Design and Optimization of a Digital Baseband Receiver ASIC for GSM/EDGE

Christian Benkeser, Qiuting Huang

▶ **To cite this version:**

Christian Benkeser, Qiuting Huang. Design and Optimization of a Digital Baseband Receiver ASIC for GSM/EDGE. 18th International Conference on Very Large Scale Integration (VLSISOC), Sep 2010, Madrid, Spain. pp.100-127, 10.1007/978-3-642-28566-0_5 . hal-01515990

HAL Id: hal-01515990

https://inria.hal.science/hal-01515990

Submitted on 28 Apr 2017

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Distributed under a Creative Commons Attribution 4.0 International License

Design and Optimization of a Digital Baseband Receiver ASIC for GSM/EDGE

Christian Benkeser[1,2] and Qiuting Huang[1,2]

[1] Integrated Systems Laboratory
ETH Zurich
8092 Zurich, Switzerland
http://www.iis.ee.ethz.ch
[2] Advanced Circuit Pursuit AG
8702 Zollikon, Switzerland
http://www.newacp.ch
{benkeser,huang}@iis.ee.ethz.ch

Abstract. This paper addresses complexity issues at algorithmic and architectural level of digital baseband receiver ASIC design for the standards GSM/GPRS/EDGE, in order to reduce power consumption and die area as desired for cellular applications. To this end, the hardware implementation of a channel shortening pre-filter combined with a delayed decision-feedback sequence estimator (DFSE) for channel equalization is described. The digital receiver back-end including a flexible Viterbi decoder implementation is presented and hardware savings that can be achieved by using hard-decisions are discussed. Design trade-offs are highlighted to prove the efficiency of the implemented 2.5G multimode architecture. The ASIC in 0.13 μm CMOS technology occupies 1.0 mm^2 and dissipates only 1.3 mW in fastest EDGE data transmission mode.

Keywords: Mobile communication, GSM, EDGE, baseband, pre-filter, Levinson, equalizer, DFSE, Viterbi decoder, low-power, ASIC, VLSI.

1 Introduction

Cellular wireless communications has changed our lives in the past 20 years. Today, with 3.5 billion subscribers [1] the far most important cellular communication system GSM is used by all sorts of people all around the world. Simple 2G data communication is possible with the standard extension GPRS, but good user experience during wireless web-browsing and other non-voice applications can only be achieved since 2.5G EDGE data-services have been introduced.

EDGE increases 2G peak data rates to 240 kbps, mainly due to a higher modulation order (8PSK) and stronger channel coding. The symbol rate of 270 kbps is identical with basic GSM, but 8PSK modulation improves the spectral efficiency, which comes at the cost of receiver complexity. Especially channel

equalization and symbol demodulation is a challenging task in EDGE, and requires sophisticated low-complexity solutions to preserve the power-advantage and cost-advantage of 2G receivers. Although basic GSM can be implemented on advanced signal processors with tolerable power-levels today, EDGE requires efficient ASIC accelerators to reduce the power consumption to the required level in modern cellular phones.

In basic GSM with GMSK modulation typically the (optimum) Maximum-Likelihood Sequence-Estimator (MLSE) is used for channel equalization and demodulation. The complexity of an MLSE algorithm for 8PSK signals, however, would even exceed what can be implemented on ASICs. Therefore, in recent years a lot of work on channel equalization for EDGE has been performed. It has been shown that sub-optimum variants of the MLSE algorithm with acceptable computational complexity, like reduced-state sequence estimation [2] or decision-feedback sequence estimation [3], are suitable for channel equalization of 8PSK modulated EDGE signals (e.g., [4,5]). Turbo equalization has also been discussed [6] and proposed as a solution for EDGE [7]. However, all these explorations are simulation-based system analyses where implementation aspects and crucial design metrics like power consumption are not considered. Further, a practical multi-mode solution supporting GMSK as well as 8PSK modulation has not been treated so far.

This work describes the design of a low-power and low-cost digital baseband receiver for GSM/GPRS/EDGE with focus on the most challenging block, the channel equalization. The entire receiver chain has been mapped to dedicated hardware in order to achieve best design metrics. Hardware realizations for key components of the receiver ASIC realization have been optimized to provide the flexibility required for the specified modulation types and coding schemes. Costly hardware resources are shared to keep the silicon area low, and algorithmic/architectural optimizations have been applied to achieve very low power consumption.

1.1 Outline

The paper is organized as follows. In Section 2 the GSM baseband transmitter and channel model are introduced. In Section 3 the high-level architecture of the receiver ASIC realized in this work is presented. Section 4 describes the implemented channel equalizer solution in detail. In Section 5 the implemented channel decoder solution is explained together with an analysis of the impact of hard-/soft-decision equalizer outputs on receiver implementation complexity. Measurement results and key figures of the receiver ASIC implementation are presented in Section 6. The paper is concluded in Section 7.

2 System Overview

Fig. 1 shows the GSM/EDGE baseband transceiver system model used in this work. On the transmitter side the input data is first encoded with a convolutional

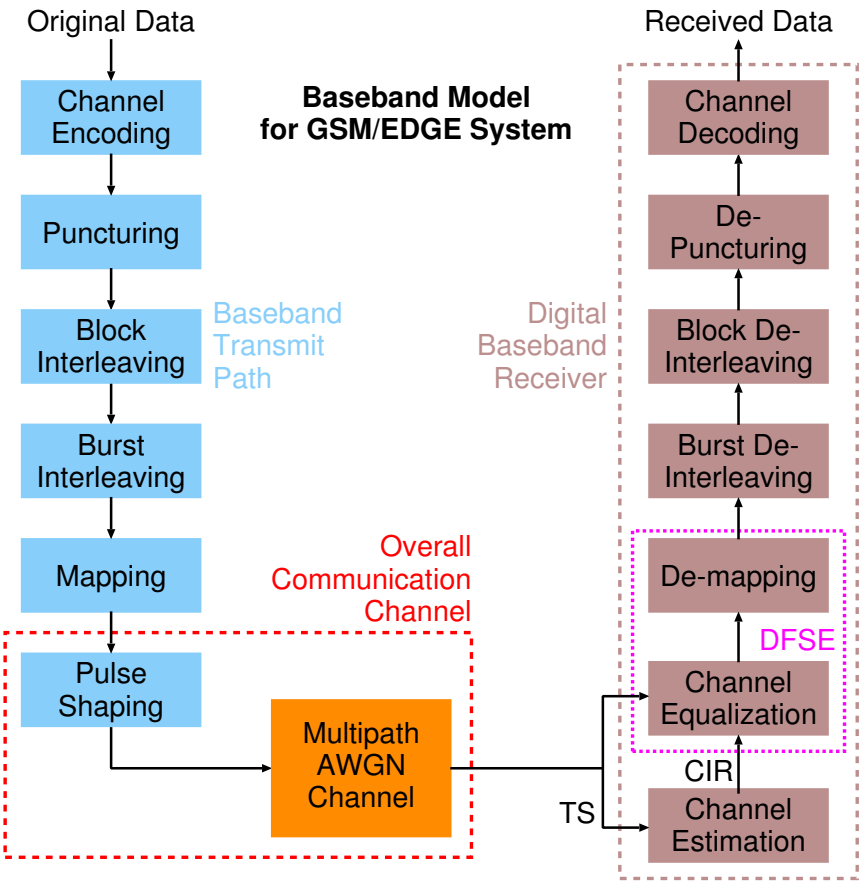

Fig. 1. System model of the GSM/EDGE baseband transceiver model.

encoder (rate 1/2 for GSM/GPRS or rate 1/3 for the coding schemes specified for EDGE). Then, selected bits are removed in the puncturing unit to adjust the code rate as required for the specific transmission mode. After that, the order of bits is scrambled in the block interleaver to distribute *radio blocks* of up to 1184 bits across 4 or 8 bursts[3]. The resulting bits (information bits) are mapped in interleaved order on bursts with a length of 156.25 symbols (3 bits per symbol in 8PSK mode), which includes fixed tail bits for channel equalization, a fixed training sequence (TS) for channel estimation and a guard period[4] as shown in Fig. 2. Finally, in the symbol mapper and pulse shaping filter the bursts are GMSK or 8PSK modulated with a symbol period of $T_s = 3.69\,\mu$s as

[3] Depending on the specific modulation and coding scheme the bits can be interleaved over 4 or 8 bursts [8].

[4] The guard period of 8.25 symbols ensures that subsequent bursts do not interfere with each other due to inter-symbol interference.

specified [9]. Typically the bursts are transmitted every 8 time slots as indicated in Fig. 2 according to the time division multiple access scheme defined for GSM, and the remaining 7 slots are reserved for other users. However, several slots can be reserved for data traffic of a single user in high-speed EDGE data transmission modes.

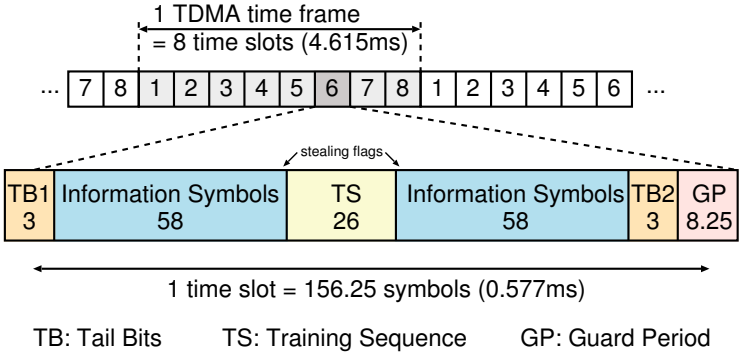

Fig. 2. Time division multiple access and the *normal burst* in GSM.

