A Flexible ASIC for Time-Domain Decision-Directed Channel Estimation in MIMO-OFDM Systems

Andreas Minwegen, Dominik Auras, Gerd Ascheid

A Flexible ASIC for Time-Domain Decision-Directed Channel Estimation in MIMO-OFDM Systems

Andreas Minwegen, Dominik Auras, Gerd Ascheid

Institute for Communication Technologies and Embedded Systems
RWTH Aachen University, 52056 Aachen, Germany
{minwegen,auras,ascheid}@ice.rwth-aachen.de

Abstract. Channel estimation is a crucial task for the overall communication performance of a wireless receiver. Compared to traditional approaches the estimation of the wireless channel can be improved by using iterative estimation with feedback from other receiver components, however the VLSI implementation of such iterative channel estimation in multiple-input multiple-output (MIMO) orthogonal frequency division multiplexing (OFDM) systems is challenging due to the high computational complexity. In this chapter we introduce the first ASIC for Decision-Directed MIMO-OFDM channel estimation which tracks channel variations using feedback from a decoder and supports M-QAM. Furthermore, timing and power dissipation trade-offs are analyzed.

Keywords: MIMO-OFDM, VLSI, ASIC, channel estimation, expectation maximization, SAGE

1 Introduction

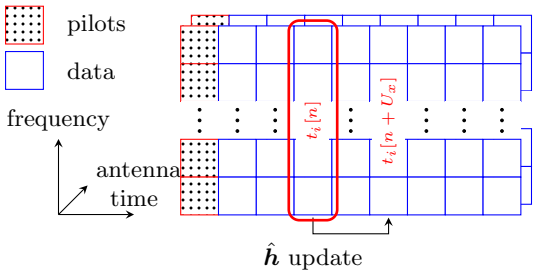

Fig. 1. The frame structure of the system

Orthogonal frequency division multiplexing (OFDM) and spatial multiplexing over multiple-input multiple-output (MIMO) transmissions schemes are adopted by several recent wireless communication standards such as 3GPP Long Term Evolution (LTE) or IEEE 802.11n and beyond. Due to the concept of coherent

detection in MIMO-OFDM receivers the channel estimation (CE) is a crucial and computational intensive part of the overall system and has a significant impact an the communication performance in terms of frame error rate and thus influence directly the maximal achievable throughtput. Traditionally, pilot-aided channel estimation (PACE) is applied where the channel is estimated at predefined pilot positions. The complete channel over all subcarriers is obtained via interpolation. Iterative channel estimation can be used to improve the estimates which delivers promising SNR gains [1]. In the case of iterative channel estimation algorithms, a priori knowledge from the detector or the decoder is used to improve the channel estimates iteratively.

However, the gain in terms of algorithmic performance is payed for in terms of high latencies since each block that participates in the iterations has to complete its processing before the next one can start using the improved input. An alternative that does not increase the latency but still provides significant benefits is the channel tracking approach. This approach ([2] and [3]) provides channel estimation updates for time instance $n + U_x$ based on the detector or decoder decisions for time instance n, as shown in Fig. 1.

Especially, fast fading channels are interesting scenarios for such a solution. An algorithmic investigation for a communication system similar to LTE is performed in [4] using the simplified frequency domain (FD) SAGE (space-alternating generalized expectation-maximization) algorithm, which calculates updates of the channel impulse response (CIR) estimates for M-QAM constellations. The authors of [3] present a modification that provides a gain of 2 dB for 64-QAM. This modification requires a non trivial matrix inversion. Fortunately, this matrix inversion can be avoided by iteratively processing each tap of the CIR in the time domain, as proposed by the time-domain (TD) SAGE algorithm presented in [5].

Furthermore, the authors of [6] present an analysis of the TD-SAGE algorithm in the context of LTE-Advanced. The results show that it has the potential to double the system throughput at high user mobility due to a better channel estimate and therefore, a lower frame error rate. Apart from that the authors present simulation results about the impact on the system performance of usage of variable number of feedback symbols from the decoder to the channel estimation block. These results show interesting trade-offs between the computational complexity and the algorithmic performance for the TD-SAGE algorithm. Therefore, the number of feedback symbols seems to be interesting parameter for a trade-off analysis between energy dissipation and algorithmic performance of a dedicated VLSI architecture.

Contributions: This chapter introduces an extension of the first ASIC implementation of the TD-SAGE algorithm which is presented in [7]. The TD-SAGE algorithm is transformed to a novel variant, further reducing the computational complexity, to what is termed the tap alternating (TA) SAGE. Apart from that this work discusses two options to further reduce the computational complexity with the penalty of a loss in algorithmic performance. Either reducing the number of feedback symbols as discussed in [6] or reducing the frequency of updating

the channel estimate. A suitable VLSI architecture is presented and area, timing and power numbers are provided. The support for a variable number of feedback symbols introduces a modification to the state machine that has an impact on the critical path. Therefore the implementation results for a architectures with and without variable feedback support are presented. This work evaluates the hardware costs of channel tracking in a MIMO-OFDM system.

Outline of the chapter: The remainder of the chapter is organized as follows. In section 2 the system setup is introduced. Section 3 presents the implemented, modified algorithm followed by Section 5 deriving the ASIC architecture and details about the processing units and the memory are given. Finally Section 6 discusses post-layout implementation results.

