their dynamic range to the proposed 8-bit quantization. Based on this, $\tilde{y} = y'_i/f'$ and $\tilde{r_{ij}} = r_{ij}/f'$ are additionally defined. To ensure flexibility of the proposed detector implementation, fixed-point representation has to be additionally independent of the system configuration. For this purpose, adjusting f and f' according to the transmission scheme is proposed [3], as shown in Table 1.[4] As a result, definition of different fixed-point representations depending on transmission characteristics (as e.g. in [27]) is avoided. By applying the described normalization, 8-bit fixed-point representation is enabled, independent of transmission conditions and adaptable to different N_T and Q by simple adjustment of f and f'. The resulting detector presents lower hardware complexity and higher flexibility than comparable sphere detector (SD) realizations [21, 27], typically requiring greater precision (10–16 bits) to achieve comparable BER performance.

Table 1 Scaling factors f and f', to enable flexibility of the proposed fixed-point representation.

Q	4	16	64
f	12	6	3
f'	8	16	32

4.2 Simulation Results

Figure 4 depicts SNR↔ $\sharp n$ achieved at 10^{-5} BER, using different transmission and detection schemes operating in a non-iterative receiver.[5] Results corresponding to unclipped ($L_{max} = \infty$) $STS_{(FLP)}$ detection are depicted representing full *max-log-APP* SNR performance bound. $SIC_{(FXP)}$ detection has been implemented using the proposed $TSD_{(FXP)}$ realization with limited $\sharp n = N_T$, hence representing the lowest $\sharp n$ bound. Results corresponding to the proposed TSD for a given SNR↔ $\sharp n$ trade-off are also included [3]. As depicted, fixed-point arithmetic leads to ~0.3dB SNR degradation with respect to the analogous TSD floating point approach. Metric underestimations resulting from the limited precision lead, in addition, to a $\sharp n$ reduction of ~20 %. It should be noticed that SNR↔ $\sharp n$ difference between $TSD_{(FLP)}$ and $TSD_{(FXP)}$ is approximately constant for the considered transmission configurations, hence validating the flexible normalization approach proposed in Section 4.1. Additonally, it should be observed that SNR↔ $\sharp n$ relationship is adjustable according to transmission requirements, ranging from close to full *max-log-APP* SNR performance at significantly reduced $\sharp n$, to SIC SNR performance and complexity. For the depicted SNR↔ $\sharp n$ trade-offs, $TSD_{(FXP)}$ achieves considerable complexity reduction with regard to unclipped $STS_{(FLP)}$ (~1–6 orders of magnitude less nodes are extended during the tree search), while providing considerable SNR performance gain with respect to SIC (~4dB in most of the considered scenarios).

[3] Integer and fractional word lengths have been individually determined for each of the variables involved in the detection. Integer word lengths have been selected to cover the variables range, while fractional lengths have been determined based on BER performance analysis through simulations. Further details can be found in [16].

[4] Values determined through simulation for the given system model.

[5] FLP denotes floating point, while FXP stands for fixed point.

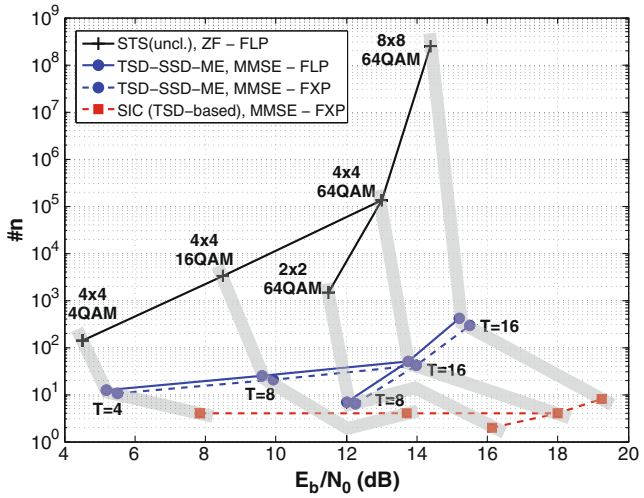


Figure 4 $\sharp n$ required for 10^{-5} BER, corresponding to different transmission and detection schemes ($It = 1$).

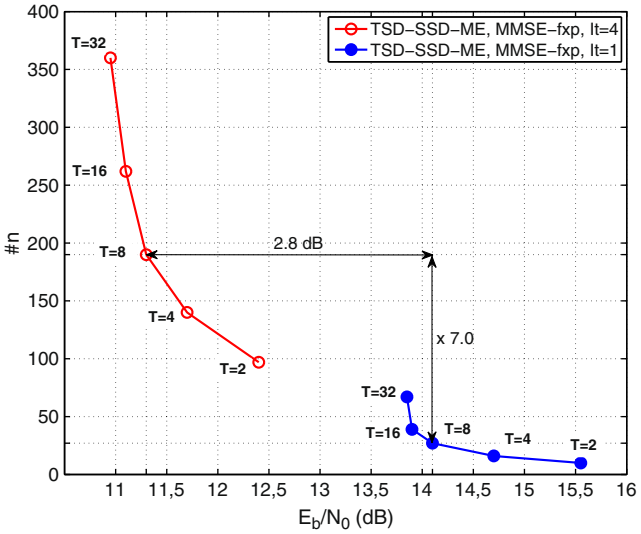
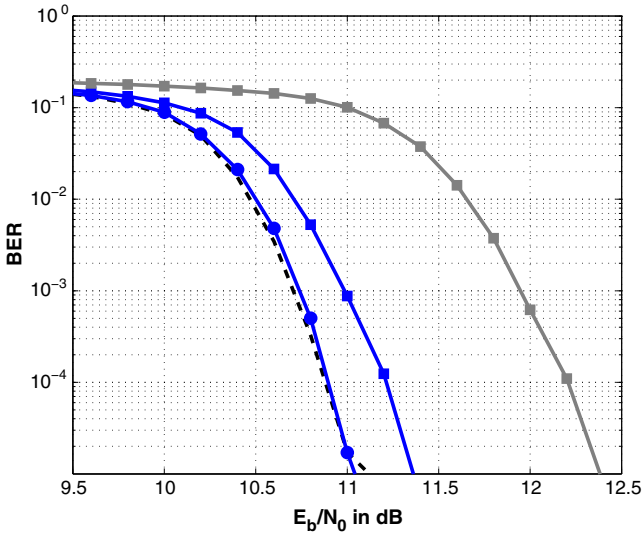
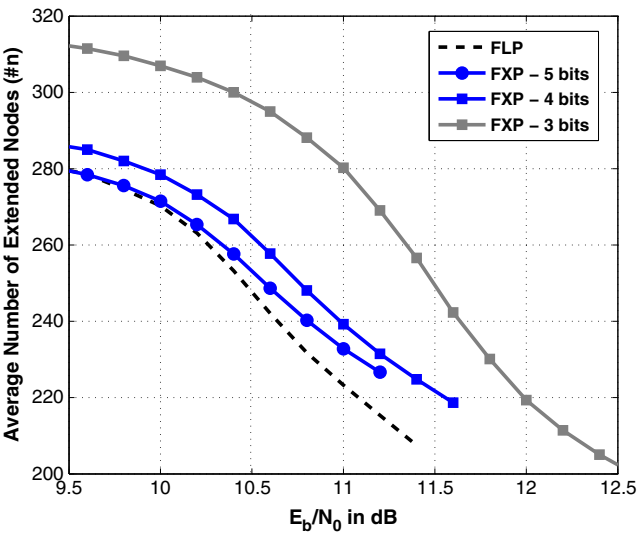


Figure 6 $\sharp n$ required for 10^{-5} BER, corresponding to different tuple sizes T for a 4×4 64-QAM transmission with 3 ($It = 4$) and without ($It = 1$) detection-decoding iterations.

