A Parallel MCMC-Based MIMO Detector: VLSI Design and Algorithm

Dominik Auras, Uwe Deidersen, Rainer Leupers, Gerd Ascheid

A Parallel MCMC-based MIMO Detector: VLSI Design And Algorithm

Dominik Auras, Uwe Deidersen, Rainer Leupers, and Gerd Ascheid

Institute for Communication Technologies and Embedded Systems,
RWTH Aachen University, 52056 Aachen, Germany
auras@ice.rwth-aachen.de
http://www.ice.rwth-aachen.de

Abstract. Stochastic detection for multi-antenna (MIMO) systems promises communications performance close to max-log detection for certain SNR regimes, especially when the system iterates between detector and channel decoder following the Turbo Principle. In this work, we propose a parallel VLSI architecture for soft-input soft-output Markov chain Monte Carlo based stochastic MIMO detection. It features run-time adaptibility to varying channel conditions, effectively allowing us to adjust the invested effort. Besides the details of our area-throughput efficient design, like the low-level algorithm and micro-architecture design, we also provide an extensive data set from our experiments regarding the detector's communications performance and relate it to our VLSI implementation results. The provided data analysis highlights the architecture's run-time adaptibility and demonstrates how we can trade off throughput for improved communications performance.

1 Introduction

With the wide-spread adoption of MIMO (multi-antenna) technology in current and future wireless communication systems, such as those based on the IEEE 802.11n standard [1], academia and industry are searching for MIMO detectors with reasonable implementation complexity and algorithmic performance. Especially for systems using bit-interleaved coded modulation with iterative decoding (BICM-ID) [2], a major challenge for VLSI implementation is the required soft-input soft-output (SISO) detector, since optimal detection has an exponential complexity.

Iterative MIMO decoding can yield impressive algorithmic performance gains in terms of significantly reduced signal-to-noise ratio (SNR) requirements to achieve a certain fixed error rate [3]. This SNR gain has several possible uses amongst others: we can extend the transmission range, we can serve more users (i.e. tolerating more interference), we can lower the transmission power to save energy (and at the same time reduce interference to other users), or transmit at a higher throughput in the same bandwidth.

Possible detectors can be roughly put into two categories: linear detectors, e.g. MMSE-filter based [4–6], and non-linear detectors e.g. [3, 7–9]. Basically, linear detectors try to suppress noise using linear filtering, then decode the estimate.

In contrast to this, non-linear detectors perform a search, e.g. a randomly guided one, in the space of possibly transmitted data vectors. Stochastic detection based on Markov chain Monte Carlo (MCMC) methods [10] belongs to this class. It enables small configurable detectors that can cover a large design space. Furthermore, when iterating between detector and channel decoder, MCMC detection shows a communications performance close to max-log detection for certain SNR regimes [10].

To date, only some research effort has been directed towards this field. There exist only a handful of publications on MCMC detector architectures at the moment [11–14]. None of them correlates communications performance with VLSI implementations results.

Related Work An MCMC-based SISO MIMO detector ASIC design supporting independent parallel Gibbs Samplers is presented in [12]. Amongst other things, [12] introduces an initialization scheme for the completely recursive, and thus simplified, computation of the detector states, and shows how to reuse the circuitry to draw independent first samples. However, a multiplier in the timing critical path yields a limited throughput and a relatively large area consumption.

In [11], the authors propose an MCMC-based SISO MIMO detector architecture mapped on an FPGA. It features one multiplier-free Gibbs Sampler pipelined at the symbol vector level. The architecture uses a simple recursive metric computation, but requires one dot-product per cycle. The first sample of every chain needs to be generated externally.

The hybrid soft-output only MCMC detector architecture [14] combined with a hard-output fixed-complexity sphere detector (FSD) features parallel multiplier-free Gibbs Samplers that start with the best candidates found by the FSD. However, the design requires the QR-decomposition of the channel matrix, and the results are only given in terms of operation counts.

Contribution We present a complete redesign of the MCMC-based MIMO detector architecture presented in [12], with multiplier-free Gibbs Samplers and further architectural improvements that result in a significant area reduction and timing improvement. Post-layout area and clock period reduce by about 50% and 40% respectively. In extension to our previous publication [13], we additionaly provide our detector's communications performance results and present an analysis showing how to trade off throughput for improved communications performance at run-time.

Outline First, we introduce the general concept of MCMC-based MIMO detection (Sec. 3), describe the implemented algorithm (Sec. 4), then we propose the redesigned architecture (Sec. 5). Subsequently, we explicitly highlight the differences to the reference design [12] in Sec. 6. Our implementation results are presented in Sec. 7. The analysis of the communications performance results is explained in Sec. 8.

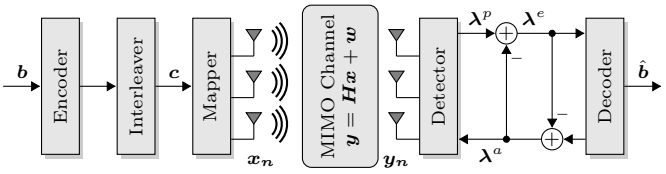


Fig. 1. Assumed MIMO BICM-ID System Model. Detector and decoder iteratively exchange information to improve the final decoding result.

2 System Model

We consider a spatial-multiplexing $N_t \times N_r$ MIMO system with BICM-ID, as depicted in Fig. 1. A message $\boldsymbol{b} \in \{0,1\}^{N_b}$ is encoded with rate $r = N_b/N_c$ and interleaved, yielding the code word $\boldsymbol{c} \in \{0,1\}^{N_c}$. Let $\mathcal{X} \subset \mathbb{C}$ be a modulation alphabet with $K = \log_2 |\mathcal{X}|$ bits per symbol. The code word is partitioned into multiple subvectors $\boldsymbol{c}_n \in \{0,1\}^{KN_t}$. They are subsequently mapped to symbol vectors $\boldsymbol{x}_n \in \mathcal{X}^{N_t}$ that are transmitted independently. Assuming a frequency-flat fading channel characterized by $\boldsymbol{H}_n \in \mathbb{C}^{N_r \times N_t}$, the received symbol vector at time n is $\boldsymbol{y}_n = \boldsymbol{H}_n \boldsymbol{x}_n + \boldsymbol{w}_n$ where $\boldsymbol{w}_n \in \mathbb{C}^{N_r}$ is a white Gaussian noise process with $\mathbb{E}[\boldsymbol{w}_n \boldsymbol{w}_n^H] = N_0 \boldsymbol{I}_{N_r}$. In the remainder, the time index n is dropped for convenience. Using iterative MIMO decoding following the Turbo Principle [15], detector and channel decoder exchange extrinsic information $\boldsymbol{\lambda}^e = \boldsymbol{\lambda}^p - \boldsymbol{\lambda}^a$ in terms of log-likelihood ratios (LLRs), where $\boldsymbol{\lambda}^p$ are the detector's posterior LLRs and $\boldsymbol{\lambda}^a$ are the prior LLRs fed back from the decoder.

3 MCMC-based MIMO Detection

The Markov chain Monte Carlo based MIMO detector class that we consider performs a randomly guided search in the space $\boldsymbol{c} \in \{0,1\}^{KN_t}$. It starts with a random candidate, then walks around randomly. On its way, it evaluates and saves metric values of the current candidates, which are later used to approximate the posterior LLRs. The random process (Monte Carlo) from which it draws new candidates evolves recursively (Markov chain). By design the search converges towards candidates of high probability [10].

We select independent first samples $\boldsymbol{c}^{(q,0)} \in \{0,1\}^{KN_t}$, one per chain $q = 1 \ldots N_q$, either randomly from the prior distribution $\boldsymbol{c}^{(q,0)} \sim p(\boldsymbol{c}) = f(\boldsymbol{\lambda}^{\boldsymbol{a}})$ or given by an external hard-output detector $\boldsymbol{c}^{(q,0)} = \boldsymbol{c}^{\text{ext}}$ (usually for at most one chain). Every sample $s = 1 \ldots N_s$ is drawn in KN_t steps. The algorithm sequentially replaces every bit with 0 and 1, computes the metric for those two candidates, then selects one of them as the next partial sample.

