LUND UNIVERSITY

Design and Measurement of a Variable-Rate Viterbi Decoder in 130-nm Digital CMOS

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

Link to publication

Design and measurement of a variable-rate Viterbi decoder in 130-nm digital CMOS

Matthias Kamuf [a,b,*], Joachim Neves Rodrigues [a], John B. Anderson [a], Viktor Öwall [a]

[a] Dept. of Electrical and Information Technology, Lund University, 221 00 Lund, Sweden
[b] Ericsson AB, Scheelevägen 19 C, 223 63 Lund, Sweden

ABSTRACT

This paper discusses design and measurements of a flexible Viterbi decoder fabricated in 130-nm digital CMOS. Flexibility was incorporated by providing various code rates and modulation schemes to adjust to varying channel conditions. Based on previous trade-off studies, flexible building blocks were carefully designed to cause as little area penalty as possible. The chip runs down to a minimal core supply of 0.8 V. It turns out that striving for more modulation schemes is beneficial in terms of power consumption once the price is paid for accepting different code rates viz. radices in the trellis and survivor path units.

© 2009 Elsevier B.V. All rights reserved.

1. Introduction

In a mobile environment, transmission with variable data rates is required, that is, coding rate and modulation scheme have to be adjustable to adapt to varying channel conditions [1,2]. Consider high-rate wireless personal area networks (WPANs) [3], which provide short-range ad-hoc connectivity for mobile communication devices. A flexible channel decoding platform for such devices should be able to provide at least two decoding modes, one when good error-correcting capability is required at low SNR, and one supporting high data throughput if the channel is good. According to this requirement, IEEE 802.15.3 [3] suggests coded modulation schemes to gradually adjust data throughput. These schemes range from QPSK to 64-QAM and are all based on a symbol rate of 11 Mbaud/s.

As part of the mentioned standard, trellis-coded modulation (TCM) [4] enables transmitting information at high rates per Hertz of bandwidth. With this scheme, Ungerböck addressed the issue of bandwidth expansion in traditional coding schemes, which was due to the redundancy added by the channel code, by treating coding and modulation as a single entity. According to him, "redundancy" is now provided by using an expanded signal set and coding is done directly on the signal sequences.

The key idea of TCM is to successively split a so-called master constellation with $\mathscr{M}$ symbols into subsets of smaller constellations. The increase in minimum distance between constellation symbols of the same subset (same-subset minimum distance d_{ss}) reduces the overall error probability of the transmission. What is left is to provide a means to determine the sequence of subsets.

The inter-subset minimum distance d_{is} is a measure similar to the Hamming distance between two code sequences, where the two outputs are in different subset sequences. In case of convolutional coding, the number of differing bit positions determines the code's free distance d_f. Now, the Euclidean distance between the signal sequences has to be evaluated. Since the performance of the TCM code is determined by $\min\{d_{is}, d_{ss}\}$, the goal is to design a clever enough sequencer that achieves $d_{is} \geqslant d_{ss}$.

The sequencing of the subsets is done by a tapped shift register, a convolutional encoder. In the context of TCM, this entity is called subset selector. Fig. 1 shows a rate-R TCM encoder. It takes in R bits per symbol time. b bits are input to the subset selector that puts out $c > b$ bits z_i that determine the subset to be used for that symbol time. Note that branches in a trellis diagram now carry subsets, not code symbols as in the case of convolutional coding.

The remaining $R - b$ bits, sometimes called "uncoded" bits, are used to choose the signal sent in a subset. This implies that the trellis diagram has parallel transitions. As a consequence, decoding complexity is increased since the most likely of these subset signals has to be determined before one can calculate the branch transition probabilities. This process is called subset decoding.

TCM is most efficient for higher (quadrature) constellations beyond QPSK, which carry more than two data symbols per two-dimensional channel use. The subset selectors of the TCM codes in [3] are rate 1/2 for QPSK and rate 2/3 for 16-QAM, 32-CR, and 64-QAM constellations. For QPSK, one data bit is transmitted per channel use, and the data rate becomes 11 Mbit/s. The higher modulations cover data rates in multiples of the symbol rate. Trellis-coded 64-QAM is the highest constellation considered in the standard and carries five data bits per channel use. Thus, the maximum data rate is 55 Mbit/s. Generally, the number of data bits per symbol time (data or transmission rate) is
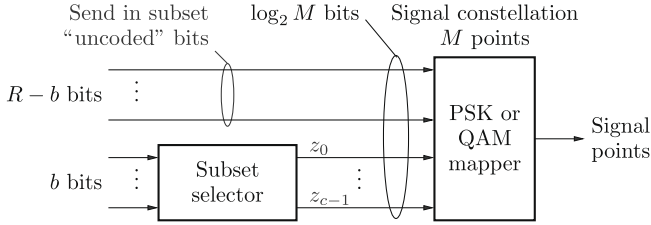
Fig. 1. A generic rate-R TCM encoder.

$$R = \begin{cases} \log_2 \mathscr{M} \cdot R_c & \text{for convolutional codes,} \\ \log_2 \mathscr{M} - 1 & \text{for TCM codes,} \end{cases} \quad (1)$$

where $\mathscr{M}$ is the number of constellation symbols.

To emphasize the importance of TCM in the high signal-to-noise ratio (SNR) region, consider Fig. 2. It compares rate $R = 1, 3, 5$ transmission schemes using QPSK, 16-QAM, and 64-QAM constellations, respectively. The TCM subset selectors are rate $R_c = 1/2$ for QPSK and $R_c = 2/3$ for the two multi-level constellations. The competing convolutional codes are rate $R_c = 1/2, 3/4$, and $5/6$. They use the same constellations as the TCM schemes, albeit with Gray-mapping. Here, the higher rate codes are achieved by puncturing the rate $R_c = 1/2$ code. Corresponding puncturing patterns are found in [5]. In all cases, the encoders have eight states. For the multi-level constellations, the gain at BER of 10^{-5} of TCM compared to the Gray-mapped system is around 1.3 dB. However, in the QPSK case with rate 1/2 coding, that is, for low transmission rates at low SNRs, the TCM code is about 0.3 dB weaker, which depends on the different coding polynomials employed for the two encoders.

In this example, BPSK or QPSK are used together with rate $1/c$ convolutional codes, or punctured codes derived thereof. The trellis diagram of these codes can be decomposed into a radix-2 (R2) butterfly state interconnect structure. For TCM, the most practical codes used together with two-dimensional modulation schemes appear for $b = 2$. Puncturing, however, is not applicable if code performance is to be fully maintained. This degradation stems from altering the minimum inter-subset distance d_{is} [6]. Thus, the trellis of the TCM subset selectors consists of radix-4 (R4) butterflies.

