VLSI Design of a Parallel MCMC-based MIMO Detector with Multiplier-Free Gibbs Samplers

Dominik Auras, Uwe Deidersen, Rainer Leupers, Gerd Ascheid
Institute for Communication Technologies and Embedded Systems
RWTH Aachen University, 52056 Aachen, Germany
email: auras@ice.rwth-aachen.de

***Abstract*—Little consideration has so far been dedicated to the investigation of the implementation complexity of stochastic detectors for multi-antenna (MIMO) systems although they promise communications performance close to max-log detection for certain SNR regimes. In this work, we propose a complete redesign of the only reported parallel VLSI architecture for soft-input soft-output Markov chain Monte Carlo based MIMO detection to date. Using multiplier-free Gibbs Sampler implementations, dynamic scaling of the noise density and multiple further architectural optimizations, we significantly reduce the area and improve the timing, yielding AT-efficiency improvements of as much as 3.6 times.**

I. Introduction

In multi-antenna (MIMO) systems using bit-interleaved coded modulation with iterative decoding (BICM-ID), the soft-input soft-output (SISO) detector constitutes one of the main challenges for VLSI implementation, as the optimal detection has an exponential complexity. Stochastic detection based on Markov chain Monte Carlo (MCMC) methods enables particularly small detector implementations due to the simple randomly guided search. Furthermore, when iterating between detector and channel decoder, MCMC detection shows a communications performance close to max-log detection for certain SNR regimes [2].

Related Work: The only MCMC-based SISO MIMO detector ASIC design truly supporting independent parallel Gibbs Samplers to date is presented in [1]. Amongst other things, [1] 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 [3], 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 [4] 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 [1], 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.

Outline: First, we introduce the general concept of MCMC-based MIMO detection (Sec. III), describe the implemented algorithm (Sec. IV), then we propose the redesigned architecture (Sec. V). Subsequently, we explicitly highlight the differences to the reference design [1] in Sec. VI. Our implementations results are presented in Sec. VII.

II. System Model

We consider a spatial multiplexing $N_t \times N_r$ MIMO system with BICM-ID. 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, 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.

III. 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

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

recursively (Markov chain). By design the search converges towards candidates of high probability [2].

We select independent first samples $\boldsymbol{c}^{(q,0)} \in \{0,1\}^{KN_t}$, one per chain $q = 1 \dots N_q$, 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 $s = 1 \dots N_s$ 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

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

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.

IV. Low-Level Algorithm

The presented algorithm implements the max-log variant of the Rao-Blackwellized MCMC detection algorithm with uniform sampling described in [2]. Its basic idea is to recursively compute the metric in Eq. (1) by tracking the changes while drawing bits [1]. 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 [1], [2].

A. Basic Concepts

1) Matched Filter: The algorithm in [1] 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 $\boldsymbol{y}^{\text{mf}} = \boldsymbol{H}^H\boldsymbol{y}$ in the metric. This does not influence the posterior LLR calculation, however it allows to use the symmetry $\boldsymbol{R} = \boldsymbol{R}^H$.

2) 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 $P(c_b|c_1, \dots, c_{b-1}, c_{b+1}, \dots, c_{KN_t})$. The state of the q-th GS at the s-th sample after drawing the b-th bit is denoted as

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

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

3) 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 [1].

4) 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.

5) Recursive Dot-Product: The algorithm tracks the current value of

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

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

6) 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

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

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.

7) 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 [2]. For the conversion to the linear domain, we apply a piece-wise linear approximation to $\text{logistic}(\gamma) = 1/(1 + e^{-\gamma})$ as in [1], [3]. 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.

B. Overall Algorithm Design

Fig. 1 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.

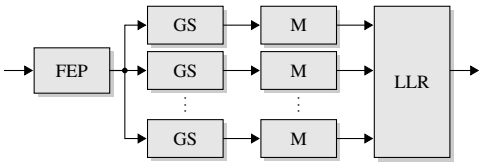

Fig. 1. Partitioning of the low-level algorithm: Front-end Processing, Gibbs Sampler, Metric Update, LLR Computation.

C. 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. IV-A5 but scaled by Γ.

D. 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. IV-A7). 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 15-18).