Let $\varphi : \{0,1\}^{N_t K} \mapsto \mathcal{X}^{N_t}$ be a rule that maps bit labels onto symbol vectors $\boldsymbol{x} \in \mathcal{X}^{N_t}$. We define the metric

$$\mu(\boldsymbol{c}) = -\frac{1}{N_0} \|\boldsymbol{y} - \boldsymbol{H}\varphi(\boldsymbol{c})\|^2 - \boldsymbol{c}^T \boldsymbol{\lambda}^a \tag{1}$$

for the candidate $\boldsymbol{c} \in \{0,1\}^{KN_t}$, which is related to the posterior probability $P(\boldsymbol{c}|\boldsymbol{y}, \boldsymbol{H}, \boldsymbol{\lambda}^a)$. Furthermore, let

$$\boldsymbol{c}_{b\beta} = (c_1, \ldots, c_{b-1},\ \beta\ , c_{b+1}, \ldots, c_{KN_t}) \tag{2}$$

be the vector $\boldsymbol{c}$ with the b-th bit replaced by β. The detector approximates the posterior LLRs as

$$\lambda_b^p \approx \max_{q,s} \mu(\boldsymbol{c}_{b0}^{(q,s)}) - \max_{q,s} \mu(\boldsymbol{c}_{b1}^{(q,s)}) \tag{3}$$

where we search for the two maxima for every bit over all chains and samples.

4 Low-Level Algorithm

The presented algorithm implements the max-log variant of the Rao-Blackwellized MCMC detection algorithm with uniform sampling described in [10]. Its basic idea is to recursively compute the metric in Eq. (1) by tracking the changes while drawing bits [12]. First, we introduce the basic concepts required for understanding the algorithm, then describe the algorithm in detail. For the theoretic background, the reader is kindly referred to [10, 12].

4.1 Basic Concepts

Matched Filter The algorithm in [12] replaces $\boldsymbol{H}$ with the Gram matrix $\boldsymbol{R} = \boldsymbol{H}^H\boldsymbol{H}$ and the received symbol vector $\boldsymbol{y}$ with the matched filter output $\boldsymbol{y}^{\mathrm{mf}} = \boldsymbol{H}^H\boldsymbol{y}$ in the metric. This does not influence the posterior LLR calculation, however it allows to use the symmetry $\boldsymbol{R} = \boldsymbol{R}^H$.

Gibbs Sampler (GS) We realize the Markov chains with Gibbs Sampling. To this end, the GS draw bits sequentially according to an approximation of the marginal distribution $P(c_b|c_1, \ldots, c_{b-1}, c_{b+1}, \ldots, c_{KN_t})$. The state of the q-th GS at the s-th sample after drawing the b-th bit is denoted as

$$\boldsymbol{c}_b^{(q,s)} = (c_1^{(q,s)}, \ldots, c_b^{(q,s)}, c_{b+1}^{(q,s-1)}, \ldots, c_{KN_t}^{(q,s-1)}) \tag{4}$$

and thus contains bits from the previous sample $\boldsymbol{c}^{(q,s-1)}$ and the current sample $\boldsymbol{c}^{(q,s)}$.

Common Starting Point All chains start with $\boldsymbol{c}^{(-1)}$, which maps onto $\boldsymbol{x}^{(-1)}$ with $x_t = 1 + j$, i.e. we have $\varphi(\boldsymbol{c}^{(-1)}) = \boldsymbol{x}^{(-1)}$. This concept enables the initialization of parallel independent Gibbs Samplers [12].

Symbol Deltas When the GS state changes, at most one bit is different. We introduce the notation

$$\begin{aligned} |\Delta|_b^2(\boldsymbol{c}) &= |\varphi_n(\boldsymbol{c}_{b1})|^2 - |\varphi_n(\boldsymbol{c}_{b0})|^2 \\ \Delta_b(\boldsymbol{c}) &= \varphi_n(\boldsymbol{c}_{b1}) - \varphi_n(\boldsymbol{c}_{b0}) \end{aligned} \tag{5}$$

where φ_n is the mapping rule for the n-th antenna, and the b-th bit belongs to the n-th antenna.

Recursive Dot-Product The algorithm tracks the current value of

$$\boldsymbol{S} = \boldsymbol{y}^{\mathrm{mf}} - \tilde{\boldsymbol{R}}\varphi(\boldsymbol{c}_b^{(q,s)}) \tag{6}$$

where $\tilde{\boldsymbol{R}}$ is the matrix $\boldsymbol{R}$ with the diagonal set to zero. Starting from $\boldsymbol{S}^{(-1)} = \boldsymbol{y}^{\mathrm{mf}} - \tilde{\boldsymbol{R}}\boldsymbol{x}^{(-1)}$, it updates $\boldsymbol{S}$ recursively when $\boldsymbol{c}_b^{(q,s)}$ changes.

Recursive Metric Computation We introduce an arbitrary offset such that $\mu(\boldsymbol{c}^{(-1)}) = 0$, which cancels out in Eq. (3). Let the distance update be

$$\delta_b^{(q,s)} = \mathrm{Re}\{r_{nn}\}|\Delta|_b^2(\boldsymbol{c}^{(q,s-1)}) - 2\mathrm{Re}\{S_n^*\Delta_b(\boldsymbol{c}^{(q,s-1)})\} \tag{7}$$

where the b-th bit belongs to the n-th antenna, then the metric update is

$$\Delta\mu = \frac{1}{N_0}\delta_b^{(q,s)} + \lambda_b^a \tag{8}$$

which we either subtract from or add to the current metric $\mu(\boldsymbol{c}^{(q,s)})$, depending on the bit flip direction, if the b-th bit changes.

Log-domain Bit Probability The term

$$\gamma = \frac{1}{\eta N_0}\delta_b^{(q,s)} + \lambda_b^a \tag{9}$$

expresses the probability of the next bit being 1 in the log-domain, where the temperature parameter η mitigates lock-in effects in the high-SNR regime [10]. For the conversion to the linear domain, we apply a piece-wise linear approximation to logistic$(\gamma) = 1/(1+e^{-\gamma})$ as in [11,12]. To this end, the GS simply limits γ to the range $[-4, 4)$ and compares $-\gamma$ to a uniformly distributed pseudo-random number $u \sim U(-4, 4)$ in the same range.

4.2 Overall Algorithm Design

Fig. 2 depicts the algorithm partitioned into four different parts: the *Front-end Processing* (FEP), that transforms the channel observations, the parallel *Gibbs Samplers* (GS) realizing the Markov chains, the *Metric Update* (M) tracking the current metric state, and the *LLR Computation*, which searches for the two maximum metric values per bit.

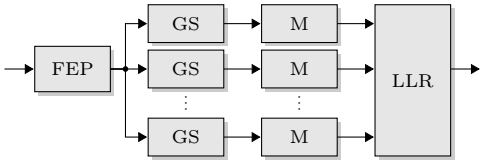


Fig. 2. Partitioning of the low-level algorithm: Front-end Processing, Gibbs Sampler, Metric Update, LLR Computation

4.3 Front-end Processing

First, choose $\Gamma = 2^{\alpha}/(\eta N_0)$ with α such that $\Gamma \in [0.5, 1)$. We assume $\eta = 2$. The FEP computes

$$\begin{aligned} \boldsymbol{R} &= \Gamma \boldsymbol{H}^H \boldsymbol{H} \\ \boldsymbol{S}^{(-1)} &= \Gamma \boldsymbol{H}^H \boldsymbol{y} - \tilde{\boldsymbol{R}} \boldsymbol{x}^{(-1)} \end{aligned} \tag{10}$$

as described in Sec. 4.1 but scaled by Γ.

