Optimization and implementation of a Viterbi decoder under flexibility constraints

Kamuf, Matthias; Öwall, Viktor; Anderson, John B

Published in:
IEEE Transactions on Circuits and Systems Part 1: Regular Papers

DOI:
10.1109/TCSI.2008.918148

2008

Optimization and Implementation of a Viterbi Decoder Under Flexibility Constraints

Matthias Kamuf, *Member, IEEE*, Viktor Öwall, *Member, IEEE*, and John B. Anderson, *Fellow, IEEE*

***Abstract*—This paper discusses the impact of flexibility when designing a Viterbi decoder for both convolutional and TCM codes. Different trade-offs have to be considered in choosing the right architecture for the processing blocks and the resulting hardware penalty is evaluated. We study the impact of symbol quantization that degrades performance and affects the wordlength of the rate-flexible trellis datapath. A radix-2-based architecture for this datapath relaxes the hardware requirements on the branch metric and survivor path blocks substantially. The cost of flexibility in terms of cell area and power consumption is explored by an investigation of synthesized designs that provide different transmission rates. Two designs are fabricated in a digital 0.13-μm CMOS process. Based on post-layout simulations, a symbol baud rate of 168 Mbaud/s is achieved in TCM mode, equivalent to a maximum throughput of 840 Mbit/s using a 64-QAM constellation.**

***Index Terms*—Convolutional codes, flexibility, quantization, subset decoding, Trellis-coded modulation (TCM), Viterbi decoding, VLSI, wireless personal area network (WPAN).**

I. Introduction

With growing application diversity in mobile communications, the need for flexible processing hardware has increased. As an example, consider application-diverse high-rate wireless personal area networks (WPANs) [1], which provide short-range ad hoc connectivity for mobile consumer electronics and communication devices. In this environment, different transmission schemes and code rates are required in order to adjust to varying channel conditions [2], [3]. Thus, a flexible channel decoding platform should be able to handle at least two decoding modes, one when a high error-correcting capability is required at low E_b/N_0, and another one supporting high data throughput if the channel is good. How flexibility constrains the design process and its overall cost in terms of hardware are issues that are often overlooked. A design goal could be stated as follows: increase flexibility with as little sacrifice as possible in area, throughput, and power consumption.

Trellis-coded modulation (TCM) [4], [5] is considered as a means of transmitting coded information at high data rates. It is most efficient for higher (quadrature) constellations beyond quadrature phase-shift keying (QPSK), which carry more than two information symbols per 2-D channel use. The subset selectors of the TCM codes in [1] are rate 1/2 for QPSK and rate 2/3 for 16-QAM, 32-CR, and 64-QAM constellations. In Fig. 1, we plot their bit error rate (BER) in the additive white Gaussian noise (AWGN) channel since these curves are not generally available in literature. For comparison, the Shannon limit for equivalent rate R (bits/dimension) systems is shown in this figure. For example, since there is one coded bit per constellation point, the rate for 16-QAM TCM transmission is 3 bits per 2 dimensions, equivalent to $R = 1.5\ \mathrm{bits/dimension}$. The target BER in the WPAN standard is around 10^{-5}; here the transmission schemes using higher constellations are around 5 dB from the Shannon limit.

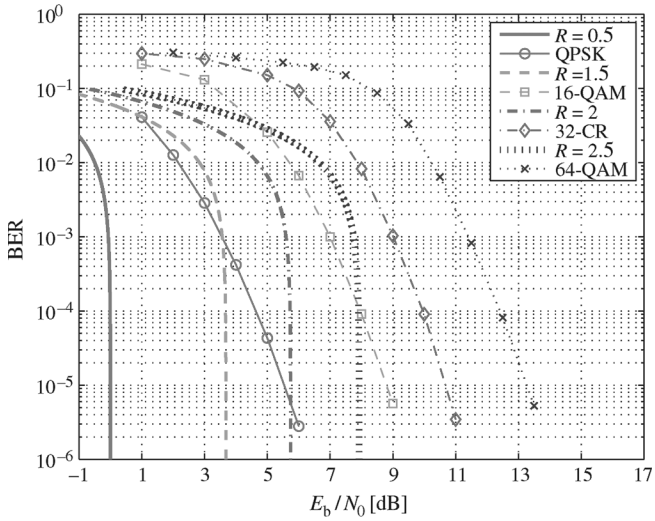


Fig. 1. BER performance in AWGN of TCM codes from [1] together with the Shannon limit of equivalent rate R (bits/dimension) systems. Target BER for transmission is 10^{-5}.

The QPSK scheme is considered as a dropback mode for low E_b/N_0. In such a small-constellation scenario, rate 1/2 binary convolutional codes are usually preferred since they can achieve the same transmission rate as TCM with better BER performance. Simulations show that at the target BER, the best 8-state rate 1/2 convolutional code together with Gray-mapped QPSK is about 0.3 dB better than the TCM QPSK scheme. Apparently, subset partitioning is not very effective at this low rate. In the following, the coding polynomials (octal notation, right justified) considered are $\mathbf{g} = (1, 17/13)$ for the rate 1/2 convolutional encoder in controller form and $\mathbf{h} = (15, 02, 04)$ for the rate 2/3 TCM subset selector in observer form.

The Viterbi algorithm (VA) [6], [7] is a maximum-likelihood (ML) decoding scheme that is used, among others, to recover encoded information corrupted during transmission over a noisy channel. Its processing complexity increases both with

Manuscript received September 12, 2006; revised August 15, 2007. First published February 7, 2008; current version published September 17, 2008. This work was supported by the Competence Center for Circuit Design at Lund University. This paper was recommended by Associate Editor I. Verbauwhede.

M. Kamuf was with the Department of Electrical and Information Technology, Lund University, SE-22100 Lund, Sweden. He is now with the Ericsson Mobile Platforms, SE-22370 Lund, Sweden (e-mail: Matthias.Kamuf@ericsson.com).

V. Öwall and J. B. Anderson are with the Department of Electrical and Information Technology, Lund University, SE-22100 Lund, Sweden (e-mail: Viktor.Owall@eit.lth.se; John_B.Anderson@eit.lth.se).

Digital Object Identifier 10.1109/TCSI.2008.918148

the number of trellis states $N = 2^m$ and branches 2^b per node that connect these states. Here, m denotes encoder memory and there are b information bits per trellis stage.

Flexible Viterbi decoding processors were studied and presented in, for example, [8] and [9]. However, they were intended solely for use with rate $1/c$ binary convolutional codes, c an integer, and provided flexibility by varying the encoder state space and thus error performance. No attempt was made to investigate the cost of this flexibility.

The trellis diagram of rate $1/c$ codes can be decomposed into a radix-2 (R2) butterfly state interconnect structure. To reduce bandwidth expansion, higher rate codes up to rate 1 are obtained by puncturing [10] the basic $1/c$ code, which preserves the R2 structure of the trellis. In case of TCM, however, puncturing is not applicable if code performance is to be fully maintained. This degradation stems from altering the minimum inter-subset distance [11]. The trellis of the considered subset selector consists of radix-4 (R4) butterflies, where 4 branches are leaving and entering each state. Almost all practical trellis codes, whether TCM or not, are based on either R2 or R4 butterflies. Therefore, the presented architecture easily extends to other codes.

To summarize these considerations, a flexible channel decoding architecture has to be tailored to efficiently process both R2 and R4 butterflies, while limiting overhead in both area and power consumption, that is, both modes should utilize the same computational kernel. For the trellis processing blocks such an architecture is presented in [12]. However, flexibility is also required in other processing blocks, namely branch metric (BM) and survivor path (SP) units. In this paper, we consider all processing blocks of the decoder and evaluate the cost of flexibility for our design example.

The paper is organized as follows. In Section II we briefly review the building blocks in the VA. Section III–V describe optimization steps for the different blocks to achieve the desired flexibility with minimum hardware penalty. In particular, it will be seen how the architecture choice of the main processing block enables the other blocks to reuse hardware resources as efficiently as possible. In Section VI, the contribution of flexibility to the overall cost in cell area and power consumption is evaluated based on synthesized hardware. It also presents a chip implementation in a digital 0.13-μm CMOS process.

II. Overall Architecture

For the scope of this paper, we briefly revisit the basic building blocks used in the VA. A more thorough, hardware-oriented description is found in [13]. As shown in Fig. 2, there are three main processing blocks in a Viterbi decoder, that is, BM, trellis, and SP unit. The BM unit provides measures of likelihood $\boldsymbol{\lambda}$ for the transitions in a trellis. These measures are consumed by the trellis unit, where add-compare-select (ACS) operations on the state metrics (SMs) $\Gamma(\mathbf{S}', k)$ at instant k form a new vector of SMs $\Gamma(\mathbf{S}, k+1)$ at instant $k+1$. This operation is equivalent to discarding unlikely branches in the trellis diagram. Here, $\mathbf{S}'$ denotes the vector of states in a trellis and $\mathbf{S}$ is its permutation according to the given state interconnection.

The trellis unit outputs an $N \times b$ matrix $\mathbf{D}$ of decision bits about surviving branches. These bits are processed by the survivor path (SP) unit to reconstruct the information bits that caused the transitions. Additionally, in case of TCM, the most likely transmitted signals $\boldsymbol{\varsigma}$ for all subsets have to be stored in the subset signal memory. These signals, together with the reconstructed subset sequence from the SP unit, form the final decoded sequence.

