Advanced Implementation Issues of Turbo-Decoders

A. Worm, H. Michel, F. Gilbert, G. Kreiselmaier, M. Thul, N. Wehn

Institute of Microelectronic Systems
University of Kaiserslautern, Erwin-Schrödinger-Straße, D-67663 Kaiserslautern, Germany
Tel: (+49) 631 205 4435, Fax: (+49) 631 205 4437
E-mail: worm@eit.uni-kl.de

Abstract*: This paper presents a new renormalization scheme for Turbo-decoders and a novel early stop criterion which can be used to considerably reduce implementation complexity.*

Keywords: Turbo-decoder, implementation.

1. INTRODUCTION

Efficient hardware and software Turbo-decoder [2] implementations are mandatory for widespread use, e. g. 3G and WLAN. Although some papers on Turbo-decoder implementation have been published (e. g. [11], [6]), they lack major issues or do not address them in a satisfying manner. To obtain efficient decoder implementations, the implementation space has to be explored on multiple levels [1], trading off communication performance versus VLSI performance. This paper puts emphasis on some important issues not yet covered in our previous paper [1]: We systematically investigate the application of modulo renormalization to Turbo-decoders and present a new optimized scheme based on a combination of modulo renormalization with rescaling. Second, an early iteration stopping criterion is proposed and applied within an iteration control scheme to considerably reduce implementation complexity.

We assume conventional Turbo-decoding of parallel concatenated convolutional codes (PCCC) with constituent decoders using the Log-MAP or the suboptimal Max-Log-MAP algorithm [9]. However, our results also apply to serial concatenation (SCCC). The decoder model is described in [8]. We refer the reader to [9] for a discussion of MAP algorithms and adopt the notation of [9] and [5].

The remainder of this paper is structured as follows: Section 2. studies previously undiscussed aspects of renormalization within Turbo-decoders, Section 3. presents an early stop mechanism and shows how this can be exploited to considerably reduce implementation complexity.

2. RENORMALIZATION

Fixed-point number representation is mandatory for most target architectures in hardware and software. Thus, the transformation from floating to fixed-point is an important step towards an efficient implementation. The interested reader is referred for [8], [7] for further discussions.

Another important implication of fixed-point number representation is the necessity to renormalize. In principle, the path metrics $\overline{\alpha}_k(S_k)$ and $\overline{\beta}_k(S_k)$ can accumulate during (Max-)Log-MAP algorithm forward and backward recursion without bounds and therefore require periodic renormalization to prevent arithmetic overflow[1]. This can be achieved by a rescaling approach (Section 2.1.) or by using modulo arithmetic (Section 2.2.). Modulo arithmetic was devised in [4] and [10] as an efficient renormalization method for Viterbi decoders. However, it is not quite obvious how to apply modulo arithmetic to MAP decoders. The applicability is even denied in [11]. In [6], modulo arithmetic is used, but the implications on the soft-output calculation are not mentioned. In this paper, we investigate for the first time in more detail how modulo arithmetic can be applied in MAP decoder implementations including the soft-output (Λ) calculation and its implications.

2.1. Rescaling

As stated above, the path metrics $\overline{\alpha}_k(S_k)$ and $\overline{\beta}_k(S_k)$ of a (Max-)Log-MAP decoder would increase without limits and effect arithmetic overflow in a fixed-point implementation if not periodically renormalized. This renormalization can be achieved by subtracting the respective minimum path metric in each time step. The relevant information is contained within the differences of the path metrics and remains therefore unaltered. Admitting tolerable decrease in communication performance, rescaling can be easily combined with saturating the path metrics in order to minimize bit-widths. Let (q, f) be a fixed-point number with a total bit-width of q and with f bits as fractional part. Our simulation results show that $(5, 2)$-quantized input data lead to negligible bit-error rate (BER) degradation of a 3GPP Turbo-decoder. Using rescaling and saturating path metrics, this yields an implementation with $(7, 2)$ quantization of internal data. Further details are discussed in [7].

The major drawback of renormalization by subtracting the minimum metric of the current trellis step is a prolonged critical path for a hardware implementation and

[1] In a sense, normalization is also necessary for numerical stability when formulating the MAP algorithm with probabilities.

therefore a degraded maximum clock frequency. Counteraction by pipelining is not possible due to the well known add-compare-select (ACS) bottle-neck problem and can be only solved on a higher abstraction level, i. e. decoding several sub-blocks in parallel. In a software implementation, rescaling results in additional instructions inside the inner loop.