To summarize these considerations, the flexible channel decoding architecture has to be tailored to efficiently process both R2 and R4 butterflies. Both modes should use the same computational kernel to limit overhead in both area and power consumption. A
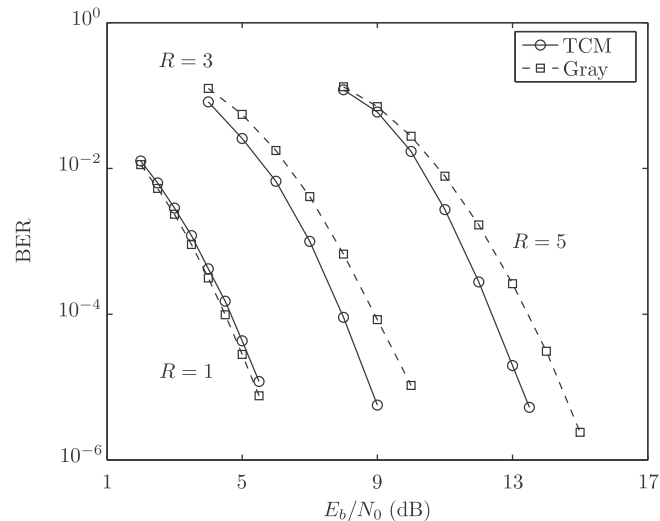

Fig. 2. Performance comparison of rate-R transmission schemes using TCM or convolutional coding with Gray-mapped constellations.
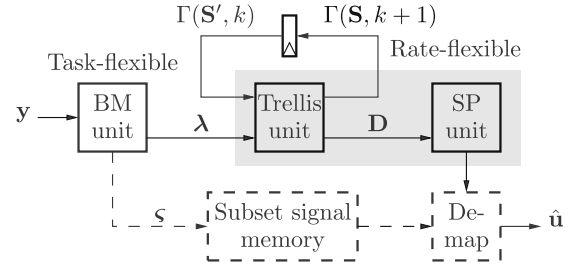

Fig. 3. Block diagram of a flexible Viterbi decoder. Additional parts needed for decoding of TCM codes are dashed.

more general design objective could be stated as follows: increase flexibility with as little sacrifice as possible in area, throughput, and power consumption.

Maximum-likelihood decoding is provided by the Viterbi algorithm (VA) [7]. We denote by m the number of memory elements in the encoder which is excited by b bits per time step. The number of trellis states is $N = 2^m$ and there are 2^b branches per node that connect these states. Shown in Fig. 3 is a principal architecture for a flexible Viterbi decoder. Let us revisit the three main processing blocks of this decoder, that is, BM, trellis, and SP units, from a flexibility perspective.

Based on the demodulated channel values $\mathbf{y}$, the branch metric (BM) unit provides measures of likelihood λ for the transitions in a trellis stage. This unit 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 convolutional codes, TCM codes require an additional subset decoder as discussed earlier.

These BMs 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 suboptimal 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, which is determined by the encoder. The architecture of this unit depends on the code rate and number of states in a trellis diagram.

The trellis unit produces 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 data bits that caused the transitions. Depending on the algorithm used for the SP unit, the architecture becomes more or less related to the number of bits b and states N per trellis stage. For example, the register-exchange algorithm requires the trellis to be directly mapped onto hardware [8], which gives a stronger connection to b and N.

Additionally, in case of TCM, the most likely transmitted signals ς for all subsets have to be stored in the subset signal memory. These signals, together with the reconstructed subset sequence from the SP unit, lead to the final decoded data sequence $\hat{\mathbf{u}}$.

We now begin with an overview and a classification of other flexible decoders to point out where in the application and performance space our design is located. Then, our flexible architecture is presented in Section 3. Design and silicon implementation, together with measurements of the fabricated chip that lead to an overall cost estimation of flexibility, are described in Section 4. A previous evaluation that laid the foundation for the chip [9] solely relied on simulated data. Finally, Section 5 discusses an alternative architecture and evaluates its hardware cost.

2. Classification of flexible trellis decoders

Several approaches have been made to incorporate flexibility in the design of trellis decoders. We divide these attempts into the

following categories: the first two (*m*- and algorithm-flexible) are expected to operate in the low energy region and, therefore, employ small constellations such as BPSK or QPSK. Coding is based on rate $1/c$ convolutional codes, including punctured or concatenated versions thereof. The last category (bandwidth-flexible) inherently supports larger constellations and different coding schemes, thus facing other design challenges, as mentioned in the previous section for the case of TCM.

2.1. m-flexible solutions

These approaches use one decoding algorithm and provide flexible error correction by varying the encoder memory *m*. For example, Chadha [10], Zhu [11], and Hocevar [12] designed flexible VA-based architectures.

Chadha's implementation provides a fully parallel solution (up to $m_{max} = 6$) and shuts down unnecessary parts when processing trellises with fewer states. The extra hardware spent in the flexible designs is compared to a fixed design with the same m_{max}. This overhead is at most 2.9%, which is not surprising since it mainly accounts for shut-down logic and routing resources. What is missing is an evaluation of the provided flexibility compared to fixed designs with $m < m_{max}$. In this case, an increasing relative overhead should be encountered as *m* decreases. Supported code rates are 1/2 and 1/3. Code rate is not a critical design parameter when used with antipodal constellations such as BPSK and QPSK since the calculation of distances to the 2^c code sequences is very simple [13]. However, the design does not explicitly provide puncturing resources to enable high-rate transmission.

Zhu's reconfigurable decoder ($R_c = 1/2, m = 6,\dots,9$) works in a folded manner; that is, starting from a trellis unit with eight ACS units working in parallel, the processing is carried out time-multiplexed. According to the sequential access scheme of the SMs, 5-level pipelining can be introduced in the feedback loop of the ACS units. Reconfigurability is achieved by 4×4 switches that shuffle the SMs between the ACS units and the global SM memory. These switches are steered by a controller that provides the necessary schedule for a given *m*. This controller is probably the most complex part of the implementation since access patterns change in every iteration and for every encoder memory.

Hocevar's design is a DSP coprocessor, which supports a variety of code rates that are achieved by puncturing the basic 1/2, 1/3, and 1/4 convolutional codes. *m* is variable from 4 to 8, and 16 states can be processed in parallel. Generally, a processor's flexibility is inherently larger than a tailored design such as Chadha's. As discussed in [14], the price for this flexibility is paid by throughput degradation.

A different class of reconfigurability is the dynamically adaptive Viterbi decoder investigated by Tessier [15]. It is an FPGA-based approach that can be reconfigured externally to cope with varying channel SNR. The designs that can be loaded range from $m = 3,\dots,13$. Power is saved compared to a static decoder by adaptively updating only a predefined portion of the trellis with paths that have the least cumulative metrics.

A flexible max-log-MAP decoder with $m_{max} = 4$ is presented in [16]. The evaluation is carried out similar to [10], and hence the information about the introduced overhead does not cover aspects missing in Chadha's investigation.