4.4 Gibbs Sampler

Alg. 1 describes how the GS sequentially draws bits of the candidate sequence $\boldsymbol{c}^{(q,s)}$. GS and Metric Update share the term $\delta_b^{(q,s)}$ computed in line 6. Note the back-shifting with α to compensate the normalization of Γ. For the first sample ($s = 0$), only the prior LLRs are used, in order to draw $\boldsymbol{c}^{(q,0)} \sim \boldsymbol{\lambda}^a$ (line 7). The saturation in line 8 produces a threshold in the range $[-4, 4)$ (cf. Sec. 4.1). The comparison to a uniformly distributed pseudo-random number in the same range (line 13) yields the new bit value. Afterwards, we need to update the $\boldsymbol{S}$ state (lines 14-16).

Algorithm 1: Gibbs Sampler

input: $\boldsymbol{S}^{(-1)}, \boldsymbol{R}, \boldsymbol{c}^{\text{ext}}, \boldsymbol{\lambda}^a$, Chain Index q
output: $c_b^{(q,s)}, c_b^{(q,s-1)}, \delta_b^{(q,s)}$

1 $\boldsymbol{c}^{(q,-1)} = \boldsymbol{c}^{(-1)}$
2 $\boldsymbol{S} \leftarrow \boldsymbol{S}^{(-1)}$
3 **for** $s = 0$ **to** N_s **do**
4 **for** $b = 1$ **to** $N_t K$ **do**
5 $n \leftarrow \lfloor (b-1)/K \rfloor + 1$
6 $\delta_b^{(q,s)} = \left[\mathrm{Re}\{r_{nn}\} |\Delta|_b^2(\boldsymbol{c}^{(q,s-1)}) - 2\mathrm{Re}\{S_n^* \Delta_b(\boldsymbol{c}^{(q,s-1)})\} \right] 2^{-\alpha}$
7 $\gamma \leftarrow \lambda_b^a + \begin{cases} 0 & \text{if } s = 0 \\ \delta_b^{(q,s)} & \text{otherwise} \end{cases}$
8 $\gamma \leftarrow \mathrm{saturate}(-4, 4, \gamma)$
9 draw $u \sim U(-4, 4)$
10 **if** $s = 0$ **and** $q = 1$ **then** /* first sample, first chain */
11 $c_b^{(q,s)} = c_b^{\text{ext}}$
12 **else**
13 $c_b^{(q,s)} = \mathrm{sign}(u + \gamma)$
14 $\Delta S_t \leftarrow r_{tn} \Delta_b(\boldsymbol{c}^{(q,s-1)}) \quad \forall\, t = 1 \ldots N_t,\ t \neq n$
15 **if** $c_b^{(q,s-1)} \neq c_b^{(q,s)}$ **then**
16 $S_t \leftarrow S_t + \begin{cases} \Delta S_t & \text{if } c_b^{(q,s-1)} = 1 \\ -\Delta S_t & \text{if } c_b^{(q,s-1)} = 0 \end{cases} \forall\, t \neq n$

4.5 Metric Update

Alg. 2 recursively computes the current candidate's metric $\mu(\boldsymbol{c}_b^{(q)})$, using the state $\mu^{(q)}$, and produces the two metrics for the current bit $\mu(\boldsymbol{c}_{b0/1})$. As stated earlier, we arbitrarily set the metric for the common starting point to zero (line 1). Lines 4 to 9 show the underlying metric update. Of the two possible states, one is identical to the current state, and thus has the same metric value (line 4). The other one is updated according to the direction of the bit flip (lines 6 and 8). In line 9, we select one of the two as the new current metric. It remains unaltered if the bit does not change.

Algorithm 2: Metric Update

input: $c_b^{(q,s)}, c_b^{(q,s-1)}, \delta_b^{(q,s)}, \boldsymbol{\lambda}^a$, Chain Index q
output: $\mu(\boldsymbol{c}_{b0}^{(q,s)}), \mu(\boldsymbol{c}_{b1}^{(q,s)})$

1 $\mu^{(q)} \leftarrow 0$
2 **for** $s = 0$ **to** N_s **do**
3 **for** $b = 1$ **to** $N_t K$ **do**
4 $\mu(\boldsymbol{c}_{b0}^{(q,s)}) = \mu(\boldsymbol{c}_{b1}^{(q,s)}) = \mu^{(q)}$
5 **if** $c_b^{(q,s-1)} = 0$ **then**
6 $\mu(\boldsymbol{c}_{b1}^{(q,s)}) = \mu(\boldsymbol{c}_b^{(q,s-1)}) - (\eta\delta_b^{(q,s)} + \lambda_b^a)$
7 **else**
8 $\mu(\boldsymbol{c}_{b0}^{(q,s)}) = \mu(\boldsymbol{c}_b^{(q,s-1)}) + (\eta\delta_b^{(q,s)} + \lambda_b^a)$
9 $\mu(\boldsymbol{c}_b^{(q,s)}) = \begin{cases} \mu(\boldsymbol{c}_{b0}^{(q,s)}) & \text{if } c_b^{(q,s)} = 0 \\ \mu(\boldsymbol{c}_{b1}^{(q,s)}) & \text{if } c_b^{(q,s)} = 1 \end{cases}$
10 $\mu^{(q)} \leftarrow \mu(\boldsymbol{c}_b^{(q,s)})$

4.6 LLR Computation

Alg. 3 searches for the maximum metrics among all chains, then compares these local maxima with the current global maxima. It excludes the $s = 0$ step, which is the transition from $\boldsymbol{c}^{(-1)}$ to $\boldsymbol{c}^{(q,0)}$, from the search (line 3). The computation of the extrinsic LLRs in line 7 is included, as it can be easily implemented in hardware.

Algorithm 3: LLR Computation

input: $\mu(\boldsymbol{c}_{b0}^{(q,s)}), \mu(\boldsymbol{c}_{b1}^{(q,s)}), \boldsymbol{\lambda}^a$
output: $\boldsymbol{\lambda}^e$

1 $\mu_{b0}^{\max} \leftarrow -\infty \quad \forall\, b = 1 \ldots N_t K$
2 $\mu_{b1}^{\max} \leftarrow -\infty \quad \forall\, b = 1 \ldots N_t K$
3 **for** $s = 1$ **to** N_s **do** /* Note: ignore input for $s = 0$ */
4 **for** $b = 1$ **to** $N_t K$ **do** /* For every bit index */
5 $\mu_{b0}^{\max} \leftarrow \max(\mu_{b0}^{\max}, \max_q(\mu(\boldsymbol{c}_{b0}^{(q,s)})))$
6 $\mu_{b1}^{\max} \leftarrow \max(\mu_{b1}^{\max}, \max_q(\mu(\boldsymbol{c}_{b1}^{(q,s)})))$
7 $\lambda_b^e = \mu_{b0}^{\max} - \mu_{b1}^{\max} - \lambda_b^a \quad \forall\, b = 1 \ldots K N_t$

5 VLSI Architecture

5.1 Overview

The macro pipeline of FEP-Circuit and MCMC core, shown in Fig. 3, constitutes the proposed MCMC detector. Both components require multiple clock cycles per input vector, but double buffering between FEP and Core ensures that the computations can overlap. The MCMC core in turn contains four stages connected via registers. The stages exchange information in every clock cycle. They effectively run in a pipeline manner.

The FSM and the multiplexers (e.g. λ_b^a, and for the column of $\boldsymbol{R}$) are part of the Mux stage. There are N_p GS-Circuits implementing Alg. 1. For every GS-Circuit, there is one corresponding M-Circuit executing Alg. 2. The L-Circuit performs the LLR Computation in Alg. 3. Every GS/M-Circuit can run several chains sequentially. For example $N_q = 8$ chains can be run on $N_p = 4$ GS/M-Circuits by executing two chains sequentially per GS/M-Circuit. We can also turn off some GS/M-Circuits, e.g. run $N_q = 4$ chains on $N_p = 8$ GS/M-Circuits with four inactive circuits.