2.2. Modulo Renormalization

In contrast to rescaling, modulo renormalization avoids the normalizing subtractions by using a better suited number representation. The Max-Log-MAP algorithm can be interpreted as a combination of two Viterbi algorithms, one proceeding from the beginning to the end of the data block, the other proceeding in opposite direction. The key idea of modulo renormalization is not to invest in avoiding overflow, but instead to accommodate overflow in such a way that is does not affect the correctness of the results. This is achieved by using two's complement arithmetic on a fixed-point number representation. We use the theory of [4] as basis for our investigations. The bit-width c has to be large enough to allow unambiguous evaluation of differences Δ [4]:

$$\lceil \mathrm{ld}\,\Delta_{\max} \rceil + 1 \leq c. \tag{1}$$

Let B be an upper bound for the absolute values of the signed branch metrics:

$$\left| \overline{\gamma}_i \left[(y_k^s, y_k^p), S_{k-1}, S_k \right] \right| \leq B, \quad S_k, S_{k-1} \in S. \tag{2}$$

With $m = K-1$ being the memory order of a code with constraint length K, the difference between any pair of forward path metrics of the same trellis step k is then also upper bounded:

$$|\overline{\alpha}_k(S_1) - \overline{\alpha}_k(S_2)| \leq 2mB, \quad S_1, S_2 \in S. \tag{3}$$

The required bit widths for storing the path metrics after a recursion step is given as:

$$c_{\mathrm{pm}} = \lceil \mathrm{ld}(2mB) \rceil + 1. \tag{4}$$

It can be shown that the candidate path metrics are upper bounded as follows [4]:

$$\begin{aligned} &\left| \overline{\alpha}_{k-1}(S_{k-1,1}) + \overline{\gamma}_i \left[(y_k^s, y_k^p), S_{k-1,1}, S_k \right] \right. \\ &\quad \left. - \left(\overline{\alpha}_{k-1}(S_{k-1,2}) + \overline{\gamma}_i \left[(y_k^s, y_k^p), S_{k-1,2}, S_k \right] \right) \right| \\ &\qquad \leq 2(m+1)B, \quad S_{k-1,1}, S_{k-1,2}, S_k \in S, \end{aligned} \tag{5}$$

assuming that each state S_k has two predecessors, $S_{k-1,1}$ and $S_{k-1,2}$. Thus, we obtain the candidate path metric bit-width

$$c_{\mathrm{cpm}} = \lceil \mathrm{ld}((2(m+1)B) \rceil + 1. \tag{6}$$

All bounds and derived bit-widths hold for the backward recursion as well. It can be shown that the same bit-widths are required in case of non-negative branch metrics.

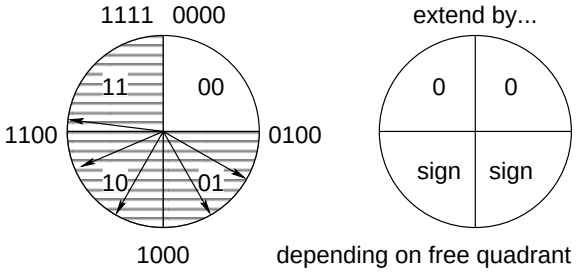

Figure 1: Proposed scheme for bit-width extension in calculation of Λ, assuming $c = 4$.

The results also apply to the Log-MAP decoder, at least in a probabilistic sense, as the correction term of the max*-operator, which is used in a Log-MAP decoder instead of plain maximum selection, mitigates the hard path decisions of the Max-Log-MAP decoder and therefore reduces path metric differences. During soft-output calculation, the Log-MAP decoder computes the sums

$$\overline{\delta}_k(i, S_{k-1}, S_k) = \overline{\gamma}_i \left[(y_k^s, y_k^p), S_{k-1}, S_k \right] + \overline{\beta}_k(S_k) \tag{7}$$

and combines them with the path metrics $\overline{\alpha}_{k-1}(S_{k-1})$ according to the trellis structure:

$$\begin{aligned} \Lambda(d_k) = &\max^*_{(S_k, S_{k-1})} \{ \overline{\alpha}_{k-1}(S_{k-1}) + \overline{\delta}_k(1, S_{k-1}, S_k) \} \\ &- \max^*_{(S_k, S_{k-1})} \{ \overline{\alpha}_{k-1}(S_{k-1}) + \overline{\delta}_k(0, S_{k-1}, S_k) \}. \end{aligned} \tag{8}$$

