Design and Optimization of an HSDPA Turbo Decoder ASIC

Christian Benkeser, *Student Member, IEEE*, Andreas Burg, *Member, IEEE*, Teo Cupaiuolo, and Qiuting Huang, *Fellow, IEEE*

***Abstract*—The turbo decoder is the most challenging component in a digital HSDPA receiver in terms of computation requirement and power consumption, where large block size and recursive algorithm prevent pipelining or parallelism to be effectively deployed. This paper addresses the complexity and power consumption issues at algorithmic, arithmetic and gate levels of ASIC design, in order to bring power consumption and die area of turbo decoders to a level commensurate with wireless application. Realized in 0.13 μm CMOS technology, the turbo decoder ASIC measures 1.2 mm^2 excluding pads, and can achieve 10.8 Mb/s throughput while consuming only 32 mW.**

***Index Terms*—Channel decoding, early termination, HSDPA, low power, turbo codes, 3G mobile communication.**

I. Introduction

MODERN wireless communication standards rely on powerful channel coding to ensure reliable (error free) transmission. Channel coding introduces a controlled amount of redundancy into the transmitted data stream, which is then exploited in the receiver to correct transmission errors induced by noise or interference present in the wireless channel. Turbo codes, first proposed in 1993 [1], represent a breakthrough in channel coding techniques, since they have the potential to enable data transmission at rates close to the Shannon limit. They have been adopted for error control coding in the high speed downlink packet access (HSDPA) standard by the third-generation partnership project (3GPP), which considerably enhance the throughput for data-centric 3G modems.

The excellent performance of turbo codes however, comes at the expense of significant computational complexity and consequently high power consumption at the receiver for proper decoding. Indeed, the computational burden of the turbo decoder far exceeds that of any other component in an HSDPA receiver, especially for the higher data rate classes up to 10.8 Mb/s. This places a considerable strain on the total power consumption which must be capped in a mobile device and efficient realization of turbo decoders for HSDPA has been the subject of considerable research in recent years. Publications to date indicate that two distinct approaches are being pursued: one focuses on realizing turbo decoders for 3GPP on general purpose digital signal processors (DSP) ([2]–[4]). Such realizations have so far failed to reach the high end of HSDPA throughput (10.8 Mb/s[1]) despite prohibitively high power consumption and hardware complexity of a fully programmable architecture. The second approach relies on dedicated circuits for the turbo decoding operation with some built-in flexibility to handle the different code-block sizes specified by the 3GPP standard [6]–[9]. Even among these circuits only one [6] met the highest HSDPA throughput, at a staggering 1 W power consumption with a core area of 14 mm^2.

Typically, the throughput of digital circuits can be increased by architectural and circuit-level transformations such as pipelining or parallel processing. For turbo decoders, the applicability of pipelining is limited due to the presence of feedback loops and the accompanying extra registers increase the energy consumption. Parallel processing, on the other hand, is limited by both data dependencies and the access bandwidth of standard memories, which causes considerable area overhead.

In this work, we pursue the increase of throughput via optimization and simplification at algorithm, arithmetic, as well as the gate level to limit the hardware requirements that give rise to higher power consumption and larger silicon area.

A. Outline

The paper is organized as follows. In Section II the fundamentals of the turbo decoding algorithm are revisited. In Section III the high-level architecture of the turbo decoder ASIC realized in this work is presented and the key ideas for optimization and complexity reduction are described. Section IV is dedicated to adapting the decoding effort to the radio link quality to reduce the power consumption of the implemented circuit. Finally, measurement results are presented and compared to reference and prior-art implementations in Section V. Section VI concludes the paper.

II. Algorithm Overview

HSDPA employs a rate 1/3 parallel concatenated turbo code. The corresponding encoder is comprised of two rate 1/2 recursive systematic convolutional encoders, as shown in Fig. 1. The first component encoder receives uncoded (systematic) data bits u_k in natural order and outputs a set of parity bits x_k^{p1}. The second component encoder receives a permutation of the data bits from a block interleaver and outputs a second set of parity bits x_k^{p2}. The systematic bits and the two sets of parity

Manuscript received April 15, 2008; revised August 31, 2008. Current version published December 24, 2008.

The authors are with the Integrated Systems Laboratory (IIS), Swiss Federal Institute of Technology (ETH), Zurich CH-8092, Switzerland (http://www.iis.ee.ethz.ch).

Digital Object Identifier 10.1109/JSSC.2008.2007166

[1]Ref. [5] specifies a maximum raw data rate of 14.4 Mb/s with 16QAM modulation and a maximum code rate of 3/4.

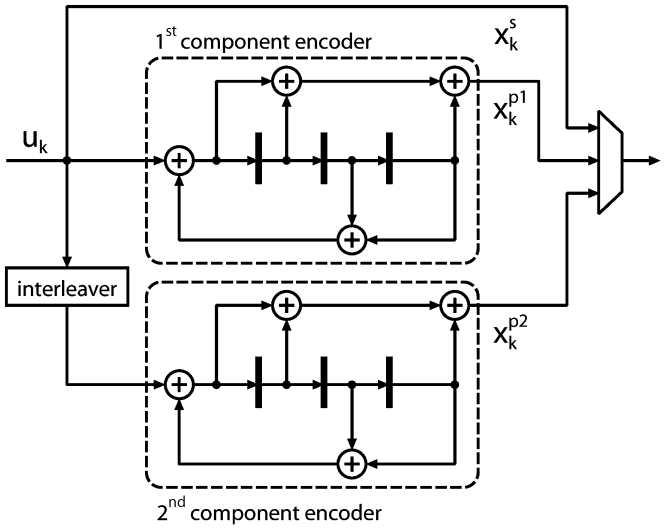

Fig. 1. Channel encoder specified for HSDPA.