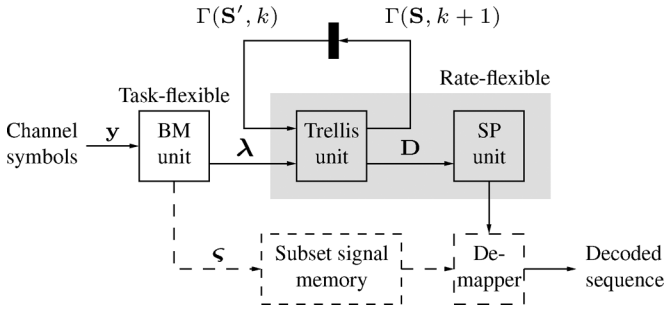


Fig. 2. Block structure of a Viterbi decoder. Flexible parts are indicated and additional parts needed for decoding of TCM codes are dashed.

Which parts of the architecture have to be flexible is seen in Fig. 2. The architecture of the trellis and SP units solely depends on the code rate and number of states in a trellis diagram. A realization in dedicated hardware, which is more energy efficient, is thus logical. The BM unit, on the other hand, is strongly related to the task the Viterbi processor is intended for. For example, apart from calculating distances between received and expected symbols as in the case of binary convolutional coding, TCM codes require an additional subset decoder. Extension of this architecture to cope with, for example, Viterbi equalization would require another processing part for finding the necessary BMs.

III. Branch Metric Unit

We first introduce an applicable distance measure and apply it in the context of TCM. An investigation of symbol quantization follows. Its impact on cutoff rate as well as error performance is evaluated, which ultimately leads to the required BM wordlength that determines the wordlength of the trellis datapath.

Throughout this section, we consider 2-D quadrature modulations that consist of two independent pulse amplitude modulations (PAMs) generated from orthogonal pulses. Dimensions $i = 0, 1$ relate to in-phase ($\mathcal{I}$) and quadrature-phase ($\mathcal{Q}$) signal components, respectively.

To provide measures of likelihood for the state transitions, different metrics are appropriate for different channel models. In the AWGN channel, the optimal distance measure is the squared Euclidean distance between received channel symbol $\mathbf{y}$ and constellation symbol $\mathbf{s}$

$$\sum_{i=0}^{1}(y_i - s_i)^2 \tag{1}$$

where s_i is from the alphabet of M-PAM. In case of binary signaling per dimension where $s_i \in \{-1, +1\}$, (1) simplifies to $-\sum_i y_i s_i$ since y_i^2 and s_i^2 contribute equally to all distances and can be neglected. For larger constellations, different s_i^2 have to be taken into account in the distance calculation.

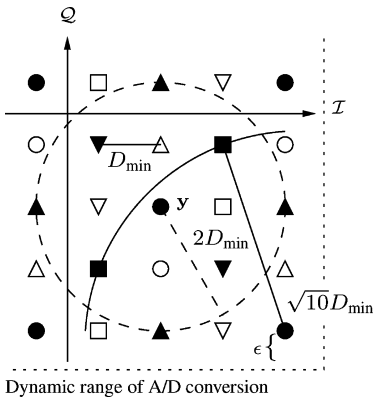


Fig. 3. Maximum distance between a received point and constellation point belonging to one of 8 subsets, which are represented by circles, squares, and triangles. For an infinite constellation (here lattice $\mathbb{Z}^2$), the largest distance (bound by the dashed circle) appears if the received point coincides with any constellation point. For a finite constellation (shown is part of 64-QAM), this distance (bound by the solid circle sector) appears if the received point is in the corner of the constellation. Depending on the maximum number range, an additional factor ϵ has to be accounted for in either dimension.

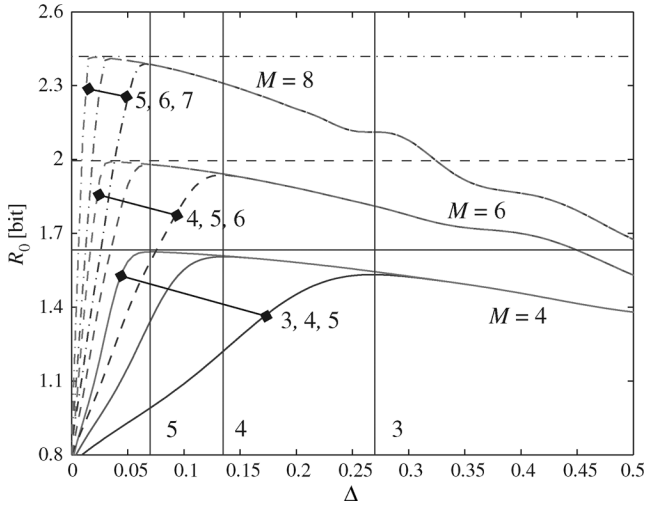


Fig. 4. Cutoff rates for a DMC with M equally spaced inputs ($\sqrt{E_s} = 1$) and 2^q uniformly quantized outputs. E_b/N_0 is chosen for desired quadrature BER $= 10^{-5}$, see Fig. 1. Connected curves are in increasing order q from right to left. Vertical lines denote the optimal Δ for a chosen q found beside the line. The respective maximum R_0 for unquantized channel outputs is indicated by the horizontal lines.

Here, possible simplifications utilize the constellation's bit mapping, commonly Gray, to determine soft bit values [14] that can be utilized by the ML decoder. Due to TCM's subset partitioning, assigning such soft bit values is not meaningful since Gray mapping is only applied to points in the same subset, not to the constellation itself. Furthermore, only the distance to the nearest constellation point in a specific subset is considered as BM for that subset in TCM decoding. This point has to be determined for each subset before the BM calculations. Fig. 3 shows part of the lattice $\mathbb{Z}^2$, where $D_{\min}$ denotes the minimum distance between lattice points and the subset distribution is represented by circles, squares, and triangles. The largest distance appears if the received symbol $\mathbf{y}$ matches a lattice point and the resulting maximum attainable BM $\lambda_{\max}$ is bound by a circle of radius $2D_{\min}$.

For the given finite constellation, though, the situation is slightly different. Assume that the values of the received symbols are limited by the dynamic range of the analog-to-digital (A/D) conversion. Receiving a corner point causes the largest distance, that is, $\sqrt{10}D_{\min}$ as in Fig. 3. Depending on the dynamic range, which ultimately determines the mapping of constellation points in respect to the maximum number range, one might have to consider an additional factor ϵ per dimension. This factor will be included in the BM calculations in the following subsection.

A. Quantization Issues

One method of wordlength design [15] is based on the cutoff rate R_0, which is a lower bound to capacity for any specific signaling constellation. Consider a communication system using M-PAM at the transmitter and a receiver that outputs noisy channel symbols quantized with q bits. This arrangement constitutes a discrete memoryless channel (DMC) with M inputs and 2^q outputs. Let the M symbols be equiprobable (symmetric cutoff rate [16]). Thus

$$R_0 = \log_2 M - \log_2 \left\{ \frac{1}{M} \sum_{i=0}^{2^q-1} \left[\sum_{j=0}^{M-1} \sqrt{P(i|j)} \right]^2 \right\} \quad (2)$$

where the transition probabilities for AWGN are

$$P(i|j) = \int_{r_i} \frac{1}{\sqrt{\pi N_0}} e^{-(y-s_j)^2/N_0} dy \quad (3)$$

and $\{s_j\}$ is the set of M equally spaced constellation points over the interval $[-\sqrt{E_s}, +\sqrt{E_s}]$. A quantization scheme for this DMC is considered optimal for a specific choice of q and N_0 if it maximizes (2). For simplicity, we assume a uniform quantization scheme with precision Δ. The thresholds that bound the integration areas r_i in (3) are located at either

$$0, \pm\Delta, \ldots, \pm(2^{q-1}-1)\Delta \quad \text{(truncation)}$$

or

$$\pm\Delta/2, \ldots, \pm(2^{q-1})\Delta/2 \quad \text{(rounding)}.$$

Furthermore, assume an automatic gain control (AGC) that perfectly estimates the noise variance $\sigma^2 = N_0/2$ by which the noisy channel symbols are normalized.

An information theoretical approach could involve an evaluation of $\partial R_0/\partial\Delta$ to determine the Δ that maximizes R_0 given M, q, and N_0. However, since there is no analytical solution to the integral in (3) for $r_i < \infty$, numerical calculations have to be carried out to find the optimal threshold spacing Δ, which in turn determines the dynamic range of the A/D conversion. Fig. 4

TABLE I
LOSS IN E_b/N_0 AT BER $= 10^{-5}$ FOR UNIFORM SYMBOL QUANTIZATION WITH q BITS AND OPTIMUM CHOICE OF Δ. ABSOLUTE DISTANCES ARE USED ACCORDING TO (4). AS A REFERENCE, ROW $q = \infty$ SHOWS THE REQUIRED E_b/N_0 WITH UNQUANTIZED INPUTS AND EUCLIDEAN DISTANCE AS IN (1)

q	16-QAM	32-CR	64-QAM
∞	8.7	10.7	13.2
7	≈ 0	≈ 0	0.05
6	≈ 0	0.05	0.3
5	0.15	0.31	1.15
4	0.4	0.85	n/a
3	1.4	n/a	n/a

shows the cutoff rates R_0 for different M-PAM schemes. The corresponding M^2-QAM schemes are found in Fig. 1, where E_b/N_0 is assumed to achieve a BER of 10^{-5}. Note that 32-CR corresponds to 36-QAM where the corner points are removed to achieve a lower energy constellation. The vertical lines denote the optimal Δ given q. The upper limit on R_0 for unquantized channel outputs is indicated by the horizontal lines. For example, given $M = 4$ and $q = 4$, the optimal Δ is around 0.135, which yields a dynamic range of 2.16. Considering the precision requirement on Δ, which is based on a good noise estimation, it is already seen from Fig. 4 that a deviation towards a slightly higher Δ than the optimal one is more tolerable. That is, the slope of a cutoff rate curve is rather flat after its maximum, whereas it is comparably steep before it.