The bursts are submitted through a multipath communication channel with additive white Gaussian noise (AWGN). The GSM test channel profiles [10] have a delay spread of up to $5\,T_s$, causing considerable inter symbol interference (ISI). Even more ISI is introduced by the system itself, because the GSM channel bandwidth of 200 kHz is significantly lower than the symbol rate. Therefore, an overall communication channel with a delay spread of up to $8\,T_s$ has to be considered when removing ISI in the receiver with the channel equalizer.

3 Digital Baseband Receiver

The complete receive chain shown in Fig. 1 has been implemented in our ASIC. As indicated in the corresponding simplified block-diagram (Fig. 3) two 2.5 kb RAMs allow the buffering of up to 2 GSM bursts, each comprising the in-phase and quadrature-phase component of 156 symbols. Each time a new GSM burst has been received the channel impulse response (CIR) is estimated with the training sequence by using the Least Squares technique [11]. The ISI of unknown data symbols adjacent to the training sequence causes estimation errors when considering all 26 symbols of TS. In our implementation only the central 16 symbols of TS are used to estimate the CIR. Since linear 8PSK modulation can be realized with simple FIR filtering in the transmitter, the CIR of the overall communication channel including the pulse shaping filter can be estimated at the receiver (cf., Fig. 1). In (non-linear) GMSK instead, the signal is modulated indirectly by filtering and accumulating the phase. Hence, the transmitter pulse shaping cannot simply be included in the estimated CIR.

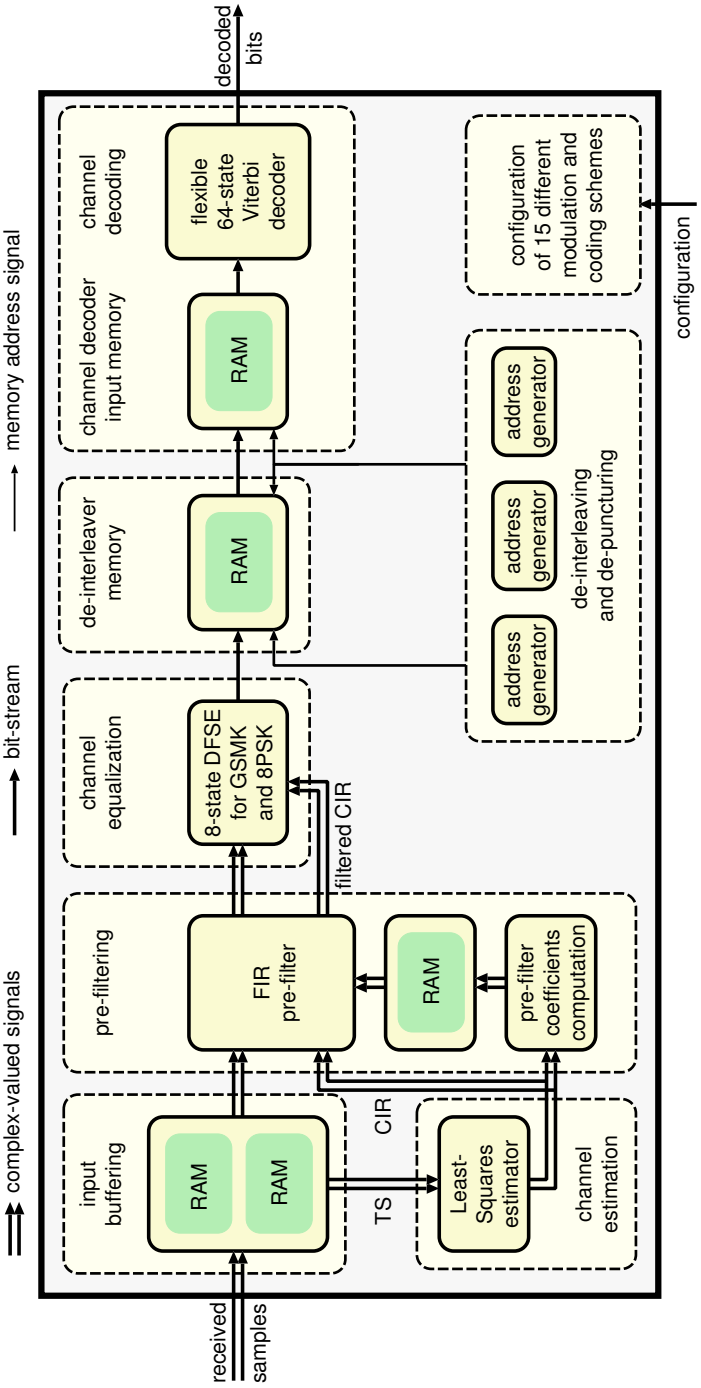

Fig. 3. GSM/EDGE receiver ASIC block diagram.

Channel equalization and demodulation are performed in our receiver implementation with the combination of a minimum-phase FIR pre-filter and a Decision-Feedback Sequence Estimator (DFSE) which are explained in detail in Section 4. The pre-filter coefficients computed with the estimated CIR are buffered to be used to filter the CIR and the stream of received samples. These are fed into the DFSE for channel equalization and demodulation.

High-speed EDGE transmission modes require continuous burst reception, because more than one burst can be allocated to a specific user in each time frame. Therefore, the demodulation of each burst has to be finished in one burst period of $T_b = 577\,\mu$s. When a radio block (4 or 8 bursts) has been demodulated, the de-interleaver can begin to reorder the bits. Both, burst and block de-interleaving are performed by generating the specific RAM addresses for writing to and reading from the de-interleaver memory. The de-puncturing unit introduces zeros (no a-priori information on the demodulated symbol) into the data stream to provide data blocks to the channel decoder with the correct code rate. A flexible Viterbi decoder for trellises with up to 64 states has been implemented to support 15 specified coding schemes. The channel decoder processes all branches in parallel to keep the receiver latency low.

4 A Combined Pre-filter and DFSE Solution

In an MLSE, as can be used for optimum equalization in GSM, all possible symbol combinations of length L that could have been transmitted are generated as reference symbol sequences. These sequences are modulated and convolved with the CIR to emulate the effect of ISI. The resulting reference signals are compared with the received samples to generate branch metrics which are a measure for the likelihood that the specific symbol sequence has been transmitted, given the received signal. The branch metrics are used by the Viterbi algorithm in a trellis diagram to find the most probable transmitted symbols. Contrary to implementations of Viterbi decoders, the branch metric generation is the most challenging part in an MLSE. The computational complexity can be characterized with the number of trellis branches $B = M^L$ to be processed, where M denotes the modulation order (alphabet size) with $M = 1$ for GMSK and $M = 3$ for 8PSK. The length of the reference symbol sequences L defines the maximum tap delay (delay spread) in the channel profile that is considered in the equalization. The storage requirements are defined by the number of states in each trellis stage according to $S = M^{L-1}$. When considering the impact of 7 previously transmitted symbols, i.e. $L = 8$, as required for GSM (cf., Section 2), the MLSE trellis is large in the GMSK case, and even beyond what can be realized with any hardware platform when processing 8PSK signals.

Contrary to the MLSE algorithm, in the DFSE not all possible symbol combinations of the reference sequence are used for the comparison with the received samples. Decisions of the Viterbi algorithm on previous symbols, so-called per-survivor estimates, are taken as fixed assumptions to reduce the number of branch metrics to be computed. Only the permutations of the first D symbols are

considered as separate state transitions in the trellis, which reduces the number of branches to be processed to $B = M^D$ and the required number of trellis states to $S = M^{D-1}$. Therefore, the taps of the CIR which correspond to these first D symbols have greatest impact on the equalizer performance. In channels with strong delayed taps and comparably weak main taps, correct symbol demodulation can be difficult or even impossible for the DFSE algorithm. Therefore, minimum-phase pre-filtering is required to transform such channels into channels better suited for the DFSE. Such a pre-filter concentrates the energy of the channel taps in the front, in order to maximize the impact of the first D symbols during equalization, thus improving the DFSE performance (cf., Fig. 4).

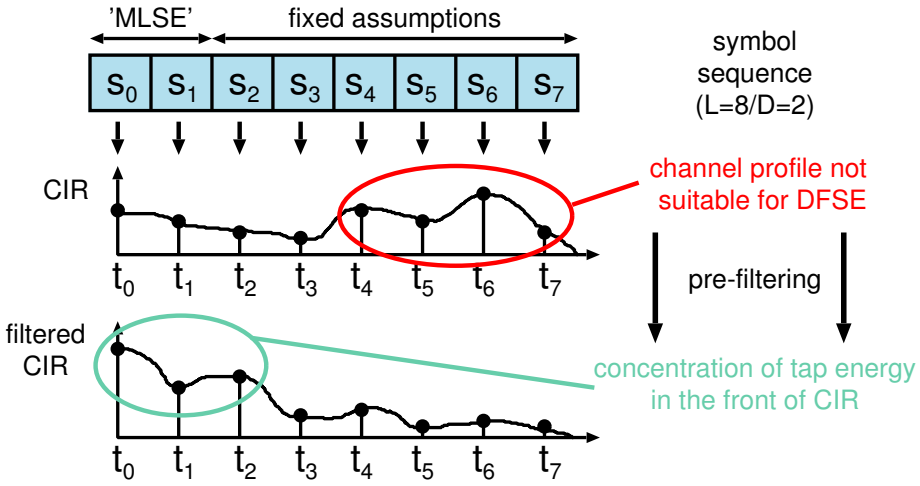

Fig. 4. The concept of pre-filtering required for the DFSE.