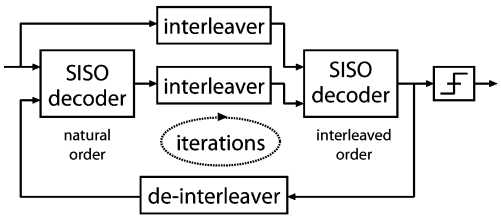

Fig. 2. Simplified block diagram of a turbo decoder.

bits are then modulated onto an analog waveform (according to the employed communication standard) and sent over the radio channel. On the other side of the wireless link, a demodulator is responsible for the reconstruction of the transmitted bits from the received signal. However, since this signal is usually distorted by noise and interference, the demodulator can only obtain estimates of the systematic and two sets of parity bits. These estimates are provided to the subsequent turbo decoder in the form of log-likelihood ratios (LLRs) L_k^s, L_k^{p1}, and L_k^{p2} which express the ratio between the probabilities of the transmitted bits being 0 and being 1, given the received analog signal.

A. The Turbo Decoding Algorithm

The basic idea behind the turbo decoding algorithm is to iterate between two soft-input soft-output (SISO) component decoders as illustrated in Fig. 2. A turbo-iteration corresponds to one pass of the first component decoder followed by a pass of the second component decoder. The operation performed by a single component decoder is referred to as a half-iteration.

The component decoders compute a-posteriori probabilities $L(u_k)$ of the transmitted systematic bits from the LLRs of the (interleaved) systematic bits, the associated parity bits and the a-priori information $L_a(u_k)$. The latter is set to zero for the first half-iteration in the first turbo iteration. In subsequent iterations, each component decoder uses the so-called extrinsic information output $L_e(u_k)$ of the other component decoder in the preceding half-iteration as a-priori information:

$$L_e(u_k) = L(u_k) - L_k^s - L_a(u_k)$$

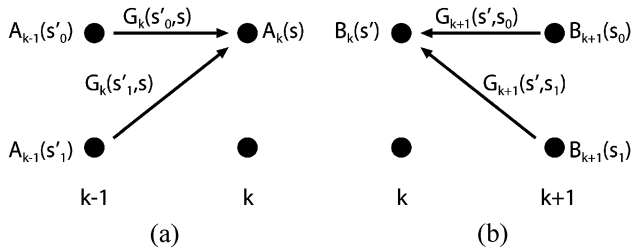

Fig. 3. Example of the calculation of the forward and backward state metrics $A_k(s)$ and $B_k(s')$.

With each half-iteration, the bit error rate (BER) performance improves, with diminishing returns after 6 to 8 turbo iterations.

B. The Maximum A-Posteriori (MAP) Algorithm

In the turbo-decoder under consideration in this paper, the maximum a-posteriori (MAP) algorithm of Bahl, Cocke, Jelinek and Raviv [10] (BCJR algorithm) is used for the SISO component decoders, because it shows the best error rate performance. Unfortunately, the exact MAP algorithm requires the computation of the log of sums of exponential functions, which is clearly not suitable for integration in hardware. Hence, we shall only consider a slightly suboptimal, low-complexity variation known as max-log-MAP algorithm in the following.

The operation of this max-log-MAP algorithm is similar to the Viterbi algorithm. The decoder traverses a trellis diagram in which the nodes represent the 8 possible states of the specified encoder and the branches indicate the admissible state transitions. For the code under consideration, each transition from a state s' to a state s is associated with one of four possible transition metrics[2]

$$\Gamma_k(s',s) = \begin{cases} 0 \\ L_k^{p0/1} \\ L_a(u_k) + L_k^s \\ L_a(u_k) + L_k^s + L_k^{p0/1}. \end{cases} \tag{1}$$

The max-log-MAP algorithm traverses the trellis in both forward and backward directions to compute the state metrics $A(s_i)$ and $B(s_i)$. Loosely speaking, these two metrics reflect the probabilities that the encoder has taken a particular path to arrive at a state or has continued from a state to the end of the trellis, respectively. In the max-log-MAP algorithm the state metric update is simplified compared to the computations in the MAP algorithm using a max-log approximation [11]. The resulting forward and backward state metric recursion requires only add-compare-select (ACS) operations to compute

$$A_k(s) \approx \max\left[A_{k-1}\left(s'_0\right) + \Gamma_k\left(s'_0, s\right), A_{k-1}\left(s'_1\right) + \Gamma_k\left(s'_1, s\right)\right] \tag{2}$$

$$B_k(s') \approx \max\left[B_{k+1}(s_0) + \Gamma_{k+1}(s', s_0), B_{k+1}(s_1) + \Gamma_{k+1}(s', s_1)\right] \tag{3}$$

where s'_0 and s'_1 are the two possible predecessor states of s in the forward direction and where s_0 and s_1 are the two successor

[2]Since only the difference between these transition metrics is relevant for the max-log-MAP algorithm, one of the four possible values [11] of $\Gamma_k(s', s)$ can be turned into a zero by subtracting it from the other transition metrics.

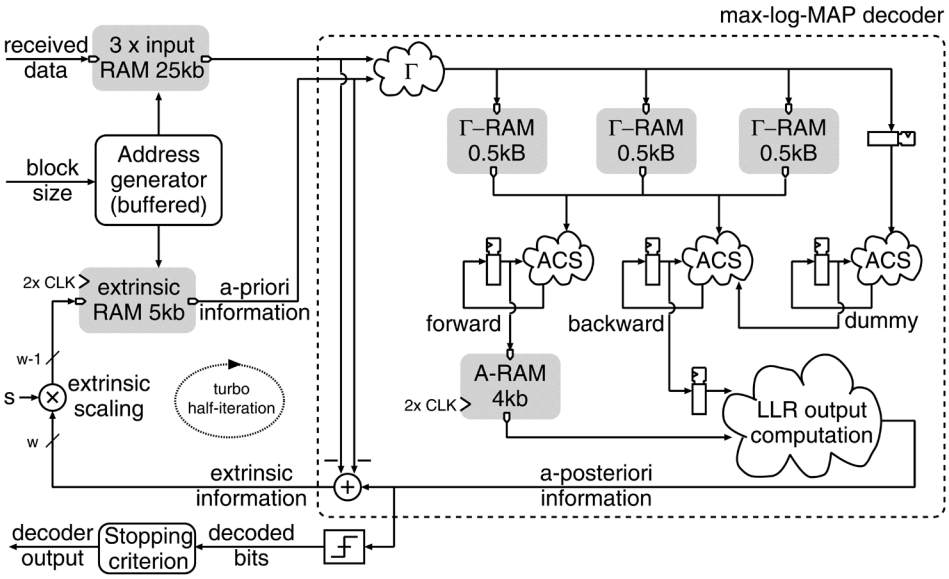

Fig. 4. Architecture of the implemented turbo decoder with max-log-MAP decoder core.

states of s' in the backward direction. An example for the procedure is shown in Fig. 3.