Having found a range of feasible Δ for the discrete channel, one can verify these results with BER simulations for the TCM schemes. To lower the complexity of the distance calculation in (1), which involves squaring, a suboptimal metric is used. It considers only the absolute distances in either dimension and (1) is replaced by

$$|y_0 - s_0| + |y_1 - s_1|. \tag{4}$$

Table I summarizes the expected loss in E_b/N_0 compared to (1) and unquantized channel outputs. As expected, for a certain constellation the loss becomes smaller as q increases, and larger constellations generally require finer quantization. QPSK together with binary convolutional coding is not listed in Table I since it is well known that 3 bits symbol quantization is usually sufficient for good performance [17]. A rate 1/2 convolutional code therefore requires 4 bits for the BM.

In order to determine $\lambda_{\max}$, one can consider two approaches. First, to guarantee negligible performance degradation for all schemes, one could choose the q that leads to the largest tolerable degradation for the largest constellation, in this case 64-QAM. Then, the overall loss in E_b/N_0 becomes the least in all transmission modes. Or secondly, one could vary the wordlength of the A/D samples to achieve the required tolerable degradation for each mode.

Pursuing the first approach, assume $q = 7$ bits performs close to optimal with 64-QAM, see Table I. This choice provides negligible degradation for the other two constellations. If the largest q-bit number exceeds the number assigned to the largest constellation point in either $\mathcal{I}$ or $\mathcal{Q}$, an additional factor ϵ has to be added per dimension. Considering Fig. 3 and (4), the largest BM becomes

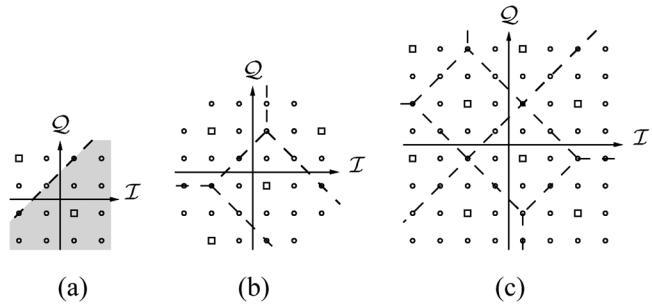


Fig. 5. Decision boundaries for subset D_0 (squares) for 16-QAM (a), 32-CR (b), and 64-QAM (c) constellations. For example, the shaded part in (a) shows the region where $y_1 - y_0 < 2/3$.

$$\lambda_{\max} = 4D_{\min} + 2\epsilon. \tag{5}$$

Choosing Δ according to Fig. 4 to maximize R_0 and achieve equidistant spacing between constellations points yields $D_{\min} = 18$. Then $\lambda_{\max}$ requires at least $\lceil \log_2 72 \rceil = 7$ bits. If the A/D-conversion uses its dynamic range efficiently, ϵ is small compared to the first term in (5) and the number of bits will be sufficient. Using the same Δ also for 16-QAM and 32-CR increases the number of levels between two constellation points so that 8 and 7 bits are now required for the BMs. That is, the lowest constellation needs the largest BM range, which means that the architecture is overdesigned.

Considering the second approach, that is, adjusting the wordlength of the A/D-samples, assume that 16-QAM and 32-CR employ $q = 5$ and $q = 6$ bits, respectively. This yields a $D_{\min}$ of 9 and 13, and the largest BM becomes at least 36 and 52. $\lambda_{\max}$ can now be represented by 6 bits, and the largest BM range applies to the highest constellation. The candidate q are shaded in Table I.

B. Subset Decoding and Signal Memory

In contrast to binary convolutional codes, which carry code symbols along trellis branches, TCM codes carry subsets that themselves consist of signals. Before BM calculations can be done one has to determine the most likely transmitted signal for each subset. This process is called subset decoding. For example, the decision boundaries for subset D_0 are depicted in Fig. 5 for the different constellations. In order to find the most likely subset point, comparisons with $y_1 - y_0$ and $y_1 + y_0$ to boundaries are needed. Furthermore, if there are more than two points per subset, as for 32-CR and 64-QAM, additional comparisons with y_1 or y_0 are required to resolve ambiguities, which are indicated by the horizontal and vertical boundaries. To determine the most likely points for the other subsets, one can either translate the input symbols relative to D_0 or simply adjust the comparison values for the boundaries.

To evaluate the computational effort, consider the lattice $\mathbb{Z}^2$ with an M^2-point constellation that is divided into $1 < P \leq M^2$ subsets with M^2/P points per subset. With $\mu = \log_2 M^2/P$, this setup requires

$$\kappa = \begin{cases} (2\mu - 1) + 2\min\{\mu - 1, 2\}, & \text{for } P = 2^{2k-1} \\ 0 + 2(\mu - 1), & \text{for } P = 2^{2k} \end{cases} \tag{6}$$

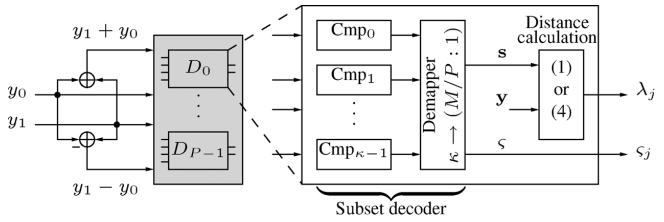


Fig. 6. Architecture of the BM unit for the flexible decoder. The gray part is the overhead due to TCM.

slicing operations (comparisons with a constant) along the decision boundaries. The number of comparisons in (6) is split into two terms: the first relates to diagonal boundaries and the second to horizontal/vertical boundaries. The boolean comparison results are mapped to a unique point in the subset.

The complete architecture for the BM unit is depicted in Fig. 6. In total there are κP slicing operations and P BM calculations according to (1) or (4). These BMs appear as $\boldsymbol{\lambda} = \{\lambda_j\}$ for $j = 0, \ldots, P-1$ in Fig. 2. The overhead due to TCM decoding is indicated in gray in Fig. 6. A subset decoder for D_j consists of κ comparisons and a demapper that chooses from these comparison bits one out of M^2/P constellation points. This point $\mathbf{s}$ is used for distance calculation to yield the BM λ_j. The calculations needed for subset decoding, $y_1 - y_0$ and $y_1 + y_0$, can be reused in case of binary rate 1/2 convolutional coding. These results are equivalent to the BMs for code symbols $\{+1 - 1\}$ and $\{-1 - 1\}$, respectively. The remaining two metrics are derived from these by negation.

As already mentioned in Section II, TCM gives rise to overhead in this flexible architecture: the unit that stores the candidate surviving signals. These "uncoded" bits represent subset points at each trellis stage and together with the reconstructed survivor path form the decoded output sequence. The unit comprises a memory that stores the $\log_2 M^2/P$ bits ς_j from the subset decoder for all P subsets. The length of this first-in first-out (FIFO) buffer equals the latency of the SP unit. For the architecture to be power-efficient, this part has to shut down when TCM is not employed.

To summarize the requirements for the BM unit, it appears that the initial price for flexibility is high due to TCM's subset decoding in combination with its larger constellations.

IV. Trellis Unit

The trellis unit consists of ACS units that are arranged and connected with each other in a butterfly fashion. These units deliver updated SMs $\Gamma(\mathbf{S}, k+1)$ and decisions $\mathbf{D}$ based on previous SMs $\Gamma(\mathbf{S}', k)$ and present BMs $\boldsymbol{\lambda}$. The kernel has to cope with two different code rates, 1/2 and 2/3, and hence both R2 and R4 butterflies are to be processed. Our optimization objective is the following: given a fixed R2 butterfly architecture, how can the R4 butterflies be efficiently mapped onto this architecture, so the existing interconnections can be reused? According to [12], such a flexible processing architecture is preferably based on a modified R2 butterfly block. For convenience, we briefly repeat the concept of this R2-based flexible trellis processing.

A. Processing Framework

We consider an R4 butterfly which utilizes a set of BMs λ_i for $i = 0, \ldots, 3$. As in Fig. 7, there are $N/2^b$ butterflies in a

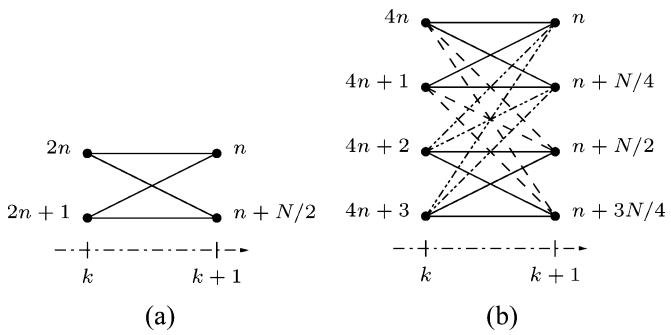


Fig. 7. An R2 butterfly (a) and an R4 butterfly (b) with state labels for encoders in controller form. N is the number of states in the trellis and $n \in [0, N/2^b - 1]$ in the respective cases, $b = 1, 2$. The R4 butterfly can be decomposed into four R2 butterflies, indicated by the different line styles.

trellis, that is, given $N = 4$ and $b = 2$, there is only one R4 butterfly. Since $n \in [0, N/2^b - 1]$, we have $n = 0$ and the state labels become 0,. . ., 3.