Simulations have shown that, depending on the channel profile, varying D from 2 to 4 brings a maximum performance gain of 1.6 dB in terms of SNR required to achieve an uncoded DFSE bit error-rate (BER) of 10^{-3} with both GMSK and 8PSK modulated signals. The simulation results for the specified Hilly Terrain (HT) channel [10] profile for different DFSE configurations with pre-filtering are shown in Fig. 5. As can be seen increasing D from 2 to 4 improves the performance in this case by only 0.9 dB and 0.6 dB for GMSK and 8PSK modulated signals respectively. Almost optimum (MLSE) performance can be achieved with $D = 4$. Further increasing of D introduces additional complexity without a noticeable performance gain.

Instead, the impact of the pre-filter can be dramatic. The uncoded BER of demodulated 8PSK signals, impaired by the HT channel are shown in Fig. 6. Different filter orders p for the pre-filter in front of a DFSE with $D = 2$ have been simulated. As can be seen, without pre-filtering the symbol sequence cannot be equalized with a DFSE properly. A pre-filter order p of only 16 causes the curves

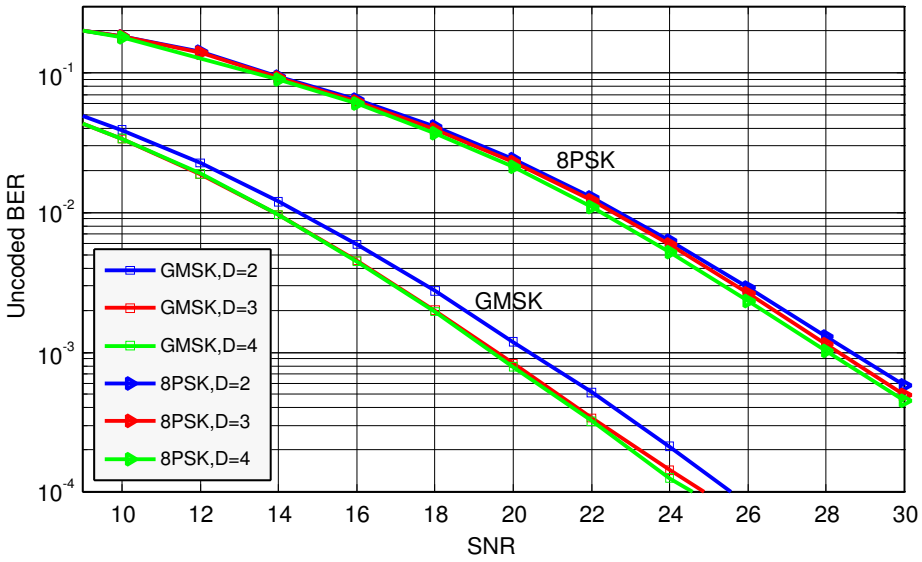

Fig. 5. Uncoded BER of GMSK and 8PSK modulated signals after pre-filtering with pre-filter order $p = 32$ and $D = \{2, 3, 4\}$. The specified Hilly Terrain (HT) channel profile (no vehicle speed) has been used for the simulations.

to saturate at a BER$\geq 10^{-3}$, whereas the BER does not improve significantly with increasing the filter order beyond 32.

The most critical parts of our GSM/EDGE channel equalizer solution, i.e., the matrix inversion for the pre-filter coefficients computation and the DFSE realization, are described in detail in the following.

4.1 Pre-filter Implementation

A suitable pre-filter transforms the CIR to its minimum-phase equivalent, where the energy of the first filter taps is maximum compared to the impulse response of all other causal and stable filters with the same magnitude response [12, 13]. Such pre-filtering can be realized with a hardware-friendly FIR filter, however, the computation of the corresponding pre-filter coefficients is costly. Hence, a thorough complexity analysis of techniques for the generation of these filter coefficients is required.

In order to be able to make a fair comparison between algorithms, we have implemented the Linear Prediction (LP) method [14] and performed floating-point simulations for a filter order of $p = 32$ with our framework. Then, the MMSE-DFE [12] and cepstrum method [15] (recently proposed for GSM/EDGE in [16]) have been implemented, and the parameters have been optimized, such that these algorithms provide the same BER performance at certain SNR points as the simulations with the LP method. It has been shown, that the cepstrum

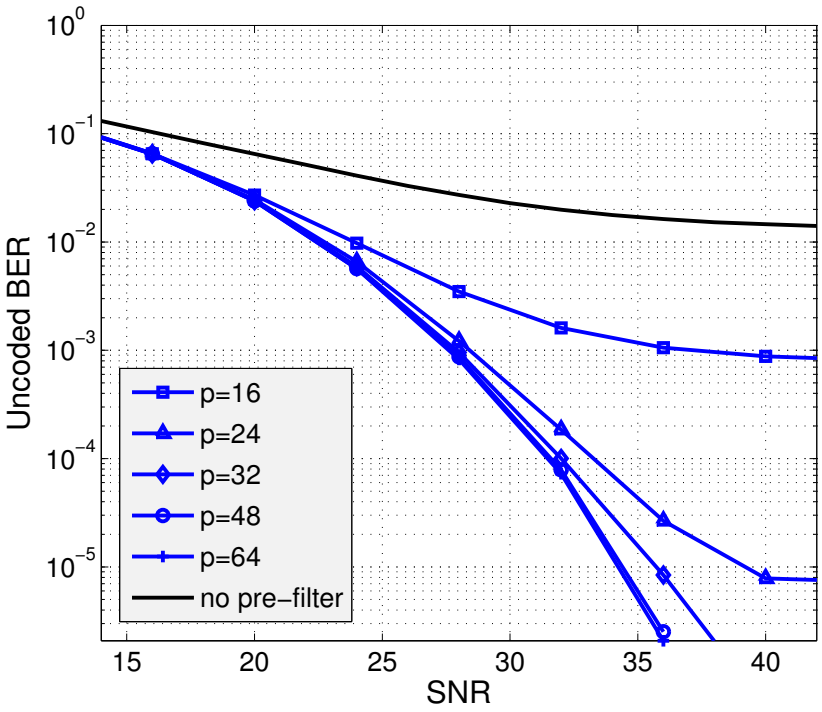

Fig. 6. Uncoded BER of 8PSK modulated signals after pre-filtering and channel equalization with DFSE ($D = 2$) with variable pre-filter order p. The specified Hilly Terrain (HT) channel profile (no vehicle speed) has been used for the simulations.

(MMSE-DFE) method requires $p = 32$ and $p = 24$ ($p = 32$ and $p = 20$) at SNR points corresponding to an uncoded BER of 1% and 0.1% respectively.

In Fig. 7 the complexity of the different algorithms in terms of number of complex-valued multiplications and additions is compared[5]. The complexity has been evaluated at different BER operating points, because the MMSE-DFE method shows slightly better performance in low SNR regimes when compared to the LP and cepstrum approach. As can be seen, the computation of the pre-filter coefficients via LP is significantly less complex than using MMSE-DFE or the cepstrum method.

Consequently, in our digital baseband receiver ASIC we have implemented a pre-filter with filter coefficients computation based on LP, which will be depicted shortly in the following (a detailed description can be found in [14]). We denote the discrete time received baseband signal as follows:

[5] Real-valued multiplications/additions have been counted as 0.25/0.5 complex-valued operations here.

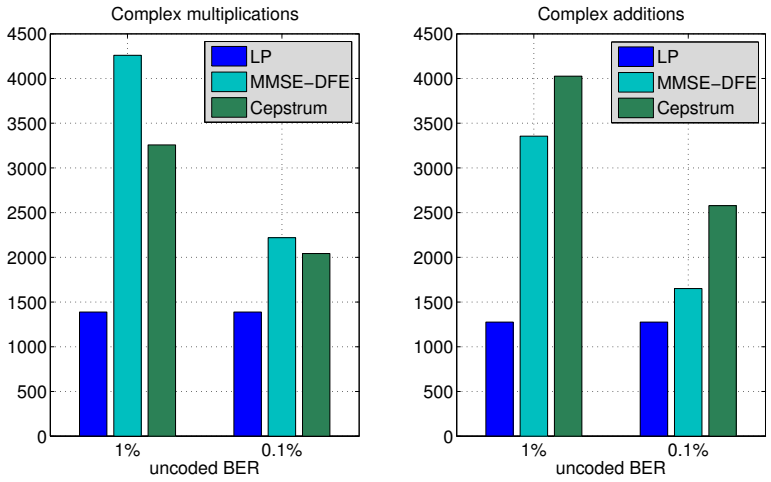

Fig. 7. Complexity comparison of different algorithms for channel shortening pre-filters.

$$r[n] = \sum_{k=1}^{L} h[k] \cdot s[n-k] + w[n] \tag{1}$$

where $h[n]$ denotes the complex baseband response of the overall channel with delay spread L (cf., Section 3), $s[n]$ the GMSK or 8PSK modulated transmitted symbols, and $w[n]$ complex additive white Gaussian noise (AWGN). $H(z)$ denotes the transfer function that corresponds to the channel:

$$H(z) = \sum_{k=1}^{L} h[k] \cdot z^{-k}. \tag{2}$$

Ideally, a discrete-time pre-filter $A(z)$ transforms $H(z)$ to its minimum-phase equivalent $H_{min}(z)$ according to

$$H_{min}(z) = A(z) \cdot H(z), \tag{3}$$

with all-pass characteristic

$$|A(z)| = 1. \tag{4}$$

Due to (4) the transfer function $A(z)$ of the desired pre-filter can be written as a cascade of two filters according to

$$A(z) = \underbrace{H^*(1/z^*)}_{A_{MF}} \cdot \underbrace{\frac{1}{H^*_{min}(1/z^*)}}_{A_{WF}}. \tag{5}$$

A_{MF} is a matched filter, matched to the channel $h[n]$, where the filter coefficients are simply the complex-conjugated and time-reversed impulse response taps. Unfortunately, obtaining the filter coefficients of the whitening[6] filter $A_{WF}(z)$ is a much more complex task that requires a costly matrix inversion.