In order to avoid storing all $A_k(s)$ and $B_k(s)$ of the entire code-block, small sub-blocks (windows) are usually processed in a max-log-MAP decoder realization. In the forward recursion the state metrics at the end of a specific window can be used as initial probabilities for the next window. Because the backward state metrics are computed in the opposite direction, a *dummy-backward recursion* must be performed on some trellis steps in advance to generate a reliable set of state metrics as starting points. The constraint length of the convolutional encoder hereby defines a lower bound for the required number of trellis steps. Simulations have shown that for the codes in HSDPA a dummy-backward recursion on less than 24 bits degrades decoding performance significantly. As an alternative to the dummy-backward recursion, the starting points for each window between subsequent turbo iterations can be stored [12]. However, this approach is less attractive for HSDPA with code-block sizes of up to 5114 due to the remarkable memory demand.

Once all $A_k(s)$ and $B_k(s)$ of a particular window have been obtained, the a-posteriori output $L(u_k)$ of the max-log-MAP decoder can be computed. To this end, the decoder must consider the state transitions $s' \Rightarrow s$ associated with $u_k = 0$ and the ones associated with $u_k = 1$ separately and then computes

$$L(u_k) \approx \max_{\substack{s' \Rightarrow s,\\ u_k=0}} \left[A_{k-1}(s') + \Gamma_k(s', s) + B_k(s)\right] - \max_{\substack{s' \Rightarrow s,\\ u_k=1}} \left[A_{k-1}(s') + \Gamma_k(s', s) + B_k(s)\right]. \quad (4)$$

C. Extrinsic Scaling

The approximation used in the max-log-MAP algorithm causes an overestimation of the extrinsic information in the turbo decoder, which results in a decoding performance degradation (c.f., Section V). This error can be reduced by attenuating the extrinsic information with a scaling factor smaller than one [13]. Furthermore, the corresponding multiplication between subsequent half-iterations reduces the correlation between the inputs of the component decoders.

III. Turbo Decoder ASIC Architecture

With low power and small chip size in mind, our realization of the 10.8 Mb/s HSDPA turbo decoder is based on the architecture shown in Fig. 4. In this design, three input RAMs, each with a size of 25 kb, store one block of the LLRs of the systematic and both sets of parity bits using a 5-bit quantization. In each clock cycle, a pair of inputs consisting of the LLR of a systematic bit and the LLR of an associated parity bit enter the max-log-MAP decoder core together with the corresponding a-priori information obtained from a fourth RAM that stores the extrinsic information that is passed between the successive component decoder iterations. The max-log-MAP decoder outputs the a-posteriori information and the extrinsic information. The a-posteriori information is used to derive the binary output of the turbo decoder and to check criteria for early termination (c.f., Section IV) of the turbo decoding process. The extrinsic information is multiplied with a 6 bit scaling factor (Section II-C). The result of this scaling is then written back to the 30 kb[3] extrinsic information memory which must be run at twice the clock rate of the max-log-map decoder to allow for a read- and a write-operation in each system clock cycle.

A. Max-Log-MAP Decoder Core

The high-level architecture of the max-log-MAP decoder is similar to what has been described in [14]. It is designed to process, on average, one trellis step per clock cycle using three state metric recursion units as shown in Fig. 4. These three units perform the forward, backward and dummy-backward calculations on three successive windows in parallel. To be

[3]Note that simulations show that without scaling the word length, the extrinsic memory needs to be 7-bit instead of 6-bit wide, which would result in 5 kb more on-chip memory.

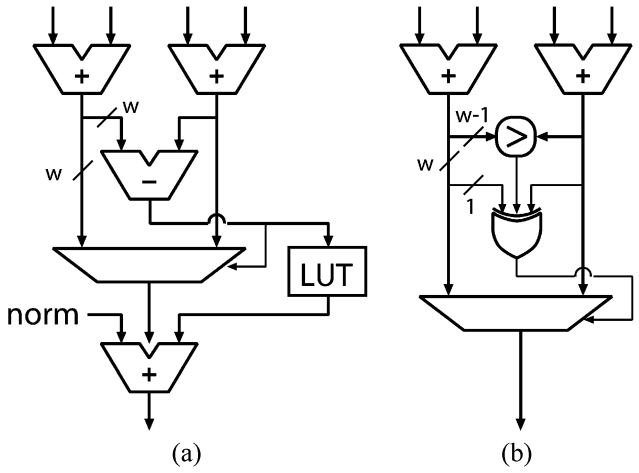

Fig. 5. Comparison of add-compare-select unit implementation in the state metric recursions.

on the safe side a conservative approach with a window size of 40 trellis steps has been adopted in our design to guarantee reliable results from the dummy-backward computation (c.f., Section II-B). Similar to what is described in [14], a cache-memory is employed to overcome the memory-access bandwidth bottleneck to the input and the extrinsic-information memories. However, in contrast to the design described in [14], the present implementation first computes $L_a(u_k) + L_k^s$ and stores only this intermediate result together with $L_k^{p0/1}$ [15] in three single-port Γ-RAMs. Compared to storing all three individual terms ($L_a(u_k)$, L_k^s, and $L_k^{p0/1}$) that contribute to the computation of the Γ-metrics in (1), this measure reduces cache memory storage requirements and avoids redundant computations. The forward state metrics are computed on one window in advance and stored in the A-RAM, making them available for the LLR output computation when the backward state metrics of the particular window are calculated. To achieve a decoding rate of 1 bit/cycle, the A-RAM must operate at twice the clock rate to enable a read- and a write-operation in each system clock cycle.