2 System Model

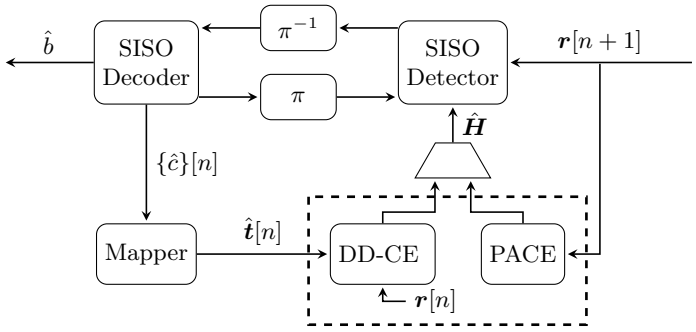

Fig. 2. Receiver model

The system considered in this chapter is a MIMO-OFDM system using $N_\mathrm{T} = 2$ transmit antennas and $N_\mathrm{R} = 2$ receive antennas, with $N_\mathrm{K} = 512$ sub-carriers and a cyclic prefix length of $N_\mathrm{L} = 32$. First, information bits $\{b\}$ are encoded using a convolutional encoder with the generator polynomial $[133_o 171_o]$. These code bits $\{c\}$ are interleaved by a random interleaver, mapped to complex symbols using a 4-, 16- or 64-QAM modulation and then multiplexed over N_T spatial streams, each corresponding to a transmit antenna. Each symbol stream is expressed as a vector $\boldsymbol{t}_i[n] = [t_i[n,0], \ldots, t_i[n, N_\mathrm{K}-1]]^T \in \mathbb{C}^{N_\mathrm{K}\times 1}$, where $t_i[n,k]$ is the complex symbol at time n on sub-carrier k transmitted over the ith antenna. Each of the spatial FD streams is processed by an OFDM modulator which outputs the final TD vector $\boldsymbol{s}_i$ transmitted by the ith transmit antenna.

The channel model used in this chapter is a frequency-selective Rayleigh fading channel with a power delay profile according to the typical urban COST259 model. It is time variant with the correlation according to Jake's model with a normalized Doppler frequency of $f_d = 1.4468 \cdot 10^{-5}$, a sub-carrier spacing of 15 kHz, a user velocity $v = 50\,\mathrm{km/h}$ and a carrier frequency $f_\mathrm{c} = 2.4\,\mathrm{GHz}$.

The frame structure of the system setup used throughout this chapter is shown in Fig. 1. The first OFDM symbol of a frame is a preamble following an orthogonal preamble scheme over the transmit antennas as proposed in [8]. The subsequent OFDM symbols are only consisting of data symbol vectors across all sub-carriers.

After the DFT processing of the received TD samples the $N_{\mathrm{K}} \times 1$ vector $\boldsymbol{r}_j$ is obtained at the jth receive antenna and can be written as:

$$\boldsymbol{r}_j[n] = \sum_{i=0}^{N_{\mathrm{T}}-1} \boldsymbol{H}_{i,j}[n]\boldsymbol{t}_i[n] + \boldsymbol{w}_j[n] \tag{1}$$

with $\boldsymbol{H}_{i,j}[n] = [H_{i,j}[n,0], \ldots, H_{i,j}[n,N_{\mathrm{K}}-1]]^T \in \mathbb{C}^{N_{\mathrm{K}}\times 1}$ being the channel frequency response between the ith transmit antenna and the jth receive antenna. $\boldsymbol{w}_j[n]$ is additive white complex Gaussian noise. For the sake of a shorter notation the time index n will be omitted in the remainder of the chapter.

The receiver model considered throughout this chapter is depicted in Fig. 2. Each received data symbol vector is iteratively processed by a soft-in soft-out (SISO) detector and a SISO decoder. A data symbol vector is defined as the vector over all receive antennas at the kth sub-carrier. Detection is performed by a max-log MAP SISO sphere detector (SD) with QR decomposition [9], while the channel decoder is a BCJR decoder providing soft information of the coded bits $\{c\}$. For the simulation results each processing block is executed twice, corresponding to one complete iteration between the detector and the decoder is performed.

The channel estimation provides the required estimate of the channel frequency response to the detector. As depicted in Fig. 2 the CE is split into two parts. First, the PACE processing block calculates an initial CIR estimate based on the preamble. This initial estimate is used to decode the second OFDM symbol of the frame. Second, the DD-CE block uses the decoder decisions of time n in order to provide an updated estimate at time $n + U_x$ for the detection of the next OFDM symbol. Thus, the third OFDM symbol is detected and decoded based on the updated channel estimate. The update frequency (U_x) can vary depending on the considered Doppler frequency.

There is an option to adjust the computational complexity of calculation of an update of the CIR. It was presented in [6]. The idea is to reduce the feedback from the decoder and therefore reduce the number of computations. Either the full feedback is used, meaning that all modulation symbols of the previous OFDM symbol are used to calculate the CIR estimate or only every second, third and so on symbol is used in the calculation.

3 The TA-SAGE Algorithm

The $\boldsymbol{H}_{i,j}$ from (1) can be expressed as the DFT of the CIR: $\boldsymbol{H}_{i,j} = \mathrm{DFT}(\boldsymbol{h}_{i,j})$, where $\boldsymbol{h}_{i,j}$ is the CIR between the ith transmit antenna and the jth receive antenna. Only the calculation of the CIR estimate $\hat{\boldsymbol{h}}_{i,j} = (\hat{h}_{i,j}[0], ..., \hat{h}_{i,j}[N_{\mathrm{L}} - 1])^T$ will be investigated in the remainder of this chapter.

The description of the TD-SAGE algorithm in [5] is modified in this chapter to remove redundant calculations for an efficient VLSI implementation.