Computation of filter coefficients with LP A linear predictor is based on the assumption that samples of a sequence can be (approximately) written as a linear combination of past samples. The linear prediction filter $P(z)$ used to compute the filter coefficients of $A_{WF}(z)$ is defined as

$$\hat{h}[n] = \sum_{i=1}^{p} a_i \cdot h[n-i], \tag{6}$$

with the linear predictor coefficients a_i and the filter order p. The prediction-error can be computed according to

$$e[n] = h[n] - \hat{h}[n]. \tag{7}$$

The filter coefficients are obtained by minimizing the mean-squared prediction error which leads to the *normal equations* or *Yule-Walker equations* [17,18]:

$$\boldsymbol{\Phi a} = \boldsymbol{\varphi}$$

with the filter coefficients vector

$$\boldsymbol{a} = \begin{bmatrix} a_1 \; a_2 \cdots a_p \end{bmatrix}^T$$

and the Toeplitz matrix

$$\boldsymbol{\Phi} = \begin{bmatrix} \varphi_0 & \varphi_{-1} & \cdots & \varphi_{-(p-1)} \\ \varphi_1 & \varphi_0 & \cdots & \varphi_{-(p-2)} \\ \vdots & & \ddots & \vdots \\ \varphi_{p-1} & \varphi_{p-2} & \cdots & \varphi_0 \end{bmatrix} \tag{8}$$

with

$$\boldsymbol{\varphi} = \begin{bmatrix} \varphi_1 \; \varphi_2 \cdots \varphi_p \end{bmatrix}^T$$

The coefficients φ_k are given by the autocorrelation function (ACF) of $h[n]$ according to

$$\varphi_k = h[k] * h^*[-k].$$

As can be seen in Fig. 8 the prediction-error filter $F(z)$ that corresponds to the linear predictor $P(z)$ is given by

[6] Details on the noise-whitening characteristic of this filter can be found in [17].

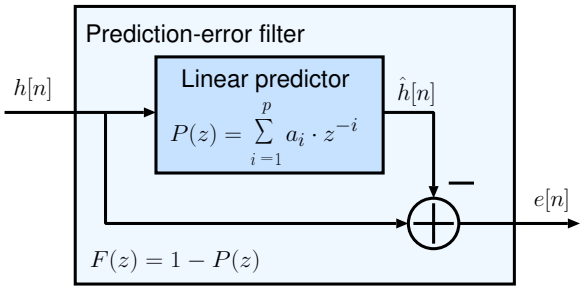

Fig. 8. Block diagram of prediction-error filter used for the pre-filter coefficients computation in the LP approach.

$$F(z) = 1 - P(z) \tag{9}$$

The generation of the A_{WF} filter coefficients exploits the fact that such prediction-error filters are always minimum phase [17]. It has been shown in [14], that

$$A_{WF}(z) = F^*(1/z^*) \tag{10}$$

is valid for an infinite filter order $p \to \infty$ and that

$$A_{WF}(z) \approx F^*(1/z^*) \tag{11}$$

is a good approximation for high filter orders p.

4.2 Matrix Inversion

In a pre-filter implementation with LP the far most complex part is the inversion of the complex-valued $p \times p$ Toeplitz matrix (8) required to generate the pre-filter coefficients. In our digital baseband receiver ASIC a pre-filter with order $p = 32$ has been implemented to guarantee high DFSE performance, even when the received signals have been impaired by channels with strong delayed taps. The required inversion of a 32×32 matrix with an internal precision of 2×20 bits has been implemented in our design by excessively applying time-multiplexing and resource sharing to curtail silicon area. To this end, the recursive Levinson-Durbin (LD) algorithm shown in Alg. 1 is used. Its recursive nature is highly suitable for sequential implementations and the computational complexity is close to optimum matrix inversion algorithms for Toeplitz matrices [19].

To improve the efficiency of an implementation of the LD recursion, the algorithm has been modified as follows:

– Initialization of the (real) scaling factor E

$$E = \varphi_0 \tag{12}$$

– Recursive generation of the (complex) coefficients a_i by looping over $1 \leq i \leq p$

- Computation of the unscaled a'

$$a' = -\left[\varphi_i + \sum_{j=i}^{i-1} a_j \cdot \varphi_{i-j}\right] \tag{13}$$

- Scaling with E to generate the new a_i

$$a_i = \frac{a'}{E} \tag{14}$$

- Updating a_j by looping over $1 \leq j \leq i-1$

$$a_j = a_j + a_i \cdot a^*_{i-j} \tag{15}$$

- Generation of the new scaling factor

$$E = (1 - |a_i|^2)E \tag{16}$$

– The LP filter coefficients of P(z) are given by a_i with $1 \leq i \leq p$

Alg. 1. LD recursion for complex coefficients.

– The initialization in equation (12) has been replaced with

$$E_{inv} = \frac{1}{\varphi_0}$$

– The scaling of a' in equation (14) has been replaced with

$$a_i = a' \cdot E_{inv}$$

– The generation of the new scaling factor in equation (16) has been replaced with

$$E_{inv} = \frac{E_{inv}}{(1 - |a_i|^2)}$$

Working with the inverse of the scaling factor $E_{inv} = 1/E$ can be exploited to process the algorithm more efficiently in hardware (cf., the block diagram of the LD algorithm implementation in our receiver ASIC in Fig. 9): the division of the complex-valued a' in (14) is replaced by a multiplication, and the multiplication in (16) is replaced with a real-valued division. Thus, one costly real-valued

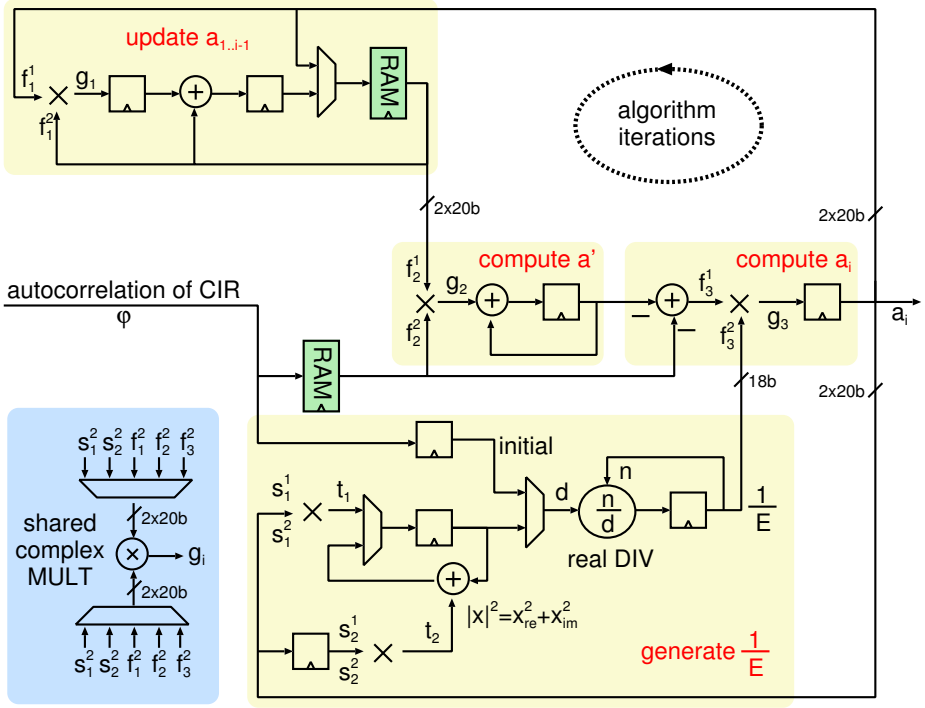

Fig. 9. Block diagram of the LD recursion implementation.

division is saved in each iteration of the LD recursion. Further, the division now has to be performed for the generation of the new (inverse) scaling factor, and therefore can be performed mostly in parallel to updating a_j. It can be realized with a low-complexity sequential divider without stalling the LD recursion loop for many clock cycles. The sequential divider implementation in our design takes 9 clock cycles per division and requires only 1.7 k gate equivalents (kGE). It significantly reduces the logic depth of the costly divide operation such that timing closure of our receiver ASIC is not affected. In our design the matrix inversion of the complex-valued 32×32 matrix takes 2630 clock cycles.

Costly hardware resources like multipliers, register arrays and RAMs, used in the LD recursion implementation, are also shared with other units of the pre-filter block. As in the implementation of the LD recursion, the sequential data flow in the pre-filter coefficients computation allows time-multiplexing of costly hardware resources. Thus, the total pre-filter coefficients computation together with the actual FIR pre-filtering requires only 32 kGE and 2.5 kb memory. The total pre-filter coefficients computation takes less than 3000 clock cycles, and the FIR pre-filter of the received burst provides one filtered symbol per 16 clock cycles to the DFSE.

4.3 DFSE Design for GMSK and 8PSK

To support processing both GMSK and 8PSK modulated signals with little overhead, our implementation of the DFSE algorithm always operates on the same number of trellis states. In GMSK mode 8 states allow $D = 4$ symbols to be considered in building the trellis, such that near MLSE performance can be achieved. In 8PSK mode with $D = 2$ the high-order pre-filter allows for good performance without having to introduce additional trellis states as previously shown. The binary alphabet in GMSK requires only 2 branches per trellis state to be processed, whereas 8PSK has 8 branches in the trellis going to each trellis state, which leads to a 4x higher computational complexity.

