VLSI-IMPLEMENTATION ISSUES OF TURBO TRELLIS-CODED MODULATION

Frank Kienle, Gerd Kreiselmaier, Norbert Wehn

Institute of Microelectronic Systems, University of Kaiserslautern
Erwin-Schrödinger-Straße, 67663 Kaiserslautern, Germany
{kienle, kreiselmaier, wehn}@eit.uni-kl.de

ABSTRACT

Turbo Trellis-Coded Modulation (TTCM) is a very promising approach for future communication systems. It combines the advantages of channel coding with multilevel signals and the powerful Turbo-Codes concept.

In this paper we consider VLSI implementation aspects of TTCM. We show that techniques, known from binary Turbo-Decoders, can be applied to TTCM to reduce the implementation complexity significantly. In detail we explore iteration control, quantization, scaling, and the MAP architecture using a bit-true model of an 8-state TTCM decoder.

1. INTRODUCTION

Future communication systems demand bandwidth efficiency and good coding gain. There are several approaches for bandwidth efficient Turbo-Codes which fulfills these demands: Pragmatic TCM [3] and Turbo Trellis-Coded Modulation (TTCM) [7] are promising approaches. From an implementation point of view the pragmatic TCM is a straight forward extension to the binary Turbo-Codes (TC) [1]. However TTCM implementation is not so evident and requires a thorough exploration with respect to its implementation complexity.

The TTCM encoder consists of two recursive systematic component codes, parallel concatenated by an interleaver. The iterative decoder consists of two soft in/soft out (SISO) component decoders and corresponding interleaver, de-interleaver. The SISO decoders, typically based on the maximum a posteriori (MAP) algorithm, exchange information while the coding gain improves from iteration to iteration. The overall structure of a TTCM decoder is similar to a binary Turbo-Decoder.

Many papers are published on implementation issues of binary TC e.g. [2][8][9]. The major techniques for an efficient TC implementation are:

algorithm transformation of the MAP algorithm in the logarithm domain [6]; the use of an *extrinsic scaling factor* (ESF) to reduce communication degradation [4]; *quantization, modulo re-normalization* [9] allows the recursion unit to operate at higher clock frequency; *windowing technique*

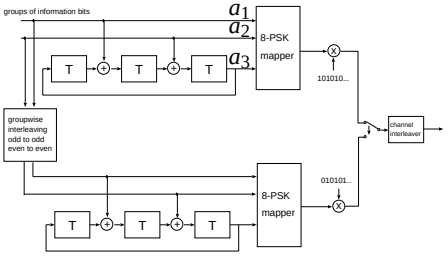

Fig. 1. Turbo TCM Encoder

reduces the memory requirement and allows to parallelize the architecture [2]; *iteration control* [5] [9] increases average throughput and/or saves power.

These techniques, known from binary Turbo-Codes, are now explored with respect to its applicability for TTCM.

The remainder of this paper is structured as follows: The system model for TTCM with algorithm simplifications is presented in Section 2. An iteration control for TTCM is introduced in Section 3 . The architecture of the MAP unit is presented in Section 4. In Section 5 simulation results are presented. Section 6 concludes this paper.

2. SYSTEM

The TTCM encoder depicted in Fig. 1 consists of two recursive component codes. Each component code is a Trellis Coded Modulation (TCM) encoder of memory size 3. The encoders are parallel concatenated by a special interleaver, which permutes under the condition that pairs of bits are mapped from an even position of the sequence stream to an even position (odd to odd respectively). The output stream of every component encoder is punctured.

The decoder has a similar structure to a binary Turbo-Decoder. The main building blocks are the two MAP decoders and the corresponding interleaver and de-interleaver. The MAP decoder has two inputs: a channel input and an 'a priori' input, which is fed back from the other MAP decoder. Only the channel information generated by the upper TCM encoder (Fig. 1) is passed to MAP 1. The symbols generated by the lower TCM encoder are set to "0", indi-

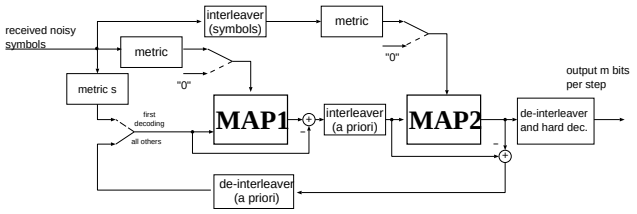

Fig. 2. Turbo TCM Decoder

cated by the switch (vice versa for MAP2). It must be guaranteed that every systematic information is used only once per iteration.