To update one state in an R4 butterfly, one can carry out all six possible partial comparisons in parallel [18]. Four such operations are needed to calculate a complete R4 butterfly as in Fig. 7(b). However, this leads to inefficient hardware reuse in a rate-flexible system due to the arithmetic in the 4-way ACS units. In [19], area-delay complexity of R2- and R4-based butterfly processing is evaluated. Two cases are considered; one where an R2-based trellis is processed with R2 processing elements (PEs), and one where two R2 trellis stages are collapsed into one R4 stage [18], which is processed with the 6-comparator approach. For a standard cell design flow (this includes FPGA implementations), R2 PEs are found to be more cost-efficient, whereas a full-custom datapath as in [18] benefits the R4 6-comparator method. This is due to the achieved speed-ups compared to the area overhead for these approaches. In a standard cell design flow, the achieved speed-up was 1.26, whereas for the full-custom design it was 1.7. On top of this, introducing a redundant number representation and bit-level pipelining in a time-shared ACS datapath, the authors of [20] increased the speed-up to 1.9, only 5% from the optimum of 2. Again, their work reaches into the domain of full-custom tailormade datapath implementations. A similar approach was followed in [21], which is based on physical design-oriented hard macro blocks from an in-house datapath generator. The authors concur that this involves a higher design effort. Since platform independence and standard design flows are desired properties, we use an architecture that can be designed with standard cells and tools. Based on the preceding considerations, the said architecture thus consists of R2 butterfly units.

Another R4 trellis decoding architecture is found, for example, in [22] and it belongs to the domain of log-MAP decoders used in iterative decoding. The authors follow the same trellis collapsing approach as previously described. However, instead of using the 6-comparator method, they apply a straightforward two logic level approach for the 4-way ACS, similar to the one described in the following paragraphs. To summarize, their work is more concerned with efficient calculation of the logsum look-up table needed in log-MAP decoding and approximations that increase implementation efficiency without degrading decoding performance.

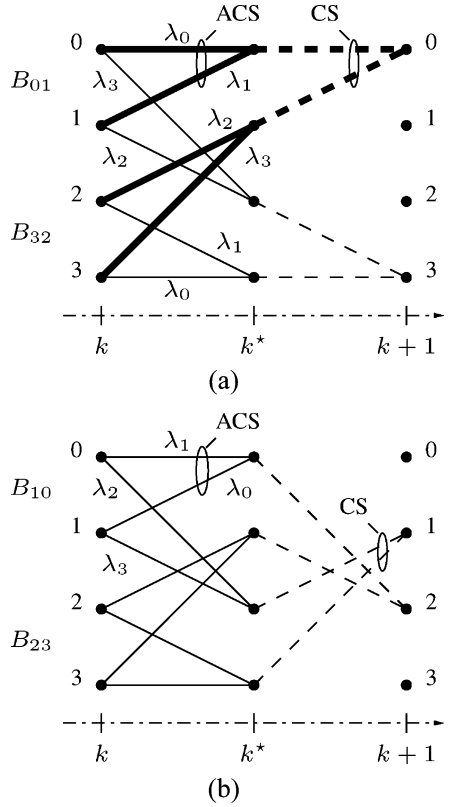


Fig. 8. A decomposed R4 butterfly and the two partial computations leading to the updated metrics for states 0 and 3 in (a) and 1 and 2 in (b). As an example, the necessary operations to update SM 0 are drawn bold.

Generally, a 4-way ACS operation can be carried out in two successive steps: in the first step a pair of cumulative metrics (ACS) is evaluated and discarded; then in the second step, one of the surviving metrics is discarded, which corresponds to a compare-select (CS) operation in Fig. 8. This procedure is a decomposition into R2 operations that are separated in both state and time. Considering step $(k \rightarrow k^\star)$, the split into four R2 butterflies achieves the cumulation of the SMs with all four BMs. Then, in step $(k^\star \rightarrow k+1)$, the partial survivors are compared and the final survivors selected. Here, Fig. 8(a) updates states 0 and 3, and Fig. 8(b) updates states 1 and 2.

To capture these processing steps formally we use the following definition. The state connectivity of an R2 butterfly is defined in Fig. 7(a). Assume that the two states at time k are named u' and v' with SMs $\Gamma(u',k)$ and $\Gamma(v',k)$, respectively. The two ACS operations leading to two updated state metrics for states u and v at stage $k+1$ are expressed as butterfly operation $B_{u'v'}$. Without loss of generality, the λ_i are distributed as in

$$B_{u'v'} \equiv \begin{pmatrix} \Gamma(u,k+1) \\ \Gamma(v,k+1) \end{pmatrix} = \begin{pmatrix} \min \begin{pmatrix} \Gamma(u',k)+\lambda_0 \\ \Gamma(v',k)+\lambda_1 \end{pmatrix} \\ \min \begin{pmatrix} \Gamma(v',k)+\lambda_2 \\ \Gamma(u',k)+\lambda_3 \end{pmatrix} \end{pmatrix}. \tag{7}$$

We have already seen from Fig. 8 that there are four such R2 butterflies between k and $k^\star$, so four operations as in (7) are needed. For example, B_{01} is shown in Fig. 8(a), that is, $u'=0$ and $v'=1$.

Processing the R4 butterfly based on (7) preserves the compatibility with the base R2 architecture. The scheme for obtaining all partial survivors is then expressed as

$$\mathbf{B}' = (B_{01} \quad B_{10}) \tag{8}$$

where the columns determine the instance of an iteration. So far we have only computed half of the partial survivors needed; to complete the R4 butterfly in parallel another unit has to carry out

$$\mathbf{B}'' = (B_{23} \quad B_{32}). \tag{9}$$

The operations in (8) and (9) guarantee that all SMs at stage k are added to all BMs, that is, 16 partial sums are reduced to 8 partial survivors at intermediate stage $k^\star$ by means of CS operations. The final surviving SMs at stage $k+1$ are obtained by CS operations on the hitherto surviving metric pairs. Note that the partial survivors are not altered and therefore the final SMs are not changed compared to a straightforward implementation.

B. R2-Based Approach

According to the preceding considerations, all partial survivors are calculated during two cycles, and in the third cycle the final update takes place. As an example, the operations to update state metric 0 are drawn bold in Fig. 8(a). The partial survivors needed for the final CS are created at instance 0 by operation $\mathbf{B}'$ and at instance 1 by $\mathbf{B}''$. These operations are carried out in different butterfly units; that is, the partial survivors have to be stored temporarily. The appropriate routing for the final CS is according to the required ordering of the updated SMs. Here, the partial survivors are brought together by means of I/O channels between adjacent butterfly units as indicated on the left side of Fig. 9.

Fig. 9 also shows the rate-flexible butterfly unit. Its arithmetic components, adders and the CS units, are identical to the ones in a R2 butterfly unit, that is, if the gray parts are removed, one yields a standard R2 butterfly unit. To cope with a decomposed R4 butterfly, routing resources (shaded in gray) are provided to distribute the partial survivors as dictated by the BM distribution and the state transitions. The input MUXes shuffle the two input SMs to guarantee their cumulation with all four BMs. The 4:2 MUXes in front of the CS units select whether the partial survivors at stage $k^\star$ are to be captured into the routing unit PERM or the final comparison at stage $k+1$ is to be performed. When carrying out (8) or (9), PERM is fed during two cycles and in the third and final cycle the partial survivors are compared. Here, the signals I and O provide the connections to the adjacent butterfly unit to carry out the comparison with the desired pairs of partial survivors. For example, O_0 is connected to I_0 in the adjacent butterfly unit. The CS operations at steps $(k \rightarrow k^\star)$ and $(k^\star \rightarrow k+1)$ in Fig. 8 are executed by the same CS unit, thus saving hardware.

