# A PARAMETRIZABLE LOW-POWER HIGH-THROUGHPUT TURBO-DECODER

*Gordian Prescher, Tobias Gemmeke and Tobias G. Noll*

Chair of Electrical Engineering and Computer Systems, RWTH Aachen University, Germany
{prescher,tgn}@eecs.rwth-aachen.de gemmeke@ieee.org

## ABSTRACT

This paper presents a high performance turbo decoder. Its major building blocks, the maximum-a-posteriori decoder and the interleaver, are optimized from architecture to layout level to achieve high-throughput at low-power. This includes a novel architecture for parallel interleaving, that sustains any interleaving scheme. Moreover, the key features of the major building blocks are analyzed and modeled for quick design space exploration e.g. achieving 760Mb/s at 570mW in a 0.13µm-CMOS-technology. Finally, the characterized implementations are benchmarked.

## 1. INTRODUCTION

With the advent of turbo codes in 1993 for the first time a feasible channel coding technique was available which almost reaches the Shannon limit [1]. Today turbo codes are used in 3rd generation mobile radio telephone services (3GPP) at a rather moderate throughput of 2 Mb/s. However upcoming applications in wireless and wireline communication systems as well as magnetic recording operate at much higher data rates of up to several Gb/s. These applications require new architectures that speed-up the decoding operation at low-power and reasonable silicon area.

The principle block diagram of a turbo decoder for parallel concatenated convolutional codes is shown in Fig. 1 The maximum a posteriori (MAP) algorithm is applied recursively to input symbols $r^{sys}$, $r^{par,i}$ and extrinsic information $L^{i,ext}$, which is scrambled by interleavers and deinterleavers referred to as $\pi$ and $\pi^{-1}$. High-throughput turbo decoding not only demands for an appropriate MAP decoder implementation but also for a matching interleaver architecture.

This paper presents a high performance turbo decoder implementation approach and analyzes key features of the major building blocks in a 0.13µm CMOS technology. The following section sketches architectures suitable for high-throughput turbo decoding. Section 3 presents a novel parallel interleaver architecture. Architectural, circuit and layout level optimization of the MAP unit is described in section 4. Finally in the last section the results of parameterized power, area and performance models are summarized and compared to those of other leading edge implementations known in literature.

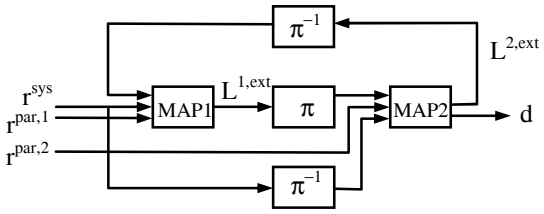


Fig. 1: Principle block diagram of a turbo decoder .

## 2. HIGH-PERFORMANCE ARCHITECTURES FOR TURBO-DECODING

For high throughput decoding there are basically two approaches: either the recursive decoding loop can be unrolled (serial approach) or parallelism is introduced within the MAP decoders (parallel approach).

In a serial architecture the whole decoder resources including memories are duplicated resulting in a large silicon area. On the contrary, a parallel architecture makes use of a single parallel MAP decoder unit which executes both decoding steps MAP1 and MAP2 associated with one turbo decoding iteration sequentially in time. Moreover, a parallel approach has the advantage that it reduces decoding latency significantly. However, it requires a parallel interleaver circuitry. To save area it can be used alternately as interleaver and deinterleaver, respectively.

Parallel MAP decoding is made possible by the sliding window principle [2]. According to this approach, state metric calculation can be started at an arbitrary position within the block. Therefore *N* sub-blocks of size *K*/*N* can be processed concurrently by *N* so-called workers [3]. To reduce memory capacity inside the workers, sub-blocks can in turn be divided into smaller units of size *W*, called windows, which are sequentially processed [4]. A corresponding parallel architecture with sub-blocks and windowing scheme is shown in Fig. 2. Here, each worker determines branch metrics and calculates forward and backward metrics for all *S* states in parallel (solid lines). Further each worker produces one extrinsic information symbol per cycle (thick line). As the worker unit itself is

organized as a single-flow MAP decoder with window size $W$ optimized for low memory requirement [5], it only utilizes two memory banks of size $2W$ for branch metric storage (hatching) and one memory bank of size $S \times W$ for state metric storage (double hatch).