As long as *a-priori* information is considered, the complexity of the metric computation increases with increasing Q (e.g. for 64-QAM modulation 6 further additions are required for each partial metric λ_i, as derived from Eq. 4). It should be additionally noticed that due to the inherent range limitation, $\sharp n$ increases as the bit-width of $L_a(c_{i,j})$ is reduced, as illustrated in Fig. 5b. In order to keep the added computational complexity as low as possible at no SNR degradation (Fig. 5a), 5-bit quantization is selected. A comparison between non-iterative ($It = 1$) and iterative detection with 3 iterations ($It = 4$) is depicted in Fig. 6 for different T values. For the SNR$\leftrightarrow \sharp n$ trade-off given by $T = 8$, a detection performance gain of ~3 dB is achieved by performing 3 detection-decoding iterations ($It = 4$) with respect to the non-iterative case ($It = 1$). This significant performance improvement comes at the cost of ~7 times higher $\sharp n$. Note that despite this increase, complexity of SISO-TSD with $It = 4$ is still ~3 orders of magnitude lower than unclipped STS with $It = 1$ (Fig. 4). As previously observed for the non-iterative case [4], increasing T beyond a certain value (e.g. 32) does not lead to significant further performance improvement.



(a) BER



(b) Complexity ($\sharp n$)

Figure 5 Impact of the fixed-point representation of $L_a(c_{i,j})$ on BER performance (SISO-TSD, 4×4 MIMO, 64-QAM, $It = 4$).

5 Computational Complexity and Critical Path Estimation

In order to estimate the hardware implementation costs, computational complexity and latency will be analyzed in the subsequent. Resulting from the application of the complexity reduction techniques in Section 3.4 only basic operations (addition, comparison and shift operations) are required to carry out the detection (Eqs. 1, 4–9). The cost of comparison and addition operations is assumed to be approximately equivalent, whereas the shift operation cost is neglegted. Table 2 summarizes the amount of computations c (add-equivalent operations, i.e. addition and comparison) required by each task block for the SISO-TSD. Despite inclusion of *a-priori* processing (Eq. 6), the metric determination tasks (§c, §g, §j) still present the lowest computational complexity c. However, dependance on Q is now observed, in contrast to SO-TSD. Except task block §l (approx. 50 % more complex due to processing of *a-priori* information), c of remaining task blocks remains unaffected in comparison to SO-TSD, presenting §l and §m (i.e. handling the bit-specific metrics) the highest c. Computational complexity depends hence in general on system order (Q, N_T), except for the radius update task §d (growing linearly with T), the mentioned metric determination tasks §c, §g, §j (only depending on Q) and the node selection task blocks §a, §b, §i (independent of N_T, Q and T). Regarding the resulting overall computational complexity c_total, an increase of 10 % is observed in comparison with SO-TSD (except for 2×2 64-QAM, where the increase reaches up to 30 %). It should be noticed that doubling N_T or Q implies increasing c_total by a factor of $\sim$1.6 ($\Delta c_\mathrm{total} \approx 4/5 \Delta N_\mathrm{T} \approx 4/5 \Delta Q$), which represents a marginal increase compared to the SO-TSD implementation (where a factor $\sim$1.5 was observed).

Table 2 Computational complexity c (in number of add-equivalent operations) and latency of each task block, for different transmission schemes.

Task	4×4 4-QAM ($T = 4$)	4×4 16-QAM ($T = 8$)	4×4 64-QAM ($T = 16$)	2×2 64-QAM ($T = 8$)	8×8 64-QAM ($T = 16$)	l
§a	4	4	4	4	4	1
§b	11	11	11	11	11	1
§c	4	6	8	8	8	1
§d	9	13	21	13	21	1
§e	18	18	18	6	42	1
§f	14	20	32	12	72	4
§g	3	5	7	7	7	1
§h	14	20	32	12	72	4
§i	15	15	15	15	15	1
§j	3	5	7	7	7	1
§k	14	20	32	12	72	4
§l	33	62	102	98	110	6
§m	55	125	227	131	419	1
§n	8	16	24	12	12	1
c_total	205	340	540	348	872	l_cp

Concerning latency l, the cost of shift operations is neglected, while the cost of addition and comparison operations is assumed to be equivalent to $l = 1$ clock cycle. This consideration is a quite conservative design constraint, which ensures achieving the target throughput (depicted in Section 8.1) in a worst-case hardware implementation. For compatibility with the SO-TSD implementation (especially concerning the task scheduling and pipeline-interleaving schemes shown in Section 7) the computations included for processing of *a-priori* information are also condensed in one clock cycle. This assumption relaxes the previously mentioned conservative design constraint for the metric computation blocks (§c, §g, §j and §l). As a consequence, slight reduction of the maximum achievable clock frequency should be expected. The latency $l = 1$ can be hence assumed for the metric determination in both SO and SISO detectors. According to the data dependency analysis,[6] most of the computations enclosed within each task block may be performed in parallel. Considering sufficient parallelization level (varying with N_T, Q and T), fixed l can be assumed (shown in Table 2). Resulting from this, latency of the regularized loop (Fig. 3) becomes independent of the system order (N_T, Q) and T, in contrast to computational complexity. Loop latency can be further reduced by partially overlapping execution of the comprehended task blocks, resulting in the task scheduling scheme illustrated in Fig. 7c. Since output from blocks §h, §l, §m and §n is never required in the immediately subsequent loop [16], the critical path is comprised by blocks §a – §e. Considering the latencies given in Table 2, the critical path presents therefore a latency $l_\mathrm{cp} = 5$ clock cycles. Similar assumptions and analysis can be applied to SD and SIC detector implementations, resulting in the schemes shown in Fig. 7a and b, respectively [16]. As illustrated, l_cp of the proposed (8-bit precision) SO/SISO-TSD implementation is $<$1/4 of l_cp corresponding to typical 16-bit precision SD realization and $<$1/2 of 16-bit precision SIC detector l_cp. In this regard, SSD and ME approaches represent, together with the proposed task scheduling, the main strategies contributing to this considerable reduction of the critical path latency.

[6] The $l = 1$ assumption takes into account existing data dependencies. Detailed data dependency analysis is considered to be out of the scope of this work and thus is not presented.

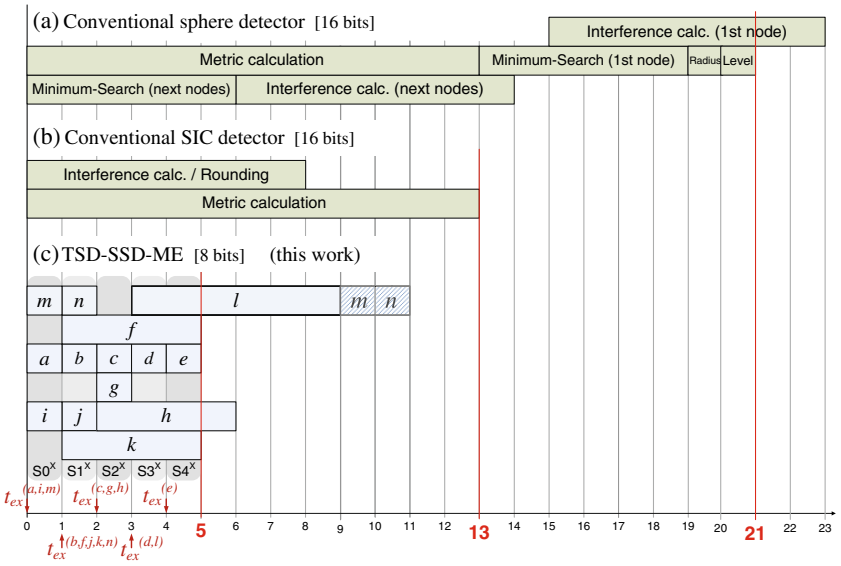


Figure 7 Loop latency (in clock cycles), corresponding to different detection strategies.