First, a permutation matrix $\boldsymbol{P}^l$ is defined such that $\boldsymbol{P}^l\hat{\boldsymbol{t}}_i$ cyclically shifts the vector $\hat{\boldsymbol{t}}_i$ by l elements with $\hat{\boldsymbol{t}}_i$ containing the N_K complex remapped decoder hard decisions in the frequency domain for the ith antenna. Second, the TD vectors $\hat{\boldsymbol{s}}_i$ and $\boldsymbol{z}_{i,l}$ are defined as follows:

$$\hat{\boldsymbol{s}}_i = \boldsymbol{F}_{N_\mathrm{K}}^H \hat{\boldsymbol{t}}_i \qquad \in \mathbb{C}^{N_\mathrm{K}} \tag{2}$$

$$\boldsymbol{z}_{i,l} = \boldsymbol{P}^l \hat{\boldsymbol{s}}_i \qquad \in \mathbb{C}^{N_\mathrm{K}} \tag{3}$$

where $\boldsymbol{F}_{N_\mathrm{K}}$ is the N_K dimensional DFT matrix, $\boldsymbol{s}_j$ is the vector with the time-domain samples of the decisions from the decoder that were sent via the ith transmit antenna. $\boldsymbol{z}_{i,l}$ is the by l elements cyclically-shifted time-domain vector. With these definitions the signal model from (1) can be reformulated in the time domain to

$$\boldsymbol{y}_j = \boldsymbol{F}_{N_\mathrm{K}}^H \boldsymbol{r}_j = \sum_{i=0}^{N_\mathrm{T}-1} \sum_{l=0}^{N_\mathrm{L}-1} h_{j,i,l} \boldsymbol{z}_{i,l} + \tilde{\boldsymbol{w}}_j \tag{4}$$

where $\tilde{\boldsymbol{w}}_j$ is the transformed noise. The estimate of $h_{j,i,l}$ at SAGE iteration m is denoted as $\hat{h}_{j,i,l}^{(m)}$. A SAGE iteration is defined as the calculation of an update for one single tap of the CIR. Therefore, the SAGE iteration range is $m = 1, \ldots, N_\mathrm{L} N_\mathrm{T} N_i$, where N_i specifies how often the algorithm iterates over the complete CIR. The initial estimate provided by the PACE is $\hat{h}_{j,i,l}^{(0)}$.

The iterations are done for each receive antenna independently. The whole processing of the algorithm is done on the TD samples and can be split into four steps that have to be executed for each receive antenna. For these steps a new variable is introduced. The residual $\epsilon_\mathrm{j}^{(m)}$ is the vector of the values that are remaining after subtracting the reconstruction of the observation given the current CIR estimate from the real observation $\boldsymbol{y}_j$, which is basically the current estimation error. Then, the steps of the modified SAGE algorithm are the following:

Step 1: Initialize the residual and calculate the norm for each transmit antenna vector:

$$\epsilon_\mathrm{j}^{(0)} = \boldsymbol{y}_j - \sum_{i=0}^{N_\mathrm{T}-1} \sum_{m=0}^{N_\mathrm{L}-1} \hat{h}_{j,n,m}^{(0)} \boldsymbol{z}_{n,m} \tag{5}$$

$$||\hat{\boldsymbol{s}}_i||^2 = \sum_{k=0}^{N_\mathrm{K}-1} \left(\mathrm{Re}\{\hat{s}_i[k]\}^2 + \mathrm{Im}\{\hat{s}_i[k]\}^2 \right) \tag{6}$$

Step 2: Select the current tap by

$$l = (m-1) \mod N_L \tag{7}$$

$$i = \left(\left\lfloor \frac{m-1}{N_L} \right\rfloor \mod N_T \right). \tag{8}$$

Step 3: Calculate a new δ based on the decoder decisions and the previous ϵ.

$$\delta^{(m)} = \frac{\boldsymbol{z}_{i,l}^{H} \boldsymbol{\epsilon}^{(m-1)}}{||\hat{\boldsymbol{s}}_i||^2} \tag{9}$$

Step 4: Update the selected tap and the residual by

$$\hat{h}_{j,i,l}^{(m)} = \hat{h}_{j,i,l}^{(m-1)} + \delta^{(m)} \tag{10}$$

$$\boldsymbol{\epsilon}_j^{(m)} = \boldsymbol{\epsilon}_j^{(m-1)} - \delta^{(m)} \boldsymbol{z}_{i,l}. \tag{11}$$

Step 2 to 4 are repeated N_i times for every tap of the estimated CIR.

A variable feedback from the decoder was evaluated in [6]. There the authors explained that they use for a reduced feedback either every second, fourth symbol and so on. In the time domain this means that only the first $N_{K_F} = N_K/F_b$ time-domain samples are used to calculate an update of the CIR estimate. The variable F_b is defined to be a power of two, where F_b greater 8 is not considered. This can be directly realized using the formulars (5) - (10) with the first elements of the vectors $\boldsymbol{z}$, $\boldsymbol{s}_i$ and $\boldsymbol{\epsilon}_j$ respectively.

4 Algorithm Evaluation

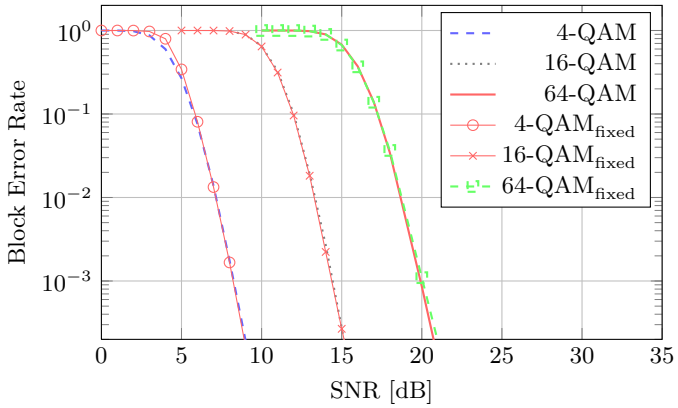

Fig. 3. Block error rate for the TA-SAGE with 4-,16- and 64-QAM. Using all available time-domain samples (N_K)

Fig. 3 shows the block error rate (BLER) for the investigated modulation schemes 4-,16- and 64-QAM using a floating-point implementation. A block is defined as one code word which is spread over one OFDM symbol. The simulations for 4-, 16- and 64-QAM were performed with $N_i = 3$ iterations. The number of estimated taps is $N_L = 32$ which equals the length of the cyclic prefix. This is a worst case assumption for the presented OFDM system which is used throughout this work. Besides the floating-point simulations the results for

a fixed-point implementation are shown in Fig. 3. The degradation due to the fixed-point arithmetic is negligible.