1) ACS Units: In high-speed MAP decoder implementations the maximum operating frequency is usually limited by the recursive state metric computation, which can not simply be pipelined or parallelized due to the presence of the feedback loop. Hence, we shall focus on measures for reducing the complexity of the state metric recursions to shorten the critical path and to reduce area and power consumption. A prior-art implementation of the ACS unit for state metric recursions ([7]) is shown in Fig. 5(a), where an error correction term is selected from a look-up table (LUT) and added to the output of the ACS unit to mitigate the performance loss associated with the use of the max-log approximation. Because state metrics increase with time, normalization is performed by subtracting a bias that is computed outside the ACS recursion, further increasing complexity.

The present realization avoids the significant cost of the lookup table approach by using extrinsic-scaling to close the error rate performance gap between the MAP and the max-log-MAP algorithm. To solve the problem of increasing state metrics, the max-log-MAP algorithm has been implemented with modulo normalization [16] using a 12-bit quantization for the state metrics (c.f., [17]). This technique, known from Viterbi decoder implementations, achieves the renormalization with a controlled overflow in the data path and requires only an additional 3-input XOR gate in each ACS unit. In summary, the reduced-complexity state metric recursion with the ACS implementations shown in Fig. 5(b) achieves a 50% improvement in speed at a 45% reduction in area compared to the prior-art implementation in Fig. 5(a).

B. Interleaver

Interleavers for turbo codes scramble the data in a pseudo-random order to minimize the correlation between the outputs of component encoders. Hence, by design, the interleaved address pattern specified for HSDPA [18] exhibits little regularity. Not surprisingly, the rules for generating interleaved addresses require complex, hardware expensive operations. The complexity of the address generation rule is compounded by the need to support large code-block sizes up to 5114 bits per block. Hence, simply storing the interleaved order of addresses requires up to 65 kb of dedicated interleaver address memory [6] and separate logic to load this memory quickly when the code-block size (i.e., the interleaver address pattern) changes.

In order to avoid such overhead, our design relies on on-the-fly address generation that is more economic in terms of area and that facilitates fast switching between different code-block sizes. Unfortunately, implementation of such a real-time address generator typically introduces a timing bottleneck for the turbo decoder, which has to be removed.

In the interleaver specification for the turbo codes in HSDPA, an interleaver matrix with $R = 5$, 10 or 20 rows and $C \leq 257$ columns, depending on the code-block size, is first filled with ascending numbers. Then, intra-row and inter-row permutations are performed to generate the desired interleaved order of addresses, which can be read out of the matrix column-by-column. The interleaver matrix does not have to be precomputed and stored, because a closed-form expression for the interleaved address I at a specific position (R_j, C_j) in the matrix can be derived from

$$I(R_j, C_j) = C \cdot T(R_j) + s\left[\underbrace{(C_j \cdot r\,(Q(R_j)))\ \bmod p'}_{X}\right] \quad (5)$$

using standard-specific patterns Q and T that are stored in lookup tables. The derivation requires a preparatory phase, in which a prime number $p \leq 257$ ($p' = p - 1$) is derived directly from the code-block size, the base-sequence for the intra-row permutation s is computed and stored in a 256 × 9b s-RAM, and the sequence r of R permuted 7 bit prime numbers is pre-calculated and stored in registers to be available during the on-the-fly address generation.

Our solution to the main challenge of the preprocessing block, the recursive generation of s, is to exploit interdependencies and irregular calculation rules given by the standard, as described in [19]. The preparatory phase in our turbo decoder ASIC can be completely hidden during the first non-interleaved

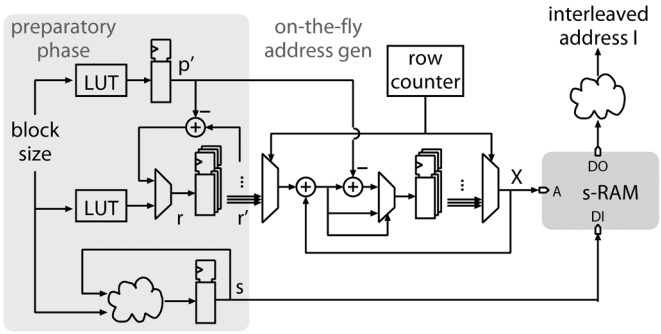

Fig. 6. Low-complexity implementation of the interleaver address generation.

max-log-MAP decoder run and therefore does not slow down the decoding process when the code-block size is switched.

During the on-the-fly address generation the computation of addresses X for the s-RAM access is resource intensive due to the 9b × 7b multiplication and 15b × 9b modulo operation (c.f. (5)). Our solution to this critical speed bottleneck uses a similar approach to that used in the processor-based turbo decoder recently published in [8].

With $Y \triangleq r(Q(R_j))$, the distributivity of the modulo operation

$$\begin{aligned} X &= (C_j \cdot Y) \bmod p' \\ &= \left(C_j \cdot (Y^Q \cdot p' + Y^R)\right) \bmod p' \\ &= (C_j \cdot Y^R) \bmod p' \\ &= (C_j \cdot (Y \bmod p')) \bmod p' \end{aligned} \tag{6}$$

can be exploited together with the fact, that C_j is an incrementing counter, to reduce the generation of X to a recursive structure ($r' \triangleq r(Q(R_j)) \bmod p'$):

$$\begin{aligned} X_{n+1} &= (X_n + r') \bmod p' \\ &= \begin{cases} X_n + r' & : \quad X_n + r' < p' \\ X_n + r' - p' & : \quad X_n + r' \geq p' \end{cases} \end{aligned} \tag{7}$$

With a maximum of 20 different r, which repeat with the incrementing row counter R_j, the modulo operation for generating r' can be realized with a simple recursive subtraction in the preparatory phase. The pre-computed r' are stored in a register bank as can be seen in Fig. 6. According to (7) the modulo operation in X of (5) can then be replaced with an ACS structure for updating and a register bank for storing the addresses.

The resulting address generator provides 1 address per system clock cycle and consumes 1/3 of the logic gates of the turbo decoder. The share of the interleaver's power consumption is considerably lower, less than 2% of the total power, thanks to logic depth reduction by the arithmetic transformations described above, as well as extensive gating of unused sub-blocks.