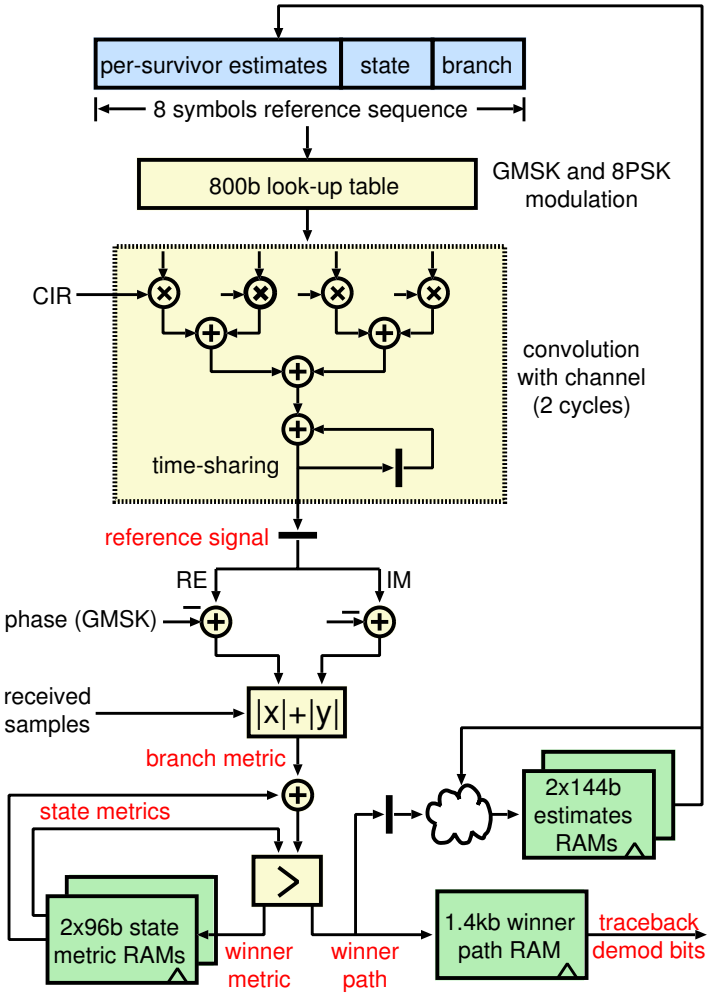

Fig. 10. Block diagram of the DFSE implementation.

The symbols of the training sequence do not have to be demodulated since they are known at the receiver[7]. Therefore, computations can be saved by simply skipping the received symbols that correspond to the training sequence. Our DFSE implementation builds two separate sub-trellises per burst: one trellis for the first 58 information symbols, using TB1 and the first symbols of TS as tail bits, and one trellis for the second 58 information symbols, using the last symbols of TS and TB2 as tail bits (cf., Fig. 2). Consequently, tail bits for both sub-trellises are known in the DFSE, which enables reliable demodulation of the information symbols with 20% less symbols to be processed.

Fig. 10 shows the block-diagram of the implemented DFSE algorithm. To enable the equalization of all specified channel profiles, channel taps with a maximum delay of $L = 8$ symbol periods are considered in our DFSE implementation (cf., Section 2). Therefore, the reference symbol sequences have a length of 8 symbols, and the (pre-filtered) CIR fed into the DFSE block consists of 8 symbol-spaced taps.

To modulate the symbols each reference sequence (one for each trellis branch) is fed into a look-up table (LUT) to generate the corresponding constellation points. In GMSK mode all $2^L = 256$ possible constellation points which arise from ISI due to the pulse shaping filter have to be stored. In 8PSK mode, where the estimated CIR also comprises transmitter pulse shaping, only 2×8 different constellation points have to be stored[8]. With a precision of 6 bit and 8 bit for the GMSK and 8PSK constellation points respectively, the LUT can be reduced to only 0.8 kb by exploiting symmetries.

The vector product of the modulated reference symbol sequence with the pre-filtered CIR, both complex-valued and with a length of 8, is by far the most computation-intensive part in a DFSE implementation: 8 multiplications and 7 additions are required for each trellis branch, leading to high implementation complexity. To keep the silicon area low we have time-multiplexed the vector product implementation for 4 symbols with 4 channel taps, thus spending 2 clock cycles for the generation of each reference signal.

In GMSK mode, after the convolution with the channel a phase rotation of $\pi/2$ introduced in the transmitter modulator [9] is compensated for[9]. Finally, the trellis branch metrics typically are computed by building the Euclidean (ℓ^2-) distance

$$d_{\ell^2} = \sqrt{(x_I - y_I)^2 + (x_Q - y_Q)^2} \tag{17}$$

[7] In our prototype ASIC implementation the training sequence code (TSC) #1 has been used. The specifications allow the base station to choose among 8 different TSCs with similar autocorrelation properties.

[8] 16 instead of 8 constellation points have to be considered, because a phase rotation of $3\pi/8$ between subsequent transmitted symbols is specified [9].

[9] The sign of the GMSK phase rotation depends on the symbol to be transmitted. However, it can be shown that due to differential encoding on the transmitter side, the received symbols can be de-rotated for the comparison in the Viterbi equalizer without having to consider previous symbols.

between the reference signal and the received sample. To avoid costly square operations in this time critical path in our implementation we generate the ℓ^1-distance

$$d_{\ell^1} = |x_I - y_I| + |x_Q - y_Q| \tag{18}$$

as an approximation. In terms of SNR the performance degradation with this sub-optimal solution is less than 0.5 dB in GMSK and within 1 dB in 8PSK mode [20].

Each branch metric is added to the state metric of the state from which the specific branch is originating. The smallest metric incident on each state is selected and stored in the state metric memory, which is double-buffered with 2 RAMs to avoid that the old state metrics are overwritten. The decision on the winner branch is stored in the *winner path* and the *estimates* RAMs (cf., Fig. 10). To avoid access problems two *estimates* RAM instantiations are required for the storage of the last $8 - D$ per-survivor estimates for each of the 8 trellis states. The *winner path* RAM stores all decisions in the trellis to find the most likely symbol sequence during final back-tracing.

The total DFSE implementation requires only 18 kGE and less than 2 kb memory. The demodulation of one GMSK burst takes 6228 clock cycles, and processing an 8PSK modulated burst requires 17360 cycles. When taking into account the time required to generate the pre-filter coefficients plus 128 cycles for the channel estimation, burst-wise operation of the fastest EDGE transmission modes is possible with a system clock frequency of only 40 MHz.

4.4 Complexity of Combined Pre-filter and DFSE

The pre-filter block is almost twice as complex in terms of silicon area as the DFSE, and occupies one third of the logic cells of the total digital receiver. As previously shown, proper pre-filtering is essential to achieve high DFSE performance, especially when reducing the number of trellis states with $D < 4$. Although, the significant complexity of the pre-filter block gives rise to the question, if a pre-filter with a lower order and a DFSE with more trellis states would achieve a similar performance at a reduced total equalizer implementation complexity.

Lowering the filter order p reduces the filter's ability to compute the optimum minimum-phase equivalent of the CIR. The lower the filter order the more energy will be (in average) in the delayed taps of the filtered CIR. Further, the all-pass characteristic of the pre-filter (cf., equation 4) deteriorates for small p. Such sub-optimum pre-filtering causes higher error rates after DFSE equalization and demodulation as shown in Fig. 6. Increasing the number of trellis states processed in the DFSE can (partially) compensate for this performance loss. But increasing the number of channel taps considered in building the trellis for example from $D = 2$ to $D = 3$ increases the number of branches to be processed by a factor of 8 in 8PSK modulation. Hence, the processing delay would increase by more than 120 k cycles with our architecture. On the other

hand, with the highly sequential and resource-saving architecture of the proposed low-complexity pre-filter implementation, going down from a pre-filter order of 32 to for example[10] 16 would save less than 2 k cycles. Furthermore, the silicon area of the pre-filter cannot be reduced significantly when lowering the filter order, because hardware resources have already been extensively re-used in the proposed architecture. Concluding, the implementation of a pre-filter with a high filter order pays off because it allows for excellent BER performance at a significantly reduced implementation complexity of the DFSE equalizer.

5 Hard-Decision Receiver Back-End Realization

The TDMA-based architecture of GSM/EDGE foresees burst-wise processing in the receiver: to save power only the signal streams on time slots with data dedicated for the specific user are equalized and demodulated. However, after the de-mapping of the bursts, de-interleaving, de-puncturing and channel decoding is required on a radio block basis, because block-interleaving is performed over 4 or 8 bursts in the transmitter signal processing chain (cf., Section 2).

In GSM/GPRS radio blocks of up to 456 bits contain one radio link control (RLC) block of information bits and in specific transmission modes (CS2-4) a short sequence called uplink state flag (USF)[11]. The coding scheme used for the information bits can be determined on the receiver side by checking the demodulated *stealing flag* bits on either side of the training sequence (cf., Fig. 2). In EDGE the radio blocks comprise the USF sequence, one or two (separately encoded) RLC blocks of information bits, and an RLC/MAC *header* part. To ensure strong header protection the header is encoded, interleaved and punctured independently from the information bits [21]. In EDGE the stealing flags indicate the coding used for the header, and the header itself provides control information, e.g. the modulation and coding scheme (MCS1-9) used for the coding of the information bits and an RLC block identifier. This block identifier is used by the *incremental redundancy* (IR) management on higher layers to determine if the received RLC blocks are a re-transmission of previously transmitted RLC blocks which have not been correctly decoded. IR uses different puncturing schemes for the re-transmission of RLC blocks, such that they can be combined with the previously received and stored blocks to increase the chance of correct decoding.