6 Parallelization Strategies for Throughput Enhancement

Implementing the bit- and instruction-level parallelism mentioned in Section 5 does not require major considerations besides dependency analysis.[6] The detection throughput is dominated by (1) the number of loops required to complete the detection (or equivalently, $\sharp n$) and (2) the loop latency l_{cp} (for a given frequency). In order to enhance throughput, further parallelization approaches are considered in this work.

6.1 Pipeline-Interleaving

The pipeline-interleaving technique [13] allows increasing the throughput provided that the processed data streams are independent. In this work, this strategy is applied in a similar way as done in [7, 14], as further described in the following. Task blocks defined in Section 3.5 can be grouped according to their execution times t_{ex}, illustrated in Fig. 7c. Thereby 5 sets of tasks Sx, $\bigvee x = \{0, \ldots, 4\}$ are defined, each assigned to a different processing element (PE). Notice that those tasks executed after l_{cp} cycles ($\S m$ and $\S n$) are scheduled together with the tasks corresponding to the subsequent loop. Based on this, the throughput can be enhanced by interleaving the execution of sets Sx corresponding to different detection paths d, in pipelined fashion. Figure 8b illustrates the pipeline-interleaving scheme of Sx^d, where e.g. $S3^2$ represents the execution of task set 3 corresponding to detection path 2 (i.e. third interleaved detection path). Throughput and resource utilization are maximized when the number of interleaved detections equals the maximum number of possible pipeline stages P, i.e. $P = 5$ for the identified critical path ($l_{\text{cp}} = 5$). Assuming in addition that a new detection starts as soon as another has finished, no PE is idle after filling in the pipe. By applying these considerations, resulting average detection throughput $\overline{\tau}$ is enhanced by a factor of nearly 5 with regard to a non-pipelined implementation. It should be noticed that the operations required for processing the *a-priori* information have been distributed throughout the described pipeline stages, in order to avoid detrimenting the maximum achievable clock frequency with respect to the SO detector implementation in [3].

6.2 SIMD Vectorization

As introduced in [20], vectorization of $\sharp n$-variable SD (e.g. for parallel orthogonal frequency-division multiplexing (OFDM) subcarrier processing) results in increased average $\sharp n$ ($\sharp\overline{n}$) per vector element, in contrast to $\sharp n$-fixed detectors (M-algorithm, K-best ...). Assuming that a vector is completely processed as soon as the execution of all its q parallel slices have finished, the overall vector latency is determined by the slowest path (i.e. greater $\sharp n$). Resulting from this, $\sharp\overline{n}$ increases with the degree of vectorization q, thereby dramatically affecting the achievable speedup. A simple approach to ease this problem consists in bounding $\sharp n \leq \sharp n_{\max}$ [20], but early termination of paths with $\sharp n > \sharp n_{\max}$ leads to BER performance loss. It is nevertheless possible to find a suitable $\sharp n_{\max}$, e.g. $\sharp n_{\max} = 1.25 \times \sharp\overline{n}$, causing negligible BER performance loss while allowing to nearly achieve the maximum feasible speedup q [20]. The simplicity and effectiveness of this approach make it

Table 3 Memory specifications for $N_T = N_R = 4$ and $Q = 64$ ($bf = 40, q = 1$).

Memory	Width (bits)	Length (♯ words)	Size (bytes)	Content
IMEM	64	40	320	Program code
SMEM	64	15	120	System and channel info.
VMEMI	64	40	320	Received symbols (**y**)
VLAMEM	128	40	640	Soft input (*a-priori* info.)
VMEMO	192	40	960	Soft output (L-values)

especially interesting for vectorization of ♯n-variable SD, like TSD. The control flow regularization described in Section 3.5 enables almost straightforward application of SIMD vectorization to SO/SISO-TSD. Assuming accordingly vectorized memory access (described in Section 7.1) and bounding ♯n appropriately, $\overline{\tau}$ may be enhanced by a factor of nearly q ($\overline{\tau}^{q\text{-SIMD}} \approx q \times \overline{\tau}$).

7 VLSI Architecture

The hardware realization presented in this work is based on the SO-TSD ASIP model and implementation concepts proposed in [3, 5], extended in order to support processing of *a-priori* information (SISO-TSD). The design is based on the so-called synchronous transfer architecture (STA) template [8] and is controlled by means of very long instruction words (VLIWs). The architecture is comprised of basic modules known as functional units (FUs). The output ports are buffered with registers so that data produced by a FU can be directly consumed by connected FUs. The proposed design is depicted in Fig. 9 and described through next sections. It is mainly comprised of:

- Control unit: the instruction decoder gets VLIWs read from the instruction memory (IMEM) and maps the corresponding operations onto the FUs comprising the design. A Flow Control Unit (FCU) generates IMEM addresses [3], assisting the sequencer in handling the conditional execution flow (as further detailed in Section 7.1.2).
- Data path: it contains data memories (SMEM, VMEMI, VLAMEM, VMEMO), banks of data and address registers and the MIMO detection module, as further detailed throughout the following sections.

7.1 Memory

7.1.1 Memory Organization

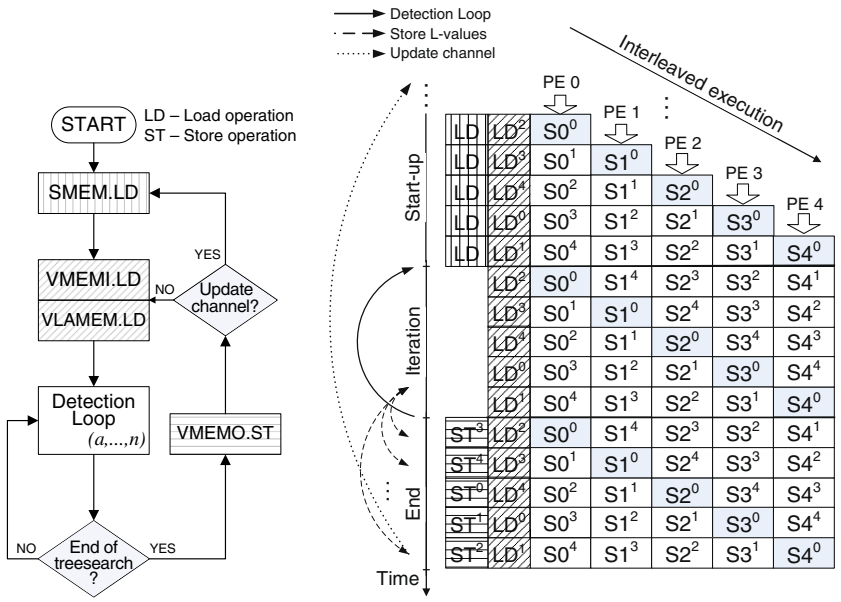
The design comprises an instruction memory (IMEM) and the four data memories specified in Table 3. VMEMI, VLAMEM and VMEMO are q-fold vector memories with regard to SIMD vectorization, while SMEM is scalar. These memories represent buffers of length bf words, with each word containing the information of one detection path. Notice that VLAMEM has been included in the SISO-TSD implementation in order to enable processing *a-priori* information. Considering 4×4 MIMO transmission with 64-QAM constellation, using buffers of size $bf = 40$ words (i.e. 40 individual detection paths) and disregarding vectorization ($q = 1$), a total of ∼2 KB data memory is required.[7]

7.1.2 Memory Access

The latency cost inherent in memory access frequently penalizes the design achievable throughput. In order to ensure data availability without increasing the detection latency, data access is performed in parallel to the execution of detection tasks. As depicted in Fig. 8a (where *Detection Loop* represents the regularized loop of Fig. 3), the memory access is conditional, randomly triggered by detection termination and channel update events. Consequently, joint control flow of the interleaved detection paths is frustrated. In order to satisfy the individual data access demands of each data path without disrupting the pipelined execution of remaining ones, the pipeline-interleaving scheme illustrated in Fig. 8b is employed, combined with the following approaches [3]:

- **Distributed AGU (Address Generation Unit):** due to the different nature and frequency of memory access, the address generation functionality has been separately defined and distributively implemented for each of the defined memories. VMEMI, VLAMEM and SMEM access is sequentially performed, thus relying on simple pointers and a register file for address storage. Moreover, VMEMI and VLAMEM present identical access patterns and consequently, data can be loaded simultaneously based on a unique pointer. As proposed in [3],

[7] Values stored in data memories are represented using the fixed-point representation described in Section 4.