2.2. Algorithm-flexible solutions

A straightforward combination is the (soft-output) VA together with the (max-)log-MAP algorithm since they share the main processing engine, the ACS operation. For example, Bickerstaff et al. [17] provide 256-state Viterbi and 8-state log-MAP decoding. The trellis block processes eight states in parallel, that is, the Viterbi

decoding is carried out time-multiplexed. Since this design is to be (commercially) used in third generation mobile systems, it includes several communication interfaces and an evaluation of the cost of flexibility was not of main interest.

In Cavalloro's work [18] the combination of VA with an augmented soft-output unit is investigated. Encoder memory is variable up to $m_{max} = 8$. The approach is in principle based on the work of Chadha [10], that is, the contribution of flexibility lies mainly in shut-down logic and routing resources.

2.3. Bandwidth-flexible solutions

A Viterbi processor for TCM codes is presented by Lou in [19]. It is widely programmable and executes the main decoding operations sequentially. Flexibility is achieved by look-up tables and some dedicated computational blocks. 32 states and both R2 and R4 processing are supported. Codes with a maximum of eight subsets are allowed and two- or four-dimensional symbols can be processed.

Miyauchi et al. [20] describe a fully integrated dedicated soft-input soft-output processor, whose focus is on iterative decoding. It has many features such as arbitrary coding polynomials, interleaving patterns, and constellation configurations. Different classes of coding approaches are supported, parallel/serial concatenated convolutional codes, turbo TCM, and serial concatenated TCM. The highest constellation considered is 8-PSK. Design challenges in this approach are mainly concerned with the incorporation of R4-processing to log-MAP decoding.

Our flexible Viterbi decoder [9], also belongs to the bandwidth-flexible class. It covers a wider range of transmission rates in order to adapt to both low- and high-energy scenarios.

2.4. Performance evaluation

The different approaches are compared in an energy–bandwidth sense. Note that energy here is the required received energy per bit E_b at the input to the decoder to achieve a certain bit error rate (BER), not the energy consumed by these implementations. Fig. 4 shows some implemented flexible trellis decoders and their energy–bandwidth performance to achieve a BER of 10^{-5} in the AWGN channel. It is assumed that an AWGN channel use with
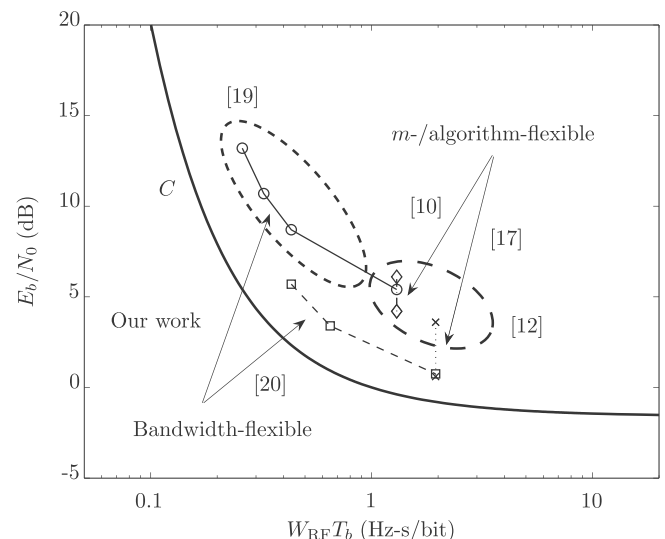

Fig. 4. Energy–bandwidth performance of some flexible trellis decoders for BER of 10^{-5}. Shannon AWGN capacity *C* is drawn bold for comparison. Designs become more complex to implement the closer to *C* they are, that is [20] is more complex than ours.

two independent dimensions occurs every symbol time $T_s = T_b \cdot R$, where T_b is time per data bit and the number of data bits per channel use (transmission rate) is determined by (1). RF bandwidth $W_{RF} = 1.3/T_s$ is normalized to T_b and includes excess bandwidth introduced by 30% root-raised-cosine pulses. In other words, the x-axis in Fig. 4 shows the inverse of the spectral efficiency of the transmission system. That is, a system's potential throughput grows as $W_{RF}T_b$ becomes smaller. The Shannon AWGN capacity C is shown for comparison. Flexible decoders to decode convolutional codes that principally support larger constellations beyond QPSK but do not provide a (soft-output) demapper [21] belong in a bandwidth sense to QPSK-systems. Also, punctured codes to increase bandwidth efficiency on a small scale (up to $R = \log_2 \mathcal{M}$ for uncoded transmission) are only considered if the necessary hardware is provided.

From Fig. 4 note that m- and algorithm-flexible designs provide no increased throughput as the channel SNR improves. They just trade required energy for bandwidth, which is indicated by vertical lines. Throughput can only be varied on a small scale. One solution is to employ higher constellations together with (punctured) convolutional codes to gradually increase data rates. However, compared to TCM, which was intended for higher constellations, these systems are not as energy-efficient for the same BER, throughput, and complexity.

It is noteworthy that systems become more complex to implement the closer to capacity they get. Consider Miyauchi's design [20] in Fig. 4, which apparently provides a good energy-bandwidth trade-off with help of iterative decoding. One must bear in mind, however, that iterative decoding schemes run multiple times over a trellis, increasing latency and raw computational cost per decoded bit. That is, for low-power low-cost applications as in WPAN, this design is certainly overdesigned. The flexible Viterbi decoder described in our work is a lower-complexity solution that adjusts to varying channel conditions by providing several transmission rates using different constellations.

Programmable trellis processors such as [12 and 19] provide the highest flexibility. In Fig. 4, these systems would realize design points that are bound by an ellipse. However, this flexibility degrades processing speed and power consumption by several orders of magnitude compared to dedicated solutions [14].

There could be many more dimensions added in Fig. 4, for example, latency, computational complexity, energy consumption per decoded bit, and so on. Such measures ultimately give an overall cost per decoded bit and depending on what is crucial for a certain application, a design choice becomes evident.