The sums $\overline{\delta}_k(i, S_{k-1}, S_k)$ are structurally identical to candidate metrics during forward and backward recursion. A bit-width of $c_{\mathrm{cpm}} = \lceil \mathrm{ld}\, 2(m+1)B \rceil + 1$ is therefore required. Note that combining these sums with the respective $\overline{\alpha}_{k-1}(S_{k-1})$ can result in detrimental overflow, because the condition in equation (1) does not hold for the sum $\overline{\alpha}_{k-1}(S_{k-1}) + \overline{\delta}_k(i, S_{k-1}, S_k)$. It is necessary to provide an additional bit and have $c_\Lambda = c_{\mathrm{cpm}} + 1$. A viable solution is to use c_Λ bits also during forward and backward recursion. However, it is often desirable to store the metrics with minimum bit-width. Thus, an efficient bit-width extension scheme before soft-output calculation is required. One solution is to exactly determine the minimum and the maximum number and to extend by either sign-true extension or by appending a leading '0' or '1', depending on the positions of these on the number circle. This solution is costly and not necessary. A more efficient solution is to use equation (1), i. e. we assume that the largest difference $\Delta_{\max}$ is not larger than half of the size of the number circle. We divide the number circle in quadrants; this means that at least one of them is free. Extension is therefore performed in dependence on the position of the free quadrant. This scheme renders quite easy to implement, because the quadrant a number belongs to is solely determined by the two most significant bits. Figure 1 depicts the situation.

In the previous paragraphs, we have derived the theoretical minimum bit-widths of the path metrics (c_{pm}), the

	Theory	Simulation
	c_{pm} / c_{cpm} / c_Λ	c_{cpm} / c_Λ
$K=3, B=16$	7 / 8 / 9	8 / 8
$K=4, B=16$	8 / 8 / 9	8 / 9
$K=5, B=16$	8 / 9 / 10	8 / 8
$K=3, B=64$	9 / 10 / 11	10 / 10
$K=4, B=64$	10 / 10 / 11	10 / 11
$K=5, B=64$	10 / 11 / 12	10 / 10

Table 1: Simulation results for a $R=1/3$ PCCC with constraint length $K=3$, $K=4$, $K=5$ (generator polynoms 5/7, 13/15, 27/31) constituent decoders, block size $L_{\text{block}}=600$, 3GPP interleaver. $B=16$ corresponds to a $(5,1)$ and $B=64$ to a $(7,2)$ branch metric quantization. The results have been validated over a large SNR range by simulating $N_{\text{block}}=20000$ blocks at each SNR both for an AWGN and a Rayleigh fading channel.

candidate path metrics (c_{cpm}) and the intermediate values during soft-output calculation (c_Λ). For reasonable values of m and B (e. g., $m \in \{3\dots7\}$, $B \in \{2^4\dots2^8\}$), the relations $c_{\text{cpm}} \geq c_{\text{pm}}$ and $c_\Lambda = c_{\text{cpm}+1}$ hold. However, from a practical point of view, bit-widths have to be adequate in a probabilistic sense only. We demonstrate in Table 1 by simulation that often smaller bit-widths are sufficient without any degradation in communication performance.

Modulo renormalization avoids the major drawback of renormalization by subtraction, because the metrics stay implicitly normalized. A hardware implementation can operate at a higher clock frequency, and a software decoder has to execute less instructions. A disadvantage might be the larger bit-widths: 1–2 bit penalty in case of $K=4$ and $B=16$, when compared to subtractive rescaling with saturation. We can show that for a high-speed MAP decoder one additional bit results in approx. 25% higher area and power consumption.

