Low Power Implementation of a Turbo-Decoder on Programmable Architectures[1]

Frank Gilbert, Alexander Worm, Norbert Wehn
Institute of Microelectronic Systems
Department of Electrical Engineering and Information Technology, University of Kaiserslautern
Email: {gilbert,worm,wehn}@eit.uni-kl.de

Abstract—**Low Power is an extremely important issue for future mobile radio systems. Channel decoders are essential building blocks of base-band signal processing units in mobile terminal architectures. Thus low power implementations of advanced channel decoding techniques are mandatory. In this paper we present a low power implementation of the most sophisticated channel decoding algorithm (Turbo-decoding) on programmable architectures. Low power optimization is performed on two abstraction levels: on system level by the use of an intelligent cancellation technique, on implementation level by the use of dynamic voltage scaling. With these techniques we can reduce the worst case energy consumption to 55% using data of state-of-the-art processors. Our approach is also applicable for hardware implementations. To the best of our knowledge, this is the first in-depth study of low power implementations of Turbo-decoders based on voltage scheduling for third generation wireless systems.**

I. Introduction

Channel coding and decoding are essential components in mobile terminal architectures. Channel coding is necessary to correct errors which are induced by noisy channels. Forward error correction (FEC) is a very common technique to correct these errors as much as possible: on the transmitter side, a channel encoder adds redundancy to the data to be transmitted. This redundancy is exploited on the receiver side by the channel decoder to correct the errors. A common measure for the performance of a coding scheme is the bit-error-rate (BER) as a function of the signal-to-noise ratio (SNR) E_s/N_0. Turbo-Codes [1], first published in 1993, show the best forward error correction performance known up to now. Thus Turbo-Codes became part of the third generation wireless systems (UMTS) [2] standard which offers high data-rate services (up to 2MBit/s) for internet and multimedia applications. Turbo-Codes are also in discussion for other wireless standards, e. g. WLAN applications.

Software implementation of the baseband functionality is an important issue in future multistandard mobile terminal architectures (Software Radio). Low power implementation of Turbo-Codes on programmable architectures is therefore a relevant matter.

The complexity of a Turbo-decoder is much higher than the complexity of the encoder (see Section III-A). Thus we put emphasis only on the decoder. Turbo-Codes work on block level, i. e. the data is separated into blocks (a typical block length is, e. g., 600 symbols). On the decoder side such a block is decoded in an iterative procedure [1]. It can be shown that for practical implementations a fixed number of iterations is sufficient to obtain the desired BER at a given SNR. Thus the majority of the published implementations use a fixed number of iterations (typical values are in the range of 5 to 10). However as the channel characteristic (e. g. SNR) in mobile systems can change over time (fading), some blocks can be decoded with less iterations (better channel characteristics), whereas other blocks cannot be decoded (worse channel characteristics), even if the number of iterations is increased. The last case is especially important in packet oriented network protocols like TCP/IP. Intelligent power control mechanisms as in wideband CDMA (air interface in UMTS [3]) try to compensate this fading, however these techniques are imperfect and induce latency. This "dynamic throughput behavior" can be exploited to save power. To achieve this, *stopping criteria* on *system level* are necessary which stop the iteration process as soon as possible.

Assuming that such stopping criteria exist, energy can be saved on *implementation level* by either shutting down the processor or using dynamic voltage scheduling which is much more energy efficient. Recent developments in microprocessor technology, e. g. the Crusoe Processor from Transmeta [4], offer the possibility to operate over a range of supply voltages and can vary their clock speed (f_{CLK}) and supply voltage (V_{DD}) at run time depending on actual throughput requirements ("Dynamic Voltage Scaling").

II. Related Work

In [5], dynamic voltage scheduling (DVS) is referred to as "the most revolutionary low power technique to date" and is superior to a processor power-down mode due to the square dependency of the supply voltage on the energy. Since variable voltage processors like the Transmeta processor enter mainstream technology, this technique will become more and more important. Recent results on the theory and technical viability of DVS have been published in [6], [7], [8], [9]. DVS can only be beneficial if the maximum CPU speed is needed just in a fraction of time. Thus, a fixed iteration implementation of a Turbo-decoder processing at maximum speed can not benefit from DVS.