The unit PERM, which is needed for permutating the partial survivors, simply consists of two tapped delay lines. If the global SM memory is implemented as a bank of registers, they can be reused to store the second intermediate survivors. Hence, PERM would be reduced to only two storage elements to capture the

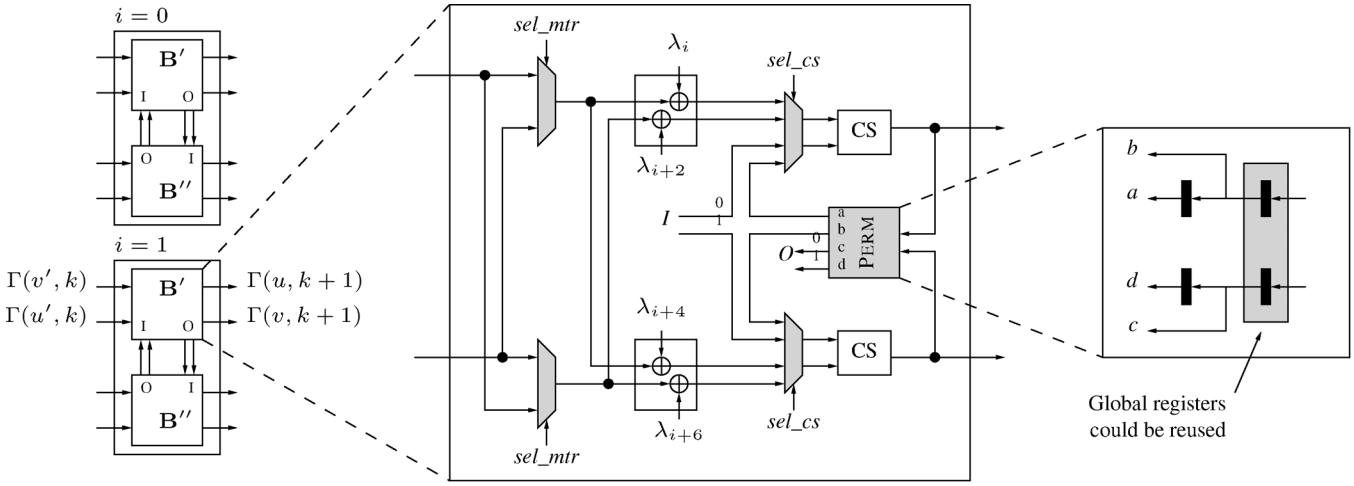


Fig. 9. Shown on the left are two butterfly unit pairs ($i = 0, 1$) that carry out the trellis processing in either R2 or R4 manner. The basic building block of these pairs is a rate-flexible butterfly unit and is depicted in the middle. An R4 butterfly is updated in three clock cycles. The shaded blocks are the overhead compared to an R2 butterfly unit. The connections for the routing block PERM apply to pair $i = 0$ in the design example.

first intermediate survivors. If it turns out that the metric in the global register is the surviving one, the final update can be suppressed and no switching power would be consumed. On average, 50% of the final updates are superfluous.

Given the BM assignments of the TCM code, PERM carries out the same permutation in a pair of adjacent butterfly units. For the codes considered, there are two such pairs in total. For the pair $i = 0$ the partial survivors on the top rail, a and b, are devoted to the same butterfly unit, whereas the bottom rail survivors, c and d, are assigned to the adjacent butterfly unit. For the pair $i = 1$, it is vice versa. Now the design fits seamlessly into the base architecture; that is, the feedback network ("perfect shuffle" state interconnection is assumed) in Fig. 2 is reused as is. Furthermore, the survivor symbols to be processed by the SP unit become equivalent to the information symbols. It will be seen that this is beneficial for the chosen implementation of the SP unit.

Additionally, a controller is needed to provide control signals to the MUXes (sel_mtr and sel_cs) and clock enable signals to the registers. Clocking is only allowed when input data is valid so that no dynamic power is consumed unnecessarily. In R2 mode, these signals are kept constant. Neglecting the controller, the rate-flexible butterfly unit only adds six 2:1 MUXes and four (two) registers on top of a R2 butterfly unit, and there is no arithmetic overhead.

V. SURVIVOR PATH UNIT

We start this section by discussing some basic algorithms for SP processing, namely register-exchange (RE) and trace-back (TB); see [13] for an overview or [23], [24] for in-depth coverage. Let L denote the necessary decoding depth of the convolutional code after which the survivor paths are expected to have merged with sufficiently high probability. The necessary depth is determined by simulations. Furthermore, to cope with rate flexibility, specific architectural trade-offs have to be investigated for this unit.

A. Existing Algorithms

Register-Exchange: This method is the most straightforward way of survivor path processing since the trellis structure is directly incorporated in the algorithm. Every trellis state is linked to a register that contains the survivor path leading to that state. The information sequences of the survivor paths are then continuously updated based on the decisions provided by the ACS units.

In a parallel implementation, bNL bits need to be stored and the latency of this algorithm is simply L. Since all these bits must be read and written in every trellis stage, an implementation in high density random access memory (RAM) is impractical due to the high memory bandwidth requirement. Instead, the algorithm is preferably realized by a network of multiplexers and registers that are connected according to the trellis topology. For a larger number of states, though, the low integration density of the multiplexer-register network and the high memory access bandwidth of bNL bits per cycle become the major drawback of this algorithm.

Trace-Back: This method is a backward processing algorithm and requires the decisions from the ACS units to be stored in a memory. After having found the starting state of a decoding segment, typically after an L-step search through a segment where all survivor paths merge into one, this surviving state sequence is reconstructed in a backward fashion by means of the stored decisions. The corresponding information symbols are output time-reversed and, therefore, a last-in-first-out (LIFO) buffer has to be introduced to reverse the decoded bitstream.

Considering an architecture with two read pointers, where decode, merge, and write segments are each of length L, we find the memory requirement is $3bNL$ bits for the decision memory plus bL bits for the LIFO, and latency is $4L$.

Only bN compared to bNL decision bits are written every cycle, lowering the memory access bandwidth. Another advantage compared to RE is the storage of the decisions in a much denser memory, typically RAMs. The cost is higher memory requirement and latency.

In order to lower both memory requirement and latency in TB-based architectures the trace-forward (TF) procedure [25]

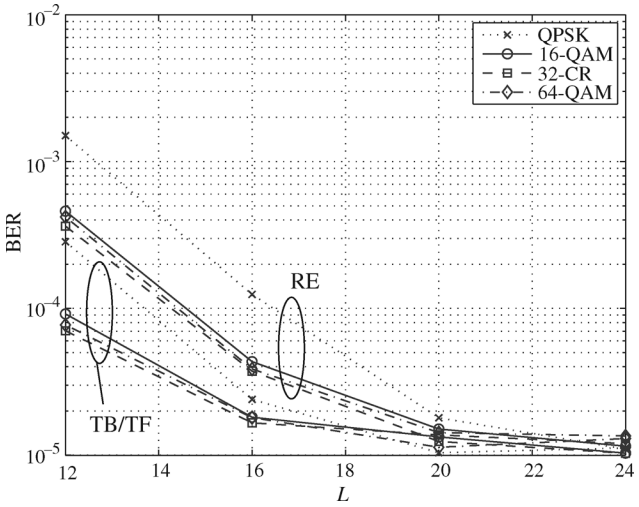


Fig. 10. BER performance for the RE and TB/TF algorithms and decoding depth L. Assumed E_b/N_0 for the TCM modulation schemes that use rate 2/3 subset selectors appears in Table I in row $q = \infty$. For the Gray-mapped QPSK scheme with rate 1/2 convolutional coding, the E_b/N_0 is 5.4 dB.

can be applied. This is a forward-processing algorithm to estimate the starting state for a trace-back decode such that the merging segment can be omitted. It relies on the fact that all tail states of the survivor path are expected to coincide after L steps. Compared to the two-pointer architecture, the memory depth decreases from $3L$ to $2L$ and latency from $4L$ to $3L$. The number of bits for RAM and TF unit is $2bNL$ and Nm.

B. Decoding Depth

An estimation for the decoding depth L for convolutional codes is given in [17]. Rate 1/2 codes need to be observed over a length of around five times the constraint length of the code. To estimate the largest necessary L for both code rates (1/2 and 2/3), we compare TB/TF and RE approaches and their expected performance degradation at E_b/N_0 required for a BER of 10^{-5} for the different transmission schemes, as shown in Fig. 10. It is seen that both approaches do not need more than $L = 20$. The degradation of RE compared to TB/TF for smaller decoding depths is caused by using fixed-state decoding, where the decoded bit is always derived from a predefined state. This is less complex than taking a majority decision among all RE outputs and saves memory by neglecting bits connected to states that cannot reach this predefined state at step L.

C. The Designed Rate-Flexible Survivor Path Unit

For this design, we employ the RE approach since the number of states is rather low, and, considering the additional subset signal memory needed for TCM, the least overhead is introduced since the decoding latency is by far the lowest. If $R_{\max}$ denotes the maximum rate of the TCM transmission, $L \times (R_{\max} - b)P$ extra bits are required, whereas three times more are needed for a TB/TF approach because its latency is three times higher. In this design $R_{\max}$ equals 5 for the 64-QAM constellation.

Additionally, for TCM a demapper has to be employed that delivers the most likely subset signal at a certain time. This is a multiplexer which chooses a subset signal depending on the decoded subset number from the SP unit. Recall that for convolutional decoding, b information bits are decoded every cycle. This is not sufficient for this task since the subset number consists of $b + 1$ bits. In this case, the RE algorithm must store in total $(b+1)NL$ bits. More precisely, $(b+1)\sum_{i=1}^{m-1}(N-i)$ bits can be neglected due to fixed-state decoding.

To reduce memory, an alternative approach only considers sequences of b information bits in the RE network as in the case of convolutional decoding. These are the estimated uncoded bits of a subset number; that is, only the two most significant bits (MSB) $\hat{x}_2, \hat{x}_1$, are decoded. A demapper has to choose the correct subset based on the MSBs in order to decide the most likely subset signal. This is achieved by an additional encoder fed by $\hat{x}_2, \hat{x}_1$. Together with the resulting coded bit $\hat{x}_0$, the subset number is now complete.