(a) Control flow diagram (b) Pipeline-interleaving scheme

Figure 8 Extended control flow and pipeline-interleaving schemes, including data memory access.

VMEMO addresses are generated based on tags which ensure that **y** and the corresponding L-values occupy homologous memory areas in VMEMI and VMEMO, respectively.

- **SMEM cache:** a partial copy of the highly frequently accessed SMEM into a register file is proposed in [3], thereby allowing faster data access and saving costly memory access operations. SMEM access is thus only required when channel information has to be updated (depending on the channel coherence time).
- **VMEMI/VLAMEM prefetching:** periodic and unconditional read operations are performed during the complete detection process, temporary storing prefetched data (24 bytes) in registers. Internal signaling detects when data has been effectively consumed (i.e. a new detection begins) in order to update VMEMI/VLAMEM pointer accordingly and prefetch new data.
- **Flow Control Unit (FCU):** despite the above described considerations, VMEMO access (triggered by termination of up to $P = 5$ different detection paths) as well as SMEM access (depending on the channel coherence time) are still conditional. Resulting from this, multiple conditional branch operations have to be performed by the sequencer, as illustrated by the arrows in Fig. 8b. Based on control signals received from the detector module, the FCU (Fig. 9) identifies the trigger events and determines the corresponding IMEM address, subsequently transferring it to the sequencer. As a result, the sequencer controls the execution flow by uniquely performing unconditional branch operations.

By applying these strategies, no overhead (i.e. additional clock cycles) is caused and control flow regularization (Section 3.5) is not disrupted due to memory access. It should be noticed that the control-flow and pipeline-interleaving schemes illustrated in Fig. 8 have been extended with respect to [5] in order to include the VLAMEM access operations required for processing *a-priori* information (SISO-TSD). Resulting latency depends uniquely on the detection process, as described in Section 8.1. The achievable speedup provided by the parallelization approaches considered in Section 6 remains therefore unaffected.

7.2 MIMO Detector Module

The task blocks §a – §n (Fig. 3) defined by the proposed algorithm partitioning (Section 3.5) are directly mapped to the STA FUs comprising the MIMO detector module (Fig. 9) as detailed in the following.

7.2.1 Node Enumeration Unit (NEU) and Interference Computation Unit (ICU)

The SSD strategy (Eq. 8) is implemented by the NEU (comprised by task blocks §a, §b and §i in Fig. 3). The predefined sequences are stored in small (∼32 bytes) look-up-tables (LUTs). The operations involved in the determination of the interference-reduced received

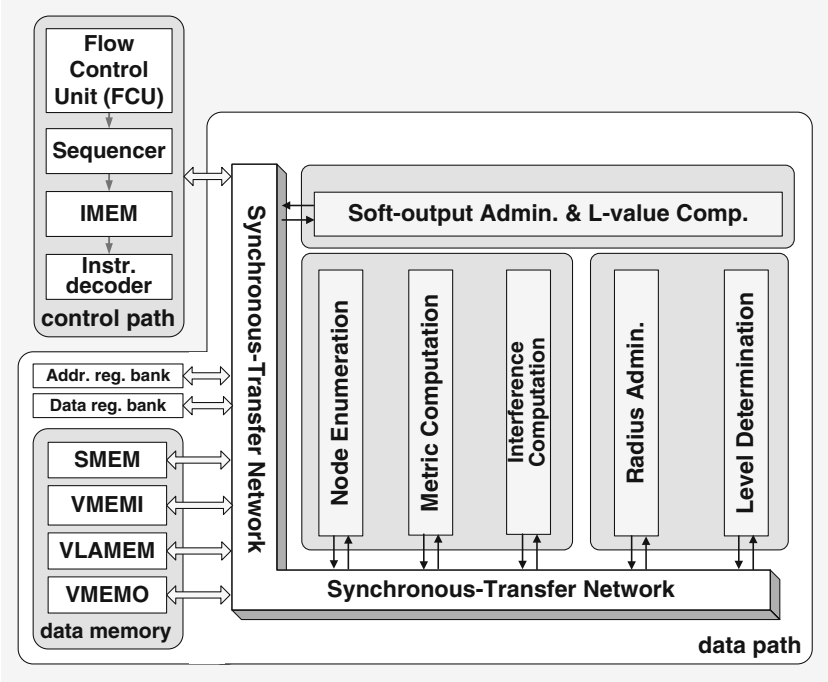


Figure 9 MIMO detector architecture.

signal y'' (Eq. 5) are performed in the ICU (comprised by task blocks §f, §h and §k).

7.2.2 Metric Computation Unit (MCU)

As described in Section 3.4.2, $r_{ii}^2 d_{m,n}^2$ values are pre-calculated and stored in SMEM, thus reducing the complexity of Eq. 4 to a simple addition operation (performed in task blocks §c, §g and §j in Fig. 3). This unit has been extended with respect to [5], in order to include the up to L addition operations (Eq. 6) required to take *a-priori* information from channel decoder into account, as shown in Fig. 10.

7.2.3 Radius Administration Unit (RAU) and Level Determination Unit (LDU)

The RAU (block §d in Fig. 3) is responsible for updating and sorting the list of leaf metrics λ_0 comprising the search tuple $\mathscr{T}$. The search radius R is adapted according to Eq. 7 by this entity, as detailed in Section 3. Subsequently, the LDU (block §e) makes a decision on the next tree level to be explored, implementing depth-first tree traversal and prunning subtrees which lead to metrics exceeding the search radius R. Both entities require logic and few comparators to carry out their respective tasks.

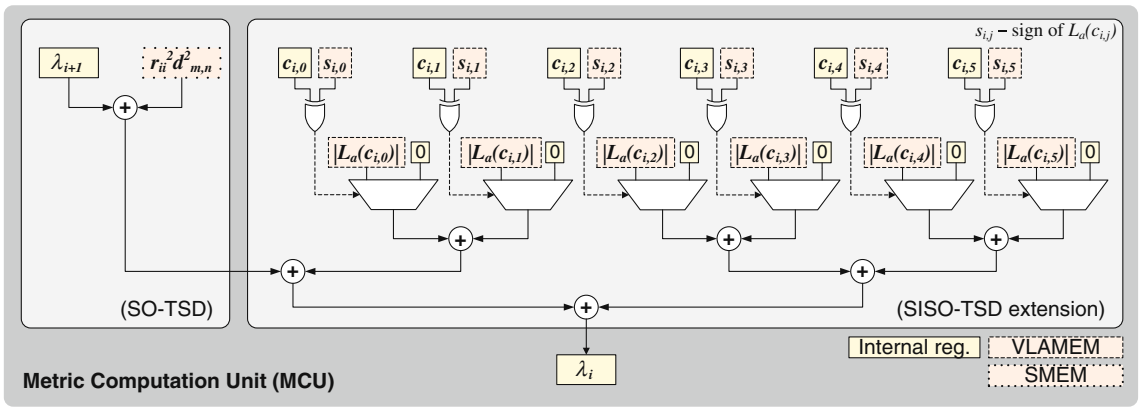


Figure 10 Block diagram of the proposed MCU architecture for SISO-TSD (dimensioned for 64-QAM).