In the case that the channel charactistics *improve* temporarily compared to the working point, the iterations can be stopped as soon as the overall quality of the decoded bits is sufficient. Techniques presented in [10], [11] can be used to derive such a criterion. However, the most efficient technique is the use of cyclic redundancy codes (CRC) [12]. This requires that the data frames are protected by a CRC-check. Depending on the CRC-check result, the iteration can be stopped or has to be continued. CRC-checks are common techniques used in standards like UMTS [2]. The computational complexity of this check is very low.

The reverse case, i. e. temporarily *worse* channel characteristics compared to the working point, is more complex to treat. It can be shown that in such cases (under realistic wireless scenarios) the required BER cannot be maintained even with the maximum number of iterations. To the best of our knowledge

[1]This work has been supported by the *Deutsche Forschungsgemeinschaft (DFG) under Grant We 2442/1-1 within the* Schwerpunktprogramm "Grundlagen und Verfahren verlustarmer Informationsverarbeitung (VIVA)"

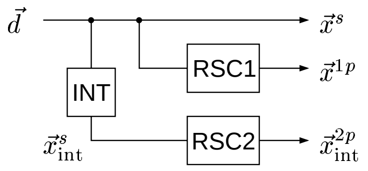
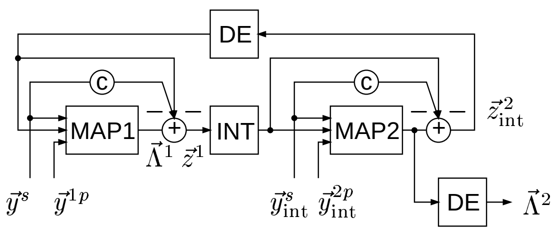


Fig. 1. Turbo-Code Encoder



DE
MAP1
INT
MAP2
DE
c
$\vec{\Lambda}^1$ $\vec{z}^1$
$\vec{z}^2_{\text{int}}$
$\vec{y}^s$ $\vec{y}^{1p}$
$\vec{y}^s_{\text{int}}$ $\vec{y}^{2p}_{\text{int}}$
$\vec{\Lambda}^2$

Fig. 2. Turbo-decoder

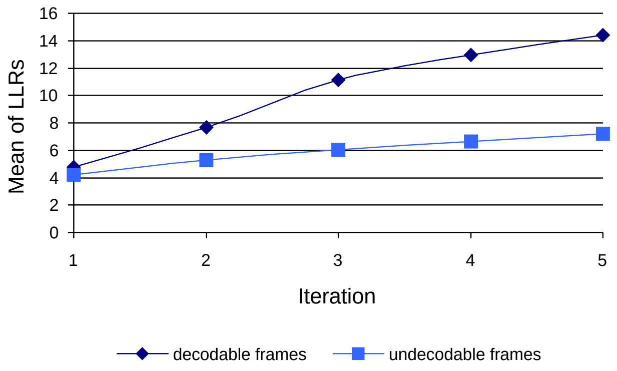


Fig. 3. Mean of LLRs for decodable and undecodable frames at SNR = -2.00 dB

a stop criterion for this case has not been addressed by current literature. Simply using an estimated channel SNR as a criterion is too coarse and may trash too many decodable blocks.

Only one approach in literature applies DVS to Turbo-Codes [13]. CRC-protection is added to each data frame and is checked after every iteration. An ASAP (as-slow-as-possible) and an SNR-assisted DVS-heuristic is presented which adjusts the decoding speed and supply voltage. However this paper lacks several issues: only temporarily *improved* channel characteristics are considered. Thus a lot of optimization potential is lost. The DVS heuristic is based on SNR estimates, but SNR estimation in the context of Turbo-codes is unnecessary [14]. The paper lacks simulations of realistic wireless environment scenarios and power models.