In the following de-mapping, block de-interleaving, de-puncturing and channel decoding as implemented in our receiver ASIC are described, highlighting the enormous complexity reduction that can be achieved by generating hard-decisions (instead of soft-decisions) in the DFSE equalizer implementation.

[10] Simulations have shown that the performance degradation due to reducing the filter order from 32 to 16 cannot even be compensated for completely by increasing D to 4.

[11] The USF indicates to the user equipment (UE) in which time slot a radio block can be transmitted in the case that two UEs share the same physical channel.

5.1 De-mapping, de-interleaving and de-puncturing

The implemented DFSE equalizer demodulates the received data stream and generates hard-decisions for each transmitted information symbol. Due to the nature of the Viterbi algorithm [22] the DFSE provides the demodulated symbols d_k with $k = 0..115$ for each sub-trellis corresponding to 58 information symbols i in inverse order, more specifically

$$d_k = i_{58}^{sub1}, i_{57}^{sub1}, .., i_1^{sub1}, i_{58}^{sub2}, i_{57}^{sub2}, .., i_1^{sub2}. \tag{19}$$

The symbols are de-mapped to actual bits and written into the de-interleaver memory. Burst de-interleaving is a simple task and can be performed by applying the specific memory write addresses. When a radio block of 4 or 8 bursts has been buffered in the de-interleaver memory block de-interleaving can begin.

The rules for the generation of the interleaved bit order (as defined for block interleaving on the transmitter side) seem to be complex and not suitable for an integration in hardware. Modulo and integer divisions are used in the specifications to compute the interleaved addresses, e.g., for MCS-9 the standard specifies the interleaving according to

$$\Pi(k) = 306(k \bmod 4) + 3\left(44k \bmod 102 + \left\lfloor \frac{k}{4} \right\rfloor \bmod 2\right) + \left(k + 2 - \left\lfloor \frac{k}{408} \right\rfloor\right) \bmod 3 \tag{20}$$

with $k = 0..1223$, such that each encoded bit of a radio block that was not punctured is interleaved to the address $\Pi(k)$. Obtaining the de-interleaved address $\Pi^{-1}(k)$ for a specific bit position k is even more complex and difficult to realize. However, when generating the interleaved addresses in ascending order, i.e., $k = 0, 1, 2, ..$, complex modulo and division operations can be replaced by simple add-compare-select stages and counters (an example for such a simplified implementation is illustrated in Fig. 11 for a modulo and for a divide operation of (20)). Therefore, the de-interleaving can be performed most efficiently by applying the interleaved positions (generated in ascending order) as read addresses to the de-interleaver memory and writing the corresponding data items to the channel decoder input memory in natural order. With de-/puncturing the situation is similar: generating the bit positions which have to be punctured can be implemented efficiently with counters, whereas the inverse operation of determining if a specific address has been punctured is a complex task.

In order to minimize implementation complexity in our ASIC design, we apply the write addresses to the decoder input memory in natural (ascending) order and perform the de-puncturing and de-interleaving in a combined step:

- Initialization of the de-interleaver counter to $q = 0$.
- For all $k = 0, 1, 2, ..$ in ascending order check if the kth bit has been punctured in the transmitter.
  - If 'yes', write a zero (no a-priori information available for channel decoding) to the kth position of the channel decoder input memory.

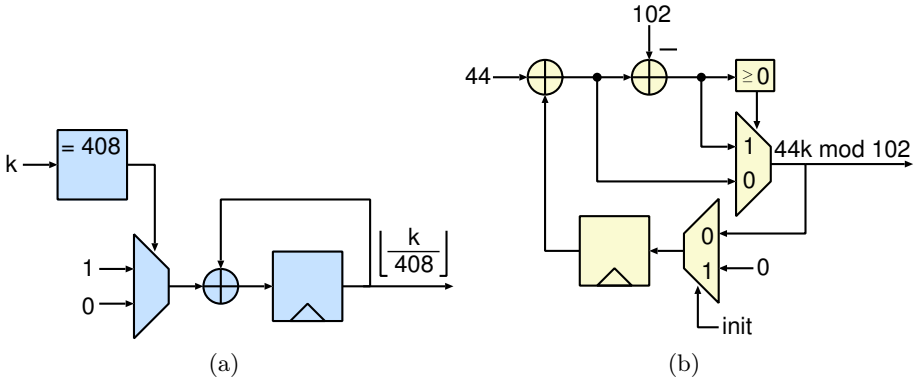

Fig. 11. Simplified implementation of integer division (a) and modulo operation (b) as required for interleaving.

- If 'no', generate $\Pi(q)$ to read the corresponding data item from the de-interleaver memory: write +1 for a stored logic '0' and -1 for a stored logic '1' to the kth position of the decoder memory, and increment the de-interleaver counter q by one.

It may seem a waste of hardware resources that memory space is provided for the zeros corresponding to the punctured bits. The de-puncturing could also be performed when reading the bits from the channel decoder input memory into the channel decoder. However, the memory space for the punctured bits is required because the received radio block could be a re-transmission initiated by the IR management. In this case, the received information and the information of the previously received and stored blocks has to be combined and written into the channel decoder input memory to be used by the channel decoder. Therefore, the capacity of the channel decoder input memory is defined by the maximum (encoded) RLC block size before puncturing.

5.2 Flexible Hard-Decision Viterbi Decoder

Channel coding for the different transmission modes in GSM, GPRS and EDGE is all based on convolutional encoding. However, the coding schemes differ in code rate r, constraint length c, decoder block size B, and trellis termination with tail bits [8]:

- The convolutional code for basic GSM and GPRS transmission modes has a code rate of $r = 1/2$ and a constraint length of $c = 5$. Block sizes vary from less than 100 up to 338.
- The RLC blocks in the EDGE modes are encoded with a $r = 1/3$ convolutional code with $c = 7$. Block sizes vary from less than 200 up to 612.

– To reduce the coding overhead in the relatively short header sequences of the EDGE modes, the headers are encoded with *tail-biting codes* [23, 24]. Here, no tail bits indicating the initial and final state of the convolutional encoder are transmitted. Instead, only the fact that the initial and final state of the tail-biting convolutional encoder are identical can be used by the decoder to find the most probable sequence.

In order to maximize the utilization of hardware resources, a single flexible Viterbi decoder solution is favorable over separate channel decoder implementations for the different requirements. Such a flexible Viterbi decoder has to support up to 64 trellis-states ($c = 7$) by taking into consideration up to 3 encoded input data items to compute the branch metrics ($r = 1/3$). Further, standard trellis-termination has to be supported as well as uninitialized trellis processing for the tail-biting codes. Finally, the channel decoder input memory has to be capable of storing up to $3 \cdot 612 = 1836$ data items.

We have implemented a fully-parallel 64-state sliding-window Viterbi decoder which supports the trellis structures required for GSM/GPRS and for EDGE. The data blocks are processed in windows of W=32 with an acquisition length of A=32 to reduce the memory requirements of the decoder (more details on Viterbi decoder design can be found, e.g., in [25]). In transmission modes that use convolutional codes with $c < 7$ only the required hardware resources are activated in order to save power. Although all trellis states that correspond to a bit to be decoded are processed in parallel, the complexity of the Viterbi decoder is comparably small. Since hard-decisions are demodulated in our receiver ASIC, the channel decoder inputs can be represented with only 2 bits which keeps the numeric range of the state metrics small[12]. The branch metric computation can be realized very efficiently when taking into consideration that

– the 2 bit inputs $i \in \{-1, 0, 1\}$
– the (ideal) encoder output corresponding to the trellis state $e \in \{-1, 1\}$
– the resulting difference is $|i - e| \in \{0, 1, 2\}$.

As can be seen in the block diagram of our Viterbi decoder implementation in Fig. 12, calculating the distance between decoder inputs and state-specific encoder outputs can be realized with only two logic gates. Depending on the code rate r, two or three units to compute the distance are enabled in each of the 128 branch metric computation units (BMUs) for the generation of the branch metrics b. The complexity of the state metric recursions can be kept low as well because the metric increase in each trellis stage is limited to

$$\Delta m_{max} = c_{max} \cdot \max |i - e| = 6. \tag{21}$$

Therefore, in our implementation with W=32 the state metrics m can be realized with only 8 bit without the need for normalization circuits. The initialization of the state metrics m_0 is configurable in our implementation to allow for

[12] Hard-decisions in the original sense require only 1 bit. Another bit has to be spent to be able to represent the zero (no a-priori information) for the punctured bits.