3. The flexible architecture

This section briefly presents the architecture of our flexible Viterbi decoder. A more in-depth coverage of the design considerations for the different building blocks and their hardware cost is found in [9].
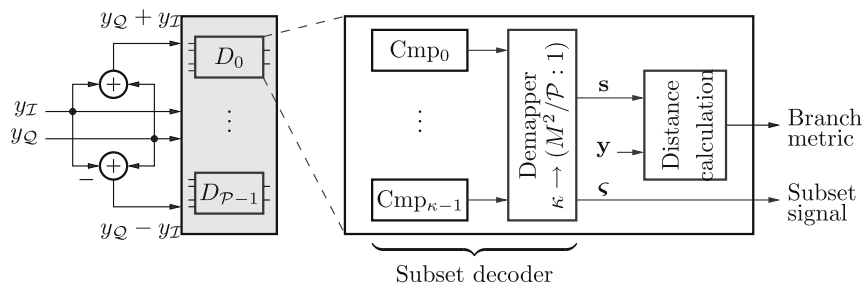
3.1. Branch metric unit

The branch metric (BM) unit shown in Fig. 5 provides measures of likelihood λ for transitions in a trellis stage. In an AWGN channel, the optimal distance measure is the squared Euclidean distance between received channel symbol $\mathbf{y} = (y_{\mathscr{I}}, y_{\mathscr{Q}})$ and constellation symbol $\mathbf{s}$, that is, $|\mathbf{y} - \mathbf{s}|^2$. For antipodal signaling this expression is reformulated to additions and subtractions of the channel symbols [13]. Channel symbols are quantized with $q = 3$ bits without causing much performance degradation [9]. Higher order modulations such as 16-QAM and 64-QAM, however, need to employ the squaring operation if optimality is to be maintained. A low-complexity workaround is to use an absolute distance measure, that is, $|\mathbf{y} - \mathbf{s}|$. Besides, more bits per channel symbol are needed to minimize BER performance degradation; see Table 1 for the expected performance losses. Note that the wordlength q of the channel values is variable (shaded in gray in the table) since this reduces the wordlength of the branch metrics [9].

Ontop of the distance calculations, TCM codes require an additional subset decoder D. The calculations needed for subset decoding of the $\mathscr{P}$ subsets, $y_{\mathscr{Q}} - y_{\mathscr{I}}$ and $y_{\mathscr{Q}} + y_{\mathscr{I}}$, can be reused in case of 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. A single subset decoder D consists of comparators, a demapper, and the actual distance calculation. The complexity in such a subset decoder stems from the use of higher order modulations, resulting in more slicing operations to find the most likely subset signal. This signal has to be stored for all subsets in a subset signal memory. Together with the reconstructed subset sequence from the SP unit, the final decoded data sequence $\hat{\mathbf{u}}$ can be established.

3.2. Trellis unit

The architecture of a trellis unit depends on the code rate R_c and number of states N. Since trellis diagrams of rate 1/2 and 2/3 encoders consist of R2 and R4 butterflies, that is, we need a rate-flexible trellis unit.

The design objective is the following: given a fixed R2 feedback network for the SMs, how can R4 butterflies be efficiently mapped onto this architecture, so the existing interconnections can be reused? In this section, the issue of rate flexibility in the trellis unit is discussed and a framework is derived for emulating R4 butterflies by means of R2 butterflies. Note that almost all practical trellis codes are based on either R2 or R4 butterflies. Therefore, the investigated architecture also extends to other codes.

3.2.1. Architectural issues

The base architecture for the flexible trellis unit is an R2 architecture for the systematic 8-state rate 1/2 convolutional code with tap set $(\mathbf{g}_0, \mathbf{g}_1) = (54, 74)$ realized in controller form, where $\mathbf{g}_0$ are the feedback taps.

Fig. 5. Architecture of the BM unit for the flexible decoder. The gray part is the additional hardware required due to the use of higher constellations and TCM. κ is the number of decision boundaries to determine a subset point in any one of the $\mathscr{P}$ subsets.

Table 1
Loss in E_b/N_0 for BER of 10^{-5} for uniform symbol quantization with q bits and absolute distances for the branch metrics. Column $q = \infty$ shows the required E_b/N_0 with unquantized inputs and Euclidean distance.

q	∞	7	6	5	4	3
16-QAM	8.7	≈ 0	≈ 0	0.15	0.4	1.4
64-QAM	13.2	0.05	0.3	1.15	n/a	n/a
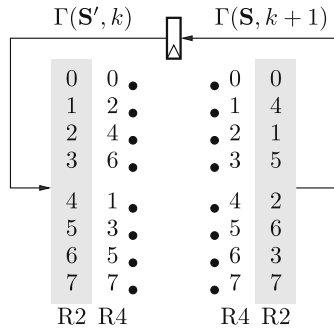
Fig. 6. Feedback connections for R2 and R4 trellis processing.
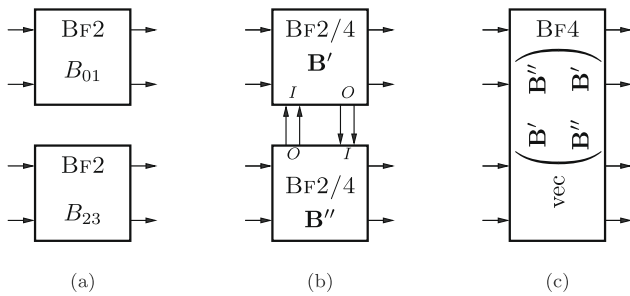
Fig. 7. Butterfly units that are instantiated inside a processing element. A setup as in (a) supports only R2 processing, setups (b) and (c) are rate-flexible.

The TCM subset selector with tap set $(\mathbf{h}_0, \mathbf{h}_1, \mathbf{h}_2) = (15, 02, 04)$ is realized in observer form ($\mathbf{h}_0$ are the feedback taps) since this maintains the number of memory elements of the underlying systematic code if $b > 1$ [22]. However, state transitions in the observer form depend on the tap set. That is, the feedback network

has to be flexible if one wants to process both R2- and R4-based codes on a single R2 architecture. This can be done by introducing additional routing resources that modify the permutation in R4 mode to feed back the state metrics in correct order in both processing modes. In this design example, though, the state transitions of the TCM subset selector allow the reuse of the trellis feedback connections of the binary code. Fig. 6 shows the interconnection structure of a trellis stage for the considered encoders. It is seen that the feedback connections for both R2 and R4 architectures are the same, for example, $\Gamma(2, k+1)$ in R4 mode is fed back along the same connection as $\Gamma(1, k+1)$ in R2 mode.

Consider a trellis unit that updates N states in parallel by means of $N/4$ processing elements (PEs), each consuming and producing four state metrics. A PE is configured with butterfly (BF) units of different radices as in Fig. 7. In Fig. 7c, vec means the stacking of a matrix columnwise. Either two BF2 units, two rate-flexible BF2/4 units, or one BF4 unit are employed. Note that the BF4-based architecture can also be configured for rate-flexible processing, whereas a BF2-based design is solely intended for R2 processing and is not discussed further. That is, the basic PEs are ACS units that consume 4 inputs at a time (4-way ACS units) and thus the cumulation is done in one step.

3.2.2. R2-based approach

Consider a rate-flexible architecture using BF2/4, which is drawn in Fig. 8. Whereas R2 processing is done in one clock cycle, R4 processing is time-multiplexed in the rate-flexible trellis unit. All partial survivors are calculated during two cycles, and in the third cycle the final update takes place. The partial survivors needed for the final compare-select (CS) are calculated in different butterfly units and have to be stored temporarily. Appropriate routing (PERM) for the final CS is according to the required ordering of the updated SMs $\Gamma(\mathbf{S}, k+1)$, where $\mathbf{S}$ is the set of states in the trellis diagram. Here, the partial survivors are brought together by means of I/O channels between adjacent butterfly units that belong to the same pair i (here, both code trellises have 8 states and thus there are two such pairs $i = 0, 1$ processing four states each).