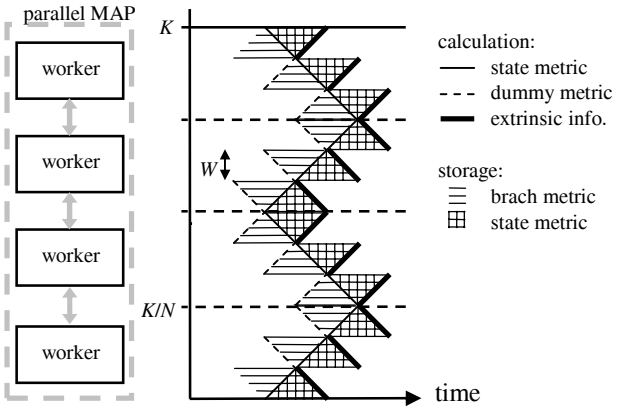


Fig. 2: Windowing scheme of a parallel MAP decoder ($N$=4).

### 3. Design of a Parallel Interleaver

When extrinsic information is exchanged between adjacent MAP decoding steps, its data sequence has to be permuted according to the interleaving scheme and is stored in a memory until the subsequent MAP decoding step begins. In a parallel decoder architecture $N$ symbols of extrinsic information are produced in parallel and have to be written into $N$ memory banks corresponding to $N$ worker units of the succeeding MAP operation. Thereby conflicts in memory access appear whenever two or more workers try to write into the same memory bank.

The complete output data block of the parallel MAP decoder can be represented by a matrix (s. Fig. 3), where each row holds data symbols of the corresponding worker of successive operations. Every symbol produced by the MAP-1 decoder is mapped onto a distinct position of the input data matrix for the following MAP-2 decoding stage according to the interleaving scheme.

To understand the underlying concept of the interleaver it is important to point out that parallel interleaving of a data block resembles a one-to-one routing problem on a two dimensional grid. This problem is well-known in parallel computing theory and can be solved in three distinct routing steps [6]:

1. a permutation of all elements within each column,
2. a permutation of all elements within each row, and
3. a permutation of all elements within each column.

Moreover, an algorithm to determine these routing steps is known [6] which provides a solution for *any* interleaving scheme. A parallel interleaver architecture based on these three routing phases requires routing networks to implement permutations of all elements within each column (s. Fig. 3). The networks in Fig. 3 feature collision-free routing of $N$ inputs to $N$ outputs. In contrast, permutations of elements within each row are mapped on the memory accesses within each of the $N$ memory banks.

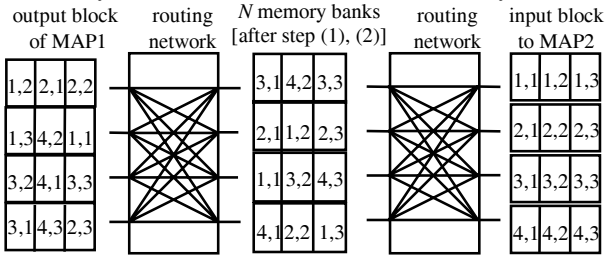


Fig. 3: Architecture of a parallel interleaver comprising two routing networks (column permutations) and $N$ memory banks (row permutations). (Indices of elements refer to their target: row,column.)

The Beneš network is well suited for collision-free routing because of its minimal number of stages. Additionally, the network's regularity and relatively good locality can be exploited by the physically oriented design approach of [7] to reduce the power dissipation. Its layout for $N$=16 is shown in Fig. 4. Its throughput can be scaled by pipelining the network and memory accesses, respectively.

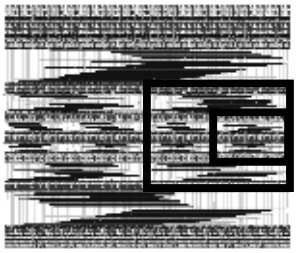

Fig. 4: Layout of Beneš network for $N$=16 (boxes depict the recursive composition with network cores for $N$=8 and $N$=4, respectively).