IV. SNR Dependent Energy Saving Techniques

In wireless communication systems the decoding effort required to achieve reliable transmission strongly depends on the quality of the radio link. Due to the iterative nature of turbo

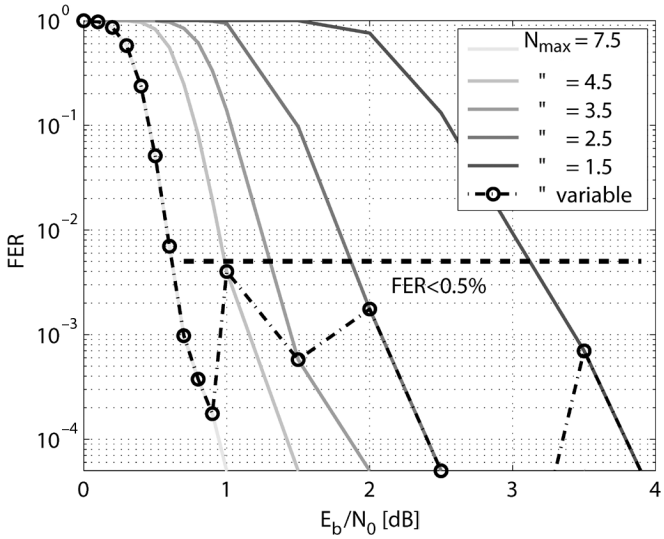

Fig. 7. Frame error rate decoding performance of the implemented turbo decoder for different N_{max}.

decoders, controlling the maximum number of turbo iterations N_{max} enables the decoding effort to be adjusted in discrete steps. With N_{max} and the decoder throughput required for a transmission data rate, the system clock frequency and the associated minimum supply voltage, as well as the resulting power consumption of the turbo decoder are defined.

In turbo decoding, the number of decoding iterations is intrinsically linked to the SNR performance. Under higher SNR conditions, the number of iterations needed to achieve a given target frame error rate (FER) of the decoded code-blocks (frames) is reduced. However, the actual number of iterations required for a particular code-block to be decoded is not known until the decoding process is finished.

We introduce an SNR-dependent constraint to set a limit for N_{max}, above which the probability of the needed number of iterations to achieve a given FER becomes negligible. In Fig. 7 the decoding performance of our turbo decoder implementation is shown with different N_{max}. It can be seen that N_{max} required for a specific FER is indeed a strong function of the SNR. An example of an FER limit is shown by the dashed line against which N_{max} can be adjusted according to the SNR. An FER limit of 0.5% will satisfy the HSDPA requirements as specified in [20].

Setting an adaptive limit to N_{max} that is SNR-dependent, allows the frequency and voltage to be scaled according to the needed decoding effort and thus, the energy efficiency of the turbo decoder can be improved. Nevertheless, due to the fact that the noise in a radio link is distributed randomly, many code-blocks require less iterations than N_{max} to be decoded. Using an on-line stopping criterion to judge when further turbo iterations are redundant enables early termination to be performed before N_{max} and the power to be subsequently gated off until the next code-block is available for decoding.

One of the stopping criteria suitable for silicon integration is the *hard-decision aided* (*HDA*) stopping criterion, where the hard-decisions of the a-posteriori decoder core output after subsequent turbo iterations are compared. When all of them match,

TABLE I
EXAMPLE OF TURBO DECODER CONFIGURATION FOR MINIMUM ENERGY DISSIPATION AT A THROUGHPUT OF $\Theta = 10.8$ Mb/s AND AN ADJUSTED NUMBER OF TURBO ITERATIONS N_{iter}

Step	Operation	Θ $[Mb/s]$	N_{iter}	f_{CLK} $[MHz]$	V_{DD} $[V]$	Power %	Energy %
1)	f_{CLK} is set according to throughput requirements	10.8	5.5	135	1.20	100	100
2)	N_{max} is scaled according to the radio link quality	10.8	3.5	90	1.20	65	65
3)	V_{DD} is reduced according to the scaled frequency	10.8	3.5	90	0.72	23	23
4)	N_{avg} is reduced due to early termination	10.8	1.5–3.5	90	0.72	23	16

it is assumed that no changes will occur with further turbo iterations and early termination is performed [21]. In a variant of the HDA criterion, the *hash HDA criterion*, the hard-decisions are fed into a linear feedback shift register (LFSR) to generate a hash sum, which is compared with the hash sum computed in the previous iteration.

In our turbo decoder a 16 bit LFSR with the characteristic polynomial $x^{16} + x^{12} + x^5 + 1$ has been implemented for the hash HDA stopping criterion. With regards to the influence of early termination, the difference between the HDA and the hash HDA criterion implemented in our ASIC is negligible. This is true in terms of the BER and average number of iterations N_{avg} at which the decoding process is stopped. Choosing a hash HDA stopping criterion reduces design complexity for the turbo decoder output stage. The implementation requires only two 16-bit register banks, one for generating the hash sum and a further one for storing the hash sum of the previous iteration. Furthermore, decoding is stopped only after half-iterations in natural order, where the bits are always in correct order and no final re-alignment of out-of-order data needs to be performed. Compared to the prior-art implementation with the HDA criterion, more than 5 kb of memory has been saved.

The combination of an economic stopping criterion implementation and the new approach of setting a variable limit to the number of turbo iterations to minimize the clock frequency and the associated supply voltage reduces energy dissipation significantly. Setting the SNR-dependent constraint on N_{max} is supported by our turbo decoder implementation, however, controlling N_{max} according to the SNR estimation is performed on system level. The proposed sequence of steps used to configure the turbo decoder for minimum energy dissipation is summarized in Table I via an example.

V. IMPLEMENTATION RESULTS

Power measurements of fabricated chips (see Fig. 9) have been performed on a digital tester and on a circuit board test environment. The fixed-point simulation model of the implemented design has been verified in the test environment through matching intermediate and final soft-outputs in a debug mode and the decoded binary bitstream during normal operation.

In Fig. 8 the BER of the integrated turbo decoder is compared with the simulated BER of floating-point models of turbo decoders using the MAP algorithm, the max-log-MAP algorithm and the implemented max-log-MAP with extrinsic scaling. It can be seen that the turbo decoder with extrinsic scaling performs 0.3 dB better than the turbo decoder with the max-log-MAP algorithm and lies within only 0.1 dB of the turbo decoder with the optimum MAP algorithm. Further, an implementation loss of less than 0.02 dB has been achieved with proper quantization and the use of modulo normalization to avoid saturation in the state metric recursions.