The arithmetic components in this rate-flexible butterfly unit are identical to the ones in a conventional R2 butterfly unit. To cope with a decomposed (time-multiplexed) R4 butterfly, routing resources are provided to distribute the partial survivors as dictated by the BM distribution (reflected in $\mathbf{B}'$ and $\mathbf{B}''$) and the state transitions. In total the rate-flexible butterfly unit only adds six
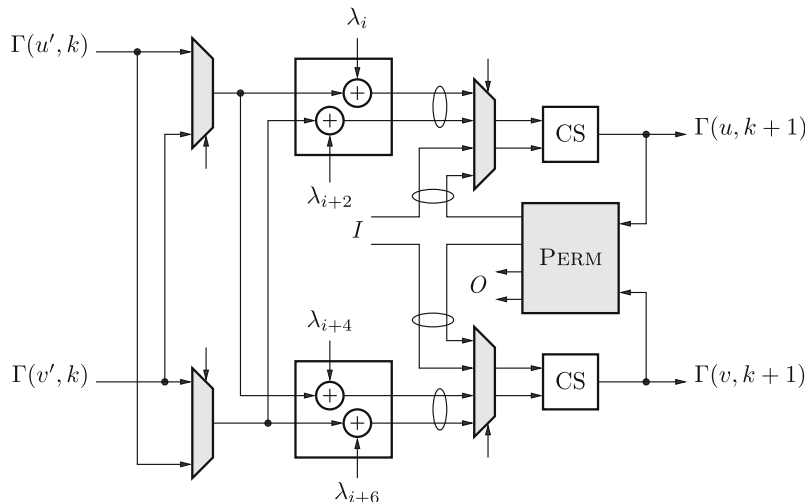
Fig. 8. The rate-flexible butterfly unit BF2/4. An R4 butterfly is updated in three clock cycles. The shaded blocks are the overhead compared to an R2 butterfly unit BF2. The routing block PERM consists of a tapped delay line.

2:1 multiplexers (MUXes) and two registers on top of an R2 butterfly unit, and there is no arithmetic overhead.

3.2.3. R4-based approach

The presented flexible R2-based approach is now compared to an R4 architecture, which is based on B$_{F4}$ units that utilize four 4-way ACS units as in Fig. 7c. To account for the intended use in a rate-flexible system, similar control mechanisms have to be provided as in the R2-based approach. Hence, a straightforward two-level-CS implementation is considered. Depending on the desired throughput, a butterfly can be updated in one or two clock cycles, which gives the well-known area–delay trade-off. Here, a two-cycle update is employed since this maintains the critical path of the R2-based approach and one CS unit can be reused.

Fig. 9 shows the flexible 4-way ACS unit. In R4 mode, two partial survivors are captured in the first cycle. The global SM register in the upper path now carries the temporary survivor from either state u' or v' and the shaded register the one from either state w' or x'. In the second cycle, these survivors are compared to yield the final state metric at $k + 1$. In R2 mode, only the upper ACS path is utilized and to be equally power-efficient, one needs to prevent switching activity in the lower ACS path. This is done by guarding the inputs of the adders with AND-gates, which is illustrated by the gray shading. The signal $\overline{r2}/r4$, which determines the processing mode, controls whether the addition block is enabled or not. Compared to a conventional 2-way ACS unit, two adders, a CS unit, a register, and a 4:2 MUX are counted as overhead.

3.3. Evaluation of synthesized trellis blocks

To show the effects of the architectural considerations in an actual implementation, the trellis blocks are synthesized using a design kit from Faraday for the United Microelectronics Company (UMC) 0.13 μm digital CMOS process. Evaluations apply to synthesized cell area, where different throughput requirements are put as design constraints.

Fig. 10 shows the required cell area for synthesized trellis blocks that process R2 and R4 butterflies. Here, $t_{k→k+1}$ denotes processing time for a trellis stage from k to $k + 1$. The B$_{F2/4}$ architecture takes three cycles for an R4 update, whereas the B$_{F4}$-based one only needs two cycles. For an R2 update, both architectures need one clock cycle. It is seen that the B$_{F2/4}$ architecture becomes somewhat larger than the B$_{F4}$ approach as the requirement on $t_{k→k+1}$ in R4 mode becomes tighter, that is, less than about 4.5 ns. However, the provided throughput at this stage is beyond the speed requirement of considered applications, for example, the high data-rate WPANs discussed in the introduction. In the figure, this means that
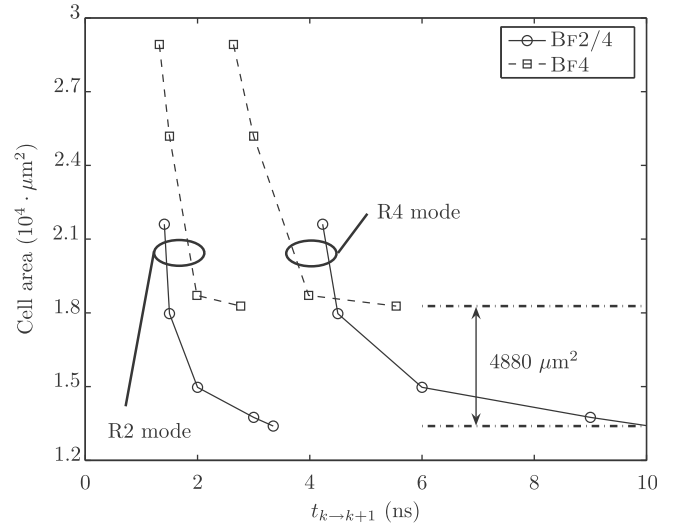

Fig. 10. Cell area versus time for a decoding stage in R2 or R4 mode for architectures based on different radices.

the actual design space to be considered is to the right hand side. Here, the B$_{F2/4}$ architecture is more suitable due to the lower area requirement of about 27% (4880 μm²). Furthermore, this approach already provides routing resources (P$_{ERM}$) to support a wider range of codes, that is, the four state metrics belonging to an R4 butterfly can be shuffled by P$_{ERM}$ in any order to maintain compatibility to the feedback connections of the basic R2 architecture. Considering R2 processing, the B$_{F2/4}$ architecture is better suited even down to a $t_{k→k+1}$ of about 1.4 ns. Therefore, the trellis unit in our implementation is based on R2-based butterfly units of type B$_{F2/4}$.

3.4. Survivor path unit

The survivor bits from the trellis unit are processed by the SP unit shown in Fig. 11 to reconstruct the transmitted data bits. The register-exchange (RE) approach is chosen since the number of states is low. Since an additional subset signal memory is needed for TCM, the least overhead is introduced since the decoding latency is the lowest compared to trace-back architectures, which have a least twice the latency [8]. Additionally, for TCM a demapper needs to be employed that delivers the most likely subset signal at a certain time. This is a MUX which chooses a subset signal depending on the decoded subset number from the SP unit. For convolutional decoding, b information bits are decoded every cycle. In case of TCM, however, $b + 1$ must be decoded per trellis stage since the subset number consists of $b + 1$ bits. Hence, the RE algorithm must store in total $(b + 1)NL$ bits, where L is the decoding depth.