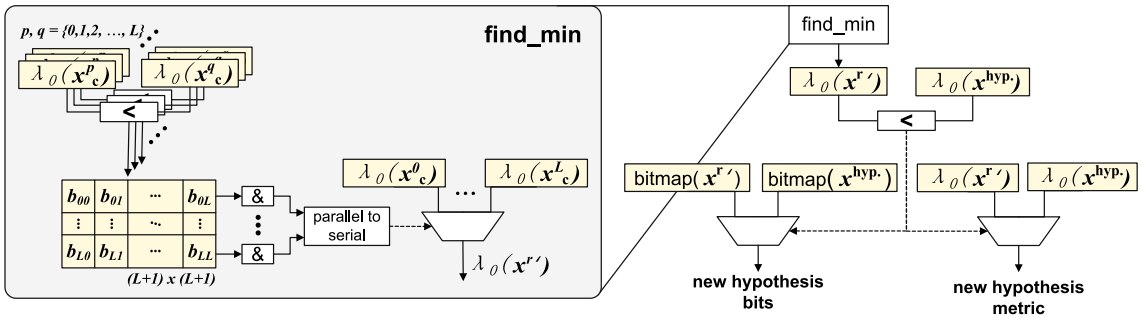


Figure 11 Block diagram of the hypothesis update procedure with *adaptive hypothesis*.

7.2.4 Soft-output Administration Unit (SAU) and L-values Computation Unit (LCU)

SAU (blocks §*l* and §*m* in Fig. 3) and LCU (block §*n*) enable generation of soft-information. In block §*l* those symbols representing the most favorable candidates for (counter-)hypotheses are firstly selected (based on the *specific leaf sequences* mentioned in Section 3.4.1) and their corresponding metrics are computed. Subsequently, in block §*m* the previously stored hypothesis and counter-hypotheses are replaced by those candidates presenting lower metrics. Resulting metric values are fed to the LCU (§*n*), calculating the L-values according to Eq. 1. These entities are internally pipelined according to the 5-stages scheme described in Section 6.1.

Block §*m* has been extended with respect to [5] in order to support the *adaptive hypothesis* approach described in Section 3.4.1. The update procedure of the hypothesis is depicted in Fig. 2 (analogous procedure is applied to update the counter-hypotheses). The metrics λ_0 of the previously stored hypothesis and the new candidate hypothesis ($x^{\text{hyp.}}$ and $x^{\text{r}'}$, respectively) are firstly compared. Subsequently, λ_0 and the binary representation (bitmap(x)) of the symbol with the lowest metric are stored. It should be noticed that in [5], the closest-to-y_l''' constellation symbol (x^{r} in Fig. 2) is directly taken as the candidate hypothesis, whereas in this work the symbol $x^{\text{r}'} \in \mathscr{B}$ with the lowest metric is considered instead (i.e. the candidate hypothesis is rectified, as explained in Section 3.4.1). For this purpose, the *find_min* block shown in Fig. 11 has been included. In this block the metrics λ_0 of all candidate x_{c}^p are compared in parallel against λ_0 of all other x_{c}^q contained in $\mathscr{B}$ (with $p, q = \{0, 1, \ldots, L\}$), requiring $(L+1)^2$ comparators. This results in a matrix of binary flags $b_{p,q}$ of size $(L+1) \times (L+1)$, where the row p containing $(L+1)$ flags set to "1"[8] corresponds to the candidate symbol $x_{\text{c}}^p \in \mathscr{B}$ with minimum λ_0. By these means, the increase in latency of block §*m* is minimized, at the cost of increasing its area in ∼2 kGE (8.5 kGE in [5], 10.6 kGE in this work).

7.2.5 Synchronous-Transfer Network

The FUs are connected according to the previously introduced STA principle. The information is intermediately stored in the FUs registered output ports, which act as pipeline registers. The data transfer is synchronously performed in feed-forward fashion among connected FUs. In particular, two data flow paths can be distinguished. Main data stream is fed forward across the units involved in the tree traversal and metric computation (NEU → MCU → RAU → LDU) and subsequently fed backward (LDU → NEU) closing the detection loop in Fig. 3. The number of ports of each of these blocks has been extended with respect to [5] to include storage and forwarding of *a-priori* information. In parallel to this, data is fed forward to SAU and finally to LCU for L-values computation.

8 Implementation Results

In order to assess the implementation complexity of the proposed detector implementation, the described VLSI architecture has been modeled in Verilog HDL and synthesized[9] using Synopsis Design Compiler. RTL and gate-level netlists are verified against the same test vectors generated from a MATLAB/C++ fixed-point model. For the analysis, a detector design configured for $N_{\text{T}} = N_{\text{R}} = 4$ with 64-QAM constellation has been instantiated. A maximum clock frequency

[8]Notice that the diagonal elements of the flag matrix ($b_{p,q}$ with $p = q$) are forced to "1" in order to satisfy this condition.

[9]Due to a version migration concerning Synopsis Design Compiler and synthesis libraries, the SO-TSD design publicated in [5] has been re-synthesized for this work. Consequently, area and power consumption results differ slightly from those presented in [5].

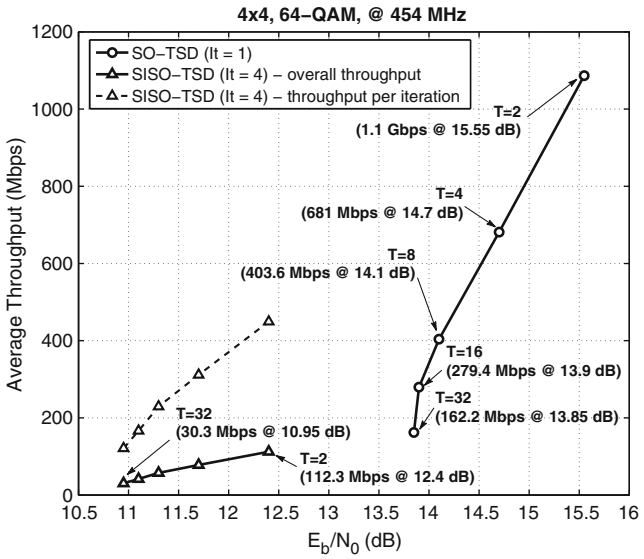


Figure 12 TSD average throughput at $f_{\max} = 454$MHz, for different tuple sizes (T) at 10^{-5} BER (4 × 4 MIMO, 64-QAM).

$f_{\max} = 454$ MHz is reached under typical case operating conditions.[10]

8.1 Throughput

Based on the considered pipeline-interleaving strategy (Section 6.1) and disregarding SIMD vectorization ($q = 1$), average detection throughput $\overline{\tau}$ at frequency f is given by

$$\overline{\tau}_{\text{TSD}} = \frac{N_{\text{T}} \cdot L}{\sharp n} f \quad [\text{bits/s}].$$

Regarding iterative receivers, formal determination of the system throughput should take into account (1) the throughput of the MIMO detector (variable with the number of detection-decoding iterations [19, 24]) and (2) the throughput of the channel decoder. For results comparability and since channel decoder realization is out of the scope of this work, the decoder throughput will be in the following disregarded.

