EFFICIENT VLSI IMPLEMENTATION OF REDUCED-STATE SEQUENCE ESTIMATION FOR WIRELESS COMMUNICATIONS

Stefan Zwicky[⋆] *Christian Benkeser*[⋆] *Andreas Burg*[†] *Qiuting Huang*[⋆]

[⋆] Integrated Systems Laboratory, ETH Zurich, 8092 Zurich, Switzerland
e-mail: {zwicky,benkeser,huang}@iis.ee.ethz.ch
[†] Telecommunications Circuits Laboratory, EPFL Lausanne, 1015 Lausanne, Switzerland
e-mail: andreas.burg@epfl.ch

ABSTRACT

Modern wireless communication systems require efficient channel equalizer implementations. This paper explores the design space of reduced-state sequence estimation (RSSE). We show how the concept of pre-computation can be applied to greatly reduce computational complexity, such that efficient RSSE architectures can be derived. As a proof of concept, an RSSE was implemented in dedicated hardware, that achieves a 1.6 times higher hardware efficiency when compared to prior art.

Index Terms— Channel equalization, RSSE, Design space exploration, Evolved EDGE, VLSI implementation

1. INTRODUCTION

Inter-symbol interference (ISI) is a challenging problem in many wireless communication systems. The strong impact of ISI due to multipath propagation or narrow bandwidth (e.g., 2G-GSM bandwidth is narrower than corresponding symbol rate) can cause severe signal distortion, which requires sophisticated equalization approaches. Maximum likelihood sequence estimation (MLSE), based on the Viterbi algorithm, is the optimum solution for channel equalization and signal detection. Unfortunately, the computational complexity of MLSE grows exponentially with the number of bits per symbol and with the delay spread of the multipath channel. The combination of a high modulation order and long channels renders MLSE infeasible in modern wireless systems.

A sub-optimal Viterbi equalizer that can achieve close-to-MLSE performance[1] at significantly lower complexity is reduced-state sequence estimation (RSSE) [2, 3]. Contrary to MLSE where *reference signals* that correspond to all possible combinations of modulated symbol sequences with channel length L are compared with the received signal, RSSE drastically reduces the number of candidate symbol sequences by applying symbol partitioning and decision-feedback with early decisions (cf. Sec. 2). Since the generation of the reference signals is the most complex part of MLSE, the reduction of candidate symbol sequences with RSSE directly translates to a reduction of corresponding implementation complexity. Unfortunately, even RSSE complexity is still huge when supporting high modulation orders. Hence, dedicated hardware is required to achieve high throughput or to achieve hardware and energy efficient solutions, as desired for mobile devices.

[1]We assume that the channel has been shortened with a channel-shortening pre-filter (e.g., [1]).

So far, VLSI implementations for RSSE have only been published for the specific application of Ethernet with trellis-coded modulation [4–6], where only costly fully-parallel architectures for very high throughput have been considered. A VLSI implementation of a delayed-decision feedback sequence estimator (DDFSE), a special case of RSSE, for 2.5G EDGE and for 2.75G Evolved EDGE (E-EDGE) [7] has been published in [8] and [9], respectively. However, in literature there is a lack of RSSE design space exploration for systems with moderate throughput requirements using higher order modulation. Especially, the impact of algorithm and architectural choices on implementation complexity can be crucial in systems that rely on efficient VLSI implementations, such as modern wireless mobile receivers.

In this paper, we explore the design space for non-fully parallel Viterbi equalizers. We apply the pre-computation approach presented in [9] for DDFSE to the more general RSSE. The presented design space exploration allows for efficient VLSI architectures tailored to given design constraints. As a proof of concept, an efficient VLSI implementation of RSSE for E-EDGE is presented and measurement results of the design are compared to prior art.