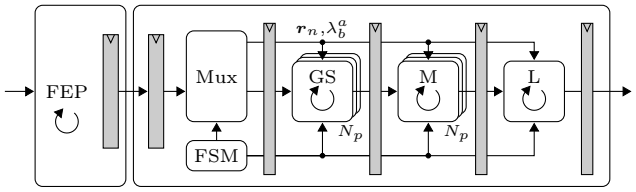


Fig. 3. Architecture design of the MCMC detector. The n-th column $\boldsymbol{r}_n$ of $\boldsymbol{R}$ and λ_b^a are selected in the Mux stage.

5.2 FEP-Circuit

The architecture, depicted in Fig. 4, contains in total five multipliers. Using four of these, the dot-product for the terms $\boldsymbol{H}^H\boldsymbol{y}$ and $\boldsymbol{R} = \boldsymbol{H}^H\boldsymbol{H}$ requires N_r cycles per complex entry. We need only the lower triangular of $\boldsymbol{R}$ due to $\boldsymbol{R}^H = \boldsymbol{R}$. The architecture computes either one complex off-diagonal entry, or two real diagonal entries in parallel. The fifth multiplier alternatingly multiplies real and imaginary parts with $\Gamma = \frac{2^\alpha}{N_0\eta}$. In parallel, we multiply the entries of $\boldsymbol{R}$ with $x_t^{(-1)} = 1 + j$ (cf. Sec. 4.1) using only adders and multiplexers, and accumulate the results to obtain $\boldsymbol{S}$.

5.3 GS/M-Circuit

Fig. 5 depicts the GS-Circuit. The $|\Delta|^2$-multiplier, depicted in detail in Fig. 6(a), exploits the limited range of $|\Delta|^2 \in \{-3, -2, \ldots, 3\} \times \{8, 16\}$ which assumes only 14 different values for 4-/16-/64-QAM. The factor Δ is either

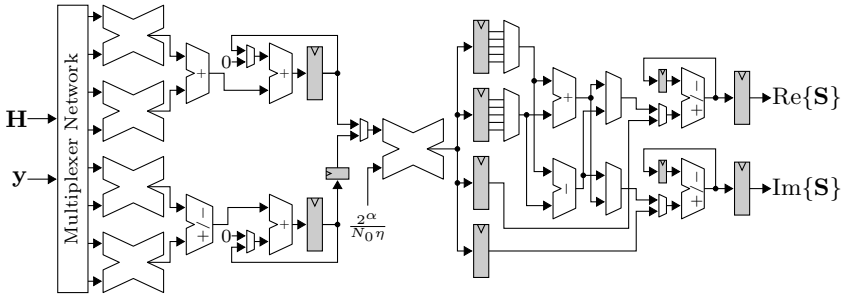


Fig. 4. FEP-Circuit

purely real or imaginary. We define $|\Delta| = |\mathrm{Re}\{\Delta\}| + j|\mathrm{Im}\{\Delta\}|$. Then we have $\mathrm{Re}\{S_n^*|\Delta|\} = \mathrm{Re}\{S_n\}\mathrm{Re}\{|\Delta|\} + \mathrm{Im}\{S_n\}\mathrm{Im}\{|\Delta|\}$. For 4-/16-/64-QAM this assumes only the four values $\{1, 3, 5, 7\} \times 2$, which greatly simplifies the Δ-multiplier (Fig. 6(b), only shifts, adders and multiplexers). The control of the subsequent adder-subtractor ① considers if $\Delta < 0$ and if Δ is imaginary to decide whether to add or subtract. To generate the independent first samples, the multiplexer ② ensures $\gamma = \lambda_b^a$. For the external initialization, we have the multiplexer ③ that selects $c_b^{(q,s)} = c_b^{\mathrm{ext}}$. The circuit uses a 32-bit maximum length Galois-LFSR that generates one 32-bit word per clock cycle. The timing critical path of the whole MCMC detector starts in the $|\Delta|^2$-control, goes through the multiplexers in the $|\Delta|^2$-multiplier towards $c_b^{(q,s)}$, then finishes in the write-enable control for the $\boldsymbol{S}$ registers.

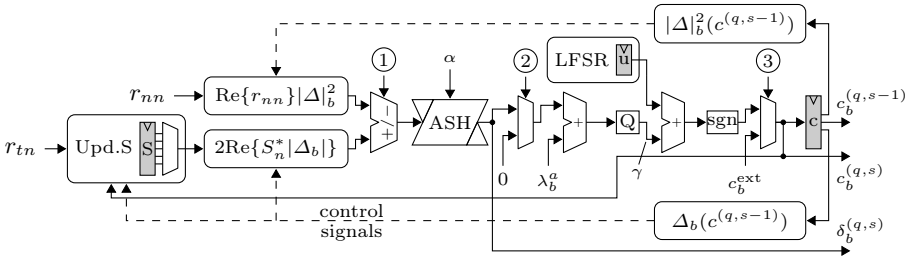
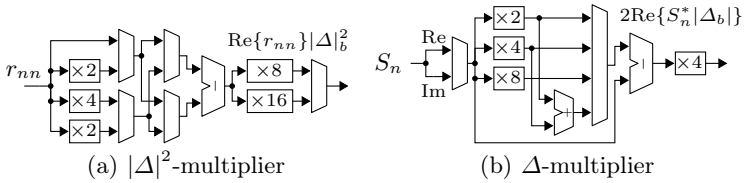


Fig. 5. GS-Circuit. The arithmetic shifter (ASH) reverts the normalization of Γ.



(a) $|\Delta|^2$-multiplier (b) Δ-multiplier

Fig. 6. Detailed view of the simplified multipliers

The M-Circuit, shown in Fig. 7, implements Alg. 2 using a write-enabled register for the current metric, which is updated when we flip the current bit. The multiplication with η is implemented as a constant shift.

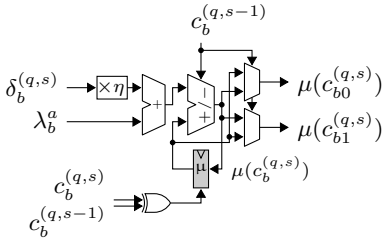


Fig. 7. M-Circuit

Update-S-Circuit The Update-S-Circuit shown in Fig. 8 has (N_t-1) complex-valued Δ-multipliers, i.e. $2(N_t-1)$ times Fig. 6(b). Using the multiplexers ③ and ④, we can update all N_t elements of $\boldsymbol{S}$, however only N_t-1 change per clock cycle. The entries of $\boldsymbol{R}$ e.g. r_{1n}, r_{2n} are selected in the Mux stage. Similar to the GS-Circuit, the adder-subtractor control ① considers $\Delta < 0$, if $|\Delta|$ is imaginary, and additionally the old bit $c_b^{(q,s-1)}$ and if the input needs to be conjugated, i.e. $\mathrm{Im}\{r_{tn}\} = -\mathrm{Im}\{r_{nt}\}$. The write-enabled $\boldsymbol{S}$ registers are updated if the current bit flips. This control ② is part of the aforementioned critical path.

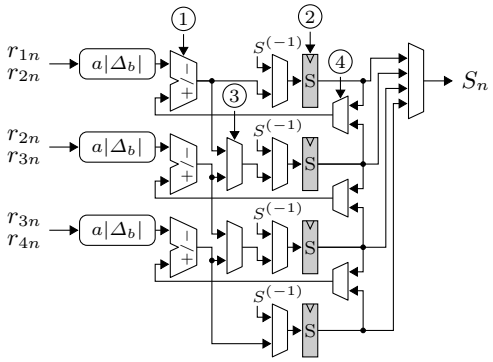


Fig. 8. Update-S-Circuit. Example for $N_t = 4$ antennas. All units exist for the real and for the imaginary parts respectively (not drawn).

5.4 L-Circuit

The L-Circuit shown in Fig. 9 contains two register files (RFs) for the current maximum metrics with KN_t entries each. We use tokens propagating alongside the data to indicate whether a value is valid. The Compare Select (CS) elements select the maximum of the valid inputs. The registers also store tokens per entry, which are reset to zero when the processing of a symbol vector starts. After the scalar subtractor, we saturate the extrinsic LLRs to limit their dynamic range. The saturation has a positive influence on the communications performance.