For the follwing evaluation the modulation 4-QAM was chosen exemplary and the number of internal iteration is $N_i = 1$. In Fig. 1 the update frequency is given by U_x. In the following two different options to reduce the computational complexity by a factor of four are evaluated for two different exemplary operation points. The plot depicted in Fig. 4 shows the algorithmic performance for 4-QAM and a mobile device speed of $v = 50\,\mathrm{km/h}$ and an $U_x = 1$ and $U_x = 4$. The loss in algorithmic performance is about 1 dB at a BLER of 1 %. Additionally the BLER for a reduced feedback from the decoder is plotted. There F_b equals 4 which means that every 4th symbol from decoder is used to calculate the update of the CIR estimate. In this case leads to the same computational complexity than using the full feedback but updating the CIR estimate only every 4th OFDM symbol. From an algorithmic perspective it can be concluded from Fig. 4 that for the given mobile speed and the same computational complexity it is a better choice to update the CIR estimate every 4th OFDM symbol using the full decoder feedback.

Fig. 5 shows a BLER plot with the same parameters of U_x and F_b but evaluated at a mobile device speed of $v = 100\,\mathrm{km/h}$. The loss in terms of algorithmic performance is in this case for an $U_x = 4$ about 7 dB at a BLER of 10 % using the full decoder feedback, compared to updating the CIR estimate every OFDM symbol. Apart from that it also shows a clear error floor at a BLER of $3 \cdot 10^{-2}$. The second option using only one 4th of the decoder feedback for the calculations shows at $v = 100\,\mathrm{km/h}$ a loss of 2 dB at a BLER of 10 %. The plot depicted in Fig. 5 also shows that at the given speed it is better to use a fourth of the decoder feedback every OFDM symbol instead, when the computational complexity should be kept constant.

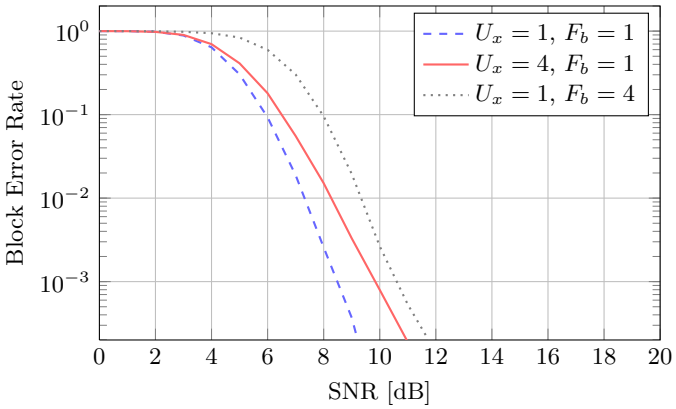

Fig. 4. Block error rate for the TA-SAGE with 4-QAM at a speed of $v = 50\,\mathrm{km/h}$

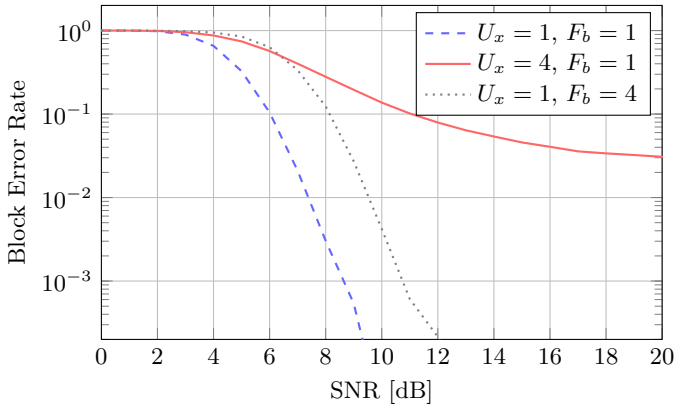

Fig. 5. Block error rate for the TA-SAGE with 4-QAM at a speed of $v = 100\,\mathrm{km/h}$

5 TA-SAGE VLSI Architecture

The architecture is split into three processing units and three different memories. The processing units are the residual update (RU) unit, the scalar product (SP) unit and the tap unit. The memories are the residual memory, the TX memory which stores the re-modulated symbol decisions from the decoder and the tap memory used to store all taps of the channel impulse response $\boldsymbol{h}$. Fig. 6 shows these blocks and depicts the memory accesses from each block.

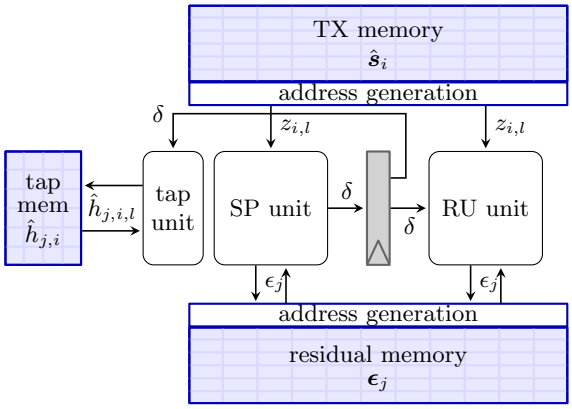

Fig. 6. High level architecture

5.1 Processing Schedule

The processing is split into four different phases: load, pre-computation, iteration and write-back. The load and the write-back phases do not include any computation but are necessary to load the input data in the memories and write back the results. These phases are considered for completeness of the hardware

complexity analysis. In the load phase the received data $\boldsymbol{y}_j$, the current CIR estimate $\hat{\boldsymbol{h}}$ and the decoder feedback $\hat{\boldsymbol{s}}_i$ are loaded into the memories depicted in Fig. 6.