2.3. New Optimized Scheme

In this section we propose a renormalization scheme for the (Max-)Log-MAP decoder which combines rescaling with modulo renormalization. The key idea is to perform the recursions using modulo arithmetics, while the path metrics are rescaled subtractively prior to being stored. This approach removes the rescaling from the critical path and therefore enables faster execution of the recursion. It comes at the expenses of wider datapaths for the recursion when compared to mere rescaling, or an additional normalization unit when compared to the modulo normalization, respectively. The advantages are: Faster execution than the standard subtractive scheme while it's desirable feature of smaller bit-widths for memories, and hence less area and power consumption, is maintained. Moreover, the rescaled forward path metrics can be compared to an absolute threshold; this enables the T-MAP algorithm [3], [13].

3. EARLY STOP CRITERION

The second new contribution of this paper is related to iteration control criteria. The number of iterations of Turbo-decoding for a given BER can largely vary, especially in wireless environments where channel characteristics can change quickly. Although this is equalized by power control mechanisms, the latency can be considerable. This can be exploited to reduce implementation complexity. In the literature, criteria have been devised which stop the Turbo-iteration when the data block is assumed to be correctly decoded. In this paper we present for the first time an approach which allows to stop iteration over the *full* SNR range, i. e. for good and for *bad* channels.

One stop criterion for correctly decoded blocks is a CRC-check performed after each iteration. When using the T-MAP algorithm as constituent decoder, the average number of alive states can serve as an efficient and easily evaluable stop criterion. However, large optimization potential exists, if iterations can be stopped when the required BER cannot be maintained with the maximum number of iterations. This issue has not been addressed in current literature. Simply using an estimated channel SNR as a criterion is too coarse and may trash too many decodable blocks. Moreover, an estimation of the channel SNR is not necessary for the Turbo-decoding process itself from a practical point of view [12]; the variance estimator can be completely omitted in favor of assuming a (set of) design-point(s) for the Log-MAP decoder or using the less complex Max-Log-MAP decoder which is independent of the channel SNR.

Next generation mobile communication systems will offer high data-rate services with packet oriented network protocols (e. g. TCP/IP). In such environments, the number of bit errors per data frame is of no interest. Thus, the frame-error rate (FER) instead of the BER in considered in the following.

The new stop criterion, which detects undecodable data frames as early as possible, works as follows: The mean $\mu_{\tilde{\Lambda}}$ of the absolute values of the saturated LLRs (Λ) serves as a measure of confidence in the decoding result of the entire frame. When comparing $\mu_{\tilde{\Lambda}}$ of decodable and undecodable data frames, the increment from iteration to iteration is quite different. The mean of a undecodable data frame is well beneath the mean of a decodable frame. Thus, the slope is much lower for undecodable frames as well. As the difference is quite remarkable, this behavior is the basis for our stop criterion for undecodable data frames. The bounds for the mean value $\mu_{\tilde{\Lambda}}$ are optimized by simulation.

Combining CRC and stop criteria for undecodable data frames, iterations of successive data frames can be redistributed on the cost of some latency. This approach improves the FER considerably compared to an implementation with a fixed number of iterations. Let T be the frame period, i. e. the time between the arrivals of suc-

cessive frames, and L be the latency requirement in terms of frame period ($L > 1$). Algorithm 1 presents the flexible iteration scheme under the assumption that the decoder is able to process the maximum number of iterations $I_{\max}$ at its peak performance within the frame period T.

Algorithm 1 Flexible Iteration Algorithm

```
{μ_th[] : threshold values for mean of μ_Λ}
{Δμ_th[] : threshold values for diff. of succ. mean val.}
{LLR : Variable LLR is ARRAY of REAL}
μ ⇐ 0,    iter ⇐ 1
δ ⇐ {remaining decoding time in terms of (1/I_max) T}
while δ > 0 do
  LLR ⇐ (Max)LogMAPIteration( frame )
  if CRCCheck( LLR ) ≠ ERROR then
    exit while    {block successfully decoded}
  end if
  μ_old ⇐ μ,    μ ⇐ Mean( LLR )
  if δ ≤ (L − 1) * I_max then
    {evaluate stop crit. only if next block avail.}
    Δμ ⇐ μ − μ_old
    if μ < μ_th[iter] or Δμ < Δμ_th[iter] then
      exit while    {block undecodable}
    end if
  end if
  iter ⇐ iter + 1,    δ ⇐ δ − 1
end while
```

Figure 2 compares the FER of the proposed algorithm ($I_{\max} = 5$, $L = 2$) to various fixed-iteration implementations for an AWGN channel. The results can be interpreted in two different ways: First, although a decoder using the flexible iteration algorithm can only perform five iterations per frame period, the FER is always superior to a fixed-iteration decoder with seven iterations. It achieves nearly the performance of fixed-iteration decoder with ten iterations, which has twice the implementation complexity. Second, based on comparable implementation complexity, an alternative interpretation is a gain of approximately 0.25 dB at a design-point of FER $= 10^{-3}$; however, when decoding performance needs to be maintained only, this algorithm can also be used to save power. We can reproduce our results for Rayleigh fading channels.