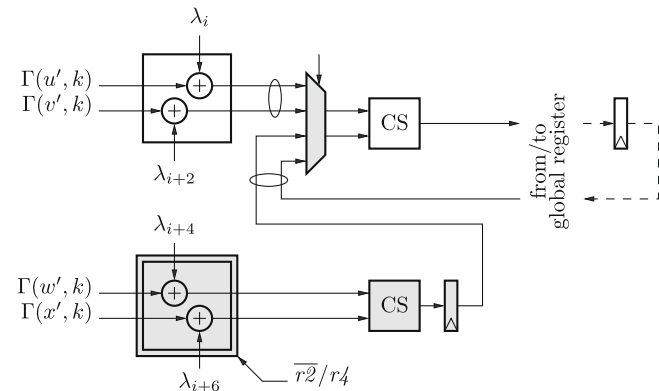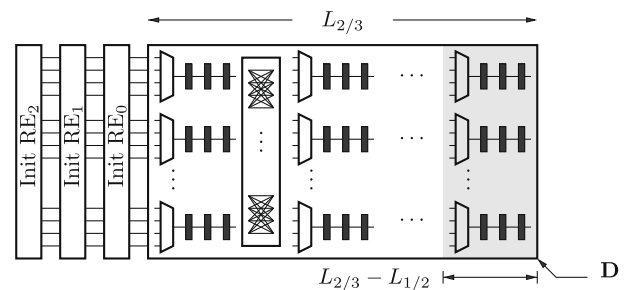**Fig. 9.** A flexible 4-way ACS unit for use in B$_{F4}$ of Fig. 7c. Four such units are needed for an R4 butterfly to be updated in two clock cycles. The shaded blocks are the overhead compared to a 2-way ACS unit.

Fig. 11. An RE architecture to suit the combined convolutional and TCM decoder. In order to save hardware, the architecture is matched to the throughput of the trellis unit.

The complexity of the RE network is lowered by matching it to the throughput of the trellis unit. Recall that R4 processing takes three clock cycles and thus the RE update can also be carried out sequentially; that is, the registers are placed in series such that three cycles are needed to update the complete survivor sequence. The hardware requirement is drastically lowered compared to a straightforward parallel approach since 66% of the MUXes and interconnections become obsolete, and utilization for both modes is effectively increased.

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 solutions 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 have easier timing and testability, the first approach is applied.

Note that rate 1/2 and 2/3 codes theoretically require different L, hence the distinction in Fig. 11, where the gray parts can be disabled during rate 1/2 processing. However, following the simulations in [9], 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 **D** and, in case of R4 processing, state numbers.

4. Silicon implementation and measurements

The chip was modeled in VHDL at register-transfer level (RTL) and taken through a flow using Synopsys Design Compiler for synthesis and Cadence Encounter for routing. A standard cell library from Faraday is used for an eight metal layer, 130-nm digital CMOS process from UMC. The RTL and gate level netlists are verified against test vectors generated from a MATLAB fixed-point model. Post-layout timing is verified using Synopsys Prime Time with back-annotated parasitics.

Fig. 12 shows the layout of the routed chip. It consists of two designs. The design to the left is called ONE, which provides a transmission rate of $R = 1$ using QPSK and an 8-state rate 1/2 systematic convolutional code. This is a fixed design tailored for operation in low-SNR regions and serves as reference to which the flexible design is compared to. The flexible design, named FIVE, additionally provides $R = 3, 5$ transmission rates using TCM with 16-QAM and 64-QAM modulation schemes. Compared to a fixed design of $R = 5$, there is no hardware overhead, that is, a higher rate design always includes the lower rate ones. This is due to the sharing of the BM calculations.

The chip is pad-limited due to test purposes and measures 1.44 mm². Designs ONE and FIVE are placed on the same die with separate V_{dd} to measure their power consumption independently. The cell area of ONE is 4.1 kGates (NAND2-equivalent), and for FIVE it is 19.2 kGates (15.7 kGates logic, 3.5 kGates memory). Design constraints are chosen such that the implementation is in the flat part of the area–delay curve. The critical path for the designs lies in the trellis unit (1.65 ns and 1.98 ns, respectively, according to post-synthesis reports).

In TCM mode, design FIVE achieves a symbol rate of 168 Mbaud/s, which gives a maximum throughput of 504 Mbit/s and 840 Mbit/s using $R = 3$ and $R = 5$ configurations. Design ONE provides a maximum 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, all these throughputs are higher than specified in [3]. Thus, the supply voltage can be lowered to save energy. Note that the throughput numbers apply to the circuit level. When considering data rates, one also has to keep in mind the
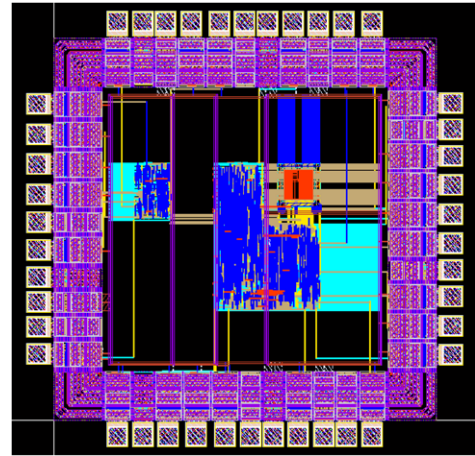

Fig. 12. Layout of the routed chip. Designs ONE and FIVE are shown on the left and right side, respectively. Row utilization is 80% in both implementations.

underlying modulation. As an example, from a transmission perspective, for ONE to achieve the highest throughput requires a transmission bandwidth of 606 MHz, whereas 64-QAM with FIVE only needs $(606/840) \cdot 168 = 121$ MHz to achieve the same data throughput.

The fabricated chip is verified up to a clock frequency of 160 MHz. This limit is due to the used pattern generator and logic analyzer. For the measurements, the core supply voltage V_{dd} was varied between the nominal value of 1.2 V down to 0.8 V, below which the voltage swings became too low for the pads to work properly. At the lower supply limit and highest clock frequency, the circuits still work properly such that more advanced investigations on energy-speed trade-offs were not carried out.

The presented designs' estimated and measured power consumption are shown in Table 2. Power estimation (numbers in parentheses) is carried out on the synthesized netlists, back-annotated with state- and path-dependent toggle information from a simulation run. Consider the following comparison scenario: convolutional decoding using either a fixed (ONE) or the flexible design FIVE to find out how much power may be sacrifice for a certain flexibility. This comparison provides a measure of the initial cost of flexibility.