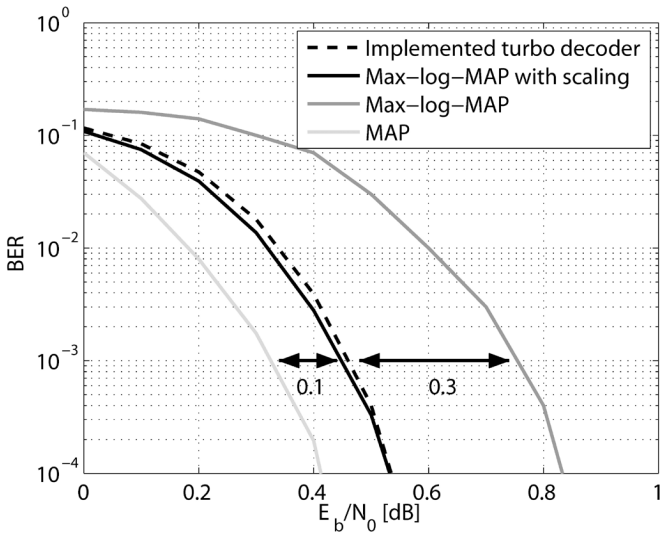

Fig. 8. Bit error rate decoding performance comparison of MAP algorithms and the implemented turbo decoder.

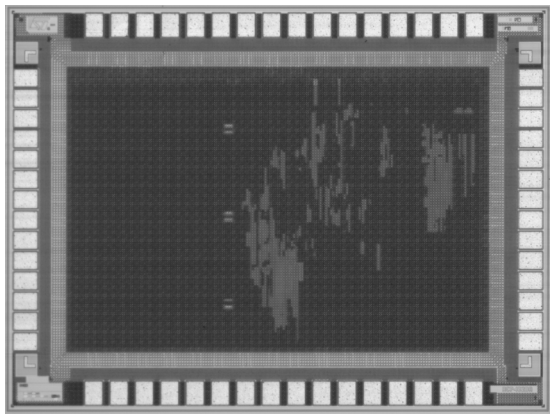

Fig. 9. Die photo of the implemented turbo decoder.

The first column in Table II provides a summary of the key characteristics of our preferred implementation of turbo decoder ASIC, while column two shows an alternative serving as reference point for comparison. Both designs are implemented in the same 0.13 μm CMOS technology and based on the max-

TABLE II
COMPARISON OF THE IMPLEMENTED TURBO DECODER WITH REFERENCE DESIGN AND PRIOR ART

Turbo Decoder	**This work**	Reference	[6]	[7]	[8]
Technology	**0.13μm**	0.13μm	0.18μm	0.18μm	0.25μm
Supply Voltage V_{DD}	**1.2V**	1.2V	1.8V	1.8V	2.5V
Core Size	**1.2mm^2**	1.5mm^2	14.5mm^2	9.0mm^2	8.9mm^2
Gate Count	**44.1k**	43k	410k	85k	34.4k
Total Memory	**120kb**	184kb	450kb	239kb	201kb
Max. System Clock Frequency	**246MHz**	132MHz	145MHz	111MHz	135MHz
Max. Throughput Θ (N_{max})	**20.2Mb/s (5.5)**	12.3Mb/s (5.5)	24Mb/s (6)	4.1Mb/s (6)	5.5Mb/s (6)
Power measured at Θ (N_{max})	Θ=10.8Mb/s (N_{max}=5.5 iter)		10.8Mb/s (8)	2Mb/s (10)	5.5Mb/s (6)
Power (nominal V_{DD})	**61.5mW**	75.6mW	956mW	292mW	n.a.
Power (scaled V_{DD})	**32.4mW**	-	n.a.	-	-
Energy efficiency	**0.54nJ/b/iter**	1.27nJ/b/iter	11.1nJ/b/iter	14.6nJ/b/iter	6.9nJ/b/iter

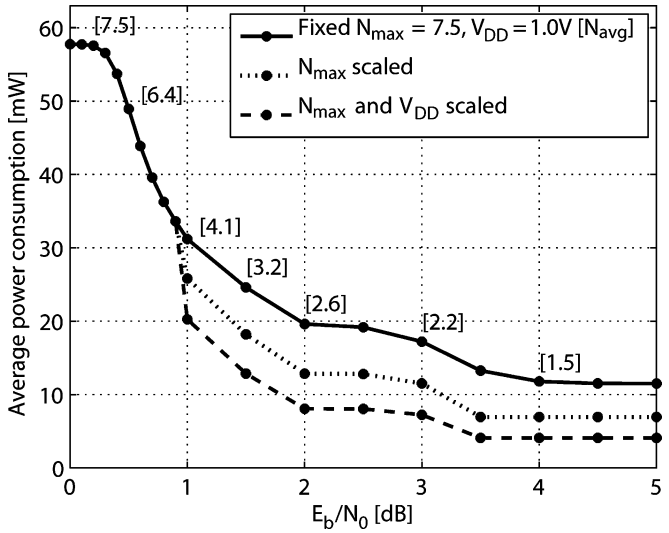

Fig. 10. Average power consumption of the implemented turbo decoder as a function of early termination and with scaling the frequency and supply voltage.

log-MAP algorithm and use the same high-level VLSI architecture of the turbo decoder and decoder core.

A. Comparison With Reference Design

The reference design in column two achieves the required throughput of 10.8 Mb/s at the nominal supply voltage of 1.2 V. However, the width of the input RAMs and the width of the extrinsic RAM have been chosen more conservatively (8 instead of 5 bits and 7 instead of 6 bits, respectively). In addition to this, the design provides no support for early termination and does not employ the algorithm optimizations discussed in Section III-B. When neglecting the influence of the increased RAM size (≈ 0.23 mm^2/ ≈ 10 mW), the die area and power of this design are comparable to the preferred ASIC implementation. However, the preferred turbo decoder shows a better decoding performance (extrinsic scaling) and provides significant timing-margin to allow voltage scaling. As can be seen from the comparison in the table, the gain in speed is achieved without suffering an increase in area and power as a result of the low-complexity implementation discussed in Section III-B. The improvement in maximum frequency permits the supply voltage to be scaled down for the target throughput to 0.87 V, which reduces the power consumption by almost 50% to only 32 mW.