Figure 12 depicts the throughput corresponding to the proposed SISO-TSD realization, both in an iterative detection-decoding system with 3 iterations ($It = 4$) as well as in a non-iterative system ($It = 1$). In both cases, different throughput↔SNR-performance tradeoffs are shown (adjustable through the tuple size T). As expected from the complexity trend observed in Fig. 6, the throughput of SISO detection with $It = 4$ decreases notably with regard to SO detection ($It = 1$), moreover depending on the value of T. For small T size ($T < 4$) a factor of approx. 10 is observed. For medium T values ($T = 8 - 16$) throughput of SO-TSD (SISO-TSD$_{It=1}$) is ≈6–7 times greater than throughput of SISO-TSD$_{It=4}$. For large T ($T > 16$), throughput of SO-TSD is ≈5 times greater. To sum up, throughput of SISO-TSD$_{It=4}$ is between 5–10 times lower than that of SO-TSD, being the throughput degradation smaller for higher T values. It should be additonally noticed that while a throughput improvement by a factor of ≈7 can be achieved by varying T in SO-TSD, only half of this improvement (≈3) is achievable by SISO-TSD$_{It=4}$.

Table 4 Area breakdown of SO and SISO TSD at 454 MHz (4 × 4 MIMO, 64-QAM).

	SO-TSD			SISO-TSD		
	kGE	mm^2	%	kGE	mm^2	%
Total	133.9	0.193	100.0	164.6	0.237	100.0
Memory	49.8	0.072	37.2	66.9	0.096	40.6
TSD core	84.1	0.121	62.8	97.7	0.141	59.4
Ctrl. path	3.1	0.004	2.3	3.2	0.005	2.0
Data path	81.0	0.117	60.5	94.5	0.136	57.4
NEU	9.9	0.014	7.4	13.8	0.020	8.4
ICU	16.6	0.024	12.4	16.4	0.024	10.0
MCU	5.8	0.008	4.3	7.7	0.011	4.6
RAU	5.9	0.008	4.4	6.9	0.010	4.2
LDU	4.8	0.007	3.6	6.1	0.009	3.7
SAU	24.2	0.035	18.1	28.9	0.042	17.6
LCU	3.9	0.006	2.9	3.9	0.006	2.4
Register file	8.2	0.012	6.1	8.2	0.012	5.0
Other logic	1.8	0.003	1.4	2.7	0.004	1.7

8.2 Area

Table 4 shows the area breakdown of the proposed SISO-TSD design, extracted from pre-layout synthesis reports. In order to provide technology-independent area characterization, the number of gate-equivalents (GEs) is additionally specified.[11] A total area of 0.24 mm^2 is required, representing an increase of ∼16 % with regard to the SO-TSD implementation (disregarding memory). In both SO/SISO-TSD, ∼40 % of the area corresponds to memory (Section 7.1) and ∼60 % to the SISO-TSD core (Section 7.2). The control path represents small area overhead (∼2 %). Regarding the SISO-TSD core, obtained results evince that those modules involved in the soft-output computation (SAU+LCU) present the highest hardware complexity, as expected from the computational complexity analysis in Section 5. Similar result is observed in [25]

[10] TSMC 65 nm low-power CMOS libraries are used. Typical case: (1.2 V, 25 °C). Worst case: (1.08 V, 125 °C).

[11] One GE corresponds to the area of a two input NAND gate synthesized using TSMC 65nm libraries.

Table 5 Total power consumption of SO and SISO TSD under typical case operating conditions, at 454 MHz (4 × 4 MIMO, 64-QAM).

	SO-TSD		SISO-TSD	
	mW	%	mW	%
Total	76.27	100.0	107.41	100.0
Memory	18.90	24.8	34.93	32.5
TSD core	58.20	76.3	73.90	68.8
Ctrl. path	1.11	1.5	1.27	1.2
Data path	57.09	74.9	72.63	67.6
NEU	8.18	10.7	12.18	11.3
ICU	12.20	16.0	12.50	11.6
MCU	4.61	6.0	6.73	6.3
RAU	4.48	5.9	5.90	5.5
LDU	4.42	5.8	5.93	5.5
SAU	15.06	19.7	20.03	18.6
LCU	2.83	3.7	2.99	2.8
Register file	4.39	5.8	4.55	4.2
Other logic	0.92	1.2	1.81	1.7

comparing area breakdowns of hard-output and soft-output sphere detector realizations. The area of ICU and NEU represents ∼1/2 of the area required by the soft-output computation modules. LDU, RAU and MCU modules present the lowest area complexity.

8.3 Power Consumption

Power analysis is performed using Synopsis PrimeTime, based on switching activity annotated during post-synthesis simulation into a value change dump (VCD) file. Simulations[12] are carried out in Mentor ModelSim under typical case operating conditions, at $f_{\max} = 454$ MHz. The reported total power consumption is detailed in Table 5. As illustrated, memory consumes ∼30 % of the total power for SISO-TSD. A slightly lower proportion is observed regarding SO-TSD, since VLAMEM is not required. Concerning the TSD core, the observed power consumption results are in line with the obtained results on area. The greatest dissipation value corresponds to the soft-output computation modules (∼20 %), followed by ICU and NEU (∼10 %). Remaining entities consume <7 % of the total power. In comparison with the SO-TSD, those units forwarding (NEU, RAU, LDU) and/or processing (MCU, SAU) *a-priori* information (Section 7.2) consume ∼30 %–50 % more power. This represents an overall increase of ∼27 % for the SISO-TSD core, while ∼40 % higher power consumption is observed for the complete design (including memory), mainly contributed by the inclusion of VLAMEM.

[12]Operation of the SISO-TSD core is simulated for a sufficiently high number of input signals.

9 Results Discussion

In this work the first implementation of a SISO-TSD detector has been presented, combining the low complexity and good BER performance of the TSD algorithm with the high throughput of highly parallel and pipelined architectures. In order to ratify the efficiency of the proposed realization, a comparative analysis including relevant state-of-the-art tree-search-based MIMO detectors from literature [22, 27–29] is summarized in Table 6. Even though comparing BER performance of the considered detection algorithms is not the focus of this analysis, BER↔SNR operating points are included as reference.[13] It should be noticed that N_T and Q focused by some of the considered references differ from those considered in this work. For comparability reasons, results corresponding to 4 × 4 MIMO with 64-QAM modulation have been extracted from literature, as far as available. Note that area results extracted from [27] correspond to 4 × 4 MIMO with 64-QAM modulation, whereas $f_{\max}$ and $\overline{\tau}$ are only reported for 16-QAM. In [28] only 16-QAM is analyzed. Analogously, $It = 4$ has been considered for the SISO scenario except in [29], where only $It = 2$ is examined. It is additionally worth mentioning that in order to allow fair comparison of designs based on different implementation technologies, presented results have been appropriately scaled (as detailed in Table 6). For simplicity and comparability with results from literature, hardware cost of the input and output memory buffers (Section 7.1) is in the following disregarded.

9.1 Throughput

In order to provide reliable throughput comparison, throughput normalized to $f_{\max}$ ($\overline{\tau}/f_{\max}$) is considered. It should be noticed that the implementations proposed in this work as well as in [27] present variable throughput (depending on SNR), in contrast to realizations in [28, 29] and [22] (fixed throughput). Variable throughput is hence compared by considering average throughput at 10^{-5} BER (1 % FER in [27]) as indicated in Table 6. As illustrated, SO-TSD outperforms in terms of $\overline{\tau}/f_{\max}$ all the considered SO-detectors, with exception of [22]. In this regard, it should be noticed that [22] reports peak throughput. Considering SO-TSD

[13]It should be noticed that presented SNR↔BER-performance trade-offs differ not only depending on the detection algorithm, but also on dissimilarities of the considered communications system model (e.g. channel fading, channel correlation, decoder type, code rate, ...). For further details, the corresponding citations are referred.

Table 6 Design comparison of different MIMO detector realizations.