### 4. Optimization of the Parallel MAP

During MAP decoding state metrics for all $S$ states of the convolutional code are computed in parallel. Due to the non-linear nature of state metric computation the time-critical path of the MAP unit is situated in the so-called add-compare-select-add (ACSA) unit, which determines state metrics according to the log-MAP algorithm [9]. A processor element (PE) that computes one state metric is shown in Fig. 5a. It consists of an add-compare-select unit as used for the Viterbi algorithm. Additionally, a variable correction factor is needed to approximate the Jacobian logarithm. Typically this operation is realized by a look-up table (LUT) and an extra adder stage.

#### 4.1 Logical Optimization

The complexity of the correction factor circuitry can be drastically reduced by using the constant-log-MAP

algorithm [10]. It features nearly identical bit error rate performance as the exact log-MAP algorithm and a 0.3 dB better performance than the simplified max-log-MAP algorithm. In this case the look-up table can be implemented solely by two logic stages leading to a regular structure of small area as well as low power (s. Fig. 6).

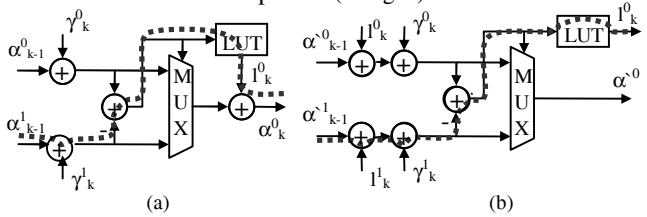


Fig. 5: Signal flow diagram of ACSA (a) and AACS (b) processor element (time-critical path is depicted by a dotted line).

In order to reduce the time critical path the last adder stage can be pushed by retiming to the input (s. Fig. 5b). Due to the nesting of carry-ripple additions the total delay of the add-add-compare-select (AACS) PE is decreased to

$$\tau_{AACS} = \tau_{CRA}(w) + 2 \cdot \tau_{FA}\,, \qquad (1)$$

where $\tau_{CRA}(w)$ is the delay of a $w$ bit carry-ripple adder and $\tau_{FA}$ that of a single full-adder. A further reduction of $\tau_{AACS}$ is possible using carry-select-adders at the cost of significantly increased area and power consumption.

### 4.2 Circuit Implementation

Low power carry-ripple-adders have been implemented using highly efficient mirror full-adder cells with output buffers as well as gate-based half-adder cells with minimum sized gates. In order to further decrease the delay of the time-critical path several optimization steps have been applied. Optimizing the carry path by applying the sizing approach of [7] to dedicated transistor stacks reduced propagation delay by about 25% while area and power dissipation remained almost constant (AACS type 1, Fig.6).

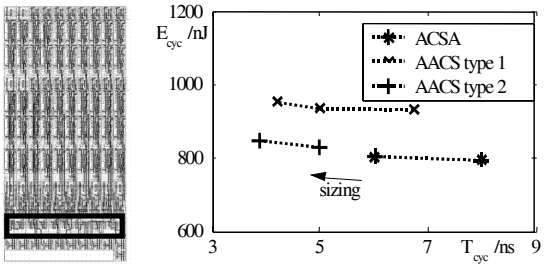


Fig. 6: Layout of AACS PE (box depicts circuitry for correction factor) and Energy per Cycle for different PEs.

Subsequently, by also optimizing the sum path of the adder the delay could further be reduced by 16%, while the increase in area and power dissipation is still negligible (AACS type 2). Finally a mirror full-adder without output buffers was used. Again the device dimensions were optimized and a minimum cycle time of 3.7ns was reached.

## 5. Performance Models and Benchmarking

Turbo decoder implementations using the presented parallel interleaver architecture can be adapted to a wide range of high throughputs. In this section an architecture comprising the parallel MAP decoder depicted in Fig. 2 and the proposed parallel interleaver is analyzed for various parameter sets.