B. Comparison With Prior Art

In the last three columns of Table II other published 3GPP-compliant turbo decoder ASICs are shown. In [6] two trellis steps are computed in each clock cycle to achieve high throughputs. Therefore, the chip can easily provide the fastest HSDPA throughput of 10.8 Mb/s, but the immense overhead in logic cells and memory causes a high power consumption and a large die size. The unified Viterbi/turbo decoder ASIC in [7] and the processor-based turbo decoder in [8] can not achieve the high throughput requirements for HSDPA. Even with scaling the throughput to 0.13 μm CMOS with a scaling factor of $\sqrt{2}$ per technology generation,[4] 10.8 Mb/s can barely be achieved with maximum 6 turbo iterations by [8]. The present turbo decoder implementation provides far more than the required HSDPA throughput at the smallest die size, lowest power consumption and best energy efficiency of all designs.

C. Average Power Consumption Over SNR

Fig. 10 shows the average power consumption of the implemented turbo decoder as a function of the E_b/N_0. In the first curve, V_{DD} and N_{max} have been fixed to 1.0 V and 7.5 iterations respectively, to show the effect of early termination alone. The power consumption decreases with N_{avg}, because the turbo decoder is shut down instead of performing unnecessary turbo iterations.[5] In the second and third curve, additionally the frequency and both, the frequency and the supply voltage are reduced according to the SNR-dependent constraint on N_{max} following the dashed line in Fig. 7. E.g., the power consumption drops from 58 mW to 20 mW at $E_b/N_0 = 2$ dB due to early termination and to only 8 mW with an additional scaling of the frequency and voltage.

VI. CONCLUSIONS

Turbo coded data transmission greatly enhances reliable data throughput but its real-time application requires exceptional computational power, which often drives integrated circuits performing such a task to the most advanced CMOS technology

[4]A detailed table with all designs scaled to 0.13 μm CMOS technology has been given in [19].

[5]The average power consumption as a function of the E_b/N_0 has been obtained as the product of the measured power consumption per turbo-iteration and the number of iterations N_{avg} obtained from bit-true fixed-point simulations.

to contain power and silicon area consumption. In an HSDPA digital receiver the turbo decoder is the most challenging component, where large block size and recursive algorithm prevent pipelining or parallelism to be effectively deployed. The complexity and power consumption issues of HSDPA turbo decoding have been addressed in this paper at algorithmic, arithmetic and gate levels of ASIC design, in order to bring its power consumption and die area down to a level commensurate with wireless application. Extrinsic scaling has been introduced to allow the max-log-MAP algorithm to retain its simplicity, yet come close to SNR performance of the MAP algorithm. The use of modulo normalization minimizes complexity in the ACS units. Interleaver address generation takes advantage of the fact that the inputs of the address generator follow a regular pattern, and hence large multiply-modulo stages can be avoided. Dynamic assignment of maximum number of iterations according to radio link SNR quality allows clock frequency to be adjusted dynamically, and supply voltage scaled down when clock frequency is low. Within a given frequency early stopping criterion has been implemented to help gate off power when not needed, and the hash sum implementation of the stopping criterion proves to be both effective and elegant in its simplicity.

Without going to the state-of-the-art CMOS technology, the 0.13 μm CMOS realization of the HSDPA turbo decoder demonstrates 30 mW power consumption and 1.2 mm^2 die size are sufficient to achieve 10.8 Mb/s throughput, the highest in its class.

REFERENCES

[1] C. Berrou, A. Glavieux, and P. Thitimajshima, "Near Shannon limit error-correcting coding and decoding. turbo codes," in *Proc. IEEE Int. Conf. Communications*, May 1993, pp. 1064–1070.

[2] Y. Lin, S. Mahlke, T. Mudge, C. Chakrabarti, A. Reid, and K. Flautner, "Design and implementation of turbo decoders for software defined radio," in *Proc. IEEE Workshop on Signal Processing Systems Design and Implementation, 2006 (SIPS '06)*, Banff, Alberta, Canada, Oct. 2006, pp. 22–27.

[3] Y. Song, G. Liu, and Huiyang, "The implementation of turbo decoder on DSP in W-CDMA system," in *Proc. 2005 Int. Conf. Wireless Communications, Networking and Mobile Computing*, Sep. 23–26, 2005, vol. 2, pp. 1281–1283.

[4] R. Kothandaraman and M. Lopez, "An efficient implementation of turbo decoder on ADI Tigersharc TS201 DSP," in *Signal Process. Commun.*, Dec. 2004, pp. 344–348.

[5] 3rd Generation Partnership Project; Technical Specification Group Radio Access Network; Channel Coding and Multiplexing Examples, TS 25.944, Rev. 4.1.0, 3GPP Organizational Partners, Jun. 2001.

[6] L. Davis, M. Bickerstaff, C. Thomas, D. Garrett, and C. Nicol, "A 24 Mb/s radix-4 log-MAP turbo decoder for 3GPP-HSDPA mobile wireless," in *ISSCC Communication Signal Processing*, 2003.

[7] M. Bickerstaff, D. Garrett, T. Prokop, C. Thomas, B. Widdup, G. Zhou, C. Nicol, and R.-H. Yan, "A unified turbo/viterbi channel decoder for 3GPP mobile wireless in 0.18 μm CMOS," in *IEEE Int. Solid-State Circuits Conf. 2002 Dig. Tech. Papers*, San Francisco, CA, 2002, vol. 1, pp. 124–451.

[8] M. C. Shin and I. C. Park, "SIMD processor-based turbo decoder supporting multiple third-generation wireless standards," *IEEE Trans. Very Large Scale Integration (VLSI) Syst.*, vol. 15, no. 7, pp. 801–810, Jul. 2007.

[9] C.-C. Lin, Y.-H. Shih, H.-C. Chang, and C.-Y. Lee, "A dual mode channel decoder for 3GPP2 mobile wireless communications," in *Proc. ESSCIRC 2004*, Sep. 21–23, 2004, pp. 483–486.