2.1. Algorithm for non-binary MAP

The non-binary MAP algorithm [7] calculates for every time step k the a posteriori probability (APP) of a possible group of info bits $d_k = i$ (for the system above $i \in \{0,1,2,3\}$) under the condition of the received block $\vec{\mathbf{y}}$:

$$P(d_k = i|\vec{\mathbf{y}}) = const \cdot \sum_{M} \sum_{M'} \gamma_i(\mathbf{y}_k, M', M) \cdot \alpha_{k-1}(M') \cdot \beta_k(M). \tag{1}$$

y_k is the received signal at time step $k \in \{1,2,\ldots,N\}$, M is the state at time k, M' at time $k-1$. $\gamma_i(\cdot)$ is the branch transition probability for $d_k = i$. α_k is a forward and β_k a backward variable which are calculated recursively.

Robertson showed that a transformation to the logarithmic domain (Log-MAP) is possible without SNR degradation [6] by using the Jacobian logarithm

$$max^*(\delta_1, \delta_2) = max(\delta_1, \delta_2) + ln(1 + e^{-|\delta_1 - \delta_2|}). \tag{2}$$

Omitting the correction term $(ln(\cdot))$ yields the so called Max-Log-MAP algorithm.

The *const*-factor in (1) can be eliminated by applying Log-Likelihood Ratios Λ. The LLRs are always calculated in relation to the a posteriori value of the first information group $P(d_k = 0|\vec{\mathbf{y}})$.

Using 1 to 2, we obtain

$$\begin{aligned} \Lambda(d_k = i|\vec{\mathbf{y}}) = \ldots \qquad (3) \\ \max_{(M,M')}{}^* \left[ln(\gamma_i(\mathbf{y}_k, M', M)) + ln(\alpha_{k-1}(M')) + ln(\beta_k(M)) \right] - \\ \max_{(M,M')}{}^* \left[ln(\gamma_0(\mathbf{y}_k, M', M)) + ln(\alpha_{k-1}(M')) + ln(\beta_k(M)) \right] \end{aligned}$$

The information fed to the next MAP decoder is the so called extrinsic information, and comprises the information of (3) without the previous a priori information. In correspondence with $\Lambda(d_k = i) = ln\frac{P(d_k=i)}{P(d_k=0)}$ the extrinsic information evaluates to

$$lnPr\{S_k|S_{k-1}\} = \begin{cases} 0 & \text{for } q(d_k = 0|S_k, S_{k-1}) = 1 \\ \Lambda(d_k = i) & \text{for } q(d_k = i|S_k, S_{k-1}) = 1. \end{cases} \tag{4}$$

The extrinsic information of the first information group is zero and is not passed between the two component decoders.

3. ITERATION CONTROL

The number of iterations needed for decoding differs from block to block. Sometimes decoding is not possible at all, thus the iteration process can be stopped immediately. For implementation issues unprofitable iteration runs waste power and time. An effective control mechanism is therefore necessary that identifies undecodable blocks or stops the iteration when the block is error free.

The presented stop criterion is based on the observation of the extrinsic information (4) passed between the two MAP decoders. It is sufficient to consider *only one* out of three extrinsic information sequences. The decision is built on the sum of the absolute extrinsic information: $\mathcal{L}_j = \sum_{k=1}^{N} |\Lambda(d_k = 1)_j|$. We refer to this value as *sum reliability*. The index $j \in \{1,2\}$ labels the output of MAP1 or MAP2 with N being the blocksize and $d_k = 1$ the first information group at time step k. As long as the sum reliability increases more than a threshold δ between two successive iterations the MAP decoder may be able to decode the transmitted block correctly. If the gain of $\mathcal{L}_j$ is smaller than δ it is a wasted effort to continue decoding and the iteration process can be stopped. This criterion gives no information whether the block is error free or not. The primer goal of the $\mathcal{L}_j$ observation is to detect undecodable blocks.