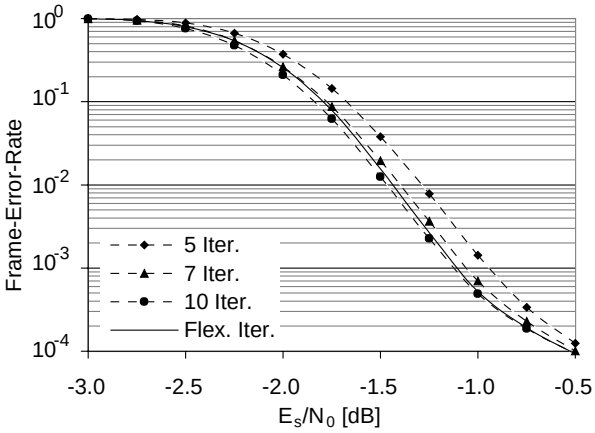

Figure 2: FER performance of flexible iteration algorithm and fixed-iteration algorithms for $R = 1/2$, PCCC, $K = 4$, $L_{\text{block}} = 600$, 3GPP interleaver, AWGN channel.

ACKNOWLEDGMENT

The authors would like to thank H. Lamm for valuable discussions on modulo normalization.

REFERENCES

[1] F. Berens, A. Worm, H. Michel, and N. Wehn. Implementation Aspects of Turbo-Decoders for Future Radio Applications. In *Proc. VTC '99 Fall*, pages 2601–2605, September 1999.

[2] C. Berrou, A. Glavieux, and P. Thitimajshima. Near Shannon Limit Error-Correcting Coding and Decoding: Turbo-Codes. In *Proc. ICC '93*, pages 1064–1070, May 1993.

[3] V. Franz and J. B. Anderson. Concatenated Decoding with a Reduced-Search BCJR Algorithm. *IEEE Journal on Selected Areas in Communications*, 16(2):186–195, February 1998.

[4] A. Hekstra. An Alternative to Metric Rescaling in Viterbi Decoders. *IEEE Transactions on Communications*, 37(11):1220–1222, November 1989.

[5] P. Hoeher. New Iterative ("Turbo") Decoding Algorithms. In *Proc. International Symposium on Turbo Codes & Related Topics*, pages 63–70, 1997.

[6] G. Masera et al. VLSI Architectures for Turbo Codes. *IEEE Transactions on VLSI Systems*, 7(3):369–379, September 1999.

[7] H. Michel and N. Wehn. Turbo-Decoder Quantization for UMTS. *IEEE Communications Letters*, 2000. To appear.

[8] H. Michel, A. Worm, and N. Wehn. Influence of Quantization on the Bit-Error Performance of Turbo-Decoders. In *Proc. VTC '00 Spring*, May 2000.

[9] P. Robertson, P. Hoeher, and E. Villebrun. Optimal and Sub-Optimal Maximum a Posteriori Algorithms Suitable for Turbo Decoding. *ETT*, 8(2):119–125, Mar.–Apr. 1997.

[10] C.B. Shung et al. VLSI Architectures for Metric Normalization in the Viterbi Algorithm. In *Proc. ICC '90*, pages 1723–1728, 1990.

[11] Z. Wang et al. VLSI Implementation Issues of Turbo Decoder Design For Wireless Applications. In *Proc. SiPS '99*, October 1999.

[12] A. Worm, P. Hoeher, and N. Wehn. Turbo-Decoding without SNR Estimation. *IEEE Communications Letters*, 4(6), June 2000.

[13] A. Worm, H. Michel, and N. Wehn. Power minimization by optimizing data transfers in Turbo-decoders. In *Kleinheubacher Berichte*, volume 43, pages 343–350, September 1999.