2. SYSTEM MODEL AND ALGORITHM

Throughout this paper, the following baseband transmission model is considered:

$$r_k = \sum_{m=0}^{L-1} s_{k-m} h_m + n_k \,. \tag{1}$$

A block of modulated symbols s_k is impaired by a multipath channel with impulse response h_k of length L. The symbols are further disturbed by complex additive white Gaussian noise n_k, resulting in complex-valued samples at symbol rate, denoted as r_k. It is assumed that an estimate $\hat{h}_k$ of the channel impulse response (CIR) is available at the receiver.

With MLSE, based on the r_k, the most probable transmitted sequence is efficiently found with the Viterbi algorithm in the logarithmic domain by searching the best path through a trellis. The states of this trellis correspond to all possible combinations of the $L-1$ modulated symbols in the channel memory, defined as the vector $\bar{\mathbf{s}}_t$.

Whenever different paths merge into a state $\bar{\mathbf{s}}_t$, the best path is selected based on a path metric which is the sum of the branch metric of the incoming branch $\Gamma(\bar{\mathbf{s}}_{t-1}, \bar{\mathbf{s}}_t)$ and the state metric of the predecessor state $A(\bar{\mathbf{s}}_{t-1})$. The branch metric is defined as

$$\Gamma(\bar{\mathbf{s}}_{t-1}, \bar{\mathbf{s}}_t) = \Big(\sum_{k=0}^{L-1} \bar{s}_{t-k}\hat{h}_k - r_t\Big)^2 \,, \tag{2}$$

with the symbols $\bar{s}_{t-(L-1)} \dots \bar{s}_{t-1}$ defined by state $\bar{\mathbf{s}}_{t-1}$, and symbol $\bar{s}_t$ defined by the branch between states $\bar{\mathbf{s}}_{t-1}$ and $\bar{\mathbf{s}}_t$. The new state metric is given as the minimal metric of all paths merging into the state $\bar{\mathbf{s}}_t$

$$A(\bar{\mathbf{s}}_t) = \min_{\bar{\mathbf{s}}_{t-1} \in \chi(\bar{\mathbf{s}}_t)} \left[A(\bar{\mathbf{s}}_{t-1}) + \Gamma(\bar{\mathbf{s}}_{t-1}, \bar{\mathbf{s}}_t) \right], \tag{3}$$

where $\chi(\bar{\mathbf{s}}_t)$ is the set of all predecessor states of $\bar{\mathbf{s}}_t$. The branch associated with the smallest metric is denoted as the *winning* branch and once the end of the trellis is reached, the symbols corresponding to the ML solution are found by tracing back through the trellis along the winning branches. The number of trellis states is given by

$$N_{\mathrm{MLSE}} = 2^{Q(L-1)} = M^{L-1}, \tag{4}$$

where Q is the number of bits per symbol and $M = 2^Q$ is the size of the symbol alphabet (or modulation order). The computational complexity of MLSE is proportional to the number of trellis states, which grows exponentially in Q and L. Therefore, it becomes prohibitively large for a typical channel length of $L = 8$ and modulation order $M > 2$, as used e.g., in E-EDGE.

2.1. Reduced-State Sequence Estimation

Compared to MLSE, the RSSE algorithm reduces the number of trellis states by dividing the symbol alphabet into subsets and defining the trellis on these subsets. The division of the M symbols into J subsets is optimally chosen such that the Euclidean distance of symbols within the same subset is maximal (e.g., through Ungerboeck partitioning [10]). Whenever a surviving path is selected, a decision within the subset is done immediately, but the selection among subsets is postponed. Each state keeps a list of surviving symbols such that the branch metric of (2) can be calculated. The trellis state is no longer defined by the past $L-1$ symbols but by the subsets corresponding to these symbols. Thus, the number of trellis states is reduced to

$$N_{\mathrm{RSSE}} = \prod_{k=1}^{L-1} J_k, \tag{5}$$

where J_k is the number of subsets considered for the symbol k time steps in the past. To have a well-defined trellis, the numbers of subsets J_k must be non-increasing (i.e., $J_1 \geq J_2 \geq J_3$ etc.) and a decrease in J_k must be achieved by merging subsets. Restricting J_k to powers of two makes the algorithm more suitable for implementation. Choosing $J_1 < M$ leads to $P = \frac{M}{J_1}$ parallel branches leading from one state to the next as depicted in the example in Fig. 1 for $M = 4$, $J_1 = 2$ and $J_2 = 2$.