Once there is a deviation from the ML path, a distorted sequence for the coded bits is created in the decoder, which in turn chooses wrong subset signals during the error event. Although the error event for $\hat{x}_2, \hat{x}_1$ can be quite short, the resulting event for $\hat{x}_0$ becomes much longer since the encoder has to be driven back into the correct state. From examination of the encoder properties, 50% of the coded bits are expected to be wrong during this event. Simulations show that this approach is quite sensitive to variations in the signal-to-noise ratio (SNR) which determines the number of unmerged paths at depth L that cause these error events. The decoding depth has to be increased, eventually to the point where the total number of registers is larger than in the previous case. For the E_b/N_0 considered, it turned out that L is 24, requiring slightly fewer stored bits than in the original approach where $L = 20$. The latter's robustness, on the other hand, is more beneficial for this implementation.

Which radix is most beneficial for a building block in the RE network? Recalling the decomposition approach from the trellis unit in Section IV, which saved arithmetic units at the cost of a slight increase in storage elements, one is tempted to apply the same approach to the R4 RE network. Again, one wants to break the R4 structure into R2 blocks. Note that in an RE network there is no arithmetic, and, contrary to the trellis unit, not only one but in total L duplicates of a trellis step are connected in series. Per trellis stage, there are $4N/2^b$ bits overhead, which is not acceptable in this implementation. Therefore, a straightforward R4 implementation of the RE network is pursued.

An architecture for the R4 RE network is depicted in Fig. 11(a). The network is visually split into three separate slices, to represent the processing of the three survivor path bits representing a subset. The basic processing element of a slice consists of a 4:1 MUX (equals three 2:1 MUXes), connected to a 1-bit register. Expressed in terms of 2:1 MUXes, the hardware requirement for this approach is $9NL$ MUXes and $3NL$ registers. However, the network can be improved by matching it to the throughput of the trellis unit. Remember that R4 processing takes 3 clock cycles and thus the RE update can also be carried out sequentially. That is, the registers are now placed in series such that three cycles are needed to update the complete survivor sequence, as in Fig. 11(b). The hardware requirement is dramatically lowered since 66% of the MUXes and interconnections become obsolete. At the same time, the utilization for both modes is effectively increased.

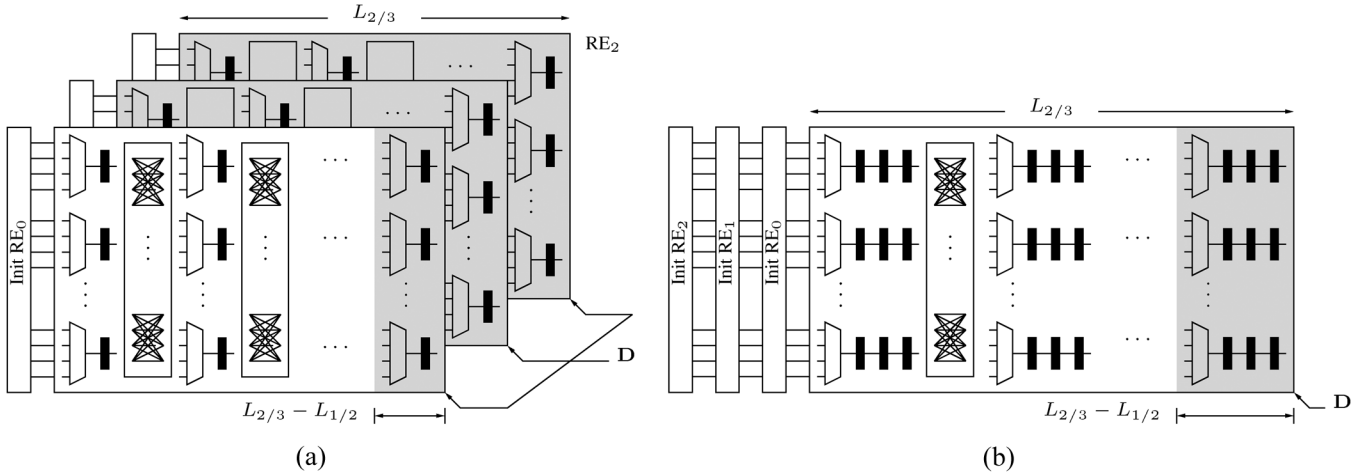


Fig. 11. Two architectures of the RE algorithm to suit the combined convolutional and TCM decoder. (a) shows the straightforward scheme, which basically consists of three RE slices in parallel. Two of these slices are considered overhead for rate $1/c$ binary convolutional codes. The second architecture in (b) is matched to the throughput of the trellis unit and executes the survivor path update in a serial fashion, thus saving 66% of the MUXes and interconnections. In both figures, the shaded parts can be disabled for convolutional decoding. The initialization Init RE_i is derived from the decision bits in $\mathbf{D}$ and state numbers.

Were it only for R4 processing, the sequential elements could be simply realized as edge-triggered master-slave flip-flops. However, R2 processing, which allows only one cycle for the survivor path update, requires the first two registers to be bypassed. There are two cures to the problem: either one introduces another 2:1 MUX in front of the third register in a stage, or the first two sequential elements in a stage are latches that are held in transparent mode. Since flip-flop-based designs are more robust considering timing and testability, the first approach is applied.

Parts of the RE network could usually be disabled since the decoding depth L of the rate 1/2 code is expected to be less than for the 2/3 code. This is indicated by the shaded parts in Fig. 11. However, following the simulations in Fig. 10, we choose $L = 24$ for both code rates to have some extra margin for varying SNR. The initial values fed into the network (Init RE_i) are derived from the decision bits $\mathbf{D}$ and, in case of R4 processing, state numbers.

VI. Hardware Evaluation and Discussion

In this section, we first study how and in which parts an increased flexibility impacts the design. Then, a chip implementation is presented.

We consider designs that provide up to three settings to adapt to varying channel conditions. For low SNR, rate $R = 1$ (data bits per 2-D channel use) is employed, which uses a rate 1/2 convolutional code together with Gray-mapped QPSK. A rate $R = 3$ mode is incorporated in the design in case there is higher SNR. In addition to the previous setup, TCM with a rate 2/3 subset selector and 16-QAM as master constellation is provided. On top of this, the third mode uses TCM with 64-QAM, thus adding $R = 5$. In the following, the different designs are named by their maximum transmission rate. Note that design ONE is fixed since it only provides $R = 1$, whereas THREE and FIVE are flexible. Furthermore, a more flexible design incorporates a less flexible one without additional hardware cost.

Recall from Section IV that the architecture of the trellis unit consists of R2 elements. Thus, the processing for the convolutional code in system ONE is one symbol per cycle. However, the other two systems need to support R2 and R4 processing. The trellis unit determines the computation rate of the whole system, which becomes one symbol per three cycles for R4 processing. We now turn to implementation aspects for the processing units, and evaluate the cost of flexibility based on the synthesized hardware blocks.

A. Branch Metric Unit

In Section III it is shown that the BM unit requires additional resources for designs THREE and FIVE because of TCM's subset decoding in combination with larger constellations. As indicated in the previous sections, additional hardware resources due to flexibility can be minimized by matching the rates of the processing units in the designs. This is done by interleaving the BM calculations and reusing the subset decoding units for the other subsets.

Subset decoding for 16-QAM is simply an MSB check of $y_1 \pm y_0$. This comparison unit is reused for 64-QAM, where the maximum number of boundaries is 9 according to (6) and Fig. 5(c).[1] Simulations show that removing the 4 extra comparisons needed to resolve ambiguities in 64-QAM has no noticeable effect on the overall BER.

With the given subset distribution, some slicing operations apply to a pair of subsets; for example, for D_1 and D_3 [1] there are three comparison results that apply to the diagonal boundaries $y_1 - y_0$. It is therefore beneficial to calculate such subset pairs together in one cycle. Subset decoding units are reused by translating the input symbols, here only y_0, for the subset pair in question. Two operations are required to form $y_0 \pm D_{\min}$; in the first cycle $y_0 + D_{\min}$ is processed and in the second $y_0 - D_{\min}$. Thus, in total it takes three cycles to decode ς_j and calculate

[1]From Fig. 5 it is also seen that 32-CR would need an additional two slicers, which are not used by the other two constellations, causing an overhead of 18%. To lower the already high complexity of the BM unit, 32-CR is thus omitted in the flexible designs.

TABLE II
THREE DESIGNS WITH DIFFERENT TRANSMISSION RATES. POWER CONSUMPTION $P(R)$ ESTIMATED AT $V_{\mathrm{dd}} = 1.2$ V AND $f_{\mathrm{clk}} = 250$ MHz

Design	R			Highest mod.	Area [μm^2]	BM unit	Trellis unit	SP unit	Subset memory	Demap.	$P(R)$ [mW]		
ONE	1	—	—	QPSK Gray	16289	3.9%	46.8%	49.3%	—	—	4.9	—	—
THREE	1	3	—	16-QAM TCM	67205	12.6%	19.3%	52.0%	14.6%	1.5%	9.9	13.4	—
FIVE	1	3	5	64-QAM TCM	76922	17.8%	18.7%	42.8%	18.5%	2.3%	10.3	14.7	15.2

λ_j for all subsets. Now the computation rate is matched to the trellis unit. Note that a latency of three cycles is introduced by this hardware sharing, which has to be accounted for in the depth of the subset signal memory.