The data rate $R$ is a function of the parameters block size $K$, window size $W$, number of turbo iterations $I$, number of parallel workers $N$ and clock frequency $f$

$$R = \frac{K \cdot f}{2 \cdot I \cdot (\frac{K}{N} + 2 \cdot W)}\,. \qquad (2)$$

Decoding latency $L$ can be determined as

$$L = \frac{2 \cdot I(\frac{K}{N} + 2 \cdot W)}{f}\,. \qquad (3)$$

As $K$, $W$ and $I$ influence decoding performance they are usually kept constant. Consequently, both data rate $R$ and decoding latency $L$ can be improved by increasing the number of parallel workers $N$ or clock frequency $f$ without influencing decoding performance. As presented in the previous sections, $N$ can be altered on architecture level using the novel parallel interleaver while $f$ can be increased by appropriate optimization on circuit and layout level while preserving low-power and small area.

In order to evaluate the proposed approach all delay, power and area critical components were laid out and characterized. Timing simulations assume worst case corner and conditions while power simulations are based on the typical case. From the characterized features of the individual building blocks conservative models for achievable throughput, power dissipation, and silicon area were derived. The figures comprise all interconnects, logic and memory blocks of the MAP decoder and the interleaving network. The models are parameterized with respect to the degree of parallelism and the clock frequency.

In Fig. 7 the silicon area $A$ and sample period $T$ of different parameter sets are shown in the $AT$-space. An "ideally" scaling architecture would feature a constant $AT$-product (dotted hyperbola), whereas the presented architecture shows a significantly decreasing $AT$-product for reasonable increases in the degree of parallelism $N$ This is due to the fact that with increased data rates $R$ (increased $N$) the parallel architecture does not need any additional memory and the memory blocks are divided into even smaller portions. As can be seen in Fig. 7, at some points of the design space it is advantageous to increase clock frequency $f$ by circuit level optimization instead of $N$ (e.g. $T_{sample} \approx 6$ns). Apparently neither parallelism nor sizing alone leads to the optimal solution in general [7]. To find the proper trade-off it is important

to account for all parameters in the design space.

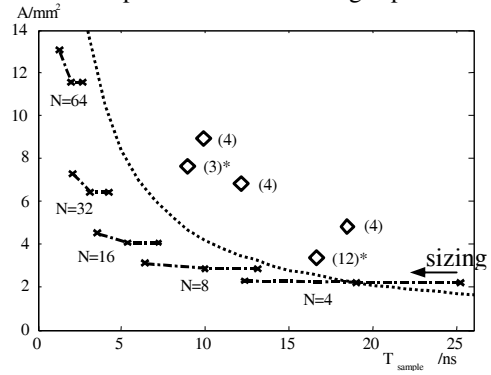


Fig. 7: *AT*-complexity of turbo decoders in comparison to published results for $K$=5120, $W$=32, $I$=6, $S$=8 with the exception of implementations marked with *. (3) features $K$=423 and an early stopping criterion. (12) requires dividable interleaving schemes, its block size $K$ was scaled to 5120.

In order to benchmark the presented architecture, silicon area, throughput and power dissipation of implementations [3], [4], [12] were scaled to a 0.13 µm technology. As shown in Fig. 7 our approach requires 40%-65% less silicon area compared to other leading edge implementations. Further it enables higher data rates at relatively small area.

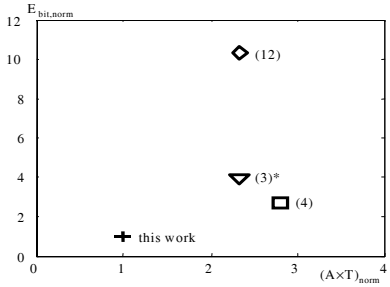


Fig. 8: Normalized energy per decoded bit vs. normalized *AT*-complexity for $K$=5120, $W$=32, $I$=6, $S$=8 (apart from *).