DDFSE corresponds to the special case where $J_k = M$ for $k \leq D$ and $J_k = 1$ for $k > D$. Choosing $D = 0$ results in a decision feedback equalizer and $D = L - 1$ in the MLSE. Especially in communication systems with large M, DDFSE implementations are usually only feasible with $D = 1$, because the complexity with $D = 2$ is already huge. With RSSE instead, the J_k can be chosen more freely, to allow for a much more fine-grained adjustment of the performance-complexity trade-off. In the following section we will describe the design space and efficient RSSE realizations, given a certain configuration of the J_k, in order to find most suitable hardware architectures for specific design constraints.

3. DESIGN SPACE EXPLORATION

In contrast to Viterbi decoding, the branch metric computation (2) is the most computationally intensive part of a Viterbi equalizer [11].

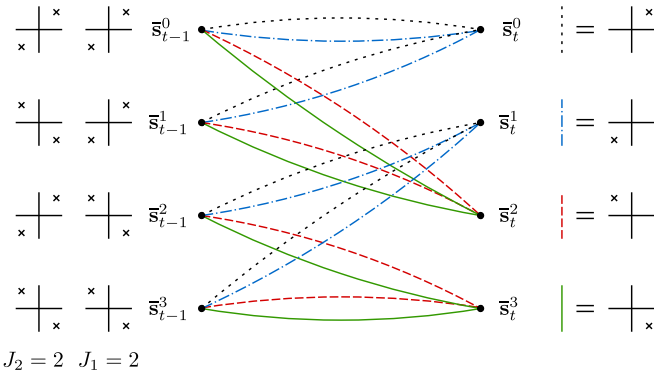

Fig. 1. Example of four-state RSSE trellis for QPSK ($M = 4$) with $J_1 = J_2 = 2$, and $P = 2$ parallel branches. The four different states of trellis stage t are labeled with $\bar{\mathbf{s}}^0_t$ through $\bar{\mathbf{s}}^3_t$.

Especially the computation of the reference signals requires many expensive complex-valued multiplications. These can be split into two parts according to

$$e(\bar{\mathbf{s}}_{t-1}, \bar{\mathbf{s}}_t) = \overbrace{\sum_{k=1}^{L-1} \bar{s}_{t-k} \hat{h}_k}^{e_s} + \overbrace{\bar{s}_t \hat{h}_0}^{e_b}. \tag{6}$$

The *partial reference signals* e_s and e_b depend only on the symbols associated with the state $\bar{\mathbf{s}}_{t-1}$ and the branch corresponding to $\bar{s}_t$, respectively. Although each of the NM different $e(\bar{\mathbf{s}}_{t-1}, \bar{\mathbf{s}}_t)$ is unique, they all can be computed by adding one of the N state-dependent e_s and one of the M branch-dependent e_b.

In fully-parallel implementations, all reference signals have to be computed concurrently. When the completion of a trellis stage takes several cycles, however, more efficient computation schemes can be derived by applying pre-computation and storing the different incarnations of e_b or e_s. The stored partial reference signals are simply combined by addition to generate the final $e(\bar{\mathbf{s}}_{t-1}, \bar{\mathbf{s}}_t)$ according to (6). It has been shown in [9] for the case of a DDFSE that the number of multiplications can be drastically reduced by pre-computation of e_s. In the following we discuss the trade-off between computational and storage complexity for the more general case of RSSE and include also the possibility of pre-computing e_b.

3.1. Trade-Off between Computational and Storage Complexity

In the following analysis we assume that the state metric update (3) of a state $\bar{\mathbf{s}}_t$ is completed before the metric of the next state is updated. This requires that the branch metrics of all branches arriving at the current state are computed by combining the corresponding e_s and e_b. Every state metric update requires a combination of different e_s and e_b, but the same e_s and e_b are used several times. Hence, whenever the partial reference signals are reused, they must be either recomputed or loaded from a memory.