The first processing phase (pre-computation) corresponds to step 1 of the algorithm description. First, the scalar $||\hat{\boldsymbol{s}}_i||^2$ is calculated for all transmit antennas in the SP unit. Second, the residual vector $\boldsymbol{\epsilon}_j^{(0)}$ for all receive antennas is calculated in the SP and RU unit. Both units are running concurrently, processing different receive antennas. In parallel to the $\boldsymbol{\epsilon}_j^{(0)}$ calculation the reciprocal of $||\hat{\boldsymbol{s}}_i||^2$ is pre-computed for all transmit antennas, since it does not change over the internal iterations.

The second processing phase is the iteration phase, which corresponds to steps 2, 3 and 4 of the algorithm. Step 2 is reflected in the dedicated address generation of each memory. Step 3 is executed by the SP unit calculating the inner product of (9) and the multiplication with the scaling factor $\frac{1}{||\hat{\boldsymbol{s}}_i||^2}$. The last steps of the algorithm are (11), executed on the RU unit and (10) calculated by the tap unit. To achieve full utilization of the processing units and account for the data dependencies between (9) and (11) the SP and RU unit are separated by a pipeline register and execute the calculations concurrently for different receive antennas. This is possible since there is no data dependency between different receive antennas, which is a property of the SAGE algorithm.

In the write-back phase the new calculated estimate of the CIR is written from the tap memory to the output ports.

5.2 Processing Units

The processing units are the parts of the architecture that are executing the calculations of (5) to (11). Apart from the algorithmic parameters defined in the previous sections, the main architectural design parameter is the data path parallelism w.

SP Unit Section 5.1 discussed that the SP unit is used in two phases and calculates (5), (6) and (9). It can be seen from (5) that all complex multiplications can be executed in parallel. Therefore, it is possible to have a data path parallelism up to N_K. In (6) and (9) it is necessary to accumulate the result of the concurrent calculations. This is implemented via an adder tree. Due to the high data path parallelism (up to $w = 32$) the maximum achievable frequency is determined by these adder trees. This leads to the design decision to have a dedicated pipeline stage as shown in Fig. 7 (third pipeline stage). The separation into the first and second pipeline stage is done to avoid two real multipliers in chain. This unit includes $6 \cdot w$ multipliers and $3 \cdot \log(w) + 7 \cdot w$ adders. The multipliers in the first pipeline stage are active in the pre-computation phase and the iteration phase. The dotted part in Fig. 7 is only active during the pre-computation phase, where first $||\hat{\boldsymbol{s}}_i||^2$ is computed and written into a register file and then the initial residual vector $\boldsymbol{\epsilon}_j^{(0)}$ is computed while the sequential divider concurrently outputs all $\frac{1}{||\hat{\boldsymbol{s}}_i||^2}$. The dashed part is active in the iteration phase calculating $\delta^{(m)}$.

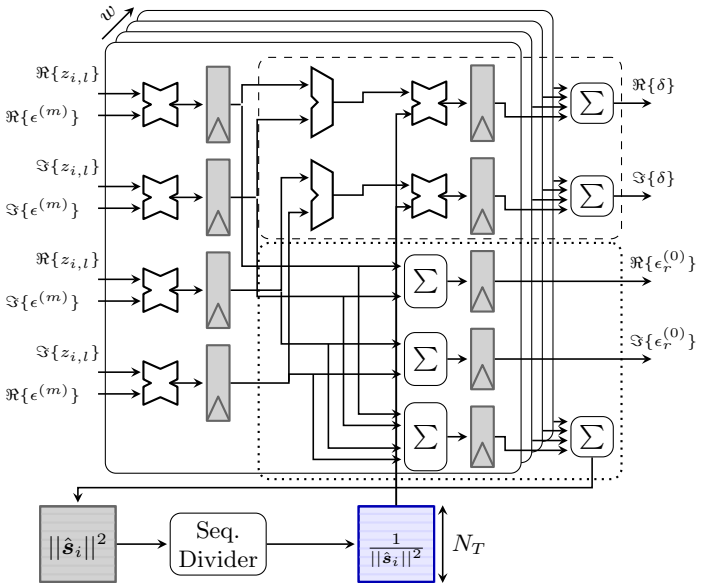

Fig. 7. SP Unit

RU Unit The RU unit also computes (5). Thus, the structure of this unit differs only slightly from the SP unit. The calculation of (11) allows for parallel complex multiplications up to N_K. The separation into three pipeline stages for this unit is done to achieve a balanced design. The output registers are added to ensure the same latency for the RU and SP unit, which eases the scheduling of the memory accesses. The complexity in terms of multipliers and adders of the RU unit is $4 \cdot w$ multipliers and $4 \cdot w$ adders.

Tap Update Unit This unit updates the current tap (10). Due to the low requirements in terms of throughput and the low complexity (2 adders) of this unit compared to the RU and SP units it is no longer discussed separately.

5.3 Memory Architecture

As shown in Fig. 6 the design has three different memories. Each of these memories has a dedicated controller that includes an address generation unit and multiplexers to realize the different data access schemes. The first memory is the tap memory, which stores the $N_\mathrm{R} N_\mathrm{T} N_\mathrm{L}$ taps of the CIR. This memory has the most relaxed constraints in the architecture. During the initialization phase it is read in every cycle from the RU and the SP unit with a linear addressing scheme. In the iteration phase the tap memory is read and written once per tap update, i.e. every $\frac{N_\mathrm{K}}{w}$ cycles. Therefore, one read/write port is sufficient.