To determine if a decoded block is error-free a cyclic redundancy check (CRC) [5] is employed. Transmitting the check sum slightly decreases the rate R, but provides an almost perfect criterion to detect error-free blocks.

The combination of the $\mathcal{L}_j$ observation and the CRC check is an effective way to detected decodable and undecodable blocks [9].

4. MAP ARCHITECTURE

The MAP architecture requires three memory blocks: the channel values for the received block have to be stored and also the a priori values passed between the MAP decoders. The memory size of these values depends on the quantization and the block-length and can not be reduced. For the LLR calculation only the alpha values have to be stored. The beta values and LLR can be calculated in one step. To reduce the alpha memory size the well known windowing technique is used [2]. In the Max-Log-MAP algorithm the extrinsic information is scaled by a factor before saturation.

4.1. Extrinsic Scaling Factor

Implementation of the Log-MAP algorithm requires the realization of the correction term according to (2), which pro-

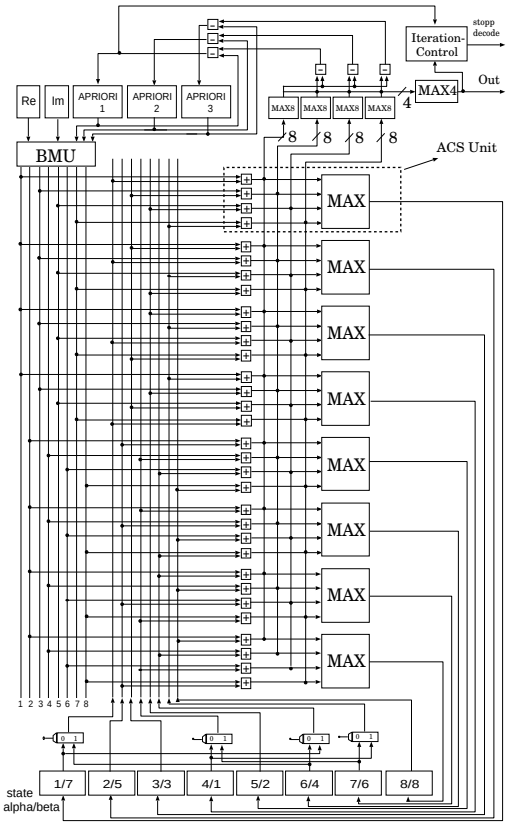

Fig. 3. Architecture of the Recursion Unit

longs the critical path in a Log-MAP implementation [9]. Since the Max-Log-MAP algorithm omits the correction term, its implementation is faster and smaller but has a degradation of its communication performance. However the performance of the TTCM system based on Max-Log-MAP decoders can be improved by using an extrinsic scaling factor (ESF) [4].

The optimal ESF value (determined by simulation) is $ESF = 0.7$. For fixed-point implementation an $ESF = 0.75$ is used which can be implemented by shift operations and the communication degradation is negligible.

4.2. Recursion Unit

The recursion unit is the basic building block of a MAP architecture (Fig. 3). For medium throughput (up to 5 Mbit/s) the recursions α and β can be folded on the same hardware. Only four additional multiplexer are needed switching between forward and backward recursion. The registers for the α and β values are shared - the mapping of these values are optimized with respect to minimum number of multiplexers (Numbers in registers in Fig.3 denote the trellis state value for forward and backward recursion respectively).

The add compare select (ACS) unit is the critical part for implementation. Pipelining has no advantage due to the recursive structure. The difference to a binary MAP architecture is the increased number of max^* calculations (4 instead of 2). Omitting the correction term in (2) reduces the critical path significantly and strengthen the usage of a simple maximum search in combination with ESF. The α and β values can accumulate during recursion without bounds and therefore require periodic re-normalization to prevent arithmetic overflow. This can be achieved by a rescaling approach or by using modulo arithmetics [9]. The rescaling approach prolongs the critical path, therefore the modulo re-normalization is applied. The key idea is to accommodate overflow in such a way that it does not affect the correctness of the results. This allows to operate at a higher clock frequency and is explained in detail in [9].