When no pre-computation is applied, both partial reference signals must be calculated for all branches of a trellis stage, resulting in

$$C_{\mathrm{full}} = NML \tag{7}$$

complex-valued multiplications per trellis stage. The number C_{full} is composed of $NM(L-1)$ multiplications for the computation of all e_s and NM multiplications for the computation of all e_b.

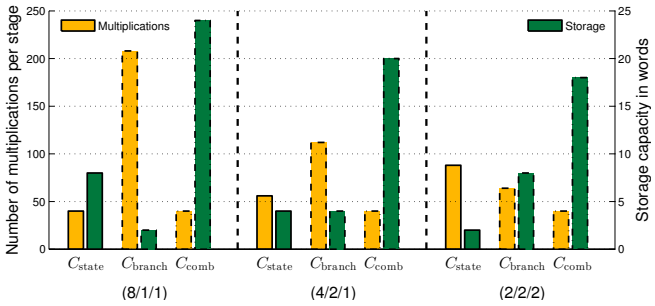

Fig. 2. Computational complexity vs. storage requirements for $M = 16$, $N = 8$ and $L = 4$ for different values of $(J_1/J_2/J_3)$. The number of multiplications without pre-computation is $C_{\text{full}} = 512$.

Applying a pre-computation strategy for the e_s requires a memory capacity of J_1 words to store the e_s corresponding to all predecessor states of a state $\bar{\mathbf{s}}_t$. In this way, every e_s needs to be calculated only once. In the example of Fig. 1 ($J_1 = 2$), pre-computation of e_s for the two states $\bar{\mathbf{s}}_{t-1}^0$ and $\bar{\mathbf{s}}_{t-1}^1$ enables the state metric update for the states $\bar{\mathbf{s}}_t^0$ and $\bar{\mathbf{s}}_t^2$. After that, the pre-computed e_s are no longer required and can be overwritten. This reduces the number of multiplications for the computation of e_s from $NM(L-1)$ to $N(L-1)$. Having the e_s of all predecessor states stored has the advantage that an e_b of a given symbol can be used for all branches that connect to the different predecessor states, before a new e_b is calculated. This reduces the number of multiplications for e_b from NM to NP. The total number of multiplications for the case of J_1 pre-computed e_s is thus given by

$$C_{\text{state}} = N(L-1) + NP\,. \tag{8}$$

Conversely, the e_b can be pre-computed such that all e_b need to be computed only once. This reduces the number of multiplications for e_b from MN to M. To achieve this, all P parallel branches must be pre-computed and stored. The availability of the e_b for all parallel branches has the advantage that the same e_s can be used for all P parallel branches before a new e_s is computed, reducing the number of multiplications for e_s from $NM(L-1)$ to $NJ_1(L-1)$. The total number of multiplications per stage can be reduced to

$$C_{\text{branch}} = NJ_1(L-1) + M \tag{9}$$

by providing a storage capacity for P words.

When the two pre-computation strategies are combined, all e_s and e_b must be calculated only once. To this end, the e_b of all possible M symbols must be pre-computed and stored in memory. Furthermore, the e_s of J_1 states must be available in memory. By providing a storage capacity of $J_1 + M$ words, the number of multiplications can be reduced to

$$C_{\text{comb}} = N(L-1) + M\,. \tag{10}$$

The trade-off between storage and computational complexity for the four strategies discussed above is visualized in Fig. 2 for an example trellis with $N = 8$ states, $M = 16$, and a CIR of length $L = 4$. This setup allows for three different RSSE configurations, i.e., $(8/1/1)$, $(4/2/1)$, and $(2/2/2)$ for $(J_1/J_2/J_3)$. While the selection of the configuration is subject to algorithmic evaluations, the most suitable hardware architecture for each configuration can be found by considering the options of Fig. 2. The figure illustrates that