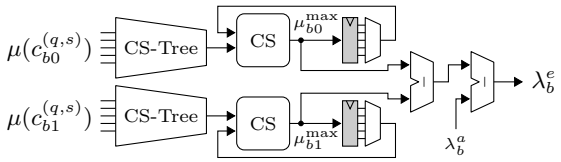


Fig. 9. L-Circuit

6 Differences to Reference Architecture

The proposed architecture is a complete redesign of [12]. This section explicitly highlights the architectural modifications. The original and new timing critical path are located in the GS-Circuit.

Multiplier-Free Gibbs Sampler: Similar to [11], we move the multiplication with $1/(\eta N_0)$ out of the GS into the FEP, by scaling $\boldsymbol{R}$ and $\boldsymbol{S}$ with Γ. This removes the multiplier from the detector's critical path, but increases the required word lengths.

Dynamic Scaling: The normalization of $\Gamma \in [0.5, 1)$ allows to use smaller word lengths, mitigating the previously mentioned increase. Consequently, we need an arithmetic shifter in the GS-Circuit at the previous location of the multiplier, which reverts the normalization.

Pipelined Input Multiplexers: Our MCMC detector selects the column of $\boldsymbol{R}$ and the entry of $\boldsymbol{\lambda}^a$ in the new Mux stage in front of the GS stage. While this removes those multiplexers from the detector's critical path, it adds an additional latency cycle.

Reduced Update-S-Circuit: We remove two Δ-multipliers (one per real and imaginary part) from the Update-S-Circuit, since in every cycle one of the entries of $\boldsymbol{S}$ does not change. This requires multiplexers for the resource sharing, which are however not in the critical path and are smaller than the removed Δ-multipliers.

Shared Maximum Metric Register File: The RFs are moved from the M-Circuit [12] to the L-Circuit. This reduces the required RFs from N_p to one. We also add a pipeline register after the L-Circuit to improve timing, which

requires another extra latency cycle. Also, our M-Circuit in Fig. 5 has one adder-subtractor instead of two adders, similar to [11].

Adder-Subtractor Units: These new units right after the Δ-multipliers in the GS- and the Update-S-Circuit, replace the original adders and the conditional negation units. The control selects addition or subtraction depending on the sign of Δ, if Δ is imaginary, the old bit $c_b^{(q,s-1)}$ and if $\mathrm{Im}\{r_{tn}\} = -\mathrm{Im}\{r_{nt}\}$.

Simplified Delta Multiplier: Our Δ-multipliers, used for γ and $\boldsymbol{S}$, compute the absolute value $|\Delta|$. This removes one multiplexer stage from the critical path.

Postponed Conjugation: We are storing only the lower half of $\boldsymbol{R}$. Due to the hermitian property of $\boldsymbol{R}$, we have $\mathrm{Im}\{r_{tn}\} = -\mathrm{Im}\{r_{nt}\}$. The control of the subsequent adder-subtractor units considers the required negation, instead of an explicit conjugation [12].

7 Results

With the word lengths given in Sec. 7.1 and the throughput equations in Sec. 7.2, we first compare our model to the reference architecture [12] based on gate-level synthesis results, then we present post-layout results for different design-time variants of our architecture. Sec. 8 presents the algorithmic evaluations.

7.1 Simulation Setup

A 802.11n-like 4×4 MIMO system is considered assuming a spatially uncorrelated Rayleigh channel, perfect channel knowledge and a max-log BCJR decoder. For all results, we assumed a rate-5/6 tail-biting binary convolutional code with generator polynomials 0133 and 0171 and puncturing, a random interleaver and 64-QAM modulation ($K = 6$). The frame length of 2160 information bits equals the interleaver's length, which is one OFDM symbol for this setup. For every data point, we simulated at least 10^5 frames. The average signal-to-noise ratio (SNR) per receive antenna is defined as $\mathrm{SNR} = \mathbb{E}[\|\boldsymbol{H}\boldsymbol{x}\|^2]/(N_r N_0)$. The required word lengths for an SNR loss of ≤ 0.1dB compared to the floating-point model at a frame error rate (FER) of 10% are: [integer.fractional] $\boldsymbol{y}$ [7.8], $\boldsymbol{H}$ [3.8], $\boldsymbol{\lambda}^a$ [5.4], $1/N_0$ [6.11], $\boldsymbol{R}$ [6.10], $\boldsymbol{S}$ [9.9], δ [17.6], μ [19.5], γ [3.29], $2^\alpha\delta$ [14.6], α [4.0], $\boldsymbol{\lambda}^e$ [8.4]. All are signed, per entry, and for real and imaginary part identical. The first chain ($q = 0$) is always initialized with the result of an hard-output zero-forcing MIMO detector. We assume $N_q = 8$ chains with $N_s = 8$ samples per chain (i.e. $N_{gs} = 64$ in [12]) for the next three sections, but vary those parameters in Sec. 8.

7.2 Architecture

Our parameterized architecture implementation currently supports up to 4×4 MIMO and 64-QAM. MIMO mode and QAM scheme can be configured at run-time within the supported range, which in turn can be configured at design-time.

Each GS/M-pair can process up to 16 chains sequentially, with up to 16 samples per chain. The FEP-Circuit requires

$$n_{\text{fep}} = N_r((N_t + 1)N_t/2 + \lceil N_t/2 \rceil) + 3 \tag{11}$$

cycles for its computation. This is slightly faster than the FEP-Circuit in [12]. The MCMC core runs for

$$n_{\text{gs}} = \frac{N_q}{N_p}(N_s + 1)KN_t + 5 \tag{12}$$

cycles. Compared to [12], we need two extra latency cycles (cf. Sec. 6). The code bit throughput of the architecture is $\theta_c = \frac{KN_t}{n_{\text{gs}}} f_{\text{clk}}$ assuming $n_{\text{gs}} \geq n_{\text{fep}}$ and sufficient input data.

7.3 Synthesis Results

We synthesized the design with Synopsys Design Compiler I-2013.12-SP2 in topographical mode using a 1.0V standard-performance standard cell library for the UMC 90nm SP-RVT LowK CMOS process. One gate-equivalent (GE) is the area of one 2-input drive-1 NAND gate. Fig. 10 compares the four instances $N_p = \{1, 2, 4, 8\}$ to [12]. While the most efficient design in [12] has an AT_{exec}-product of 181.7 kGEµs, our proposed design achieves 50.0 kGEµs, which is 3.6 times more efficient.

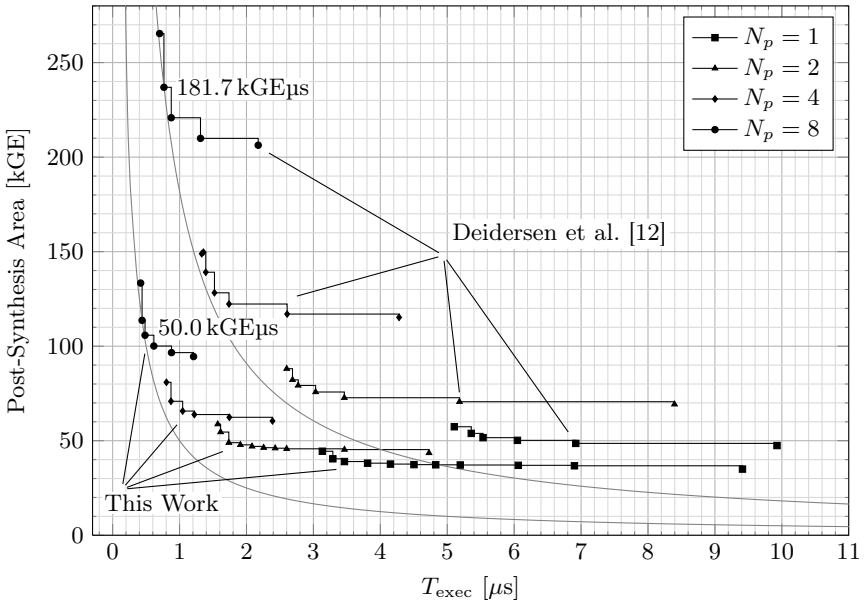


Fig. 10. Area vs. execution time based on the MCMC detector's synthesis results, comparing this work to [12], assuming $N_t = 4$, $K = 6$