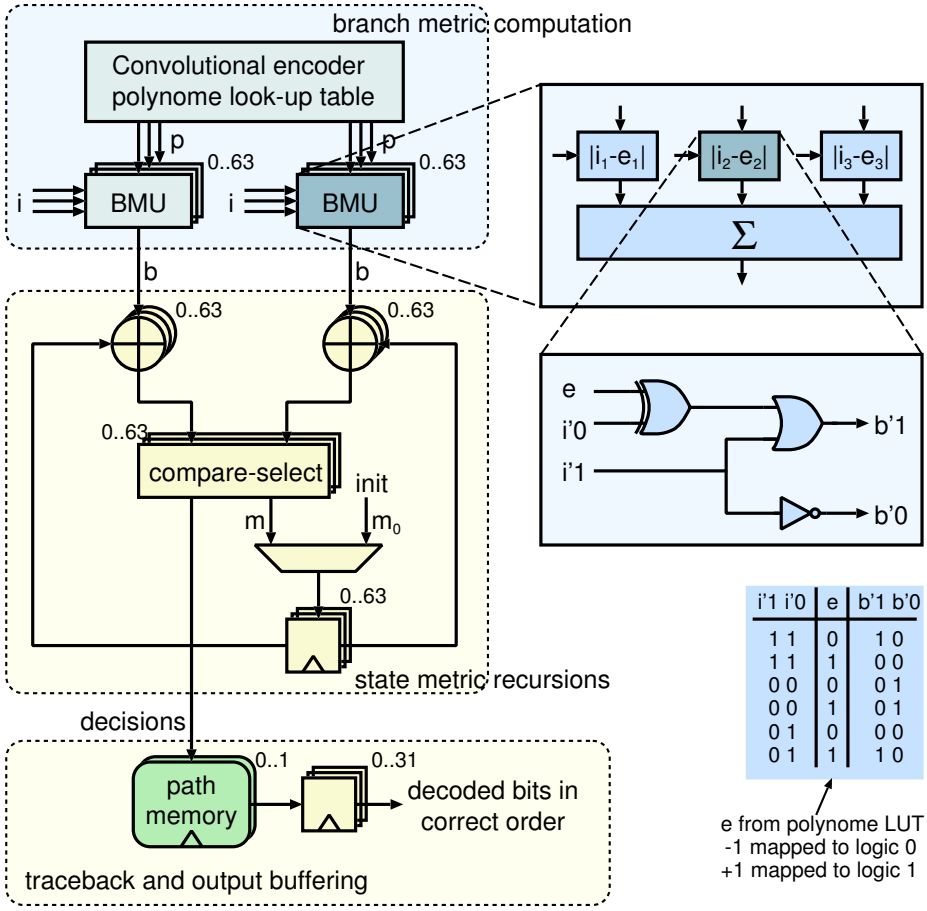

Fig. 12. Block diagram of the flexible 64-state parallel Viterbi decoder implemented in our design.

uninitialized decoder runs in order to find the most probable start- and end-state of the trellis in tail-biting mode. Then, these are used in a second decoder run to decode the (header) bit sequence (details on decoding tail-biting codes can be found, e.g., in [26, 27]).

The trace-back provides the decoded bits in windows of W=32 according to the decisions stored in the path memory in inverse order, such that an output buffer for 32 bits is necessary to bring them into natural order. Our channel decoder implementation requires about 150 cycles to decode the header and 4000 cycles to decode the two RLC blocks in the fastest EDGE transmission mode MCS-9. With a target clock frequency of 40 MHz the decoder latency translates

to only about 100 μs, which allows to compensate for potential delays between header and data block decoding[13].

5.3 Discussion: Hard-Decision vs. Soft-Decision Receiver Back-End

When designing a wireless receiver many decisions have to be taken where performance trades against complexity and cost. Often a specific performance goal (e.g., BER under specific conditions) can be met with different combinations of devices, algorithms and architectures. E.g., choosing a radio-frequency IC with a comparably high noise-figure could be (at least partially) compensated for in the digital baseband receiver by implementing the best-performing channel equalizer and decoder algorithms. To find the optimum implementation for a given application under specific constraints it is crucial to know the performance as well as the complexity increase of the different design options.

Using demodulated soft-decision values can improve the BER after channel decoding by 2-3 dB in terms of SNR (e.g., [28]). However, the generation of soft-decision outputs with a Viterbi equalizer is complex and significantly increases the required memory capacity. For each incoming branch to each trellis state the corresponding sum of the branch and state metric has to be stored. Hence, assuming an 8-state DFSE with a 58-stage trellis using 12 bit state metrics and 8 incoming branches (8PSK) to each trellis state (as implemented in our ASIC, cf., Section 4.3) the soft-output DFSE would require an additional 44.5 kb of memory capacity. To generate reliable soft-decisions several paths have to be traced back for each trellis stage [29] which is difficult to realize in hardware and which significantly increases the required number of memory accesses during back-tracing, leading to higher power consumption.

Storing soft-decisions instead of hard-decisions in the de-interleaver and channel decoder input memory requires a higher storage capacity, and processing soft-inputs in the Viterbi decoder increases the complexity of the BMU and state metric recursion units.

The implemented digital baseband receiver ASIC with hard-decision DFSE outputs contains RAMs to store 20 kb. With typical soft-decisions of 5 bit instead, 76 kb memory would be required which translates to an increase of the core size by a factor of more than 2, and to a chip where more than 2/3 of the silicon area are occupied only by memory.

Even more dramatic are the savings achieved by using hard-decision equalizer outputs when thinking of the memory required for the storage of previously received and not correctly decoded radio blocks for IR. According to the specifications, a multi-slot class 12 user equipment [14] has to store up to 24 radio blocks

[13] The header information has to be extracted and processed first (typically by higher layers) to obtain the coding scheme, puncturing pattern and RLC block identifier. When a re-transmission has occurred the previously received data block will be restored from an IR memory to combine it with the received data block in the channel decoder input memory.

[14] Such a MS is capable of receiving on 4 time slots per frame.

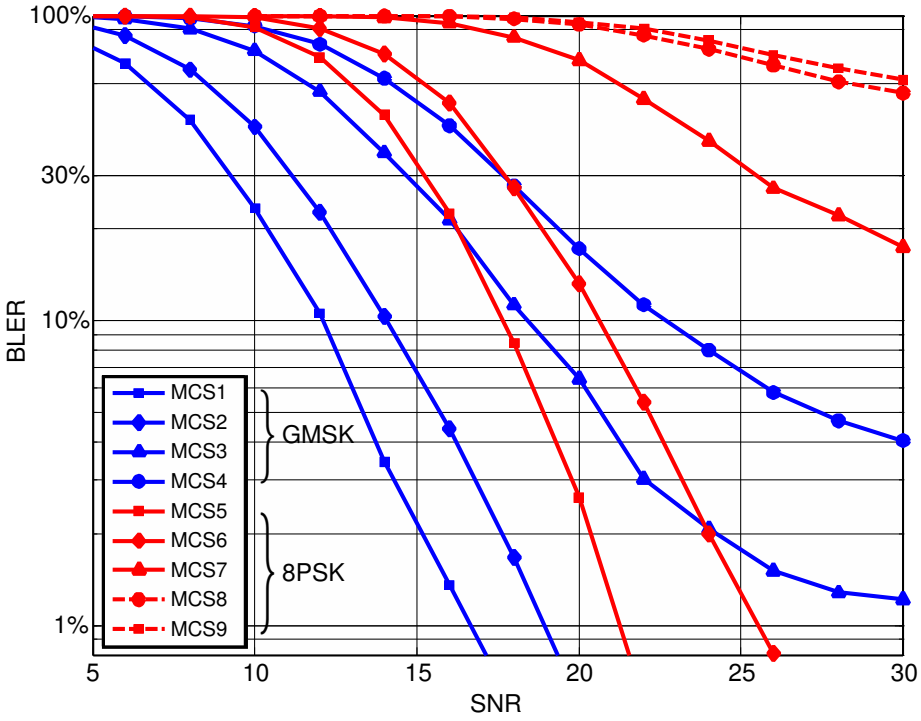

Fig. 13. Receiver block error-rates in various modulation and coding schemes (MCS) of EDGE for the specified Hilly Terrain (HT) test channel with a vehicle speed of 100 km/h.

with the modulation and coding scheme MCS-9, which translates to about 150 kb using 5 bit soft-values. With hard-decisions only the puncturing pattern and the actual bits have to be stored, which saves as much as 120 kb memory.

6 Implementation Results

Functional tests and power measurements of fabricated chips (see Fig. 14) have been performed on a digital tester. The fixed-point simulation model of the implemented design has been verified and used to generate the block error-rates (BLERs) of the receiver as shown in Fig. 13 for the various modulation and coding schemes (MCS) of EDGE with the specified Hilly Terrain (HT) test channel with a vehicle speed of 100 km/h. It should be noted that the standard requires MCS1-6 to achieve a BLER of 10 % and MCS7 to achieve a BLER of 30 % at a certain receive power level. For MCS8-9 there are no specifications for these propagation conditions. Assuming the noise-figure of a state-of-the-

art RF transceiver IC[15] these BLERs can be achieved at the specified power levels. Other key characteristics are summarized in Table 1. The ASIC occupies 1.0 mm^2 in 0.13 μm CMOS, comprising 97 kGE and 20 kb of memory. A measured maximum clock frequency of 172 MHz allows the supply voltage to be lowered to 0.6 V when operating at the target frequency of 40 MHz. During continuous burst reception the average power consumption is as low as 5.2 mW in the fastest EDGE transmission mode MCS9, and only 1.3 mW when lowering the supply voltage to 0.6 V.