III. New Approach

We consider packet oriented network protocols such as TCP/IP. In these protocols, the number of bit errors per data frame is of no interest. Faulty frames will be discarded and must be sent again. Thus we consider in the following frame-error-rates (FER) instead of BER. In this paper we present for the first time an in-depth study of low power implementations of Turbo-decoders based on voltage scheduling for third generation wireless systems [2] on processors which can operate with variable supply voltages. In detail it contributes to the following new issues:

- We present a new approach for early stopping the iterations over the full SNR range for AWGN and Rayleigh fading channels.
- A new robust voltage scheduling algorithm is introduced which adjusts the supply voltage and frequency.
- The power model used to evaluate the benefit of our approach is based on commercially available state-of-the-art processor data sheets.
- We have validated our approach by extensive simulations based on realistic wireless scenarios of UMTS and demonstrate that the worst case energy consumption can be reduced to 55% compared to a traditional approach with a fixed iteration scheme.

Our approach is not limited to software implementations, it can be also used for accelerator hardware.

A. Turbo-Decoder Model

We refer to Turbo-Codes as presented in [1]. The assumed Turbo-decoder system model (see Fig. 2) comprises two maximum a posteriori (MAP) decoders (MAP1, MAP2), an interleaver (INT), and two deinterleavers (DE). These are the building blocks of the iterative Turbo-decoding process. Details of the Turbo-decoder system model and the MAP algorithm are given in [15] and [16].

If not otherwise stated, an AWGN channel model is applied. All investigations are based on the following Turbo-Encoder/Decoder details: constraint-length $K = 4$, generator polynoms 15/13, frame length $N = 600$, code rate $R = 1/2$, 3GPP interleaver, no tailing scheme, Log-MAP decoders, a maximum of 5 iterations, and a quantization model of 9 bit total and 4 bit fractional part.

B. Cancellation Algorithm

The Turbo-decoder calculates the logarithmic likelihood ratios (LLRs) $\Lambda^2(d_k)$ as soft-outputs at the end of each iteration. The signs of these numbers indicate the respective 0/1-decision. The absolute values are measures of the confidence in the respective 0/1-decision. Our Turbo-Code model uses saturation to restrict the bit-width during calculation. As a result, all LLRs lie in the range between -16 and 16. Therefore an absolute value of 16 indicates the highest confidence in the decoded bit. Furthermore, we can calculate the mean $\mu_{\vec{\Lambda}^2}$ of the absolute values of the LLRs as a measure of confidence in the decoding of the entire frame.

Our investigations show that the mean value $\mu_{\vec{\Lambda}^2}$ increases from iteration to iteration, saturating as it approaches the maximum value of 16. Moreover, the mean value $\mu_{\vec{\Lambda}^2}$ after the first iteration is higher for higher channel qualities.

If we compare $\mu_{\vec{\Lambda}^2}$ of decodable and undecodable data frames, the increment from iteration to iteration is quite different (see Fig. 3). The mean of a undecodable data frame is well beneath the mean of a decodable frame. The slope is much lower for undecodable frames as well. As the difference is quite remarkable, this behavior may serve as a stop criterion for undecodable data frames. We optimized the bounds for the mean value $\mu_{\vec{\Lambda}^2}$ through simulation and developed an "intelligent cancellation" algorithm to detect the undecodable data frames as early as possible (Algorithm 1). The standard deviation $\sigma_{\vec{\Lambda}^2}$ may also serve as a stop criterion, but the mean is preffered, because

Algorithm 1 Intelligent Cancellation Algorithm

```
{$\mu_{th}$[] : threshold values for mean of $\mu_{\vec{\Lambda}}$}
{$\Delta\mu_{th}$[] : threshold values for diff. of succ. mean val.}
{LLR : Variable LLR is ARRAY of REAL}
$\mu \Leftarrow 0$,    $iter \Leftarrow 1$
**while** remaining decoding time **do**
    $LLR \Leftarrow$ (Max)LogMAPIteration( $frame$ )
    **if** CRCCheck( $LLR$ ) $\neq$ ERROR **then**
        **exit while**    {block successfully decoded}
    **end if**
    $\mu_{old} \Leftarrow \mu$,    $\mu \Leftarrow$ Mean( $LLR$ )
    $\Delta\mu \Leftarrow \mu - \mu_{old}$
    **if** $\mu < \mu_{th}[iter]$ **or** $\Delta\mu < \Delta\mu_{th}[iter]$ **then**
        **exit while**    {block undecodable}
    **end if**
    $iter \Leftarrow iter + 1$
**end while**
```