Since the processing in the BM unit is purely feedforward, it can be easily pipelined so the throughput of the design is determined by the feedback loop of the trellis unit. Therefore, two additional pipeline stages were introduced and the depth of the subset signal memory was adjusted accordingly.

This memory could be partitioned in order to efficiently support 16-QAM and 64-QAM, which use 1 and 3 bits to specify a subset signal, respectively. The required memory width is the number of subsets times the maximum number of subset signal bits, that is, 8×3. Based on area estimates from custom memory macros, it turns out that a single 24 bit wide implementation gives less overhead than three separate 8 bit wide blocks. Nevertheless, memory words are partitioned into 8 bit segments that can be accessed separately and shut down to save power. To account for all latency in the flexible designs, a single-port memory of size 28×24 is used. Since simultaneous read and write accesses are not allowed in single-port architectures, an additional buffer stage is required.

B. Trellis Unit

To avoid manual normalization of the otherwise unbounded wordlength increase of SMs in the trellis datapath, the modulo normalization technique from [26] is used. It relies on the fact that the VA bounds the maximum dynamic range $\delta_{\max}$ of SMs to be $\delta_{\max} \leq m \cdot \lambda_{\max}$.

If two SMs $\Gamma(u', k)$ and $\Gamma(v', k)$ are to be compared using subtraction, the comparison can be carried out according to $[\Gamma(u', k) - \Gamma(v', k)] \bmod 2\delta_{\max}$ without ambiguity. Modulo arithmetic is simply implemented by ignoring the SM overflow if the wordlength is chosen to represent twice the dynamic range of the cumulated SMs before the compare stage, namely

$$\lceil \log_2(\delta_{\max} + \lambda_{\max}) \rceil + 1 \text{ bits}. \tag{10}$$

The datapath wordlengths vary due to varying symbol quantization for the different constellations. Choosing $q = 7$ bits, which leads to acceptable degradation for 64-QAM according to Table I, 10 bits are needed to represent the SMs. For 16-QAM and 32-CR ($q = 5, 6$), 9 bits are required for the SMs, whereas in the QPSK case ($q = 3$), the wordlength turns out to be 7 bits. That is, in order to be power-efficient for both convolutional and TCM decoding, one could slice the datapath to 7 plus 3 extension bits, and prevent the latter bits from toggling in QPSK mode.

C. Impact of Flexibility

The cost of the flexible processing blocks was already characterized at the architecture level. Now, three flexible designs are considered. The numbers for the cell area (expressed as NAND2-equivalent kGates) in Table II apply to synthesized blocks at the gate level. We applied the same design constraints to allow a fair comparison. The constraints are chosen such that the implementation is still in the flat part of the area-delay curve and the resulting critical path for the designs lies in the trellis unit.

As flexibility is introduced, for example, from design ONE to THREE, note that the BM unit gets a larger share of the total area. In design ONE, it is almost negligible, whereas in design THREE, it is comparable to the size of the trellis unit; in design ONE this unit took half of the total size, declining to about a fifth in THREE. The growth of the BM unit is not only due to TCM; it is mainly due to the required slicing operations, which stem from larger QAM constellations and would have to be considered even for a Gray-mapped convolutional code.

The higher code rate of the subset selector for design THREE and FIVE impacts the trellis and SP units. These units are in theory independent of the code details. However, the size of the SP unit is partially influenced by TCM; the $b + 1$ bits that represent a subset number are processed per RE stage and state, instead of b bits in the case of conventional convolutional decoding. Contrary to the trellis unit, the SP unit now takes a larger part. The dramatic decrease of the trellis unit share certainly justifies the R4 emulation by R2 elements. Recall that this emulation would not have made sense for the SP unit, where one has to accept the R4 processing character, hence, its percent growth.

For design FIVE, the BM share grows even further, although not as much as before; it appears that the initial price for task flexibility or larger constellations has already been paid. Also, the percent cost of the trellis unit decreases slightly, although another transmission rate has been introduced. It appears that task flexibility has a larger impact on the size of an implementation than rate flexibility. That is, the underlying trellis structure of a code is much easier to reuse than BM calculations that are specialized for a certain task.

The observed trend is expected to continue for larger constellations. The BM unit takes even larger portions, whereas trellis and SP units, which are principally fixed except for the growth of the wordlength (10), drop in percent. The parts exclusive of the TCM, subset memory and demapper, consume roughly a fifth of the cell area.

Power estimation in Table II is carried out with Synopsys Power Compiler on the synthesized netlists, back-annotated with state- and path-dependent toggle information obtained from a simulation run. There are in principle two comparison scenarios that need to be distinguished: first, convolutional decoding using either a fixed (ONE) or one of the flexible designs

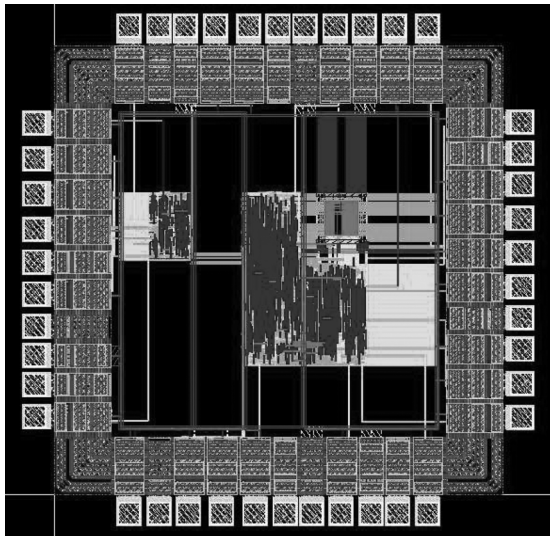


Fig. 12. Layout of the routed chip. Designs ONE and FIVE are shown on the left and right side, respectively. Row utilization is approximately 80% in both implementations.

(THREE or FIVE) to find out how much power one has to sacrifice for a certain flexibility; and second, a comparison between designs THREE and FIVE to see how much more power has to be spent for additional transmission rates. These comparisons yield a measure of the cost of flexibility.

Not surprisingly, power consumption is sacrificed for flexibility. Scenario one indicates that from design ONE to THREE, there is twice the power spent to run the flexible design with transmission rate 1. For design FIVE, the number is slightly higher but still roughly twice the amount of power for rate 1 compared to the fixed design. Comparing designs THREE and FIVE, we find there is a 4% and a 9.7% increase in power consumption for rate 1 and 3 configurations. Furthermore, rate 5 mode in design FIVE only requires an extra 3.4% power, a low number considering the additional rate provided.

To conclude, having accepted the initial impact of task flexibility in the TCM-designs, it makes sense to strive for more transmission rates. Therefore, the two designs ONE and FIVE will be implemented on a chip.

D. Silicon Implementation

The complete design was modeled in VHDL at register-transfer level (RTL) and then taken through a design flow that includes Synopsys Design Compiler for synthesis and Cadence Encounter for routing. We used a high-speed standard cell library from Faraday for the 0.13-μm process from United Microelectronics Company (UMC). The RTL and gate level netlists are all verified against test vectors generated from a MATLAB fixed-point model. Post-layout timing is verified using Synopsys Prime Time with net and cell delays back-annotated in standard delay format.

Fig. 12 shows the layout of the fabricated chip. It is pad-limited due to test purposes and measures 1.44 mm^2. Designs ONE and FIVE are placed on the same die with separate $V_{\rm dd}$ to measure their power consumption independently. In TCM mode, design FIVE achieves a symbol rate of 168 Mbaud/s, a throughput of 504 Mbit/s and 840 Mbit/s using $R = 3$ and $R = 5$ configurations. Design ONE achieves a throughput of 606 Mbit/s; flexibility causes a speed penalty in that FIVE provides 504 Mbit/s in $R = 1$ mode.

If WPANs are the application, these throughputs are higher than WPAN specification. Thus, supply voltage can be lowered to save energy. Measurements on the fabricated chip will show how much speed has to be traded for each energy reduction.

VII. Conclusion

We presented a design for a Viterbi decoder that decodes both convolutional and TCM codes to cope with varying channel conditions. Sacrifices in the speed of the trellis unit result in large hardware savings for the other processing blocks by applying computation rate matching. Synthesized designs that provide different transmission rate combinations show that task flexibility inherent in the BM unit impacts the design size far more than rate flexibility of trellis and SP units. Furthermore, power estimation figures at the gate-level indicate that the flexible designs become more cost-effective if provided with more than two transmission rates. Last, to yield a quantitative cost on flexibility, a silicon implementation is crucial. The implementation is fabricated in a 0.13-μm CMOS process. Thus, the performance of two designs, one fixed, one flexible, can be compared. For good channel SNRs, the flexible design enables a 28% higher throughput than the fixed, while it only lags by 17% when run in low SNR configuration.

Acknowledgment

The authors thank Dr. J. N. Rodrigues for help with the layout of the chip and UMC for its fabrication.

References

[1] *Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for High Rate Wireless Personal Area Networks (WPANs)*, IEEE Std. 802.15.3, 2003.

[2] J. Karaoğuz, "High-rate wireless personal area networks," *IEEE Commun. Mag.*, vol. 39, no. 12, pp. 96–102, Dec. 2001.

[3] Z. Ding and S. Lin, Channel Equalization and Error Correction for High Rate Wireless Personal Area Networks Dept. Electrical and Computer Engineering, Univ. of Calif., Davis, Tech. Rep. MICRO 01-029, 2001.

