Power minimization by optimizing data transfers in Turbo-decoders[*]

A. Worm[†], H. Michel, and N. Wehn

University of Kaiserslautern, Germany

Camera-ready Copy for

Kleinheubacher Berichte

Manuscript-No.

Offset requests to:
A. Worm
Universität Kaiserslautern
Fachbereich Elektrotechnik und Informationstechnik
AG Mikroelektronische Systeme
Erwin-Schrödinger-Straße
D-67663 Kaiserslautern
Germany
`worm@e-technik.uni-kl.de`

[*]Part of this work has been supported by the *Deutsche Forschungsgesellschaft (DFG)* under grant We 2442/1-1 within the *Schwerpunktprogramm „Grundlagen und Verfahren verlustarmer Informationsverarbeitung (VIVA)“.*

[†]Alexander Worm is the recipient of a Motorola Partnerships in Research Grant.

Power minimization by optimizing data transfers in Turbo-decoders*

A. Worm†, H. Michel, and N. Wehn
University of Kaiserslautern, Germany

*Part of this work has been supported by the *Deutsche Forschungsgesellschaft (DFG)* under grant We 2442/1-1 within the *Schwerpunktprogramm „Grundlagen und Verfahren verlustarmer Informationsverarbeitung (VIVA)“*.
†Alexander Worm is the recipient of a Motorola Partnerships in Research Grant.

Abstract. The power consumption of (embedded) microelectronic systems is dominated by energy-consuming accesses to memories, both on-chip and off-chip. An off-chip memory-access consumes up to 33 times more energy compared to an arithmetic operation. Therefore, the potential to minimize power consumption by rearranging (and eventually avoiding) memory accesses has to be exploited before addressing data-path optimizations.

Since their invention in 1993, Turbo-Codes have been a hot topic among coding theorists. The anticipated use in future mobile radio systems also raises interest among implementation specialists. However, although the computational complexity of Turbo-Code's MAP component decoders is well known, the memory accessing scheme is the real bottleneck for low-power implementations, which are essential for building cheap battery-operated mobile devices.

This paper therefore focuses on data-transfer optimizations. The number of data-transfers is a function of system-level parameters, algorithmic transformations, and implementation parameters. It will be shown how the data-transfer related energy consumption of Turbo-decoders can be significantly reduced by taking proper design choices. Especially the effects of algorithmic transformations like windowing and some control/data-flow transformations will be discussed.

Zusammenfassung. Der Leistungsverbrauch (eingebetteter) mikroelektronischer Systeme wird von energieaufwendigen internen und externen Speicherzugriffen dominiert. So ist der Energiebedarf externer Speicherzugriffe bis zu 33mal höher als der einer arithmetischen Operation. Vor der Anwendung von Datenpfadoptimierungen zur Verringerung des Leistungsverbrauchs ist deshalb das Optimierungspotential auszuschöpfen, welches sich durch Umordnen und Vermeiden von Speicherzugriffen ergibt.

Seit ihrer Erfindung im Jahre 1993 werden Turbo-Codes unter Codierungstheoretikern intensiv diskutiert. Neuerdings zieht der erwartete Einsatz in zukünftigen Mobilfunksystemen auch das Interesse von Implementierungsspezialisten auf sich. Während man sich jedoch der hohen Rechenkomplexität der MAP-Komponentendecodierer bewußt ist, liegt der wirkliche Engpaß beim Speicherzugriffsschema, besonders bei Anwendungen, die kritisch hinsichtlich des Leistungsverbrauchs sind, wie z. B. kostengünstige, batteriebetriebene mobile Endgeräte.

Der vorliegende Artikel konzentriert sich deshalb auf Datentransferoptimierungen. Die Anzahl der Datentransfers ist eine Funktion von Systemparametern, algorithmischen Transformationen und Implementierungsparametern. Es wird gezeigt, wie der durch Datentransfers verursachte Anteil am Energieverbrauch von Turbo-Decodierern durch geeignete Entscheidungen während der Implementierungsphase signifikant verringert werden kann. Dabei werden insbesondere die Auswirkungen von algorithmischen Transformationen diskutiert, wie z. B. Fensterung und einige Kontroll-/Datenflußtransformationen.