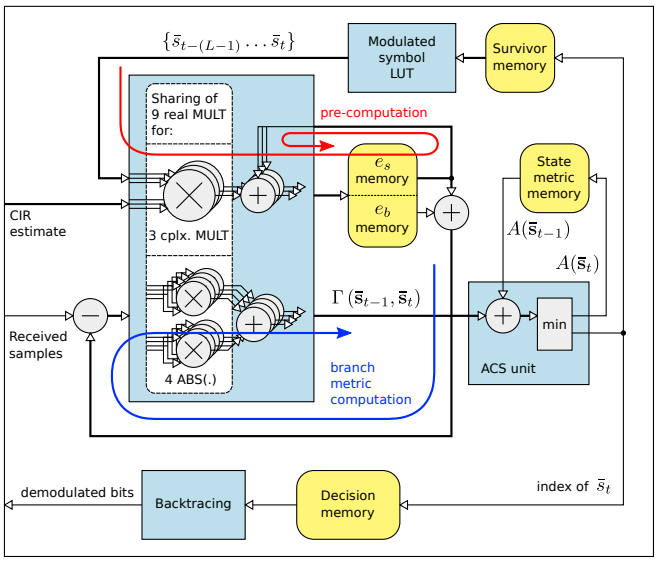

Fig. 3. Block diagram of RSSE hardware implementation.

with our presented pre-computation approach the number of multiplications per trellis stage can be greatly reduced from 512 (C_{full}) down to 40 (C_{comb}). Since storage and computational complexity are two independent optimization criteria, there are many Pareto optimal solutions. However, a visualization of the trade-off helps to find the most suitable solution for a given design goal.

4. RSSE IMPLEMENTATION FOR EVOLVED EDGE

In order to show the suitability of our design-space exploration for VLSI implementation, we have implemented an RSSE solution for a 2.75G E-EDGE baseband transceiver in dedicated hardware. To this end, the implemented RSSE supports GMSK, 8PSK, 16QAM and 32QAM modulation, and processes 2 trellises of 58 symbols per GSM burst (cf., [12]).) For 16QAM and 32QAM, the RSSE is configured $(4/2/2)$ with 16 trellis states, and for 8PSK and GMSK modulation the RSSE is configured $(2/2/2)$ with 8 states. Simulations have shown that these RSSE configurations provide close-to-MLSE performance for the different modulation types when applying a channel shortening filter [1] in front of the RSSE.

As storage requirements for a 16-state trellis are fairly low, the combined pre-computation approach was chosen and the e_s of all 16 predecessor states and the e_b of all 32 possible symbols are pre-computed (cf. Sec. 3). Thus, the number of multiplications per trellis stage has been greatly reduced from 4096 (no pre-computation) to 144 at the cost of a storage capacity of 48 pre-computed words.

4.1. VLSI Architecture

A block diagram of the corresponding RSSE architecture is depicted in Fig. 3. Inputs to the block are the estimated CIR and the received samples at symbol rate. In a first step, all e_s and e_b are pre-computed. Then, corresponding e_s and e_b are summed up to reference signals which are subtracted from the received signal r_k. The branch metric computation (2) is completed by calculating the square of the absolute value. For all calculated branch metrics $\Gamma(\bar{\mathbf{s}}_{t-1}, \bar{\mathbf{s}}_t)$, the ACS unit adds the state metric $A(\bar{\mathbf{s}}_{t-1})$ of the corresponding predecessor state to get the path metric and determines

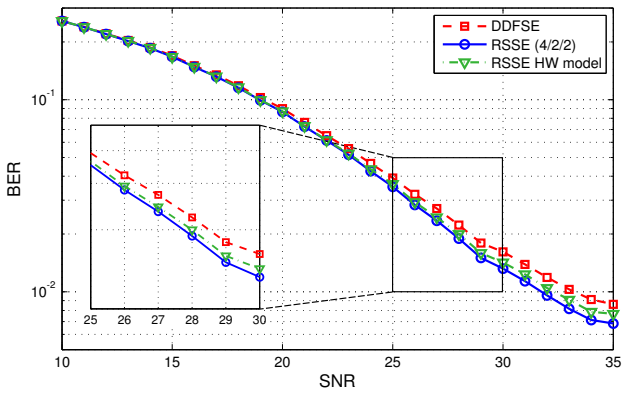

Fig. 4. BER performance of 32QAM under Hilly Terrain (HT) channel conditions [7] for DDFSE, RSSE with $J_1 = 4$, $J_2 = J_2 = 2$ and corresponding hardware model.