[4] G. Ungerböck, "Channel coding with multilevel/phase signals," *IEEE Trans. Inf. Theory*, vol. 28, no. 1, pp. 55–67, Jan. 1982.

[5] G. Ungerböck, "Trellis-coded modulation with redundant signal sets," *IEEE Commun. Mag.*, vol. 25, no. 2, pp. 5–21, Feb. 1987.

[6] A. J. Viterbi, "Error bounds for convolutional codes and an asymptotically optimum decoding algorithm," *IEEE Trans. Inf. Theory*, vol. 13, no. 2, pp. 260–269, Apr. 1967.

[7] G. D. Forney, Jr., "The Viterbi algorithm," *Proc. IEEE*, vol. 61, no. 3, pp. 268–278, Mar. 1973.

[8] D. E. Hocevar and A. Gatherer, "Achieving flexibility in a Viterbi decoder DSP coprocessor," in *Proc. IEEE Veh. Technol. Conf.*, Boston, MA, Sep. 2000, pp. 2257–2264.

[9] J. R. Cavallaro and M. Vaya, "Viturbo: A reconfigurable architecture for Viterbi and turbo decoding," in *Proc. IEEE Intl. Conf. Acoust., Speech, Signal Process.*, Hong Kong, Apr. 2003, pp. 497–500.

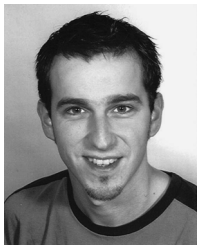
[10] J. B. Cain, G. C. Clark, Jr., and J. M. Geist, "Punctured convolutional codes of rate $(n-1)/n$ and simplified maximum likelihood decoding," *IEEE Trans. Inf. Theory*, vol. 25, no. 1, pp. 97–100, Jan. 1979.

[11] J. B. Anderson and A. Svensson, *Coded Modulation Systems*. New York: Plenum, 2003.

[12] M. Kamuf, V. Öwall, and J. B. Anderson, "Architectural considerations for rate-flexible trellis processing blocks," in *Proc. IEEE Intl. Symp. Personal, Indoor, and Mobile Radio Commun.*, Berlin, Germany, Sep. 2005, pp. 1076–1080.

[13] H. Dawid, O. Joeressen, and H. Meyr, "Viterbi decoder: High performance algorithms and architectures," in *Digital Signal Processing for Multimedia Systems*, ser. Signal Processing Series. New York: Marcel Dekker, Feb. 1999, ch. 17.

[14] F. Tosato and P. Bisaglia, "Simplified soft-output demapper for binary interleaved COFDM with application to HIPERLAN/2," in *Proc. IEEE Intl. Conf. Commun.*, New York, Apr./May 2002, pp. 664–668.
[15] I. M. Onyszchuk, K.-M. Cheung, and O. Collins, "Quantization loss in convolutional decoding," *IEEE Trans. Commun.*, vol. 41, no. 2, pp. 261–265, Feb. 1993.
[16] L.-N. Lee, "On optimal soft-decision demodulation," *IEEE Trans. Inf. Theory*, vol. 22, no. 4, pp. 437–444, Jul. 1976.
[17] J. A. Heller and I. M. Jacobs, "Viterbi decoding for satellite and space communication," *IEEE Trans. Commun.*, vol. 19, no. 5, pp. 835–848, Oct. 1971.
[18] P. J. Black and T. H.-Y. Meng, "A 140-Mb/s, 32-state, radix-4 Viterbi decoder," *IEEE J. Solid-State Circuits*, vol. 27, no. 12, pp. 1877–1885, Dec. 1992.
[19] H. T. Feldkämper, H. Blume, and T. G. Noll, "Study of heterogeneous and reconfigurable architectures in the communication domain," *Adv. Radio Science—Kleinheub. Berichte*, vol. 1, pp. 165–169, May 2003.
[20] A. K. Yeung and J. M. Rabaey, "A 210-Mb/s radix-4 bit-level pipelined Viterbi decoder," in *Dig. Tech. Papers IEEE Intl. Solid-State Circuits Conf.*, San Francisco, CA, Feb. 1995, pp. 88–92.
[21] T. Gemmeke, M. Gansen, and T. G. Noll, "Implementation of scalable power and area efficient high-throughput Viterbi decoders," *IEEE J. Solid-State Circuits*, vol. 37, no. 7, pp. 941–948, Jul. 2002.
[22] M. A. Bickerstaff *et al.*, "A 24 Mb/s radix-4 logMAP turbo decoder for 3GPP-HSDPA mobile wireless," in *Dig. Tech. Papers IEEE Intl. Solid-State Circuits Conf.*, San Francisco, CA, Feb. 2003, pp. 150–151.
[23] R. Cypher and C. B. Shung, "Generalized trace back techniques for survivor memory management in the Viterbi algorithm," in *Proc. IEEE Global Telecommun. Conf.*, San Diego, Dec. 1990, pp. 1318–1322.
[24] G. Feygin and P. G. Gulak, "Architectural tradeoffs for survivor sequence memory management in Viterbi decoders," *IEEE Trans. Commun.*, vol. 41, no. 3, pp. 425–429, Mar. 1993.
[25] P. J. Black and T. H.-Y. Meng, "Hybrid survivor path architectures for Viterbi decoders," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, Minneapolis, MN, Apr. 1993, pp. 433–436.
[26] A. P. Hekstra, "An alternative to metric rescaling in Viterbi decoders," *IEEE Trans. Commun.*, vol. 37, no. 11, pp. 1220–1222, Nov. 1989.



Matthias Kamuf (S'02–M'06) was born in Karlsruhe, Germany, in 1973. He received the Dipl.-Ing. (FH) degree in communications engineering from Karlsruhe University of Applied Sciences in 1998 and the Dipl.-Ing. degree in electrical engineering from Universität Karlsruhe in 2001. In March 2007, he graduated as Ph.D. in circuit design from Lund University, Lund, Sweden.

He is presently with the Research Department at Ericsson Mobile Platforms, Lund, Sweden, working on baseband algorithms for next-generation wireless terminals and reconfigurable VLSI architectures for baseband processing. His main research interests are channel coding and algorithm-architecture trade-offs in the implementation of communication systems.

Viktor Öwall (M'91) received the M.Sc. and Ph.D. degrees in electrical engineering from Lund University, Lund, Sweden, in 1988 and 1994, respectively.

During 1995 to 1996, he joined the Electrical Engineering Department, the University of California at Los Angeles as a PostDoc, where he mainly worked in the field of multi-media simulations. Since 1996, he has been with the Department of Electrical and Information Technology, Lund University. His main research interest is in the field of digital hardware implementation, especially algorithms and architectures for wireless communication, image processing and biomedical applications. Current research projects include combining theoretical research with hardware implementation aspects in the areas of pacemakers, channel coding, video processing, and digital holography.

Dr. Öwall was an Associate Editor of the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS—II: ANALOG AND DIGITAL SIGNAL PROCESSING (from 2000–2002) and is currently Associate Editor of the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS—I: REGULAR PAPERS.

John B. Anderson (M'72–SM'82–F'87) was born in New York State in 1945. He received the B.S., M.S. and Ph.D. degrees in electrical engineering from Cornell University in 1967, 1969 and 1972. During 1972–1980 he was on the faculty of the Electrical and Computer Engineering Department at McMaster University in Canada, and during 1981–1998 he was Professor in the Electrical, Computer and Systems Engineering Department at Rensselaer Polytechnic Institute. Since 1998 he has held the Ericsson Chair in Digital Communication at Lund University, Sweden. He has held visiting professorships at the University of California, Berkeley (1978–1979), Chalmers University, Sweden (1987), Queen's University, Canada (1987), Deutsche Luft und Raumfahrt, Germany (1991–1992, 1995–1996) and Tech. University of Munich (1995–1996). His research work is in coding and communication algorithms, bandwidth-efficient coding, and the application of these to data transmission and compression. He has served widely as a consultant in these fields. Presently, he is Director of the Swedish Strategic Research Foundation Center for High Speed Wireless Communication at Lund.

Dr. Anderson was a member of the IEEE Information Theory Society Board of Governors during 1980–1987 and 2001–2006, serving as the Society's Vice-President (1983–1984) and President (1985). In 1983 and 2006, he was Co-Chair of the IEEE International Symposium on Information Theory. He served during the 1990s as chair of Research Initiation Grants for the IEEE Foundation. In the IEEE publications sphere, he served on the Publications Board of IEEE during 1989–1991 and 1994–1996. He was a member of the IEEE Press Board during 1993–2006 and during 1994–1996 was Editor-in-Chief of the Press. Since 1998, he has edited the IEEE Press book Series on Digital and Mobile Communication. He has also served as Associate Editor for the IEEE TRANSACTIONS ON INFORMATION THEORY (1980–1984) and as Guest Editor for the IEEE TRANSACTIONS ON COMMUNICATIONS on several occasions. He is author or coauthor of six textbooks, including most recently *Digital Transmission Engineering*, IEEE Press (2nd ed. 2005), *Coded Modulation Systems*, Plenum/Springer (2003), and *Understanding Information Theory*, IEEE Press (2005). He received the Humboldt Research Prize (Germany) in 1991. In 1996 he was elected Swedish National Visiting Chair in Information Technology. He received the IEEE Third Millennium Medal in 2000.