E. Metric Update

Alg. 2 recursively computes the current candidate's metric, 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 10, we select one of the two as the new current metric. It remains unaltered if the bit does not change.

F. 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 9 is included, as it can be easily implemented in hardware.

V. VLSI Architecture

After an overview of the proposed architecture design, we present the details of the core components.

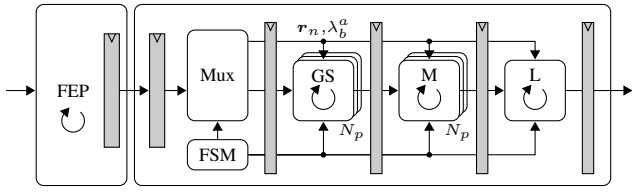

Fig. 2. Architecture design of the MCMC detector. The n-th column $\boldsymbol{r}_n$ of $\boldsymbol{R}$ and λ_b^a are selected in the Mux stage. FEP and MCMC core computations overlap thanks to double buffering.

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 $\delta_b^{(q,s)} = \Big[\mathrm{Re}\{r_{nn}\}|\Delta|_b^2(\boldsymbol{c}^{(q,s-1)}) - 2\mathrm{Re}\{S_n^* \Delta_b(\boldsymbol{c}^{(q,s-1)})\}\Big] 2^{-\alpha}$
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 **end**
15 $\Delta S_t \leftarrow r_{tn} \Delta_b(\boldsymbol{c}^{(q,s-1)}) \quad \forall\, t = 1 \ldots N_t,\ t \neq n$
16 **if** $c_b^{(q,s-1)} \neq c_b^{(q,s)}$ **then**
17 $S_t \leftarrow S_t + \begin{cases} \Delta S_t & \text{if } c_b^{(q,s-1)} = 1 \\ -\Delta S_t & \text{if } c_b^{(q,s-1)} = 0 \end{cases} \forall\, t \neq n$
18 **end**
19 **end**
20 **end**

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 $\mu(\boldsymbol{c}_b^{(q,-1)}) = 0 \quad \forall\, b = 1 \ldots N_t K$
2 **for** $s = 0$ **to** N_s **do**
3 **for** $b = 1$ **to** $N_t K$ **do**
4 $\mu(\boldsymbol{c}_{b0}^{(q,s)}) = \mu(\boldsymbol{c}_{b1}^{(q,s)}) = \mu(\boldsymbol{c}_b^{(q,s-1)})$
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 **end**
10 $\mu(\boldsymbol{c}_b^{(q,s)}) = \begin{cases} \mu(\boldsymbol{c}_{b0}^{(q,s)}) & \text{if } c_b^{(q,s)} = 0 \\ \mu(\boldsymbol{c}_{b1}^{(q,s)}) & \text{if } c_b^{(q,s)} = 1 \end{cases}$
11 **end**
12 **end**

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 **end**
8 **end**
9 $\lambda_b^e = \mu_{b0}^{\max} - \mu_{b1}^{\max} - \lambda_b^a \quad \forall\, b = 1 \ldots K N_t$

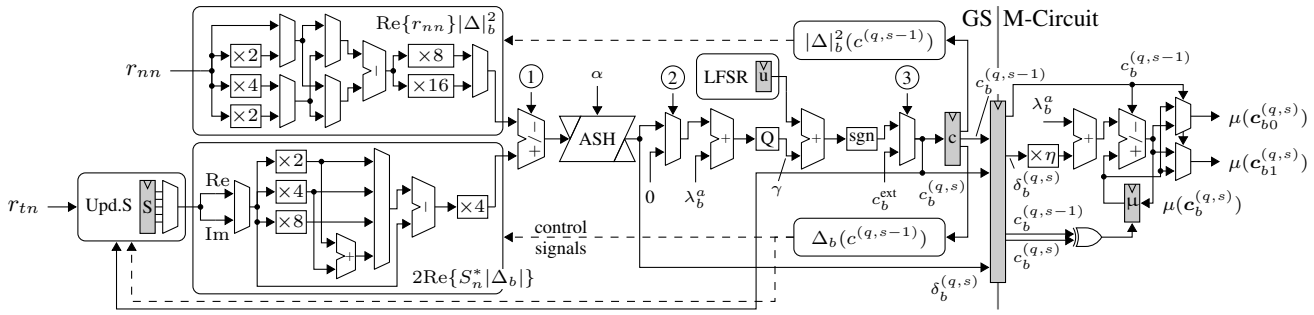

Fig. 3. GS/M-Circuit. Multiplications with Δ_b and $|\Delta|_b^2$ are realized with shifts and adders. The arithmetic shifter (ASH) reverts the normalization of Γ.