Table 1. MCMC Detector Synthesis Results

Component		This Work	[12]	
FEP-Circuit		16.0	11.0	kGE
GS-Circuit	$8\times$	10.7	16.9	kGE
M-Circuit	$8\times$	0.9	12.2	kGE
L-Circuit	$N_p = 8$	13.3	3.3	kGE
Miscellaneous		5.0	17.9	kGE
Update-S-Circuit (cont. in GS)		5.2	7.7	kGE
Total	$N_p = 8$	127.1	265.0	kGE
Clock frequency		526	312	MHz
Cycles ($N_q = N_s = N_p = 8$)		221	219	
Average throughput		57.4	34.2	Mbit/s
Area efficiency		0.45	0.13	Mbit/s/kGE

Tbl. 1 lists the synthesis results for our fastest design instance and the reference design [12]. The FEP is larger (5 kGE), while the GS is smaller (-6.2 kGE per GS), since we moved the multiplier from the GS to the FEP. The additional area of the new arithmetic shifter is partially compensated for by the other improvements. The Update-S-Circuit becomes smaller (-2.5 kGE) since we save one complex Δ-multiplier and use $|\Delta|$ now. The saving effect is larger than the additional area from the multiplexers required for the resource sharing. The M-Circuit exhibits only about 7.4% of the original area, since we moved the RFs to the L-Circuit, which consequently became larger (10 kGE). The remainder of the area (-12.9 kGE) is occupied amongst others by the $\boldsymbol{R}$ column multiplexers. The area is reduced because the multiplexers are no longer in the timing critical path.

In total, the redesigned architecture takes on only about 48% of the original area for $N_p = 8$. The saving depends on the number of GS/M-Circuits. The critical path was shortened by about 40%, i.e. the maximum clock frequency increased from 312 MHz to 526 MHz.

7.4 Layout Results

A layout was obtained with Cadence SoC Encounter 9.1 for each configuration's fastest design instance in order to further study the proposed architecture's implementation complexity and to enable more precise comparison with future related work. All following area figures are taken from the layout results, depicted in Fig. 11. The consumed area slightly increased, while the achievable clock frequency decreased. It is interesting that the throughput mainly depends on the number of parallel GS/M-Circuits and the chain parameters, i.e.

$$\theta_c = \frac{KN_t}{n_{\text{gs}}} f_{\text{clk}} \approx \frac{N_p}{N_q(N_s+1)} f_{\text{clk}} \tag{13}$$

as can be seen in Fig. 11.

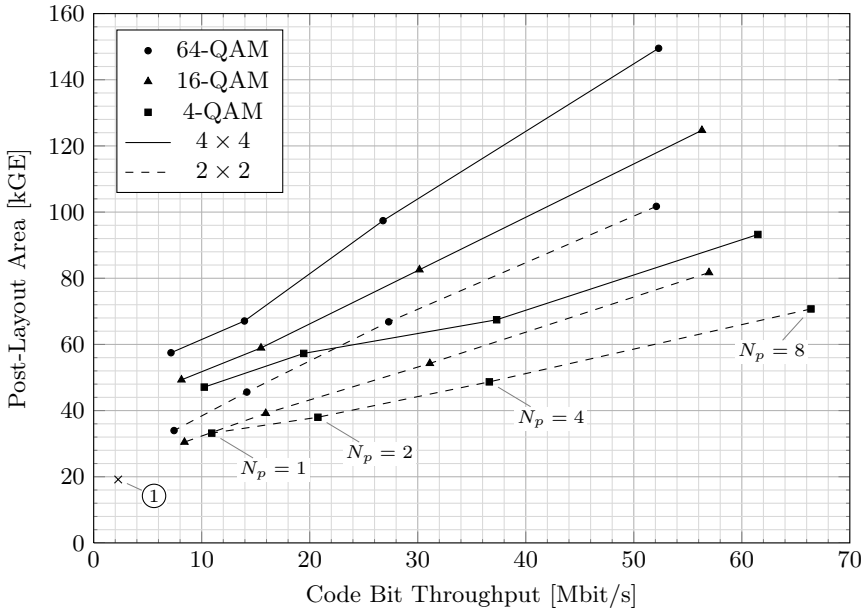


Fig. 11. Area vs. throughput based on the MCMC detector's layout results. For each design-time configuration, the ASIC with the fastest clock is shown. As an example, the 16-QAM 2×2 design supports one or two antennas and 4- or 16-QAM at run-time.

The largest instance, for 64-QAM, $N_t = 4$ and $N_p = 8$, requires 149.5 kGE or 0.47 mm^2 and achieves a maximum clock frequency of 479 MHz, yielding a code bit throughput of 52 Mbit/s. The fastest instance in terms of throughput supports 4-QAM, $N_t = 2$ and has $N_p = 8$ GS/M-Circuits. It occupies in total an area of 70.7 kGE or 0.22 mm^2 and runs at 664 MHz, which results in a throughput of 66 Mbit/s.

To determine the smallest instance, which should be the lower corner of the covered design space, ① in Fig. 11, we synthesized the detector with $N_t = 2$, 4-QAM and one GS/M-Circuit for a target of 100 MHz. This ASIC consumes 19.2 kGE or 0.06 mm^2, runs at 165 MHz and yields a 2.27 Mbit/s throughput. The FEP-Circuit and MCMC core require 10.9 kGE and 8.3 kGE respectively. Further word length optimizations could yield additional area reductions.

Tbl. 2 compares our work to a selection of reported MIMO detector implementations. We make three observations. First, in terms of hardware efficiency expressed in Mbit/s/kGE, the MCMC detector resides in about the same order of magnitude as the single-tree-search sphere decoder (STS-SD) [7], though our architecture is more than two times more efficient than our reference architecture [12]. The MCMC detector exhibits a deterministic run-time, which eases the receiver system design, while the SD can in principle always achieve near-capacity performance at the cost of a strongly varying run-time. Secondly,

Table 2. Comparison to other reported MIMO detectors

	This work	[12]	[6]	[5]	[4]	[8]	[7]	[9]	[16]
Ant. Cfg.	$\leq 4 \times 4$	$\leq 4 \times 4$	4×4	4×4	4×4	4×4	$\leq 4 \times 4$	4×4	4×4
Mod. Order	≤ 64	≤ 64	≤ 64	≤ 64	≤ 64	64	≤ 64	16	64
Algorithm	MCMC	MCMC	EPIC	MMSE	MMSE	FCSD	STS-SD	Trellis	K-Best
SO / SISO	SISO	SISO	SISO	SISO	SISO	SISO	SISO	SISO	SO
Technology	90 nm	90 nm	90 nm	90 nm	90 nm	90 nm	90 nm	65 nm	65 nm
Core Area [mm^2]	0.47	–	1.08	1.5	–	2.61	–	1.58	0.57
TP Θ_c [Mbit/s]	avg. 52[a]	avg. 34.2[a]	733	757	833	2200	avg. 51.1[b]	1228[c]	1444[c]
Preproc. [kGE]	_[d]	_[d]	345.8	384.2[e]	178.3	_[f]	_[f]	_[f]	_[f]
Detection [kGE]	149.5	265				555	175	1097	298
Eff. [Mbit/s/kGE]	0.34	0.13	2.11	1.97	4.67	3.96	0.29	1.12	4.85

[a] We assume $N_s = 64, N_{gs} = 8, N_p = 8$ [12].
[b] We assume 100 visited nodes per symbol vector [5].
[c] Scaled to 90nm CMOS technology assuming $t_{pd} \sim 1/s$
[d] Area for optional initial detection not included
[e] Area for chip IO interface excluded
[f] Area for required QRD not included

the MCMC detectors (and the STS-SD) are about one order of magnitude less efficient than the linear [4, 5], iterative-linear [6] detectors, and most notably the fixed-complexity sphere decoder (FCSD) [8], which achieves close-to-optimal communications performance at a deterministic run-time. In this perspective, the FCSD [8] is the best choice. In case that a particularly small implementation is needed, the MCMC might have an advantage, depending on how well the FCSD scales. Lastly, there are three cases for the preprocessing circuitry. Some implementations include it [4–6], it is optional for the MCMC detectors [12], and definitely required for the other reported work [7–9, 16]. This of course makes the area-throughput efficiency comparison difficult.