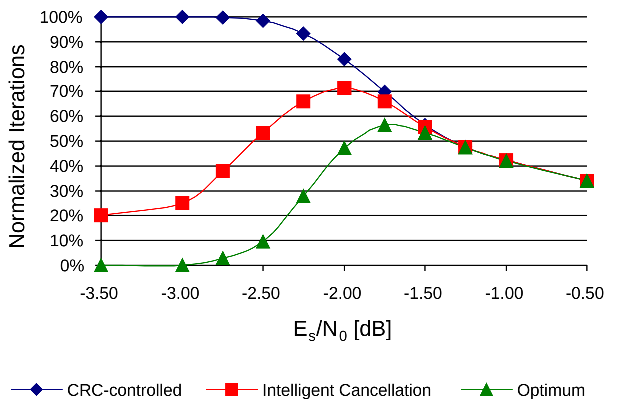


Fig. 4. Normalized number of iterations of stop-criteria

it is easier to calculate, it is preferred. Recently our approach was confirmed theoretically by [17].

Fig. 4 shows the results of our cancellation method. The normalized number of iterations is compared to a CRC-controlled scheme. Our method reduces the number of iterations over the complete SNR range. The algorithms implies some degradation in the FER. However, this is negligible: smaller than 1.5% at -2.25 dB (worst case).

We proved our method to be applicable for Rayleigh fading channels, too (data not shown for space limitation).

C. Dynamic Voltage Scheduling Algorithm

For the terminology of dynamic voltage scaling we refer to [18]. The following systems assumptions are made:

All iterations are processed on a single variable supply voltage processor, that can vary its clock speed (f_{CLK}) and supply voltage (V_{DD}) at run time. This is a fair assumption since variable supply voltage processors now enter mainstream technology, e. g. the Crusoe processor from Transmeta [4].

Energy consumption of the processor during idle cycles is neglected since DVS reduces the idle time by slowing down the clock speed (remaining idle cycles are a minority).

The system must provide the next data frame before completion of the last frame. Turbo-decoding iteration is processed in a constant number of clock cycles, which is a fair assumption, since the (Log-)MAP algorithm shows no data dependent runtime, and the number of clock cycle per iteration only depends on the frame length. The switching activity per iteration is constant over all data frames.

Let T be the frame periods, i. e. the time between the arrival of successive frames, and L be the latency requirement in terms of frame period (DVS requires $L > 1$). We assume that for a targeted system throughput the processor is able to process the maximum number of iterations at its peak performance within the frame period T.

To simplify the DVS algorithm, we consider five discrete iteration delays: $\frac{1}{5}T$, $\frac{2}{5}T$, $\frac{3}{5}T$, $\frac{4}{5}T$, and T. An iteration delay of $\frac{1}{5}T$ is equivalent to the maximum speed of 5 iterations per frame period (common value for a fixed iteration scheme). In our model these iteration delays correspond to the multiples 1, $\frac{1}{2}$, $\frac{1}{3}$, $\frac{1}{4}$ and $\frac{1}{5}$ of the maximum clock frequency f_{max}.

The DVS-heuristic presented in [13] has one major drawback: an estimation of the current SNR is required to predict the number of iterations. In a realistic environment, this information may be inaccurate, is varying quickly, or is not available at all. Thus it is not a robust algorithm. Our algorithm adjusts the processor supply voltage and clock frequency for iteration j of frame i on the basis of the remaining decoding time satisfying the latency requirement δ and the average number of iterations of the last three data frames (Algorithm 2). At first an estimated number of iterations I_s is calculated. An offset is added to the average of iterations, as a more pessimistic assumption will result in a better energy consumption. The speed is chosen to average the workload during the estimated number of iterations, taking in regard that all remaining iterations are processed at maximum CPU speed. In our normalized energy models a minimum CPU speed of $\frac{1}{3}f_{max}$ is required (i. e. $\frac{3}{5}T$ iteration delay) as the targeted processors are not specified to operate below a