A. Overview

The macro pipeline of FEP-Circuit and MCMC core, shown in Fig. 2, 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.

B. FEP-Circuit

The architecture 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 Γ. In parallel, we multiply the entries of $\boldsymbol{R}$ with $x_t^{(-1)} = 1 + j$ (cf. Sec. IV-A3) using only adders and multiplexers, and accumulate the results to obtain $\boldsymbol{S}$.

C. GS/M-Circuit

Fig. 3 depicts the combined GS/M-Circuit including the connecting register. The $|\Delta|^2$-multiplier 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 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 (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 part on the right side that implements Alg. 2 uses a write-enabled register for the current metric, which is updated when we flip the current bit. 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.

The Update-S-Circuit shown in Fig. 4 has (N_t-1) complex-valued Δ-multipliers. 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.

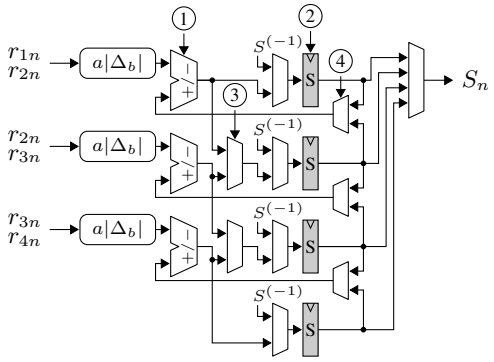

Fig. 4. Update-S-Circuit. Example for $N_t = 4$ antennas. All units exist for the real and for the imaginary parts respectively (not drawn).

D. L-Circuit

The L-Circuit shown in Fig. 5 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.

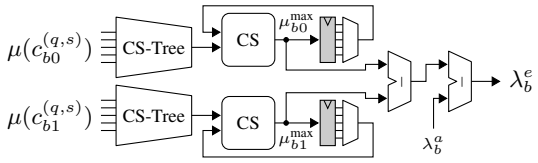

Fig. 5. L-Circuit. Data tokens propagate alongside the data values. The registers hold tokens and values. The Compare Select (CS) elements forward the data item with the larger value if both are valid.

VI. Differences to Reference Architecture

This section explicitly highlights the proposed architectural modifications to [1]. The original and new timing critical path are located in the GS-Circuit.

1) Multiplier-Free Gibbs Sampler: Similar to [3], 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.

2) 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.

3) 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.

4) 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.

5) Shared Maximum Metric Register File: The RFs are moved from the M-Circuit [1] 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. 3 has one adder-subtractor instead of two adders, similar to [3].

6) 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 $\mathrm{Im}\{r_{tn}\} = -\mathrm{Im}\{r_{nt}\}$.

7) 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.

8) 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 [1].

VII. Results

Our parameterized architecture implementation[1] 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 [1]. The MCMC core runs for

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

cycles. Compared to [1], we need two extra latency cycles (cf. Sec. VI-3 and VI-5). 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.

A. 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. Fig. 6 compares the four instances $N_p = \{1, 2, 4, 8\}$ to [1]. While the most efficient design in [1] has an AT_{exec}-product of $181.7\,\text{kGE}\mu\text{s}^2$, our proposed design achieves $50.0\,\text{kGE}\mu\text{s}$, which is 3.6 times more efficient.