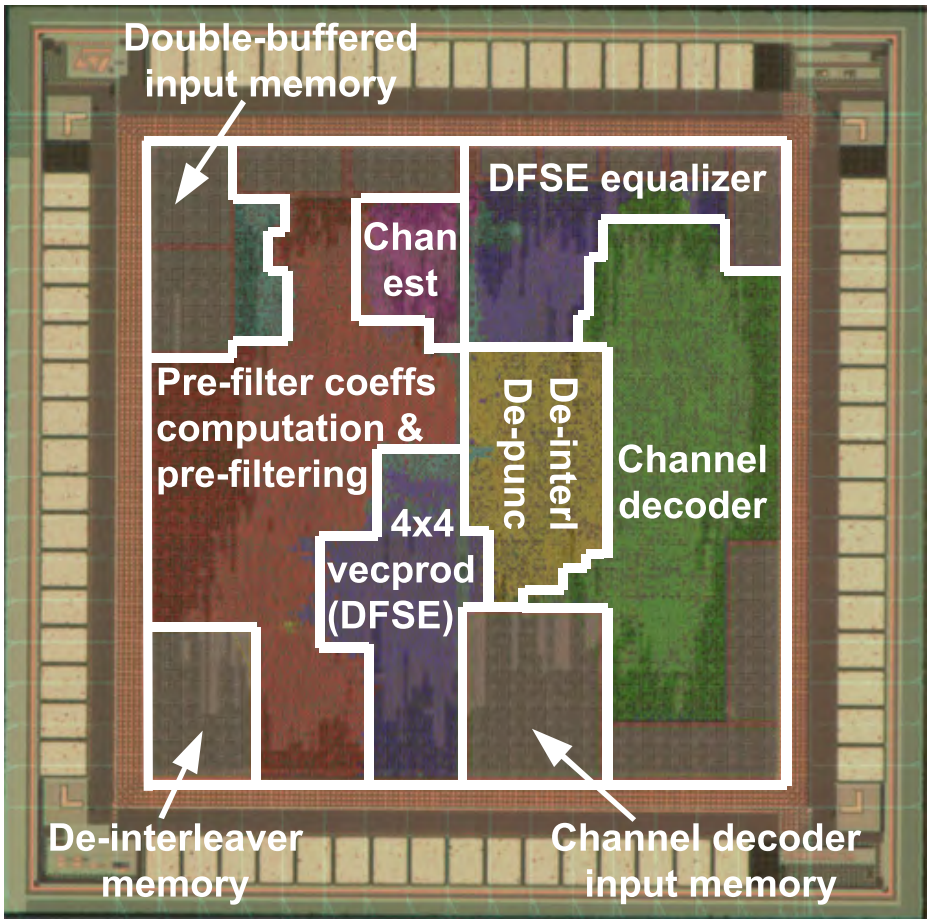

Fig. 14. EDGE receiver ASIC micrograph with highlighted units.

[15] E.g., *IRIS305* from ACP AG, a single-chip RF transceiver supporting both TD-SCDMA/HSPA (3G) and GSM/GPRS/EDGE (2.5G) standards.

Technology	**0.13 μm CMOS**
Supply Voltage V_{DD}	**1.2 V-0.6 V**
Core Size	**1.0 mm^2**
Gate Count	**97 kGE**
Total Memory	**20.3 kb**
Max. System Clock Frequency	**172 MHz**
Leakage current @ V_{DD}=1.2 V (0.6 V)	**0.49 mA (0.15 mA)**
Average power measured during continuous burst reception @ target frequency f=40 MHz, V_{DD}=1.2 V (0.6 V)	
GSM CS1 (GMSK)	**2.4 mW (0.6 mW)**
EDGE MCS9 (8PSK)	**5.2 mW (1.3 mW)**

Table 1. Key characteristics of the receiver ASIC implementation.

7 Conclusion

EDGE enables 2G data services all over the world with data rates suitable for many cellular applications. The introduction of 8PSK modulation improves the bandwidth utilization, but significantly increases equalizer complexity. Latest 65 nm signal processors provide the computational power required to perform the digital baseband signal processing for EDGE. However, power and silicon area of such implementations exceed what is desirable for a low-power, low-cost 2.5G solution required in modern cellular phones.

In this work a combined architecture for GSM/GPRS/EDGE has been presented. We have shown how the digital baseband can be mapped efficiently to dedicated hardware by using suitable algorithms and architectures to share hardware resources with little overhead. A certain amount of flexibility can support the specified transmission modes with different modulation and coding schemes. Our approach to reduce complexity in the most crucial block, the channel equalizer, is to maximize the effort in the pre-processing FIR filtering, in order to be able to reduce DFSE complexity. Using hard-decisions at the equalizer output is the key to curtail memory requirements in the back-end of a digital baseband receiver for EDGE.

With 1.0 mm^2 and only 1.3 mW average power consumption our ASIC implementation shows that cost-effective multi-mode digital baseband receiver accelerators for GSM/EDGE can be realized at ultra low-power.

Acknowledgments

The authors would like to thank Andreas Burg, Andreas Bubenhofer, Johannes Widmer, Stefan Zwicky, Harald Kröll, and Christoph Roth for their support. This project was funded by CTI (No. 11370.1), Switzerland in collaboration with Advanced Circuit Pursuit AG.

References

1. "GSM Association, Market Data, www.gsmworld.com, August 2009."
2. M. Eyuboglu and S. Qureshi, "Reduced-state sequence estimation with set partitioning and decision feedback," *IEEE Transactions on Communications*, vol. 36, pp. 13–20, Jan. 1988.
3. A. Duel-Hallen and C. Heegard, "Delayed decision-feedback sequence estimation," *IEEE Transactions on Communications*, vol. 37, pp. 428–436, May 1989.
4. J. Olivier *et al.*, "Efficient equalization and symbol detection for 8-PSK EDGE cellular system," *IEEE Transactions on Vehicular Technology*, vol. 52, pp. 525–529, May 2003.
5. W. Gerstacker and R. Schober, "Equalization concepts for EDGE," *IEEE Transactions on Wireless Communications*, vol. 1, pp. 190–199, Jan. 2002.
6. L. Szczecinski and I. Soto, "Is turbo equalization useful in EDGE systems?," *Vehicular Technology Conference*, vol. 1, pp. 80–84, May 2002.
7. C. Laot, R. Le Bidan, and D. Leroux, "Low-complexity MMSE turbo equalization: a possible solution for EDGE," *IEEE Transactions on Wireless Communications*, vol. 4, pp. 965–974, May 2005.
8. 3GPP Organizational Partners, Sophia Antipolis Valbonne, France, *3rd Generation Partnership Project; Technical Specification Group GSM/EDGE Radio Access Network; Channel Coding*, June 2009.
9. 3GPP Organizational Partners, Sophia Antipolis Valbonne, France, *3rd Generation Partnership Project; Technical Specification Group GSM/EDGE Radio Access Network; Modulation*, Dec. 2008.
10. 3GPP Organizational Partners, Sophia Antipolis Valbonne, France, *3rd Generation Partnership Project; Technical Specification Group GSM/EDGE Radio Access Network; Radio transmission and reception*, June 2009.
11. J. Proakis, *Digital Communications, 4th Edition*. New York, NY, USA: McGraw-Hill Higher Education, 2001.
12. M. Schmidt and G. Fettweis, "Fractionally-spaced prefiltering for reduced state equalization," in *Global Telecommunications Conference, 1999. GLOBECOM '99*, vol. 5, pp. 2291 – 2295, 1999.
13. J. O. III Smith, *Introduction to Digital Filters: with Audio Applications*. W3K Publishing, 2004.
14. W. Gerstacker *et al.*, "On prefilter computation for reduced-state equalization," *IEEE Transactions on Wireless Communications*, vol. 1, Oct. 2002.
15. A. V. Oppenheim and R. W. Schafer, *Digital Signal Processing, 1st edition*. Englewood Cliffs, NJ: Prentice-Hall, Inc., 1989.
16. T. Detert, "Prefiltered low complexity tree detection for frequency selective fading channels," in *Vehicular Technology Conference, 2007. VTC2007-Spring. IEEE 65th*, pp. 2364 –2368, Apr. 2007.

17. W. C. Chu, *Speech coding algorithms : foundation and evolution of standardized coders*. Hoboken, NJ: John Wiley & Sons, Inc., 2003.
18. J. Makhoul, "Linear prediction: A tutorial review," in *Proceedings of the IEEE*, pp. 561 – 580, Apr. 1975.
19. G. Golub and C. Van Loan, *Matrix computations*. Johns Hopkins University Press, 1996.
20. C. Benkeser, *Power Efficiency and the Mapping of Communication Algorithms into VLSI*. PhD thesis, ETH Zürich, Switzerland, Series in Microelectronics, vol. 209, Hartung-Gorre Verlag Konstanz, 2010.
21. E. Seurre, P. Savelli, and P.-J. Pietri, *EDGE for Mobile Internet*. Norwood, MA: Artech House, 2003.
22. A. Viterbi, "Error bounds for convolutional codes and an asymptotically optimum decoding algorithm," *IEEE Transactions on Information Theory*, vol. 13, pp. 260–269, Apr. 1967.
23. H. Ma and J. Wolf, "On tail biting convolutional codes," *Communications, IEEE Transactions on*, vol. 34, pp. 104 – 111, feb 1986.
24. E. Biglieri, *Coding for wireless channels*. New York, NY: Springer, 2005.
25. T. Truong, M.-T. Shih, I. Reed, and E. Satorius, "A VLSI design for a trace-back Viterbi decoder," *Communications, IEEE Transactions on*, vol. 40, pp. 616 –624, mar 1992.
26. R. Cox and C. Sundberg, "An efficient adaptive circular Viterbi algorithm for decoding generalized tailbiting convolutional codes," *Vehicular Technology, IEEE Transactions on*, vol. 43, pp. 57 –68, feb 1994.
27. J. Anderson and S. Hladik, "An optimal circular Viterbi decoder for the bounded distance criterion," *Communications, IEEE Transactions on*, vol. 50, pp. 1736 – 1742, nov 2002.
28. G. Clark and J. Cain, *Error-correction coding for digital communications*. New York; London: Plenum Press, 1981.
29. J. Hagenauer and P. Hoeher, "A Viterbi algorithm with soft-decision outputs and its applications," in *Global Telecommunications Conference, 1989, and Exhibition. Communications Technology for the 1990s and Beyond. GLOBECOM '89., IEEE*, pp. 1680 –1686 vol.3, nov 1989.