1 Introduction

In digital communications, errors are induced by noisy channels. To combat these errors, forward error correction (FEC) is applied. FEC comprises two parts: In the sender, a channel encoder adds redundancy to the data to be transmitted. This redundancy is exploited by a channel decoder, which is located in the receiver, to detect and correct errors. The error correction capability is theoretically limited (Shannon, 1948a,b). This limit is (still) unreached by practical coding methods. 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 (E_b/N_0). In Fig. 1, this function is plotted for

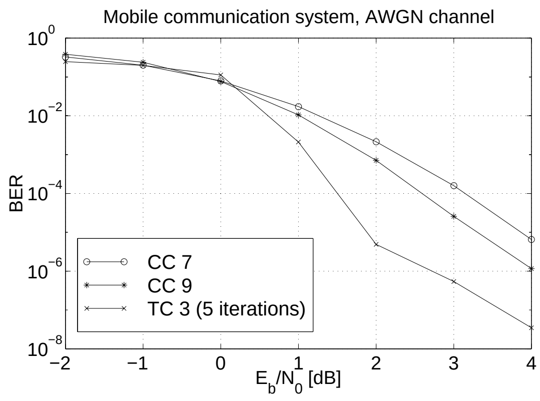

Figure 1. BER $= f(E_b/N_0)$ comparison of convolutional codes (CC 7, CC 9) and a Turbo-Code (TC 3).

convolutional codes of constraint length $K = 7$ and $K = 9$ and for a Turbo-Code (cf. Section 2) of constraint length $K = 3$. As a precondition for a fair comparison, the decoding complexity of the $K = 9$ convolutional code and the Turbo-Code can be considered approximately equivalent; the encoding complexity can be safely neglected, because the encoders consist of simple shift registers. Assuming a bit error rate of, e. g., 10^{-5}, the gain of employing the $K = 3$ Turbo-Code is about 1.5 dB, which means that the required transmission energy per bit (E_b) to sustain a demanded bit error ratio is considerably reduced. We learn from this comparison that the mere decision to employ Turbo-Codes is a step forward towards minimized power consumption.

When implementing a Turbo-decoder, simplifications for reduced power consumption can be applied at different abstraction levels (Wehn and Münch, 1999). There is a general consensus that the largest optimization potential resides on higher levels of abstraction. Design decisions on system level, e. g., algorithm selection and simplification, have the highest impact on the properties of the resulting design, including low power consumption. However, system level optimization requires a proper understanding of the operation of the algorithms under question. In Section 2 we will therefore give a brief introduction into Turbo-decoding. As suggested in the abstract, data-transfer optimization is one of the most prominent measures to reduce power consumption on system level. In Section 3 we will present a memory power model and deduce the main parameters which influence power consumption. These will be used in Section 4 to evaluate the impact of several algorithmic and control/data-flow transformations on the power consumption caused by data-transfers. Section 5 will finally conclude the paper.

2 Turbo-Codes

Turbo-Codes (Berrou et al., 1993) have been paid considerable attention since their proposal in 1993 because of their unmatched forward error correction performance. Their anticipated use in future mobile radio systems (3GPP, 1999) also raises interest for low-cost low-power implementations. For this discussion, we will use the notation of Robertson et al. (1997). Only some important results are stated here. For a detailed discussion of Turbo-Codes, the reader is referred to, e. g., Berrou et al. (1993), Robertson (1994), Robertson et al. (1997), and Hoeher (1997).

2.1 PCCC and iterative Turbo-decoding

The term “Turbo-Code” originally describes the parallel concatenation of two convolutional codes (PCCC), separated by an interleaver. Each constituent code produces parity outputs ($\vec{x}^{1p}$, $\vec{x}^{2p}_{\text{int}}$), as depicted in Fig. 2 (Hoeher, 1997). The constituent codes are recursive systematic convolutional codes (RSC) with relatively small constraint length, e. g., $K = 3$. The third output delivers the systematic information ($\vec{x}^s$). Each maximum a posteriori (MAP) decoder (Bahl et al., 1974) in the Turbo-decoder is associated with one of the constituent

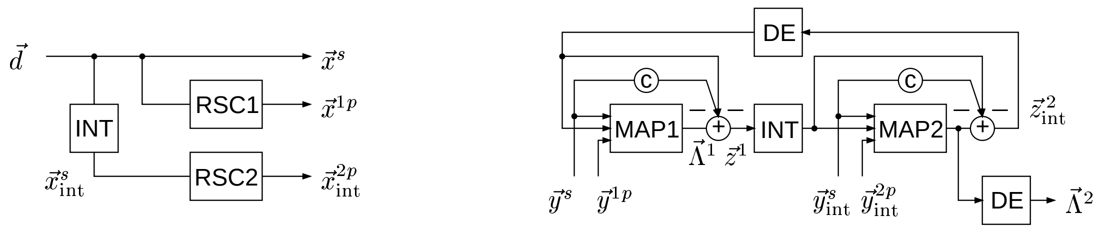

Figure 2. Parallel concatenation of two constituent codes (RSC) in the encoder (left), iterative decoding with MAP-decoders in the decoder (right). INT denotes the interleaver, DE is the corresponding deinterleaver.