8 Algorithmic Considerations

In this section, we put the code bit throughput θ_c, as an implementation property of our architecture, in relation to our design's communications performance in terms of SNR required to achieve a 10% frame error rate. With this data, we can determine for example appropriate run-time parameters, or an appropriate run-time strategy to adapt them. Depending on the optimization criterion, the parameter choices might be different. Possible criteria are for example spectral efficiency or energy efficiency (as future work, we plan to perform energy estimations). The first part of this section gives a general overview, while the second part explains in more detail the iterative receiver figures.

In the remainder, we use the post-layout implementation results of the 64-QAM, $N_t = 4$, $N_p = 8$ instance that runs at 479 MHz. The simulation setup that we select resembles the highest-throughput mode of the 802.11n standard, which requires a high SNR. However, our experiments show that the MCMC-based detection performs best in a mid-range SNR regime, in combination with lower-order modulation schemes. Thus this can be considered as kind of a worst-case scenario for the MCMC detector.

We assume the same simulation setup as in Sec. 7.1. Additionally, we perform up to two detector-decoder iterations, i.e. per frame, we execute the MCMC detector and BCJR decoder twice. This gives us four run-time parameters: the number of chains N_{q1} and samples N_{s1} in the first iteration and respectively N_{q2}, N_{s2} for the second iteration. The short-hand notation $\mathrm{GS}_1$8x6 denotes $N_{q1} = 8$ and $N_{s1} = 6$, similarly we use $\mathrm{GS}_2 N_{q2}\mathrm{x}N_{s2}$. We simulated the parameter set $N_{q1/2} \in \{8, 16\}$ and $N_{s1/2} = \{1, 2, \ldots, 16\}$. Thus all $N_p = 8$ GS/M-Circuits are always active. The total number of samples per iteration defined as $N_{gs1/2} = N_{q1/2} \cdot N_{s1/2}$ is our measure for the invested effort.

Fig. 12 shows four curves: two for the first iteration, and two for the second. The last part of this section explains how we determine the two second-iteration curves. They are pareto-optimal in terms of SNR versus throughput.

Clearly in Fig. 12, we can identify the existence of a run-time tradeoff between SNR and throughput. As could be expected, more effort (i.e. more samples, more chains) results in a better algorithmic performance (lower SNR). An SNR gain has several possible uses amongst others: we can extend the transmission range,

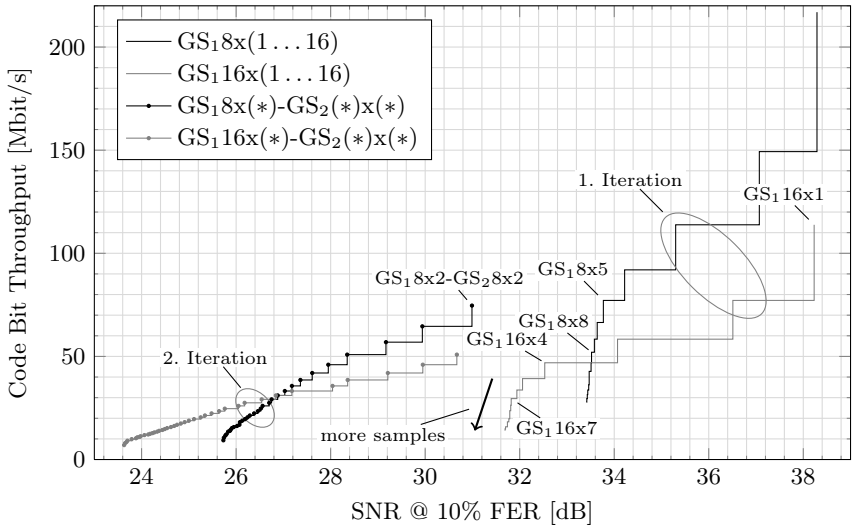


Fig. 12. Code bit throughput over SNR required to achieve a 10% frame error rate

we can serve more users (more interference), or we can also lower the transmission power to save energy (and reduce interference to other users).

In the non-iterative case (first iteration), we observe a vanishing gain beyond five samples, both for eight and 16 chains. At around 33.5 dB, it is better to use 16 instead of eight chains. Interestingly, this switches from $\mathrm{GS}_1$8x8 at 33.52 dB to $\mathrm{GS}_1$16x4 at 32.53 dB. The total number of samples for both configurations is 64, but we gain about 1 dB SNR while approximately maintaining the throughput. It is not completely identical due to the pipeline delays of the architecture.

Instead of using $\mathrm{GS}_1$8x6 after $\mathrm{GS}_1$8x5, a good decision would be to switch to the second iteration, therefore never using 16 chains in the non-iterative case. This yields a large SNR gain of about 2.7 dB at a similar throughput. At this transition point, we switch from $\mathrm{GS}_1$8x5 to $\mathrm{GS}_1$8x2-$\mathrm{GS}_2$8x2. The throughputs drops slightly from 77.15 Mbit/s to 74.65 Mbit/s. With $N_{gs1} = 40$ compared to $N'_{gs1} + N'_{gs2} = 32$, the MCMC detector's effort remains very similar.

MCMC-based detection benefits greatly from iterative MIMO decoding. Switching from one to two iterations yields SNR gains as large as 6 dB. While in the first iteration we achieve only about 31.7 dB, all SNR operating points of the second iteration are lower than 31 dB. A possible explanation is that the guidance from the channel decoder, in terms of prior LLRs, is the contributing factor for this. It helps the MCMC-based detection in two ways: we select the initial samples $\boldsymbol{c}^{(q,0)} \sim p(\boldsymbol{c}) = f(\boldsymbol{\lambda}^{\boldsymbol{a}})$, and the transition probability γ depends on $\boldsymbol{\lambda}^{\boldsymbol{a}}$. This seems to let the chains converge faster (in less samples) to interesting regions.

It follows a closer look on the second iteration. There are four parameters, $N_{q1/2}$ and $N_{s1/2}$. For a given SNR, we determine the parameter combination

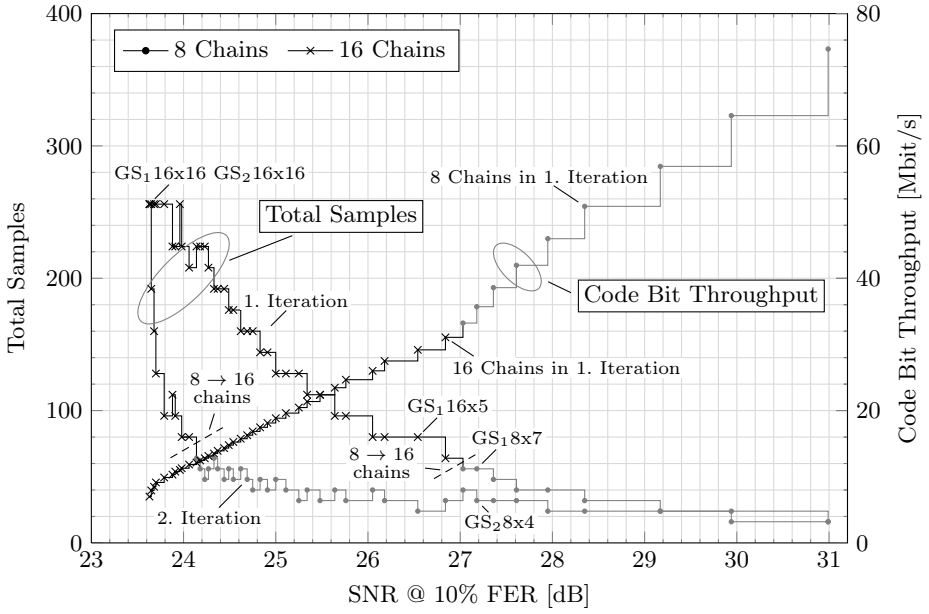


Fig. 13. Iterative receiver: two detector-decoder activations per symbol vector. For the throughput curve, dots and crosses denote if eight or 16 chains are used in the first of the two iterations. For the two total samples curves, the dots and crosses likewise denote if eight or 16 chains are used in the respective iteration. The figure shows only the pareto-optimal points in terms of SNR versus throughput determined from the data set with two iterations.