ESTIMATED KEY FEATURES OF TURBO DECODER

| Data rate $R$ / Mb/s | 121 | 284 | 758 |
|---|---|---|---|
| Latency L / µs | 42.2 | 18.0 | 10.5 |
| $P$ / mW | 76 | 178 | 573 |
| $E_{bit}$ / nJ | 0.62 | 0.66 | 0.76 |
| $A$ / mm² | 3.1 | 4.5 | 13.1 |
| $f$ / MHz | 200 | 256 | 256 |
| $N$ | 8 | 16 | 64 |
| $K$ / $W$ / $I$ / $S$ | 5120 / 32 / 6 / 8 | | |
| $w_{\alpha/\gamma}$ / bit | 10 / 7 | | |
| $L_g$ / µm / $V_{DD}$ / V | 0.13 / 1.2 | | |

Finally, the implementations are compared with respect to their normalized energy conversion per decoded bit. For this purpose energy was scaled to a 0.13µm technology: $E \sim V_{dd}^2 \cdot L_g^{0.75}$. Fig. 8 highlights the typical trade-off between *AT*-complexity and energy conversion, when additional area is used to reduce power consumption. Even further power savings are possible using an early stopping criterion and dividable interleaving schemes, as in [3] and [12]. In the latter, the data would pass the interleaver network only once.

## 6. CONCLUSION

In this paper a scalable implementation approach for low-power turbo decoders is presented suitable for throughputs of several 100 Mb/s. It features a novel parallel general purpose interleaver and an ACSA unit optimized on logical and circuit level for high throughput and low-power. With this turbo decoder a data rate of up to 760 Mb/s is feasible with a silicon area of 13 mm² and power dissipation of 570 mW in a 0.13 µm CMOS technology.

## 7. REFERENCES


[1] C. Berrou, A. Glavieux and P. Thitimajshima, "Near Shannon Limit Error-Correcting Coding and Decoding: Turbo-Codes," IEEE Int. Conf. on Communications, vol. 2, pp. 1064-1070, 1993.

[2] K.-H. Tzou and J. G. Dunham, "Sliding Block Decoding of Convolutional Codes," IEEE Trans. Commun., Vol. 29, No. 9, pp. 1401-1403, Sep. 1981.

[3] A. Giulietti et al, "A 80 Mb/s low-power scalable turbo codec core," Proc. CICC, pp. 389-392, 2002.

[4] M. J. Thul et al, "A scalable system architecture for high-throughput turbo-decoders," Kluwer Journal VLSI, Feb. 2003.

[5] G. Masera, M. Mazza, G. Piccinini, F. Viglione and M. Zamboni, "Architectural strategies for low-power VLSI turbo decoders," IEEE Trans. on VLSI Systems, vol. 10, no. 3, pp. 279-285, June 2002.

[6] F. T. Leighton. "Introduction to parallel algorithms and architectures: arrays, trees and hypercubes," San Mateo, CA, USA, Morgan Kaufmann Publishers, 1992.

[7] T. Gemmeke, M. Gansen and T. G. Noll, "Implementation of scalable power and area efficient high-throughput Viterbi decoders," IEEE J. of Solid-State Circuits, vol. 37, no. 7, pp. 941-948, July 2002.

[8] J. Kwak and K. Lee, "Design of dividable interleaver for parallel decoding in turbo codes," Electr. Letters, vol. 38, no. 22, pp 1362-1364, October 2002.

[9] P. Robertson, P. Hoeher and E. Villebrun, "Optimal and sub-optimal maximum a posteriori algorithms for turbo decoding," Europ. Trans. on Telecomm., vol. 8, no. 2, pp. 119-125, 1997.

[10] B. Classon, K. Blankenship and V. Desai, "Turbo decoding with the constant-log-MAP algorithm," Proc. 2nd Int. Symp. on Turbo Codes & Related Topics, pp. 467-470, Sept. 2000.

[11] T. Gemmeke, M. Gansen, H. Stockmanns and T.G. Noll, "Design otimization of low-power high-performance DSP building blocks," IEEE J. of Sol. State. Circ., vol. 39, no. 7, pp.1131-1139, July 2004.

[12] J. Kwak, S. M. Park, S. S. Yoon and K. Lee, "Implementation of a parallel turbo decoder with dividable interleaver," Proc. Int. Symp. on Circuits and Systems, vol. 2, pp. 65-68, May 2003.