[10] L. Bahl, J. Cocke, F. Jelinek, and J. Raviv, "Optimal decoding of linear codes for minimizing symbol error rate," *IEEE Trans. Inform. Theory*, vol. 20, pp. 284–287, Mar. 1974.

[11] J. Woodard and L. Hanzo, "Comparative study of turbo decoding techniques: An overview," *IEEE Trans. Vehicular Technol.*, vol. 49, no. 6, pp. 2208–2233, Nov. 2000.

[12] F. Raouafi, A. Dingninou, and C. Berrou, "Saving memory in turbo-decoders using the max-log-MAP algorithm," in *IEE Colloquium on Turbo Codes in Digital Broadcasting—Could it Double Capacity? (Ref. No. 1999/165)*, London, U.K., 1999, pp. 1–14.

[13] J. Vogt and A. Finger, "Improving the max-log-MAP turbo decoder," *Electron. Lett.*, vol. 36, pp. 1937–1939, Nov. 9, 2000.

[14] C.-C. Lin, Y.-H. Shih, H.-C. Chang, and C.-Y. Lee, "A low power turbo/Viterbi decoder for 3GPP2 applications," *IEEE Trans. Very Large Scale Integration (VLSI) Syst.*, vol. 14, no. 4, pp. 426–430, Apr. 2006.

[15] P. Ituero, M. Lopez-Vallejo, and S. Mujtaba, "A configurable application specific processor for turbo decoding," in *Conf. Record 39th Asilomar Conf. Signals, Systems and Computers*, Nov. 2005, pp. 1356–1360.

[16] A. P. Hekstra, "An alternative to metric rescaling in Viterbi decoders," *IEEE Trans. Commun.*, vol. 37, no. 11, pp. 1220–1222, Nov. 1989.

[17] Y. Wu, B. D. Woerner, and T. K. Blankenship, "Data width requirements in SISO decoding with module normalization," *IEEE Trans. Commun.*, vol. 49, no. 11, pp. 1861–1868, Nov. 2001.

[18] 3rd Generation Partnership Project, Technical Specification Group Radio Access Network; Multiplexing and Channel Coding (FDD), TS 25.212, Rev. 5.10.0, 3GPP Organizational Partners, Jun. 2005.

[19] C. Benkeser, A. Burg, T. Cupaiuolo, and Q. Huang, "A 58mW 1.2mm^2 HSDPA Turbo Decoder ASIC in 0.13μm CMOS," in *IEEE ISSCC Dig. Tech. Papers*, Feb. 3–7, 2008, pp. 264–612.

[20] 3rd Generation Partnership Project; Technical Specification Group Radio Access Network; User Equipment (UE) Radio Transmission and Reception (FDD) (Release 8), TS 25.101, Rev. 8.1.0 3GPP Organizational Partners, Dec. 2007.

[21] R. Y. Shao, S. Lin, and M. P. C. Fossorier, "Two simple stopping criteria for turbo decoding," *IEEE Trans. Commun.*, vol. 47, no. 8, pp. 1117–1120, Aug. 1999.

Christian Benkeser (S'08) was born in Ottersweier, Germany, in 1977. He received his Dipl.-Ing. degree in Electronical Engineering from the University of Karlsruhe (TH), Germany, in 2004. In the same year, he joined the Integrated Systems Laboratory of the Swiss Federal Institute of Technology (ETH) Zurich, (http://www.iis.ee.ethz.ch), Zurich, Switzerland, where he is working towards the Ph.D. degree as a Research Assistant. His chief areas of interest have been algorithms, circuits and systems for wireless communications.

Andreas Burg (S'97-M'05) was born in Munich, Germany, in 1975. He received his Dipl.-Ing. degree in 2000 from the Swiss Federal Institute of Technology (ETH) Zurich, (http://www.iis.ee.ethz.ch), Zurich, Switzerland. He then joined the Integrated Systems Laboratory of ETH Zurich, from where he graduated with the Dr. sc. techn. degree in 2006.

In 1998, he worked at Siemens Semiconductors, San Jose, CA. During his doctoral studies, he was a visiting researcher with Bell Labs Wireless Research for a total of one year. From 2006 to 2007, he held positions as postdoctoral researcher at the Integrated Systems Laboratory and at the Communication Technology Laboratory of the ETH Zurich. In 2007 he co-founded Celestrius, an ETH-spinoff in the field of MIMO wireless communication, where he is responsible for the VLSI development. His research interests include the design of digital VLSI circuits and systems, signal processing for wireless communications, and deep submicron VLSI design.

In 2000, Dr. Burg received the "Willi Studer Award" and the ETH Medal for his diploma and his diploma thesis, respectively. He was also awarded an ETH Medal for his Ph.D. dissertation in 2006. In 2008, he was awarded a 4-years grant from the Swiss National Science Foundation (SNF) on which he will join the ETH Zurich as an SNF Professor.

Teo Cupaiuolo received the M.S. degree in electrical engineering in 2006 from Politecnico di Torino, Torino, Italy.

He is currently working within the MIMO WLAN team at Advanced System Technologies (AST), STMicroelectronics of Agrate, where he designs VLSI architectures of advanced MIMO &detection algorithms.

Qiuting Huang (S'86–M'88–SM'96–F'02) received the Ph.D. degree in applied sciences from the Katholieke Universiteit Leuven, Belgium, in 1987.

Between 1987 and 1992 he was a lecturer at the University of East Anglia, Norwich, U.K. Since January 1993, he has been with the Integrated Systems Laboratory, Swiss Federal Institute of Technology (ETH), (http://www.iis.ee.ethz.ch), Zurich, Switzerland, where he is Professor of Electronics. In 2007 he was also appointed as a part-time Cheung Kong Seminar Professor by the Chinese Ministry of Education and the Cheung Kong Foundation and has been affiliated with the South East University, Nanjing, China. His research interests span RF, analog, mixed analog-digital as well as digital application specific integrated circuits and systems, with an emphasis on wireless communications applications in recent years. He has published widely on those topics in leading solid-state circuits conferences and journals.

Prof. Huang is a member of the technical program committees of the International Solid-State Circuits Conference (ISSCC) and the European Solid-State Circuits Conference (ESSCIRC). He is also a member of the executive committee of ISSCC.