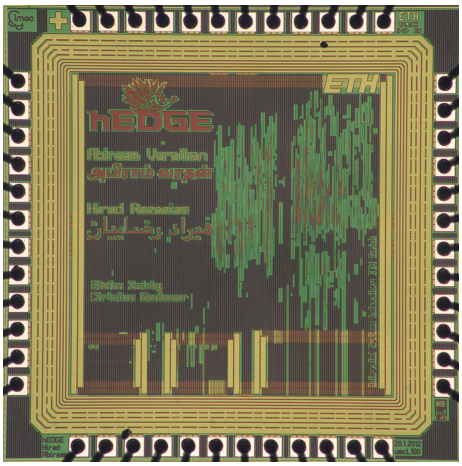

Fig. 5. Chip photograph of 16-state RSSE.

the index of the winning symbol $\bar{s}_t$ as well as the new state metric $A(\bar{\mathbf{s}}_t)$ which is then stored in the state metric memory. The index of the winning symbol is stored in both the decision memory for the final back-tracing process and in the survivor memory where the $L-1$ survivor symbols for each state are stored. Based on the survivor symbols, the modulated symbols for the pre-computation of the next trellis stage are derived by accessing a look-up table (LUT) with the indices corresponding to the survivor symbols.

In order to optimize the area utilization of our design, the multipliers for the pre-computation are reused for the calculation of Euclidean distances. To achieve the required throughput at our target clock frequency of 52 MHz, 9 real-valued multipliers are employed which can be either used for 3 complex-valued multiplications[2] or for 4 squaring operations.

Several storage elements in our architecture require parallel access to four words in the same clock cycle. Thus, they are realized most efficiently with flip-flop arrays. The large decision memory as well as the two memories used for the storage of survivor symbols are realized with single-port RAMs, in order to save silicon area.

The design has been manufactured in a 0.18μm CMOS technology and a chip photograph is depicted in Fig. 5. It has been measured on a digital tester to run at a clock frequency of 124 MHz. The clock frequency required to achieve the target throughput is more than two times lower. This cycle-time headroom provides the possibility to scale down the supply voltage for power savings.

Table 1 compares our design with the only previously published 2.75G detector [8]. The performance in terms of bit-error rate (BER) of the implemented algorithms is comparable, as can be seen in Fig. 4, and the implementation loss of our design is within 0.2 dB. It has already been shown in [13] for EDGE with 8PSK, that RSSE requires less states than DDFSE to achieve the same BER performance. When employing a hardware efficiency measure defined by the area in gate equivalents (GE) divided by the maximal throughput, our solution shows an efficiency gain of 1.6 when compared to the other E-EDGE solution based on DDFSE.

[2]With Gauss' complex multiplication algorithm, the number of real-valued multiplications can be reduced to 3.

Table 1. Published 2.75G Viterbi equalizer implementations

Publication	This work	[8]
Algorithm	RSSE	DDFSE
Number of trellis states	16	32
Technology	UMC 180 nm	ST 130 nm
Gate count[a] [kGE]	61.3	94.7
Memory [kb]	6.2	11
f_{max} [MHz]	124	151 (109[b])
f_{target} [MHz]	52	40
Maximal Θ [Mbit/s]	3.3	4.4 (3.2[b])
HW efficiency [kGE/(Mbit/s)]	18.6	21.5 (29.6[b])

[a]Including memories.
[b]Scaling to 180 nm [14]: $t_{pd} \sim 1/l$

5. CONCLUSIONS

Modern wireless communication receivers require high-performance channel equalizer solutions that meet the (moderate) throughput requirements at reasonably low complexity. RSSE is a suitable candidate that allows for a multitude of algorithmic and architectural realizations. To this end, the concept of pre-computation enables a significant reduction of computational complexity at the cost of a small amount of storage capacity in RSSE implementations. The efficiency of RSSE architectures with pre-computation has been shown with our measured ASIC implementation of a 2.75G channel equalizer, that achieves a 1.6 times higher hardware efficiency, when compared to a corresponding DDFSE design with similar performance.