The TX memory stores the $N_\mathrm{T} N_\mathrm{K}$ TD samples of the complex symbols of the remapped decisions from the decoder. The circular shift in (3) is realized

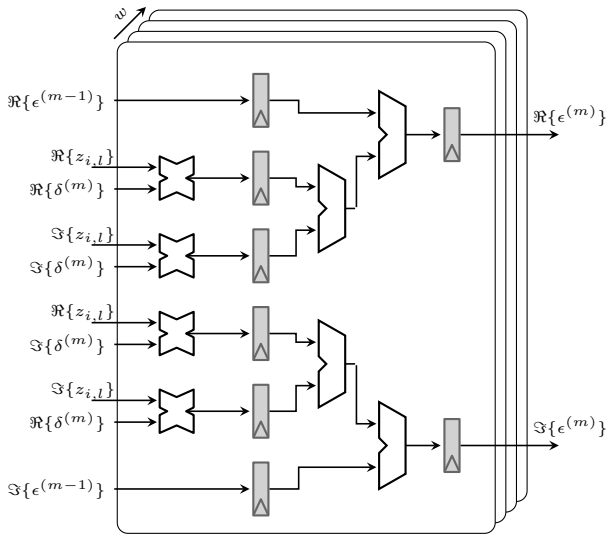

Fig. 8. RU Unit

as part of the address calculations. This memory is read by the RU and SP unit during the pre-computation phase and the iteration phase every cycle in parallel and written in the load phase. Each access reads/writes w elements in parallel. Therefore, two read/write ports with a word width of w elements are implemented.

The third memory is the residual memory. It stores the $N_\mathrm{R}N_\mathrm{K}$ ϵ-values and needs to be read by the SP unit (9) and read and written by the RU unit (11) independently and concurrently with a word width of w elements. Furthermore, during the pre-computation phase the SP and RU units read and write independently the residual memory (5). Thus, the residual memory has two read and two write ports with a data width matching the data parallelism w.

With w and the algorithmic parameters N_T, N_R, N_L, N_K and N_i, the cycle count of each phase can be calculated using the following equations.

Load phase:

$$c_\mathrm{load} = N_\mathrm{T}N_\mathrm{R}N_\mathrm{L} \tag{12}$$

Pre-computation phase:

$$c_\mathrm{precomp} = N_\mathrm{T}N_\mathrm{K}(1 + N_\mathrm{L})/w + 2 \tag{13}$$

Iteration phase:

$$c_\mathrm{iter} = \frac{N_\mathrm{K}}{w} N_\mathrm{T}N_\mathrm{R}N_\mathrm{L}N_i + 3 + 2 \tag{14}$$

Write-back phase:

$$c_\mathrm{wb} = N_\mathrm{T}N_\mathrm{R}N_\mathrm{L} + 2 \tag{15}$$

The additive constants are needed because the pipeline of the processing units needs to be empty before the next phase can start. In the pre-computation phase the RU and the SP unit are running completely independently, leading to an overhead of two cycles. In the iteration phase the complete pipeline is the concatenation of the SP and RU unit. Thus the latency is 5. The latency of two in the write-back phase is due to the pipelined access to the memory. The total cycle count for one update of the CIR is the sum of the cycle counts of the four phases.

$$c_{\text{total}} = c_{\text{load}} + c_{\text{precomp}} + c_{\text{iter}} + c_{\text{wb}} \tag{16}$$

6 Implementation Results

The architecture was synthesized using a 90 nm, 1.0 V standard-performance standard cell library with Synopsys Design Compiler 2010.12-SP2 and layouted with Cadence SoC Encounter 9.1. In the following first the implementation results for the full feedback architecture are presented assuming $N_i = 3$, $N_T = 2$, $N_R = 2$, $N_L = 32$ and $N_K = 512$. Second the results after extending the architecture to support the flexible decoder feedback is presented.

Full Feedback Architecture

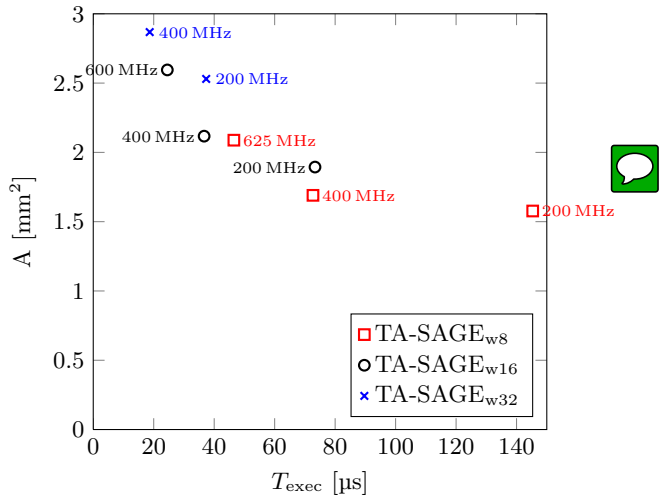

Fig. 9. Area-time trade-offs for different degrees of parallelism w and different synthesis/layout constraints. The algorithmic parameters are $N_i = 3$, $N_T = 2$, $N_R = 2$, $N_L = 32$ and $N_K = 512$.

Three different configurations of the architecture were implemented. A configuration is defined by its data path parallelism $w = \{8, 16, 32\}$. Each configuration was synthesized and layouted for its maximum achievable frequency

and additionally for 400 MHz and 200 MHz. There are only two different design points for $w = 32$ since the maximum achievable frequency is 400 MHz.

The area-time trade-off diagram for the architecture variants is shown in Fig. 9. In this diagram T_{exec} is defined as the time that the specific architecture requires to calculate a complete update of the CIR.