codes. Decoder MAP1 (Fig. 2) has three inputs: the systematic input ($\vec{y}^s$), the parity input ($\vec{y}^{1p}$) associated with the output of RSC1, and the extrinsic information ($\vec{z}^2$) of decoder MAP2, which serves as a priori information. Decoder MAP1 produces for each bit d_k of the input data sequence $\vec{d} = (d_1, \dots, d_N)$ the MAP output

$$\Lambda^1(d_k) = \log \frac{\Pr\{d_k = 1 \,|\, \vec{R}\}}{\Pr\{d_k = 0 \,|\, \vec{R}\}}, \quad \text{where } \vec{R} = (R_1, \dots, R_k, \dots, R_N) \text{ with } R_k = (y^s_k, y^{1p}_k, z^2_k), \tag{1}$$

which can be written as

$$\Lambda^1(d_k) = z^1_k + c \cdot y^s_k + z^2_k \text{ with } c = -\frac{4E_s}{N_0}, \tag{2}$$

because the systematic term $c \cdot y^s_k$ and the a priori term z^2_k are regarded as independent of the parity information for the bit d_k. The newly generated extrinsic information can therefore be computed as

$$z^1_k = \Lambda^1(d_k) - c \cdot y^s_k - z^2_k \tag{3}$$

and serves, after interleaving, as a priori information for decoder MAP2:

$$z^2_{k,\text{int}} = \Lambda^2(d_{k,\text{int}}) - c \cdot y^s_{k,\text{int}} - z^1_{k,\text{int}} \tag{4}$$

This procedure iterates several times, until the MAP estimates $\vec{\Lambda}^{1,2}$ stabilize.

2.2 The Log-MAP algorithm

From now on, the superscripts 1 and 2 denoting the decoders MAP1 and MAP2 are dropped for notational convenience. Robertson et al. (1997) pointed out that is is mandatory to implement the MAP decoders in the log-domain (Log-MAP) in order to avoid numerical problems without degrading the decoding performance. The reader is again referred to (Robertson et al., 1997) for a more detailed discussion of the Log-MAP algorithm. The Log-MAP decoder computes the log-likelihood ratio $\Lambda(d_k)$ as follows:

$$\begin{aligned}\Lambda(d_k) = &\max^*_{(S_k, S_{k-1})} \{\overline{\gamma}_1 \left[(y^s_k, y^p_k), S_{k-1}, S_k\right] + \overline{\alpha}_{k-1}(S_{k-1}) + \overline{\beta}_k(S_k)\} \\ &- \max^*_{(S_k, S_{k-1})} \{\overline{\gamma}_0 \left[(y^s_k, y^p_k), S_{k-1}, S_k\right] + \overline{\alpha}_{k-1}(S_{k-1}) + \overline{\beta}_k(S_k)\},\end{aligned} \tag{5}$$

where

$$\overline{\gamma}_i \left[(y^s_k, y^p_k), S_{k-1}, S_k\right] = \frac{2E_s}{N_0} y^s_k x^s_k(i) + \frac{2E_s}{N_0} y^p_k x^p_k(i, S_k, S_{k-1}) + \ln \Pr\{S_k | S_{k-1}\} \tag{6}$$

are the branch metrics ($i = 0, 1$), $\ln \Pr\{S_k | S_{k-1}\}$ comprises the a priori information, and

$$\overline{\alpha}_k(S_k) = \max^*_{(S_{k-1}, i)} \{\overline{\gamma}_i \left[(y^s_k, y^p_k), S_{k-1}, S_k\right] + \overline{\alpha}_{k-1}(S_{k-1})\}, \tag{7}$$

$$\overline{\beta}_k(S_k) = \max^*_{(S_{k+1}, i)} \{\overline{\gamma}_i \left[(y^s_{k+1}, y^p_{k+1}), S_k, S_{k+1}\right] + \overline{\beta}_{k+1}(S_{k+1})\} \tag{8}$$

are the recursively accumulated path metrics. The Log-MAP decoder is implemented using the definition

$$\max{}^*(\delta_1, \delta_2) = \max(\delta_1, \delta_2) + \ln(1 + e^{-|\delta_2 - \delta_1|}), \tag{9}$$

whereas the approximation $\max^*(\delta_1, \delta_2) \approx \max(\delta_1, \delta_2)$ is assumed for the therefore sub-optimal Max-Log-MAP decoder. Four sub-tasks of the (Max-)Log-MAP algorithm can be easily identified by inspection of formulas (5)–(8):