6. ACKNOWLEDGMENT

This work was funded by CTI, Switzerland (project no 11370.1 PFNM-NM) in collaboration with Advanced Circuit Pursuit (ACP) AG. In addition we want to thank Abiraam Varathan and Hirad Rezaeian for their valuable work on the hardware implementation as well as Beat Muheim and Frank Gürkaynak for their support during the back-end design of the ASIC.

7. REFERENCES

[1] C. Benkeser, S. Zwicky, H. Kröll, J. Widmer, and Qiuting Huang, "Efficient channel shortening for higher order modulation: Algorithm and architecture," in *Circuits and Systems (ISCAS), IEEE International Symposium on*, May 2012, pp. 2377–2380.

[2] M.V. Eyuboglu and S.U.H. Qureshi, "Reduced-state sequence estimation for coded modulation of intersymbol interference channels," *Selected Areas in Communications, IEEE Journal on*, vol. 7, no. 6, pp. 989–995, Aug 1989.

[3] A. Duel-Hallen and C. Heegard, "Delayed decision-feedback sequence estimation," *Communications, IEEE Transactions on*, vol. 37, no. 5, pp. 428–436, May 1989.

[4] E.F. Haratsch and K. Azadet, "High-speed reduced-state sequence estimation," in *IEEE International Symposium on Circuits and Systems*, May 2000, vol. 3, pp. 387–390.

[5] E.F. Haratsch and K. Azadet, "A 1-Gb/s joint equalizer and trellis decoder for 1000BASE-T Gigabit Ethernet," *Solid-State Circuits, IEEE Journal of*, vol. 36, no. 3, pp. 374–384, March 2001.

[6] E.F. Haratsch and Z.A. Keirn, "Digital signal processing in read channels," in *Custom Integrated Circuits Conference, Proceedings of the IEEE*, Sept 2005, pp. 683–690.

[7] "3GPP; tech. spec. group GSM/EDGE radio access network; radio transmission and reception," Nov 2009.

[8] C. Benkeser, A. Bubenhofer, and Qiuting Huang, "A 4.5mW digital baseband receiver for level-A Evolved EDGE," in *IEEE International Solid-State Circuits Conference (ISSCC)*, Feb 2010, pp. 276–277.

[9] C. Benkeser and Q. Huang, "Design and optimization of a digital baseband receiver ASIC for GSM/EDGE," in *VLSI-SoC: Forward-Looking Trends in IC and Systems Design*, vol. 373 of *IFIP Advances in Information and Communication Technology*, pp. 100–127. Springer Boston, 2012.

[10] G. Ungerboeck, "Channel coding with multilevel/phase signals," *Information Theory, IEEE Transactions on*, vol. 28, no. 1, pp. 55–67, Jan 1982.

[11] C. Benkeser, *Power Efficiency and the Mapping of Communication Algorithms into VLSI*, Ph.D. thesis, ETH Zürich, Switzerland, Series in Microelectronics, vol. 209, Hartung-Gorre Verlag Konstanz, 2010.

[12] "3GPP; tech. spec. group GSM/EDGE radio access network; physical layer on the radio path; general description," Nov 2009.

[13] W.H. Gerstacker and R. Schober, "Equalization concepts for edge," *Wireless Communications, IEEE Transactions on*, vol. 1, no. 1, pp. 190–199, Jan 2002.

[14] B. Razavi, *Design of analog CMOS integrated circuits*, McGraw-Hill, 2002.