The branch metric unit (BMU) calculates $\gamma(\cdot)$ by using the real (Re) and (Im) part of the channel values and the three feed back LLR values saved as APRIORI1-3.

5. RESULTS

The channel was modeled to be AWGN with the one-sided noise power spectral density N_0. Bit Error Rate (BER) and block error rate (BLER) simulations are based on blocksize $N = 1024$ with the TTCM system of Section 2 (8-state, 8-PSK).

Fig. 4 shows the performance of the TTCM with Log-MAP and Max-Log-MAP component decoders in the floating-point domain and two fixed-point Max-Log-MAP with $ESF = 075$ implementations. Plotted is the 0., 1. and 4. iteration. Obviously, in the floating point domain the Max-Log-MAP has a performance degradation compared to the Log-MAP which is $\sim 0.3dB$ at $BER = 10^{-4}$ after the fourth iteration. For the fixed-point implementation the input data quantization (bitwidth,fractional part) is $I : (7,4)$, $I : (8,5)$. The extrinsic information is saturated with one bit more than the bit-width of the input data. For $I : (8,5)$ the performance degradation is just 0.15dB compared to the floating-point Log-MAP with perfect SNR knowledge. A further increment of the granularity $I : (9,6)$ does not improve the decoding performance significantly. A smaller input quantization $I : (7,4)$ will lead to a degradation of the decoding performance. These results show that the smaller and faster Max-Log-MAP implementation combined with ESF becomes very attractive becomes very attractive from an implementation point of view.

Fig. 5 shows the BLER and the number of iterations for different stop criteria: a perfect curve (maximum of 8. iterations), a fixed number of 4 iterations, the CRC stopp criterion, and the combination of CRC and the mean reliability observation. The CRC check has a nearly perfect BLER performance. A fixed number of 4 iterations has a performance degradation of 0.1dB at $FER = 10^{-2}$. The new stop criterion has only a degradation of 0.05dB.

The number of iterations needed for different criteria is depicted in Fig. 5(b). The perfect criterion assumes that

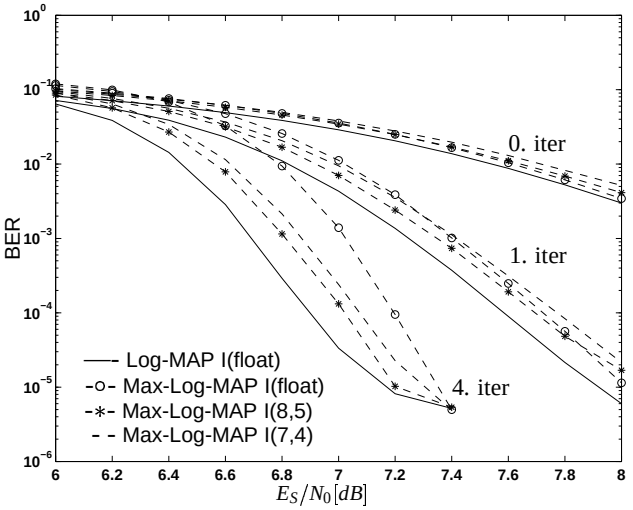

Fig. 4. Quantization of TTCM with Max-Log-MAP decoders with $ESF = 0.75$

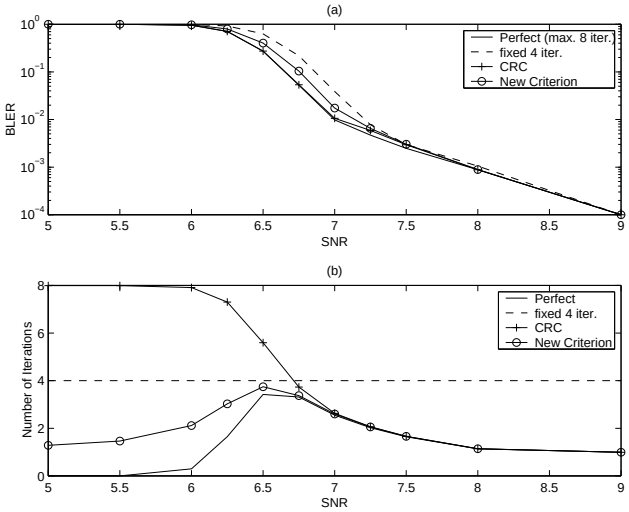

Fig. 5. (a) Block-Error-Rate for different stop criteria. (b) Number of iterations for different stop criteria.