To determine the error from the pre-layout power estimation figures compared to silicon measurements, we evaluate the impact of static power consumption P_{stat} to be able to extrapolate the estimated values onto the measured values according to $P_{dyn} \propto f_{clk}$. The measured leakage current at the nominal core supply of 1.2 V was 50 µA, which gives $P_{stat} = 60$ µW. Static power consumption is ignored since the worst case contribution appears for design ONE at 3.7 mW (see rectangular in Fig. 13) and accounts for about 1.6%. At 0.8 V, the leakage current dropped to 14 µA (equiv. $P_{stat} = 11$ µW). The total power consumption for design ONE is 1.5 mW as in Table 2, that is, the contribution of static power dropped to 0.7%. Since P_{stat} in all cases is at least two–three orders of magnitude lower than P_{dyn}, the extrapolation of the values in Table 2 results in estimation errors of 10% and 18% for designs ONE and FIVE.

Table 2
Fabricated designs and their modulation schemes. Power consumption measurements are derived at $V_{dd} = 0.8$ V and $f_{clk} = 160$ MHz. For comparison, normalized numbers of an earlier power estimation are shown in parentheses.

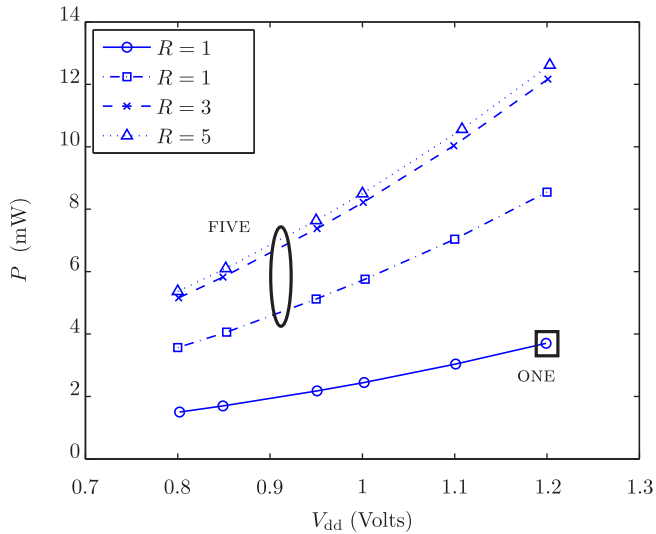| Design	$\mathcal{M}$	R			Area (kGates)	$P(R)$ (mW)		
ONE	4	1	–	–	4.1	1.5 (1.4)	–	–
FIVE	64	1	3	5	19.2	3.5 (2.9)	5.1 (4.2)	5.3 (4.3)

Fig. 13. Power consumption of designs ONE and FIVE at $f_{\mathrm{clk}} = 160$ MHz.

Power consumption is sacrificed for flexibility. For design FIVE, 2.3 times more power is consumed for $R = 1$ processing. From an earlier power estimation, comparing a design with 16-QAM as highest modulation and FIVE, the latter consumes 4% and 9.7% more power in $R = 1$ and $R = 3$ modes, respectively. Furthermore, $R = 5$ in design FIVE requires an extra 4% power compared to $R = 3$, a low number considering the additional data rate provided.

Fig. 13 shows plots of $P = f(V_{\mathrm{dd}})$ for the two designs run at different R. It confirms the initial gap between fixed and flexible design, apparently when looking at the curves $R = 1$. However, once this penalty is paid, the incremental power to be spent for higher transmission rates is small.

If QPSK is often used and power consumption is a critical factor for the application, it makes sense to accept the additional fixed design. Otherwise, one flexible design that covers all transmission rates is sufficient.

In order to conduct an overall fair comparison, one further needs to introduce larger modulation schemes also for the R2-design ONE, which to date is solely based on a rate 1/2 convolutional code using QPSK. Then the R2- and R2/R4-design can be compared based on equal transmission rate, and the contribution of task flexibility (different modulations and symbol mapping) in the BM unit related to (code) rate flexibility of trellis and SP units can be evaluated. From cell area of the different building block in the current design and their evolution using different modulations, task flexibility has a larger impact on an implementation than rate flexibility.

Comparisons to designs presented in Section 2 are difficult due to the mentioned different application areas and technologies. Chadha's [10] implementation for $m = 4$ is comparable to our design in terms of trellis complexity viz. number of branches per trellis stage. However, his implementation is for an FPGA, thus only gate count (23.5 kGates) serves as comparison measure.

5. An alternative approach

As far as throughput is concerned, the presented design is originally limited by the trellis unit, which processes an R4 butterfly in three clock cycles. In the following, throughput and size of an approach where an R4 butterfly is updated in one clock cycle are estimated.

To begin with, throughput is not only improved for R4 processing but also in R2 mode since two R2 stages can be collapsed into

one R4 stage, that is, two decoded bits per stage are put out. A reported speed-up for such an architecture compared to R2 processing is 1.7 [23]. This implementation is based on full-custom datapaths. A standard cell implementation only achieved a speed-up of 1.26 [14]. Hence, mode $R = 1$ provides a throughput of at least 767 Mbit/s, working at a symbol rate of 383 Mbaud/s. For $R = 3, 5$, a throughput of 1.1 and 1.9 Gbit/s is estimated, respectively.

The size of the R2 trellis unit from design ONE grows by a factor of 3.8 using the straightforward (not full-custom) 6-comparator approach for a 4-way ACS operation [14]. This trellis unit is now utilized in all designs, both fixed and flexible. However, the trellis collapsing implies that feedback connections are not directly reusable anymore. The inherent flexibility provided by PERM in BF2/4 is achieved by an additional routing stage instead, which slightly lowers the previously estimated throughputs.

To take advantage of the improved processing speed, the BM unit in the flexible (TCM) designs has to carry out subset decoding in one clock cycle, too. That is, eight subset decoders must work in parallel, instead of four that work in an interleaved manner such that only half the constellation points must be stored. This is not possible in the alternative approach, and thus there is a more than twofold area increase. For the SP unit, the update also has to be carried out in parallel. That is, the unit in Fig. 11 has to be unfolded [9], which increases its size by roughly 2.5 times.

Based on the preceding considerations, the sizes of the different processing units can be scaled by the mentioned factors: the size of design ONE is expected to grow by a factor of 2.5, whereas the size of flexible design FIVE increases by about 2.2 times.

Recall, though, that throughput is not the major design issue in our work. The envisioned applications never utilize the provided processing power. Furthermore, considering power consumption of shrinking process technologies, where static power consumption surpasses dynamic power consumption, this alternative approach becomes even less feasible compared to an R2-based architecture.

6. Conclusions

We presented a design of a flexible Viterbi decoder fabricated in 130-nm digital CMOS. To adapt to varying channel conditions, the flexibility applies to both code rate and modulation schemes. The cost of this flexibility is characterized on application and architectural level, which initially steered the design choices, and most importantly by measurements on the fabricated chip. Having accepted the initial impact of rate flexibility when comparing a fixed and a flexible design at their lowest transmission rate, one should incorporate more modulations since the additional cost in terms of power consumption becomes smaller.