Algorithm 2 VS-heuristic: iteration j of frame i

```
{$\delta$ : remaining decoding time in terms of $\frac{1}{5}T$}
{$t_{i,j}$ : iteration delay in terms of $\frac{1}{5}T$}
{$I_{max}$ : maximum number of iterations (=5)}
{$I_s$ : assumed number of iterations for frame i}
{$\#iter(frame_{i-k})$: number of iterations for frame $i-k$}
{Estimate number of iterations}
$I_s = \left\lfloor \frac{\#iter(frame_{i-1}) + \#iter(frame_{i-2}) + \#iter(frame_{i-3})}{3} + 1.7 \right\rfloor$
**if** $I_s > I_{max}$ **then**
    $I_s = I_{max}$
**end if**
{Calculate iteration delay}
**if** $j > I_s$ **then**
    $t_{i,j} = \delta - (I_{max} - j)$
**else**
    $t_{i,j} = \left\lceil \frac{\delta - (I_{max} - I_s)}{I_s - j + 1} \right\rceil$
**end if**
**if** $t_{i,j} > 3$ **then**
    $t_{i,j} = 3$   {4 and 5 must not be selected}
**end if**
**SetProcessorSpeed**( $t_{i,j} \times \frac{1}{5}T$ )
```

TABLE I

NORMALIZED ENERGY MODEL OF STARCORE SC140 DSP

Iter. *Delay*	f_{CLK} [MHz]	V_{DD} [V]	$P_{\mu P}$ [mW]	E_{Cycle} [nJ]	E_{Norm} [%]
$\frac{1}{5}T$	300	1.50	198.00	0.660	100.0
$\frac{2}{5}T$	150	1.00	44.00	0.293	44.4
$\frac{3}{5}T$	100	0.83	20.37	0.204	30.9
$\frac{4}{5}T$	*75*	*0.75*	*12.38*	*0.165*	*25.0*
T	*60*	*0.70*	*8.62*	*0.144*	*21.8*

minimum clock frequency and supply voltage. The normalized energy of each iteration depends on the scheduled iteration delay and the targeted processor (see Tab. I). Note that DVS reduces energy without further degradation of the FER.

D. Processor Models

Different state-of-the-art processor models are considered to quantify the advantage of our new DVS algorithm in combination with our cancellation critierion. The following processors were investigated: StarCore SC140 (Motorola/Lucent Technologies [19]) and Crusoe processor (Transmeta [4]).

The Crusoe processor is a VLIW general-purpose microprocessor optimized for low power applications. It is reported to operate over a complete range of supply voltages, from 200MHz (1.1V) to 700MHz (1.65V). The clock frequency vs. supply voltage curve is nearly a linear function of the supply voltage.

The StarCore SC140 is a state-of-the-art VLIW low power digital signal processor for advanced communications systems. The StarCore SC140 operates at 300MHz (1.5V) and 120MHz (0.9V). A continuous range of supply voltages and clock frequencies is not reported by now. We assume a linear clock frequency vs. supply voltage curve between the two specified operation modes. The StarCore is not specified to operate below 120MHz (0.9V).

The iteration delays $\frac{4}{5}T$ and T are theoretical vaules and not considered by the DVS algorithms. The normalized energy models of the StarCore are presented in Tab. I (unselectable iteration delays are separated).

IV. RESULTS

Fig. 5 shows the VS-algorithm results for $50,000$ frames, targeted at the StarCore SC140. Obviously the DVS algorithm amplifies the intelligent cancellation method over the complete SNR range, whereas a pure CRC-based DVS algorithm fails to reduce the normalized energy for low SNR.

The StarCore SC140 has the best $\frac{(V_{DD_{min}})^2}{(V_{DD_{max}})^2}$ and has therefore been chosen as the target processor. For the Crusoe the VS-gain is degraded by about 5%.

V. CONCLUSIONS AND FUTURE WORK

In mobile radio systems, channel characteristics can change quickly. An early cancellation of undecodable data frames combined with a CRC-controlled stopping of the Turbo-decoder iterations can significantly reduce the number of required iterations over the complete SNR range. The idle time is exploited by DVS reducing the worst case power consumption to 55%.