1. The *transition metrics* ($\overline{\gamma}_i\,[(y_k^s, y_k^p), S_{k-1}, S_k]$) are used for
2. *forward path metric* ($\overline{\alpha}_k(S_k)$) accumulation during forward recursion and for
3. *backward path metric* ($\overline{\beta}_k(S_k)$) accumulation during backward recursion.
4. Finally, the soft-outputs ($\Lambda(d_k)$) are calculated using the forward path metrics, the backward path metrics, and the transition metrics.

These tasks are dominated by data-transfers, especially when taking power considerations into account (cf. Section 3). In contrast, the quota of the arithmetic operations is small, especially if the correction term (see equation (9)) of the Log-MAP algorithm is implemented with a look-up table. In (Berens et al., 1999), we have published some data about a Turbo-decoder implementation using the Max-Log-MAP decoder on a 16-bit fixed-point DSP: A detailed code analysis of the assembly code revealed that 35% of the total number of executed instructions were memory $\leftrightarrow$ register transfers and 25% were register $\leftrightarrow$ register transfers. The large share of register $\leftrightarrow$ register transfers was caused by the restrictions imposed by the single-accumulator DSP architecture. The share of the memory $\leftrightarrow$ register transfers is therefore expected to be well above 35% for state-of-the-art VLIW DSP architectures and for dedicated hardware implementations.

3 Importance of optimizing data-transfers

We have pointed out in Section 1 that the highest optimization potential for power savings resides on system level. One of the most important factors to be taken into consideration during system level optimization is that of memory access and management (Wehn and Münch, 1999). Relative to a 16-bit multiplication, for instance, the energy required for a 16-bit memory transfer is one order of magnitude higher in both hardware and software (Wuytack, 1998).

3.1 Memory power consumption

	C_{x0}[fF]	C_{x1}[fF]	C_{x2}[fF]	C_{x3}[fF]
C_{read}	9707	108	1126	6
C_{write}	7994	117	759	9

Table 1. Empiricially determined capacitive constants for the black-box capacitance model of (Landman, 1994), see also (Wuytack, 1998).

Landman (1994) presented a black-box capacitance model for an on-chip SRAM which demonstrates clearly the dependency of the memory power consumption on both the memory size and the access rate: Let W be the word depth, N the bit width, and WN the number of storage cells. Then,

$$C_{\text{read}} = C_{\text{read0}} + C_{\text{read1}}W + C_{\text{read2}}N + C_{\text{read3}}WN, \tag{10}$$

$$C_{\text{write}} = C_{\text{write0}} + C_{\text{write1}}W + C_{\text{write2}}N + C_{\text{write3}}WN. \tag{11}$$

Sample capacitive constants are listed in Table 1. The memory power consumption calculates as

$$P_{\text{memory}} = \frac{1}{2}V_{\text{DD}}^2(C_{\text{read}}f_{\text{read access}} + C_{\text{write}}f_{\text{write access}}) \tag{12}$$

(Wuytack, 1998), where V_{DD} is the supply voltage, C_{read} and C_{write} are effective capacitances, and $f_{\text{read access}}$ and $f_{\text{write access}}$ are the read and the write access rate, respectively.

The following conclusion can be drawn from equations (10)–(12): The memory power consumption

1. depends on the aspect ratio (W/N) of the SRAM and grows with increasing memory size (WN),
2. is proportional to the weighted sum of the access rates.

For the sake of simplicity, we will use the unified access rate f_{access} and the unified effective capacity C_{memory} for the remainder of this paper.

4 Examples for power minimization in Turbo-decoders

Several transformations can be applied on system level in order to reduce the data transfer related power consumption. We classify them as *algorithmic transformations*, *control flow transformations*, and *data flow transformations*. Algorithmic transformations can alter the communication performance. For instance, omitting the correction term in equation (9) degrades the BER to the benefit of a reduced implementation complexity (area, throughput, power). In general, communication performance has to be traded-off against implementation performance during system design of communication systems (Berens et al., 1999). We will show that algorithmic transformations like the *sliding window technique* can also enable subsequent control flow transformations which had been impossible to apply previously. The power consumption can be further reduced by some additional algorithmic transformations and by other techniques (outside our classification) like *voltage scheduling* and proper *quantization*.