that yields the highest throughput. These pareto-optimals points are shown in Fig. 13. For the two second-iteration curves in Fig. 13, we fix the number of chains in the first iteration $N_{q1} = 8$ and $N_{q1} = 16$ respectively, then optimize over the remaining three parameters.

For our calculations, we assume that the channel decoder and the buffering between decoder and detector cause no additional delay. This is a somewhat ideal scenario, since it might give us a large area consumption e.g. of the buffers, but definitely provides us with an upper bound for the achievable throughput. Thus, the throughput is given as

$$\theta_{c,2} = \frac{KN_t}{n_{\mathrm{gs},1} + n_{\mathrm{gs},2}} f_{\mathrm{clk}} \tag{14}$$

with $n_{\mathrm{gs},1/2} = \frac{N_{q1/2}}{N_p}\big(N_{s1/2} + 1\big)KN_t + 5$ and fixing $N_p = 8$ here.

We observe that more effort is required in the first iteration. For example, around 27 dB, the two configurations $\mathrm{GS}_1$8x7 and $\mathrm{GS}_2$8x4 are in use, i.e. $N_{gs1} = 56$ total samples for the first iteration, and $N_{gs2} = 32$ for the second.

At about 26.8 dB, we switch from eight to 16 chains in the first iteration. It appears that multiple short chains are favorable for the first iteration. Only at around 24 dB, the detector should switch from eight to 16 chains in the second iteration. It is also the point where the effort significantly rises (near 23.6 dB), especially for the second iteration. This could be an indication for switching to three iterations.

From a pure SNR-throughput perspective, we can say that two iterations are better than a single. As previously stated, we observe large SNR gains from iterating, and the best non-iterative operating point is off by about 0.7 dB compared to the worst second-iteration point. However, this of course ignores the hardware cost caused by the required buffering and the increased throughput requirement on the detector and decoder architectures. A realistic comparison depends on the overall objective, i.e. lowest energy, small area, best spectral efficiency, and on additional constraints, like minimum supported bandwidth. While this is out of scope here, we think that our data outlines the run-time adaptibility of the MCMC-based MIMO detection architecture. It also shows that it performs particularly well in iterative receivers, therefore it could be a reasonable candidate to consider in the design of such a system.

9 Conclusions & Outlook

We have presented synthesis and layout results of the proposed MCMC detector architecture. The area reduction of up to 52% and the shorter clock period by up to 40% indicate that the proposed architectural modifications to the reference design are effective. Our extensive data set for the communications performance further highlights the available tradeoff between signal-to-noise ratio and architecture throughput. With its run-time adaptibility covering a large design space, our detector is effectively able to cope with a lot of channel conditions at the appropriate effort. Though being a stochastic detector, its completely deterministic run-time eases scheduling at the system level, i.e. inside a complex iterative receiver.

Still, the architecture suffers from a relatively low but deterministic throughput, which stems from the MCMC detection method itself. The main advantage appears to be its simple scalability through N_p and configurability through N_t and K. This allows the architecture to cover a large design space. Practically, only the availability of sufficient data might limit the architectural parallelism.

As future work, we plan to correlate algorithmic performance with energy consumption, which might reveal another tradeoff capability of the proposed design.

Acknowledgements This work has been supported by the Ultra High-Speed Mobile Information and Communication (UMIC) Research Centre, RWTH Aachen University.

References

1. IEEE standard for information technology– local and metropolitan area networks– specific requirements– part 11: Wireless LAN medium access control (MAC) and physical layer (PHY) specifications amendment 5: Enhancements for higher throughput. IEEE Std 802.11n-2009 pp. 1–565 (2009)
2. Guillén i Fàbregas, A., Martinez, A., Caire, G.: Bit-interleaved coded modulation. Found. Trends Commun. Inf. Theory 5(1-2), 1–153 (Jan 2008), http://dx.doi.org/10.1561/0100000019
3. Hochwald, B., ten Brink, S.: Achieving near-capacity on a multiple-antenna channel. Communications, IEEE Transactions on 51(3), 389–399 (March 2003)
4. Auras, D., Leupers, R., Ascheid, G.: A novel reduced-complexity soft-input soft-output mmse mimo detector: Algorithm and efficient vlsi architecture. In: Communications (ICC), 2014 IEEE International Conference on. pp. 4722–4728 (June 2014)
5. Studer, C., Fateh, S., Seethaler, D.: ASIC implementation of soft-input soft-output MIMO detection using MMSE parallel interference cancellation. Solid-State Circuits, IEEE Journal of 46(7), 1754–1765 (2011)
6. Auras, D., Leupers, R., Ascheid, G.: A novel class of linear mimo detectors with boosted communications performance: Algorithm and vlsi architecture. In: VLSI (ISVLSI), 2014 IEEE Computer Society Annual Symposium on. pp. 41–47 (July 2014)
7. Witte, E.M.: Efficiency and flexibility trade-offs for soft-input soft-output sphere-decoding architectures. Ph.D. thesis, RWTH Aachen University (2013)
8. Chen, X., He, G., Ma, J.: VLSI implementation of a high-throughput iterative fixed-complexity sphere decoder. Circuits and Systems II: Express Briefs, IEEE Transactions on 60(5), 272–276 (2013)
9. Sun, Y., Cavallaro, J.: Trellis-search based soft-input soft-output MIMO detector: Algorithm and VLSI architecture. Signal Processing, IEEE Transactions on 60(5), 2617–2627 (2012)
10. Senst, M., Ascheid, G.: A rao-blackwellized markov chain monte carlo algorithm for efficient MIMO detection. In: Proc. IEEE ICC. pp. 1–6 (2011)
11. Laraway, S., Farhang-Boroujeny, B.: Implementation of a markov chain monte carlo based multiuser/MIMO detector. IEEE Trans. Circuits Syst. I, Reg. Papers 56(1), 246–255 (2009)
12. Deidersen, U., Auras, D., Ascheid, G.: A parallel VLSI architecture for markov chain monte carlo based MIMO detection. In: Proc. ACM GLSVLSI. pp. 167–172 (2013)
13. Auras, D., Deidersen, U., Leupers, R., Ascheid, G.: VLSI design of a parallel MCMC-based MIMO detector with multiplier-free gibbs samplers. In: Very Large Scale Integration (VLSI-SoC), 2014 22nd International Conference on. pp. 1–6 (Oct 2014)
14. Yuan, F.L., Yang, C.H., Markovic, D.: A hardware-efficient VLSI architecture for hybrid sphere-MCMC detection. In: Proc. IEEE GLOBECOM. pp. 1–6 (2011)
15. Hagenauer, J.: The turbo principle in mobile communications. In: Proc. International Symposium on Nonlinear Theory and its Applications, Xian, China (2002)
16. Patel, D., Smolyakov, V., Shabany, M., Gulak, P.: Vlsi implementation of a WiMAX/LTE compliant low-complexity high-throughput soft-output K-Best MIMO detector. In: Circuits and Systems (ISCAS), Proceedings of 2010 IEEE International Symposium on. pp. 593–596 (2010)