References

[1] J. Karaoğuz, High-rate wireless personal area networks, IEEE Commun. Mag. 39 (12) (2001) 96–102.

[2] Z. Ding, 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.

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

[4] G. Ungerböck, Trellis-coded modulation with redundant signal sets, IEEE Commun. Mag. 25 (2) (1987) 5–21.

[5] Y. Yasuda, K. Kashiki, Y. Hirata, High-rate punctured convolutional codes for soft decision Viterbi decoding, IEEE Trans. Commun. 32 (3) (1984) 315–319.

[6] J.B. Anderson, A. Svensson, Coded Modulation Systems, Plenum, New York, 2003.

[7] G.D. Forney Jr., The Viterbi algorithm, Proc. IEEE 61 (3) (1973) 268–278.

[8] M. Kamuf, V. Öwall, J.B. Anderson, Survivor path processing in Viterbi decoders using register exchange and traceforward, IEEE Trans. Circuits Syst. II, Exp. Briefs 54 (6) (2007) 537–541.

[9] M. Kamuf, V. Öwall, J.B. Anderson, Optimization and implementation of a Viterbi decoder under flexibility constraints, IEEE Trans. Circuits Syst. I, Reg. Papers 55 (8) (2008) 2411–2422.

[10] K. Chadha, J.R. Cavallaro, A reconfigurable Viterbi decoder architecture, in: Proceedings of Asilomar Conference on Signals, Syst., and Comp., Pacific Grove, CA, November 2001, pp. 66–71.

[11] Y. Zhu, M. Benaissa, Reconfigurable Viterbi decoding using a new ACS pipelining technique, in: Proceedings of IEEE International Conference on Appl.-Specific Syst., Arch., and Processors, The Hague, June 2003, pp. 360–368.

[12] D.E. Hocevar, A. Gatherer, Achieving flexibility in a Viterbi decoder DSP coprocessor, in: Proceedings of IEEE Veh. Technol. Conference, Boston, September 2000, pp. 2257–2264.

[13] M. Kamuf, J.B. Anderson, V. Öwall, A simplified computational kernel for trellis-based decoding, IEEE Commun. Lett. 8 (3) (2004) 156–158.

[14] H.T. Feldkämper, H. Blume, T.G. Noll, Study of heterogeneous and reconfigurable architectures in the communication domain, Adv. Radio Science—Kleinheub, Berichte 1 (2003) 165–169.

[15] R. Tessier, S. Swaminathan, R. Ramaswamy, D. Göckel, W. Burleson, A reconfigurable power-efficient adaptive Viterbi decoder, IEEE Trans. VLSI Syst. 13 (4) (2005) 484–488.

[16] J.H. Han, A.T. Erdogan, T. Arslan, A power efficient reconfigurable max-log-MAP turbo decoder for wireless communication systems, in: Proceedings of IEEE International Symposium on System-on-Chip, Tampere, November 2005, pp. 247–250.

[17] M.A. Bickerstaff, D. Garrett, T. Prokop, C. Thomas, B. Widdup, G. Zhou, L.M. Davis, G. Woodward, C. Nicol, R.-H. Yan, A unified turbo/Viterbi channel decoder for 3GPP mobile wireless in 0.18-μm CMOS, IEEE J. Solid-State Circuits 37 (11) (2002) 1555–1564.

[18] J.R. Cavallaro, M. Vaya, Viturbo: a reconfigurable architecture for Viterbi and turbo decoding, in: Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing, Hong Kong, April 2003, pp. 497–500.

[19] H.-L. Lou, P. Tong, J.M. Cioffi, A programmable codec design for trellis coded modulation, in: Proceedings of IEEE Global Telecommun. Conf., Phoenix, November 1997, pp. 944–947.

[20] T. Miyauchi, K. Yamamoto, T. Yokokawa, M. Kan, Y. Mizutani, M. Hattori, High-performance programmable SISO decoder VLSI implementation for decoding turbo codes, in: Proceedings of IEEE Global Telecommun. Conference, San Antonio, TX, November 2001, pp. 305–309.

[21] F. Tosato, P. Bisaglia, Simplified soft-output demapper for binary interleaved COFDM with application to HIPERLAN/2, in: Proceedings of IEEE International Conference on Commun., New York, April/May 2002, pp. 664–668.

[22] R. Johannesson, K.S. Zigangirov, Fundamentals of Convolutional Coding Piscataway, IEEE Press, NJ, 1999.

[23] P.J. Black, T.H.-Y. Meng, A 140-Mb/s, 32-state, radix-4 Viterbi decoder, IEEE J. Solid-State Circuits 27 (12) (1992) 1877–1885.

John B. Anderson 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 1972–80 he was on the faculty of the Electrical and Computer Engineering Dept. at McMaster University in Canada, and during 1981–98 he was Professor in the Electrical, Computer and Systems Engineering Dept. at Rensselaer Polytechnic Institute. Since 1998 he has held the Ericsson Chair in Digital Communication at Lund Univ., Sweden. He has held visiting professorships at the Univ. of Calif., Berkeley (1978–79), Chalmers Univ., Sweden (1987), Queen's Univ., Canada (1987), Deutsche Luft- und Raumfahrt, Germany (1991–92, 1995–96) and Tech. Univ. of Munich (1995–96). 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.

He was a member of the IEEE Information Theory Society Board of Governors during 1980–87 and 2001–06, serving as the Society's Vice-President (1983–84) 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 1989–91 and 1994–96. He was a member of the IEEE Press Board during 1993–2006 and during 1994–96 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 (1980–84) and as Guest Editor for the IEEE Communications Transactions 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 is Fellow of the IEEE (1987) and 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 Millenium Medal in 2000.

Matthias Kamuf 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 University 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 AB, Lund, 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 received the M.Sc. and Ph.D. degrees in electrical engineering from Lund University, Lund, Sweden, in 1988 and 1994, respectively. During 1995–1996, 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. He was an Associate Editor of the IEEE Transactions on Circuits and Systems—II: Analog and Digital Signal Processing from 2000 to 2002 and is currently Associate Editor of the IEEE Transactions on Circuits and Systems—I: Regular Papers.

Joachim Neves Rodrigues (S'0-M'05) received his degree in Electrical Engineering and Computer Science from the University of Applied Sciences, Kaiserslautern, Germany, and the Ph.D. degree from the Department of Electroscience, Lund University, Lund, Sweden, in 2000 and 2005, respectively. After two years as ASIC process lead in the digital ASIC department at Ericsson Mobile Platforms, Lund, he joined the Department of Electrical and Information Technology, Lund University. He is currently holding an assistant professorship having a main research interest in energy efficient sub-threshold digital ASIC design.