The best $AT_{\text{exec}} = 53.55\,\text{mm}^2\mu\text{s}$ product is the configuration with $w = 32$ and a synthesis and layout constraint of 400 MHz. However, the following discussion will focus on the configurations with an execution time around 70 µs. The configuration with data path parallelism $w = 8$ @ 400 MHz has the $AT_{\text{exec}} = 122.81\,\text{mm}^2\mu\text{s}$ product. Doubling the parallelism $w = 16$ and halving the frequency ($AT_{\text{exec}} = 138.9\,\text{mm}^2\mu\text{s}$), leads to the same execution time and only a slight increase in terms of area. This stems from the fact that this architecture is memory dominated while an increase in the data path parallelism does not influence the memory as much as the data path (Table 1).

The memories in the presented architecture are implemented using standard cell based memories (SCM) [10]. In this work flip-flop SCMs are used. The TX memory needs to be split into w banks each providing one word to allow for non-aligned vector accesses. This would lead to 32 macro cells for the maximum configuration, rendering floor-planning difficult. Therefore, the SCMs were utilized in this architecture.

Besides area and timing analysis, post-layout simulations were performed to obtain power estimates for the different configurations. The post-layout simulations with timing annotations were executed for independent test vectors for each configuration in order to obtain statistic toggling information. Synopsys Power Compiler uses the post-layout netlist and the annotated toggling information to calculate the average power estimates.

The results of the power analysis are shown in Fig. 10. The execution time T_{exec} is the same as in the AT plot depicted in Fig. 9. The power-based analysis leads to different conclusions than the AT-based analysis. The energy for a certain configuration to compute one CIR estimate is the PT_{exec} product. Comparing the configuration $w = 8$ @ 400 MHz with $w = 16$ @ 200 MHz in terms of smallest AT product leads to choose the configuration $w = 8$ @ 400 MHz. The same comparison achieving the lowest energy dissipation leads to the choice of the configuration $w = 16$ @ 200 MHz. The main reason that a doubled data path parallelism and a halved clock frequency leads to a better design decision in terms of energy dissipation is the fact that this architecture is memory dominated. This means that halving the frequency saves more energy in the memories than it is added by doubling the data path parallelism, because the memory including the memory controller is not affected as much as the processing units by an increase of the data path.

The area requirements of the two aforementioned design points are split into the different parts of the architecture in Table 1. As expected the tap memory is the smallest one since it only has to store $N_{\text{L}}N_{\text{T}}N_{\text{R}}$ elements. From this breakdown the aforementioned memory domination of this architecture is obvious. This is also supported by the layout for the configuration with $w = 8$ @

Table 1. Area breakdown for the TA-SAGE $w = 16$ @ 200 MHz and $w = 8$ @ 400 MHz

	area $w = 16$ [µm^2]	area $w = 8$ [µm^2]
Tap mem	74284.8 (3.98%)	76822.6 (4.6%)
TX mem	515499.5 (27.62%)	518432.5 (31.08%)
Resid mem	747217.1 (40.03%)	767431.7 (46.01%)
RU unit	154939.6 (8.3%)	90029.9 (5.4%)
SP unit	374723.8 (20.07%)	215378.1 (12.91%)

400 MHz in Fig. 11. The layout shows that the SCM based memory approach makes it possible that the memory is placed where needed. In section 5.3 the discussion of the residual memory revealed that the SP and the RU unit are either writing it in parallel with w parallel accesses in the initialization phase or reading and writing it during the iteration phase. It can be seen that due to this constraints the RU unit is surrounded by the residual memory and the SP unit is placed close to it.

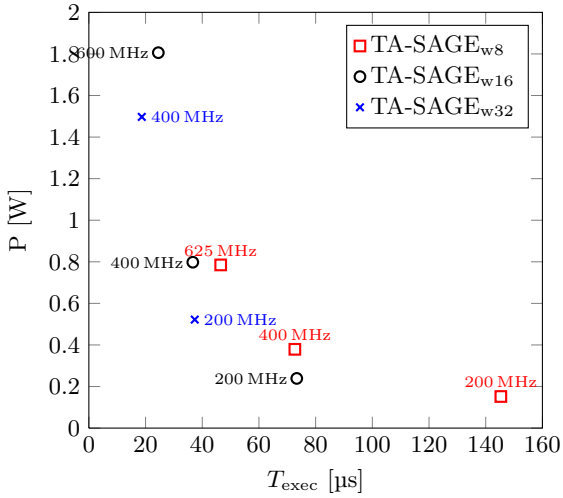

Fig. 10. Power-time trade-offs for different configurations of the TA-SAGE and different synthesis/layout constraints. The algorithmic parameters are $N_i = 3$, $N_T = 2$, $N_R = 2$, $N_L = 32$ and $N_K = 512$.

Flexible Feedback Architecture

The extension of the architecture to support the flexible feedback involved adjustements in the state machine and therefore in the schedule. Fig. 12 compares the implementation results for the full and the flexible feedback architectures.

It can be seen that for 400 MHz the area increase becomes visible in the diagram. This comes from the fact, that the modifications have an influence on

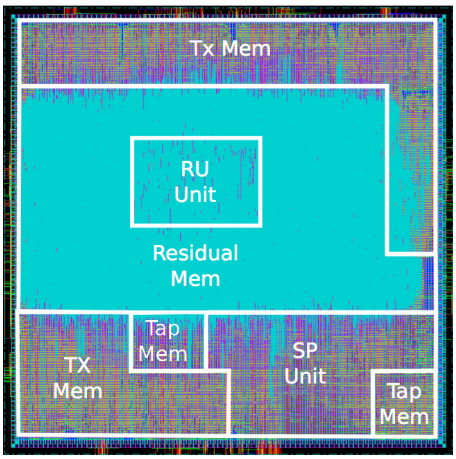

Fig. 11. Layout of the TA-SAGE ASIC for a parallelism degree of $w = 8$.

the critical path and therefore to achieve the given timing constraint more area has to be invested. The average increase in area is up to 10 %.