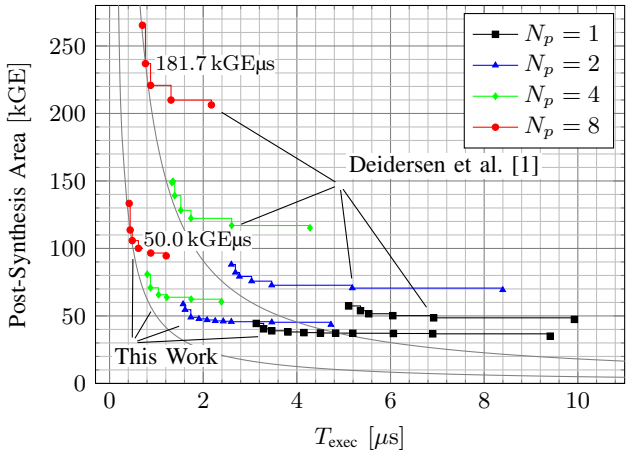

Fig. 6. Area vs. execution time based on the MCMC detector's synthesis results, comparing this work to [1], assuming $N_t = 4$, $K = 6$.

[1] 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. The frame length equals the interleaver's length. The required word lengths for an SNR loss of $\leq$ 0.1dB compared to the floating-point model at a frame error rate 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 [3.4], $2^\alpha\delta$ [14.6], α [4.0], $\boldsymbol{\lambda}^e$ [8.4]. All are signed, per entry, and for real and imaginary part identical. We always assume $N_q = 8$ chains with $N_s = 8$ samples per chain (i.e. $N_{gs} = 64$ in [1]).

[2] One gate-equivalent (GE) is the area of one 2-input drive-1 NAND gate.

TABLE I
MCMC DETECTOR SYNTHESIS RESULTS

Component		This Work	[1]	
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. I lists the synthesis results for our fastest design instance and the reference design [1]. 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.

B. Layout Results

A layout was obtained with Cadence SoC Encounter 9.1 for each configuration's fastest design instance, depicted in Fig. 7, 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. 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.

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

as can be seen in Fig. 7.

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. 7, 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.

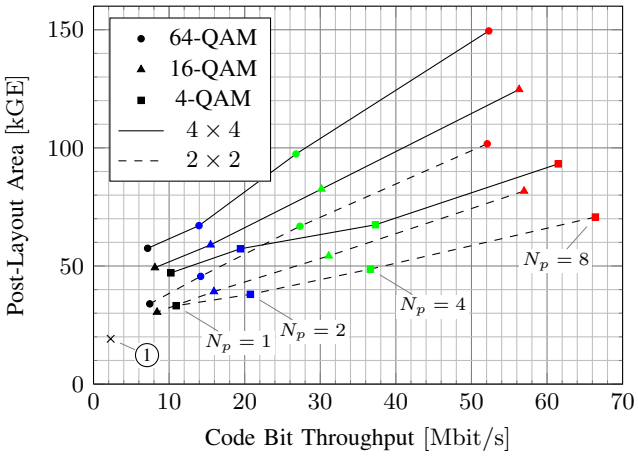

Fig. 7. Area vs. throughput based on the MCMC detector's layout results. For each design-time configuration, the ASIC with the fastest clock is shown. As an example, the 16-QAM 2×2 design supports one or two antennas and 4- or 16-QAM at run-time.

VIII. CONCLUSION

The presented synthesis and layout results show the effectiveness of the proposed architectural modifications. The area reduces by up to 52% and the timing improves by up to 40% while the communications performance remains identical. There are no known drawbacks of the proposed changes.

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 could foresee a pool of e.g. 64 Gibbs Samplers, requiring possibly less than 1 MGE, that exploits the algorithm parallelism at symbol vector and chain level.

REFERENCES

[1] U. Deidersen, D. Auras, and G. Ascheid, "A parallel VLSI architecture for markov chain monte carlo based MIMO detection," in *Proc. ACM GLSVLSI*, 2013, pp. 167–172.

[2] M. Senst and G. Ascheid, "A rao-blackwellized markov chain monte carlo algorithm for efficient MIMO detection," in *Proc. IEEE ICC*, 2011, pp. 1–6.

[3] S. Laraway and B. Farhang-Boroujeny, "Implementation of a markov chain monte carlo based multiuser/MIMO detector," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 56, no. 1, pp. 246–255, 2009.

[4] F.-L. Yuan, C.-H. Yang, and D. Markovic, "A hardware-efficient VLSI architecture for hybrid sphere-MCMC detection," in *Proc. IEEE GLOBECOM*, 2011, pp. 1–6.