Detector	[this work] DF-SD (TSD, $T = 8$)		Witte et al. [27] DF-SD (STS)		Wu and Masera [28, 29] BF-FSD		Patel et al. [22] BF-SD (K-Best)
MIMO system	4×4		2×2–8×8		4×4		4×4
Modulation	64-QAM	64-QAM	QPSK, 16-,64-QAM		16-QAM	16-,64-QAM	64-QAM
Turbo it.	SO ($It = 1$)[5]	SISO ($It = 4$)	SO ($It = 1$)	SISO ($It = 4$)	SO ($It = 1$)[28]	SISO ($It = 2$)[29]	SO ($It = 1$)
BER↔SNR performance	BER 10^{-5} @ 14.1 dB	BER 10^{-5} @ 11.3 dB	FER 0.01 @ 15 dB	FER 0.01 @ 11.2 dB	BER 10^{-5} @ 6.5 dB	BER 10^{-5} @ 9.25 dB	BER 10^{-3} @ 27 dB[a]
Technology	65 nm	65 nm	90 nm	90 nm	130 nm	130 nm	65 nm
f_{max} (MHz)	454	454	379	250	400	384.6	833
$\overline{\tau}$ (Mbps)	403.6	40	40	5	213.3	279.7	2000
$\overline{\tau}/f_{max}$	0.89	0.09	0.11	0.02	0.53	0.73	2.4
Area (mm^2)	0.12	0.14	–		0.18 (0.045[b])	–	0.57
Area (kGE)	84	98	105	185	30	121	298
NHE[c] (kGE/Mbps)	0.21	2.45	1.90	26.72	0.07	0.22	0.15
Power[d] (mW)	58.2	73.9	–		–		239[e]
Energy / bit (pJ/b)	53.4	67.8					120[e]

[a] Uncoded system.
[b] Technology scaling by the factor $(130/65)^2$.
[c] Normalized Hardware Efficiency [15].
[d] Power consumption under typical case operating conditions (1.2 V, 25 °C).
[e] Patel et al. [22] reported 280 mW and 140 pJ/b @ 1.3 V, V_{dd} scaling by the factor $(1.3/1.2)^2$ is applied here.
DF Depth-First tree traversal, *BF* Breadth-First tree traversal, *FSD* Fixed Sphere Detector.

maximum throughput of 1.1 Gbps, a $\overline{\tau}/f_{max}$ ratio close to that of [22] is then obtained. Moreover, [22] achieves a very high throughput at the cost of ∼3 times greater gate count compared to the proposed TSD. Regarding SISO-realizations, $\overline{\tau}/f_{max}$ of SISO-TSD is ∼4.5 times better than that of [27]. It should be noticed that in [29] $It = 2$ is considered, hence achieving higher throughput than that of [27] and this work ($It = 4$). SISO-TSD with $It = 2$ achieves $\overline{\tau}/f_{max} \approx 0.3$, approaching that of [29].

9.2 Area

Table 6 compares the silicon area (in mm^2) as well as the gate count (in kGEs) corresponding to the considered detectors. Both SO-TSD and SISO-TSD consume less area (gate count) than considered realizations, with exception of [28]. In this case it should be noticed that [28] supports only 16-QAM modulation, while this work supports 64-QAM. Based on the overall computational complexity c_{total} depicted in Table 2 for 16- and 64-QAM, gate count of SISO-TSD can be roughly estimated[14] as $84 \times \frac{540}{340} \approx 53$ kGE for 16-QAM, very closely approaching the area of the efficient implementation proposed in [28].

[14]Estimation based uniquely in the overall computational complexity, disregarding register savings. Further area reduction should be expected if the latter was taken into account.

9.3 Normalized Hardware Efficiency (NHE)

In order to perform a fair comparison, the normalized hardware efficiency (NHE) is additionally examined. It is defined as the ratio between the core area (kGE) and the throughput (scaled to the same technology) [15]. SO-TSD presents a good efficiency ratio, outperforming the implementation in [27]. Patel et al. [22] and [28] present a slightly better ratio due to the consideration of peak throughput and the simpler modulation scheme (16-QAM) employed, respectively. Regarding SISO detection, TSD outperforms by far the efficiency ratio of [27]. Wu and Masera [29] presents a better ratio since only $It = 2$ is considered, leading to higher throughput than for $It = 4$. For SISO-TSD with $It = 2$, a more comparable ratio is observed (0.7 KGE/Mbps).

9.4 Power Consumption

Due to lack of results on power consumption reported in literature, comparison is here limited to SO-detector realizations. The architecture presented in [22] reaches a very high clock frequency (833 MHz) and inherently very high (peak) throughput (2000 Mbps), consequently leading to high power consumption. Corresponding SO-TSD and SISO-TSD power consumption is 4 and 3 times lower, respectively. Measurement results of a system on a chip (SoC) including the SO-TSD module can be found in [26].

9.5 SISO vs. SO Detectors

As explained in previous sections, the existing SO-TSD implementation [5] has been extended in this work in order to enable processing of *a-priori* information. In contrast to comparable SISO-detectors ([27, 29]), $f_{\max}$ of the proposed SISO-TSD realization is not degraded in comparison to the SO-TSD. Likewise, the area increase (14 kGE) due to processing of *a-priori* information is small (16 %) in comparison to the designs reported in literature (∼76 % in [27] and ∼50 % in [29]). Regarding the power consumption (and inherently the energy per bit), an increase of ∼27 % is observed. As additionally depicted, SISO-TSD achieves 10 times lower throughput than SO-TSD for the considered $T = 8$. It is nevertheless possible to achieve beyond 100 Mbps with $It = 4$ and more than 1 Gbps with $It = 1$, by adjusting T of SISO-TSD as described in Section 8.1. Additionally, strategies further reducing $\sharp n$ (i.e. further enhancing the throughput) have been recently proposed for the SISO-TSD in [24].

10 Conclusion and Future Work

In this work key strategies enabling the first efficient implementation of the SISO-TSD algorithm have been presented. A novel flexible 8-bit fixed-point representation has been utilized, significantly reducing the complexity compared to state-of-the-art implementations requiring >10 bits (e.g. [27]). Presented regularization and architectural concepts permit efficient application of pipelining and parallelization approaches. The costs of processing *a-priori* information have been additionally analyzed. Considering a 4×4 MIMO system with 64-QAM, the SISO-TSD implementation presents only 16 % larger area and 27 % higher power consumption than the previous SO-TSD realization, whereas a SNR gain of up to ∼3.2 dB is provided for $It = 4$. The proposed VLSI design incorporates sophisticated concepts to reduce the algorithm computational complexity as well as parallelization strategies for throughput enhancement. In combination with the definition of a suitable architecture, a high-throughput, low-hardware-complexity and low-power-consumption implementation is enabled. Resulting implementation has been shown to be well suited for iterative receiver architectures, outperforming in most aspects similar approaches (e.g. STS-SD) and achieving data rates comparable to or even greater than less complex and more parallelizable detection strategies (e.g. K-Best, FSD). The presented realization has in general demonstrated to reduce the hardware cost with regard to comparable state-of-the-art realizations. The clearly outstanding implementation results, together with scalability and parallelizability, make the proposed design a very favorable candidate for MIMO detection in a wide range of applications.

Future work will target further throughput improvement by exploiting SIMD vectorization and possibly increasing frequency by optimizing the critical path. In addition to this, architecture optimization for further reduction of area and power consumption are planned.

Acknowledgements This work has been supported by the German Federal Ministry of Education and Research (BMBF) under grant 13N11810.