for an undecodable block the iteration process is stopped after the 0. iteration. For a decodable block the iteration is stopped with a minimum number of iterations. In the lower SNR range ($< 6dB$) the CRC needs the full number of 8 iterations. The sum reliability criterion needs at least one iteration more than the perfect number of iterations. For the high SNR range ($> 7.5dB$) the CRC criterion and the combined scheme are nearly perfect. A fixed number of 4 iterations needs over the overall SNR range more iterations and has a worth BLER than the new stop criteria.

6. CONCLUSIONS

We have demonstrated that techniques known form the binary Turbo-decoder can be applied to TTCM to reduce the implementation complexity significantly. The MAP architecture is discussed with special emphasis on the recursion unit. It is possible to process the forward and backward recursion on the same hardware with negligible overhead. For fixed-point implementation the Max-Log-MAP algorithm in combination with $ESF = 0.75$ is proposed. Simulation results show that this system with an input quantization of 8 bits and 9 bits for the LLR values is a reasonable compromise between implementation complexity and degradation of decoding performance. The performance degradation is only about 0.15 dB for AWGN channels, compared to a floating-point Log-MAP implementation. An iteration control is presented that detects decodable and undecodable blocks. The iteration control is a combination of CRC and the observation of the mean reliability in particular defined for TTCM. The new method is superior to a fixed number of 4. iterations over the overall SNR range.

7. REFERENCES

[1] C. Berrou, A. Glavieux, and P. Thitimajshima. Near Shannon Limit Error-Correcting Coding and Decoding: Turbo-Codes. In *Proc. 1993 International Conference on Communications (ICC '93)*, pages 1064–1070, Geneva, Switzerland, May 1993.

[2] H. Dawid, G. Gehnen, and H. Meyr. MAP Channel Decoding: Algorithm and VLSI Architecture. In *VLSI Signal Processing VI*, pages 141–149. IEEE, 1993.

[3] S. L. Goff, A. Glavieux, and C. Berrou. Turbo-Codes and High Spectral Efficiency Modulation. In *Proc. 1994 International Conference on Communications (ICC '94)*, pages 645–649, May 1994.

[4] A. F. J. Vogt. Improving the Max-Log-MAP Turbo Decoder. *IEEE Electronic Letters*, 36, 2000.

[5] K. R. Narayanan and G. Stüber. A Novel ARQ Technique using the Turbo Coding Principle. *IEEE Communications Letters*, 1(2):49–51, Mar. 1997.

[6] P. Robertson, P. Hoeher, and E. Villebrun. Optimal and Sub-Optimal Maximum a Posteriori Algorithms Suitable for Turbo Decoding. *European Transactions on Telecommunications (ETT)*, 8(2):119–125, March–April 1997.

[7] P. Robertson and T. Wörz. Bandwidth-Efficient Turbo Trellis-Coded Modulation Using Punctured Component Codes. *IEEE Journal On Selected Areas In Communications*, 16(2):206–218, Feb. 1998.

[8] J. Vogt, K. Koora, A. Finger, and G. Fettweis. Comparison of Different Turbo Decoder Realizations for IMT-2000. In *Proc. 1999 Global Telecommunications Conference (Globecom '99)*, volume 5, pages 2704–2708, Rio de Janeiro, Brazil, Dec. 1999.

[9] A. Worm, H. Michel, F. Gilbert, G. Kreiselmaier, M. J. Thul, and N. Wehn. Advanced Implementation Issues of Turbo-Decoders. In *Proc. 2nd International Symposium on Turbo Codes & Related Topics*, pages 351–354, Brest, France, Sept. 2000.