4.1 Sliding window technique

It can be observed that the MAP decoding of a data block in a constituent decoder can be divided into the decoding of a set of overlapping sub-blocks (Dawid, 1996). This is called the *sliding window technique*. Decoding on a window-by-window basis permits to minimize the required memory size or to increase the throughput.

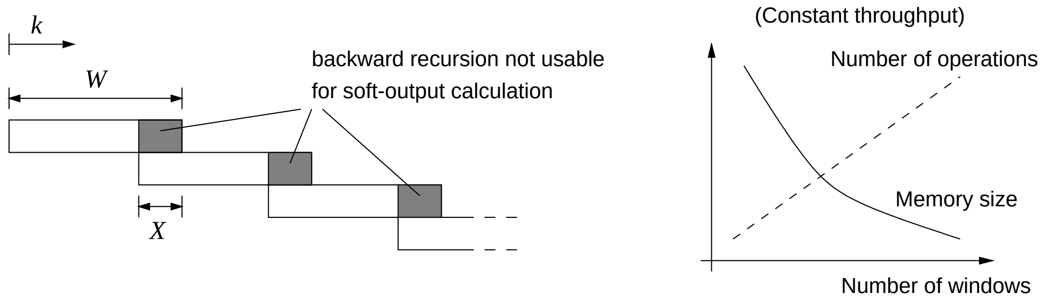

Figure 3. Serial window processing (serial MAP) with window size W and overlap width X. $k \in \{1 \dots N\}$ is the user bit index.

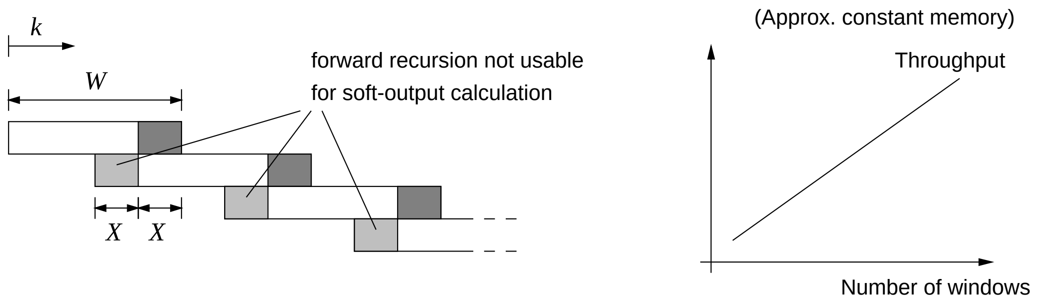
k
W
X X
forward recursion not usable for soft-output calculation
(Approx. constant memory)
Throughput
Number of windows

Figure 4. Parallel window processing (parallel MAP) with window size W and overlap width X. $k \in \{1 \dots N\}$ is the user bit index. Note that in the left part parallel window processing is *not* reflected explicitly.

– For the window overlap width (X) being small compared to the window size (W), serial window processing reduces the required memory size by the ratio of data block size (N) and window size (W), while retaining throughput ($C_{\text{memory}} \downarrow$). The memory savings is traded for a higher number of memory accesses ($f_{\text{access}} \uparrow$), because the backward recursion needs to be carried out twice for the overlap zones (indicated in Fig. 3). However, this effect can be neglected for $X \ll W$; for $W = 60$ and $X = 15$, for instance, the number of operations is increased by only 6%, according to first order complexity estimations (Berens et al., 1999).

– In contrast, parallel window processing can increase throughput by the same factor, while the memory size remains approximately constant (see Fig. 4). At the first glance, parallel window processing seems to be less suitable for saving power. But because the degree of parallelism is only limited by the number of windows inside a data block, power can be saved very effectively by application of voltage scaling (Wehn and Münch, 1999) ($V_{\mathrm{DD}} \downarrow$), if maximum throughput is not required.

4.2 MAP sub-task parallelization

We previously stated in Section 2.2 that the computations performed during MAP decoding (and derivates) can be divided into four sub-tasks: transition metric calculation, forward recursion, backward recursion, and soft-output calculation.