References

1. 3GPP *LTE, evolved universal terrestrial radio access (E-UTRA), base station (BS) radio transmission and reception*. Technical Specification 36.104 Release 10. http://www.3gpp.org.
2. 3GPP *LTE, Evolved universal terrestrial radio access (E-UTRA), user equipment (UE) radio transmission and reception*. Technical Specification 36.101 Release 10. http://www.3gpp.org.
3. Adeva, E.P., Mennenga, B., Fettweis, G. (2011). Scalable ASIP implementation and parallelization of a MIMO sphere detector. In *Proceedings of the 11th international conference on embedded computer systems: architectures, modeling, and simulation (SAMOS XI)*. Samos, Greece.
4. Adeva, E.P., Mennenga, B., Fettweis, G. (2011). Survey on an efficient, low-complex tuple search based sphere detector. In *Proceedings of the ieee 34th sarnoff symposium*. Princeton, USA.
5. Adeva, E.P., Shah, M.A., Mennenga, B., Fettweis, G. (2011). VLSI architecture for soft-output tuple search sphere decoding. In *Proceedings of the IEEE workshop on signal processing systems (SIPS'11)*. Beirut, Lebanon.
6. Burg, A., Borgmann, M., Wenk, M., Zellweger, M., Fichtner, W., Bölcskei, H. (2005). VLSI implementation of MIMO detection using the sphere decoding algorithm. *IEEE Journal of Solid-State Circuits, 40*, 1566–1577.
7. Burg, A., Wenk, M., Fichtner, W. (2006). VLSI implementation of pipelined sphere decoding with early termination. In *Proceedings of the 14th european signal processing conference, 2006 (EUSIPCO 2006)*. Florence, Italy.
8. Cichon, G. (2004). *A novel compiler-friendly micro-architecture for rapid development of high-performance and low-power DSPs*. Dissertation, Technische Universität Dresden.
9. de Jong, Y., & Willink, T. (2005). Iterative tree search detection for MIMO wireless systems. *IEEE Transactions on Communications, 53*(6), 930–935.
10. Hagenauer, J. (2002). The turbo principal in mobile communications. In *Proceedings of the international symposium on information theory and its applications (ISITA'02)*. Xi'an, China.

11. Hochwald, B., & ten Brink, S. (2003). Achieving near-capacity on a multiple-antenna channel. *IEEE Transactions on Communications, 51*, 389–399.
12. Jaldén, J., & Ottersten, B. (2005). Parallel implementation of a soft output sphere decoder. In *Proceedings of asilomar conference on signals, systems, and computers*.
13. Lee, E., & Messerschmitt, D. (1987). Pipeline interleaved programmable DSP's: architecture. *IEEE Transactions on Acoustics, Speech and Signal Processing, 35*, 1320–1333.
14. Lee, J. (2006). Area efficient pipelined vlsi implementation of list sphere decoder. In *Proceedings of the Asia-Pacific Conference on Communications, 2006 (APCC '06)*. Busan, Korea.
15. Mahdavi, M., & Shabany, M. (2012). Novel MIMO detection algorithm for high-order constellations in the complex domain. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*. doi:10.1109/TVLSI.2012.2196296.
16. Mennenga, B. (2010). *Aufwandsgünstige Detektion in Mehrantennensystemen mittels komplexitätsreduzierter Baumsuchverfahren*. Dissertation, Technische Universität Dresden.
17. Mennenga, B., von Borany, A., Fettweis, G. (2009). Complexity reduced soft-in soft-out sphere detection based on search tuples. In *Proceedings of the IEEE international conference on communications (ICC'09)*. Dresden, Germany.
18. Mennenga, B., & Fettweis, G. (2009). Search sequence determination for tree search based detection algorithms. In *Proceedings of the IEEE sarnoff symposium 2009*. Princeton, USA.
19. Mennenga, B., Fritzsche, R., Fettweis, G. (2009). Iterative soft-in soft-out sphere detection for MIMO systems. In *Proceedings of the IEEE 69th vehicular technology conference, (VTC'09-Spring)*. Barcelona, Spain.
20. Mennenga, B., Matus, E., Fettweis, G. (2009). Vectorization of the sphere detection algorithm. In *Proceedings of the IEEE international symposium on circuits and systems (ISCAS'09)*. Taipei, Taiwan.
21. Myllyla, M., Cavallaro, J.R., Juntti, M. (2010). Architecture design and implementation of the metric first list sphere detector algorithm. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 19*, 895–899.
22. Patel, D., Smolyakov, V., Shabany, M., Gulak, P. (2010). VLSI iumplementation of a WiMAX/LTE Compliant low-complexity high-throughput soft-output K-best MIMO detector. In *Proceedings of the IEEE international symposium on circuits and systems 2010, (ISCAS'10)*.
23. Robertson, P., Villebrun, E., Hoeher, P. (1995). A comparison of optimal and sub-optimal map decoding algorithms operating in the log domain. In *Proceedings of the IEEE international conference on communications, 1995 (ICC '95)*.
24. Seifert, T., Adeva, E.P., Fettweis, G. (2013). Towards complexity-reduced soft-input soft-output sphere detection. In *Submitted to the 9th international itg conference on systems, communications and coding 2013, (SCC'13)*. Munich, Germany.
25. Studer, C., Burg, A., Bölcskei, H. (2008). Soft-output sphere decoding: algorithms and VLSI implementation. *IEEE Journal on Selected Areas in Communications, 26*, 290–300.
26. Winter, M., Kunze, S., Adeva, E.P., Mennenga, B., Matus, E., Fettweis, G., Eisenreich, H., Ellguth, G., Höppner, S., Scholze, S., Schüffny, R., Kobori, T. (2012). A 335Mb/s 3.9mm^2 65nm CMOS flexible MIMO detection-decoding engine achieving 4g wireless data rates. In *Proceedings of the 59th international solid-state circuits conference (ISSCC'12)*. San Francisco, USA.
27. Witte, E.M., Borlenghi, F., Ascheid, G., Leupers, R., Meyr, H. (2010). A scalable VLSI architecture for soft-input soft-output single tree-search sphere decoding. *IEEE Transactions on Circuits and Systems II: Express Briefs, 57*, 706–710.
28. Wu, B., & Masera, G. (2010). A novel VLSI architecture of fixed-complexity sphere decoder. In *13th euromicro conference on digital system design: architectures, methods and tools (DSD)*.
29. Wu, B., & Masera, G. (2012) Efficient VLSI implementation of soft-input soft-output fixed-complexity sphere decoder. *Institution of Engineering and Technology (IET) Communications, 6*, 1111–1118.
30. Wübben, D., Böhnke, R., Rinas, J., Kühn, V., Kammeyer, K. (2001). Efficient algorithm for decoding layered space-time codes. *Electronics Letters, 37*, 1348–1350.
31. Yee, M.S. (2005). Max-log-MAP sphere decoder. In *Proceedings of the IEEE international conference on acoustics, speech, and signal processing (ICASSP '05)*, (Vol. 3).
32. Zimmermann, E., & Fettweis, G. (2006). Unbiased MMSE tree search detection for multiple antenna systems. In *Proceedings of the International symposium on wireless personal multimedia communications (WPMC'06)*. San Diego, USA.

Esther P. Adeva received her Dipl.-Ing. in Electrical Engineering from the Miguel Hernández University (Spain) in June 2009. She developed her diploma thesis at the Technische Universität Dresden, focusing on the analysis and implementation of tree-search-based MIMO detection algorithms. She is currently pursuing a PhD at the Vodafone Chair and she continues her research on the exploration of efficient architectures for MIMO detection.

Tobias Seifert received his Dipl.-Ing. in Electrical Engineering from the Technische Universität Dresden in January 2012. In his diploma thesis he dealt with implementation aspects of MIMO sphere-search-based detectors for iterative detection-decoding. He is currently pursuing a PhD at the Vodafone Chair and continues his research on strategies for efficient MIMO detection implementation.

Gerhard Fettweis earned his Ph.D. from RWTH Aachen in 1990. Thereafter he was at IBM Research and TCSI Inc., California. Since 1994 he is Vodafone Chair Professor at TU Dresden, Germany, with main research interest on wireless transmission and chip design. He is IEEE Fellow and an honorary doctorate of TU Tampere.