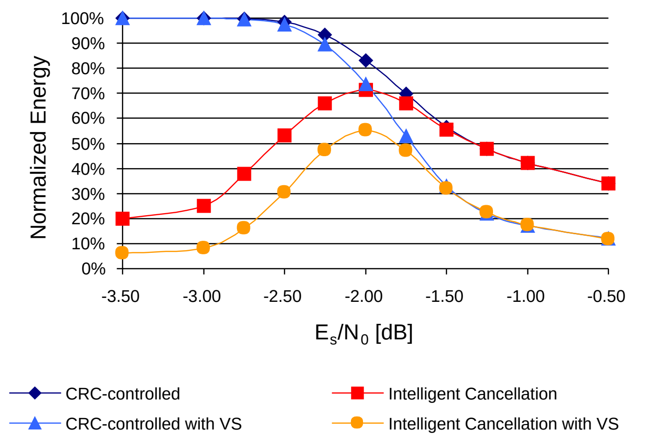


Fig. 5. Normalized energy consumption of Turbo-decoding

Our future work will concentrate on the following issues: refinement of our models and consideration of multi-processor or multi-core environments.

REFERENCES

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

[2] 3GPP, "Third Generation Partnership Project," `http://www.3gpp.org`, 2000.

[3] T. Ojanperä and R. Prasad, Eds., *Wideband CDMA for Third Generation Mobile Communications*, Artech House Publishers, 1998.

[4] Transmeta Corporation, "The Crusoe Processor," `http://www.transmeta.com/crusoe/`, 2000.

[5] B. Ackland and C. Nicol, "High Performance DSPs – What's Hot and What's Not?," in *ISLPED '98*, Aug. 1998, pp. 1–6.

[6] M. Weiser, B. Welch, A. Demers, and S. Shenker, "Scheduling for Reduced CPU Energy," in *Proc. Symposium on Operating Systems Design and Implementation*, Nov. 1994, pp. 13–23.

[7] T. Pering, T. Burd, and R. Brodersen, "The Simulation and Evaluation of Dynamic Voltage Scaling Algorithms," in *ISLPED '98*, Aug. 1998, pp. 76–81.

[8] T. Ishihara and H. Yasuura, "Voltage Scheduling Problem for Dynamically Variable Voltage Processors," in *ISLPED '98*, Aug. 1998, pp. 197–202.

[9] T. Burd, T. Pering, A. Stratakos, and R. Brodersen, "A Dynamic Voltage Scaled Microprocessor System," in *Proc. ISSCC '00*, Feb. 2000, pp. 294–295.

[10] J. Hagenauer, E. Offer, and L. Papke, "Iterative Decoding of Binary Block and Convolutional Codes," *IEEE Transactions on Information Theory*, vol. 42, no. 2, pp. 429–445, Mar. 1996.

[11] R. Y. Shao, S. Lin, and M. C. P. Fossorier, "Two Simple Stopping Criteria for Turbo Decoding," *IEEE Transactions on Communications*, vol. 47, no. 8, pp. 1117–1120, Aug. 1999.

[12] K. R. Narayanan and G. Stüber, "A Novel ARQ Technique using the Turbo Coding Principle," *IEEE Communications Letters*, vol. 1, no. 2, pp. 49–51, Mar. 1997.

[13] Leung, Yue, Tsui, and Cheng, "Reducing Power Consumption of Turbo-Code Decoder Using Adaptive Iteration with Variable Supply Voltage," in *ISLPED '99*, Aug. 1999, pp. 36–41.

[14] A. Worm, P. Hoeher, and N. Wehn, "Turbo-Decoding without SNR Estimation," *IEEE Communications Letters*, vol. 4, no. 6, June 2000.

[15] P. Robertson, P. Hoeher, and E. Villebrun, "Optimal and Sub-Optimal Maximum a Posteriori Algorithms Suitable for Turbo Decoding," *ETT*, vol. 8, no. 2, pp. 119–125, Mar.–Apr. 1997.

[16] P. Hoeher, "New Iterative ("Turbo") Decoding Algorithms," in *Proc. International Symposium on Turbo Codes & Related Topics*, Sept. 1997, pp. 63–70.

[17] D. Divsalar, S. Dolinar, and F. Pollara, "Low Complexity Turbo-like Codes," in *Proc. 2nd International Symposium on Turbo Codes and Related Topics*, Brest, France, 2000, pp. 73–80.

[18] A. P. Chandrakasan and R. W. Broderson, *Low Power Digital CMOS Design*, Kluwer Academic Publishers, 1995.

[19] Motorola and Lucent Technologies, "StarCore DSP," `http://www.starcore-dsp.com/`, 2000.