Table 2. Power comparision of the full and flexible feedback architecture with different frequencies and a $w = 8$

Frequency	TA-SAGE	TA-SAGE$_{\text{flex}}$
100 MHz	77.63 mW	82.84 mW
200 MHz	152 mW	162.24 mW
400 MHz	374 mW	400.1 mW

The same can be observed for the power dissipation depicted in Tab. 2. The procentual increase in the power is the same as the area increase for this architecture configuration. Therefore, the additional power only comes from the fact that the extension for the flexible feedback support has an influence on the critical path.

7 Conclusion

In this chapter we present to the best of our knowledge the first ASIC implementation of a decision directed channel estimation for MIMO-OFDM for M-QAM. The architecture is described and formulas for the calculation of the run-time of the algorithm depending on its parameters on the architecture are presented. The implementation is characterized in terms of area-time trade-offs and power dissipation.

It was shown, that the additional hardware costs of a channel tracking algorithm like the TA-SAGE are high compared to traditional PACE as presented in [11] but it is possible and therefore worth further investigations.

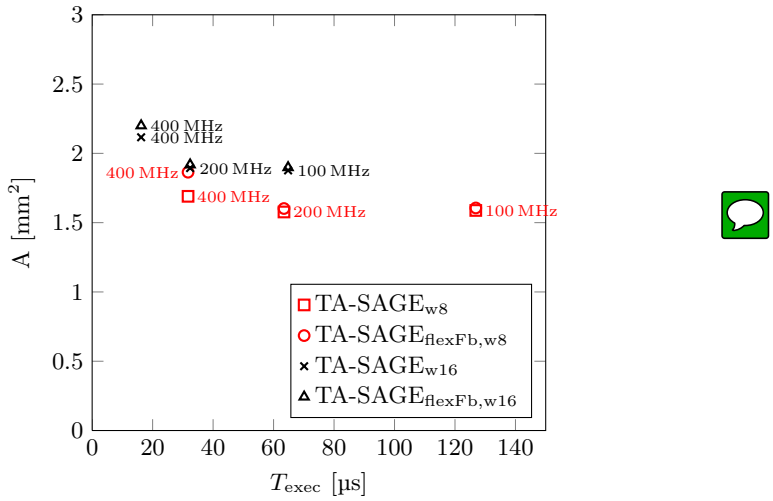

Fig. 12. Area-time trade-offs for different degrees of parallelism w and different synthesis/layout constraints. The algorithmic parameters are $N_i = 1$, $N_T = 2$, $N_R = 2$, $N_L = 32$ and $N_K = 512$.

Future work will include the influence of using latched based SCMs as presented in [10] and the evaluation of mixing macro cell memories (e.g. for the residual and the tap memory) with the SCMs approach for the TX memory. Furthermore, this architecture will be compared with simplified algorithms as for example in [12].

Acknowledgments This work has been supported by the UMIC Research Centre, RWTH Aachen University. The authors would like to thank Ernst Martin Witte, David Kammler, Martin Senst, Filippo Borlenghi and Uwe Deidersen for the valuable discussions and their feedback.

References

1. Gao, J., Liu, H.: Low-complexity MAP channel estimation for mobile MIMO-OFDM systems. In: IEEE Transactions on Wireless Communications. Volume 7. 774–780
2. Li, Y., Seshadri, N., Ariyavisitakul, S.: Channel estimation for OFDM systems with transmitter diversity in mobile wireless channels. In: IEEE Journal on Selected Areas in Communications. Volume 17. (1999) 461–471
3. Ylioinas, J., Juntti, M.: Iterative joint detection, decoding, and channel estimation in turbo-coded MIMO-OFDM. In: IEEE Transactions on Vehicular Technology. Volume 58. (2009) 1784–1796
4. Xie, Y., Georghiades, C.: Two EM-type channel estimation algorithms for OFDM with transmitter diversity. In: Communications, IEEE Transactions on. Volume 51. (Jan 2003) 106–115

5. Ylioinas, J., Raghavendra, M., Juntti, M.: Avoiding matrix inversion in DD SAGE channel estimation in MIMO-OFDM with M-QAM. In: Vehicular Technology Conference Fall (VTC 2009-Fall), 2009 IEEE 70th. (Sept. 2009) 1–5
6. Ketonen, J., Juntti, M., Ylioinas, J.: Decision directed channel estimation for improving performance in LTE-A. In: Signals, Systems and Computers (ASILOMAR), 2010 Conference Record of the Forty Fourth Asilomar Conference on. (nov. 2010) 1503–1507
7. Minwegen, A., Auras, D., Ascheid, G.: A multimode decision-directed channel estimation ASIC for MIMO-OFDM. In: VLSI and System-on-Chip (VLSI-SoC), 2012 IEEE/IFIP 20th International Conference on, IEEE (2012) 65–70
8. Li, Y.: Simplified channel estimation for OFDM systems with multiple transmit antennas. In: IEEE Transactions on Wireless Communications. Volume 1. (2002) 67–75
9. Studer, C., Bölcskei, H.: Soft-input soft-output sphere decoding. In: Information Theory, 2008. ISIT 2008. IEEE International Symposium on. (July 2008) 2007–2011
10. Meinerzhagen, P., Roth, C., Burg, A.: Towards generic low-power area-efficient standard cell based memory architectures. In: Circuits and Systems (MWSCAS), 2010 53rd IEEE International Midwest Symposium on. (Aug. 2010) 129–132
11. Simko, M., Wu, D., Mehlfuehrer, C., Eilert, J., Liu, D.: Implementation aspects of channel estimation for 3GPP LTE terminals. In: Wireless Conference 2011 - Sustainable Wireless Technologies (European Wireless), 11th European. (April 2011) 1–5
12. Qiao, X., Zhao, H., Han, Z., Sun, Y.: Decision-directed channel estimation for MIMO-OFDM systems. In: Wireless Communications, Networking and Mobile Computing, 2009. WiCom '09. 5th International Conference on, Beijing 1–4