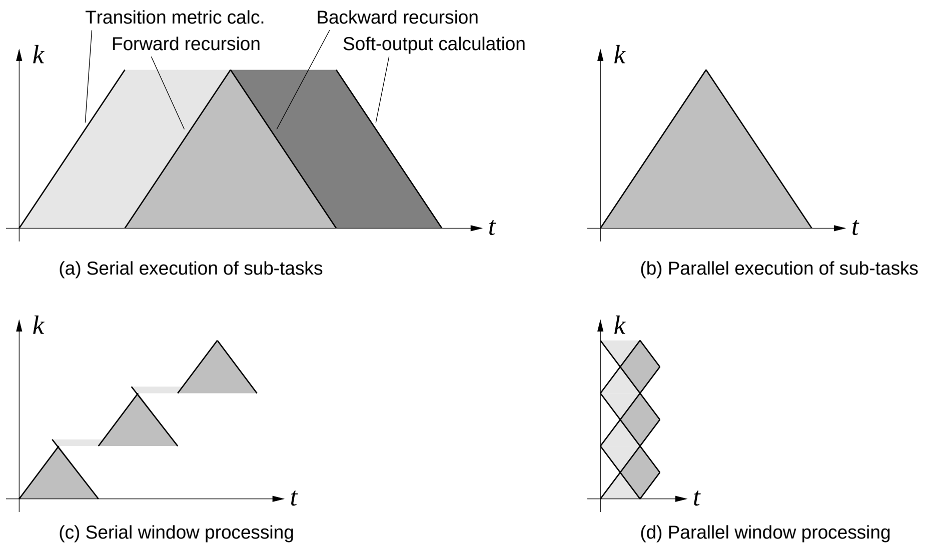
Transition metric calc.
Forward recursion
Backward recursion
Soft-output calculation
k
t
(a) Serial execution of sub-tasks
k
t
(b) Parallel execution of sub-tasks
k
t
(c) Serial window processing
k
t
(d) Parallel window processing

Figure 5. MAP sub-task parallelization. Shaded areas indicate storage of intermediate values. Darkness of shading correlates with number of intermediate values to be stored.

– Fig. 5 (a) depicts the initial solution where all tasks are executed in series. Memory is required to store the transition metrics, the forward path metrics, and the backward path metrics; this memory must not be discarded until soft-output calculation.

– Fig. 5 (b) depicts an optimized solution where the transition metric calculation is carried out in parallel with the forward recursion and where the results of the backward recursion are immediately consumed by the parallel soft output calculation. The variable life-time of the transition metrics and the forward path metrics is significantly shorter; backward path metric storage is not necessary. This reduces the memory size ($C_{\mathrm{memory}} \downarrow$) and also the access rate ($f_{\mathrm{access}} \downarrow$), as accesses to a backward path metric memory are eliminated.

– Serial window processing, depicted in Fig. 5 (c) and described in more detail above, is the solution for further reduction of memory size. This comes at the expense of increased computational complexity, which is, however, often tolerable. Another advantage of the serial MAP decoder is that the required amount of storage becomes independent of the data block size.

– A sample schedule for parallel window processing is shown in Fig. 5 (d). The overlap width has been chosen as $X = W/2$ in accordance with recently proposed high-speed MAP architectures (Dawid, 1996); power reduction can be achieved by voltage scaling, as mentioned above.

4.3 Recomputation

Recently, Schurgers et al. (1999) proposed a data flow transformation based on selective recomputation. To reduce the path metric storage, only a fraction $1/\theta$ of the metrics is stored in RAM. This transformation is therefore referred to as partial state metric storage. The key is to recalculate the missing state metrics when they are needed for $\Lambda(d_k)$ computation, see equation (5). For efficiency, the recalculated metrics should be stored temporarily in a register file (as indicated in Fig. 6) instead of recalculating each of the missing metrics

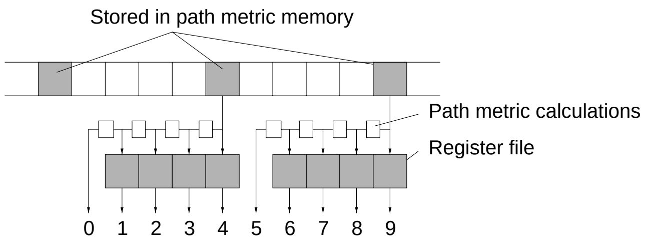

Figure 6. Partial path metric storage with recomputation.

separately starting from a stored metric. The register file is required for producing the path metrics in a proper order ($0 \to 9$) and can therefore not be omitted.

As a result of this transformation, the required path metric memory shrinks by a factor θ ($C_{\text{memory}} \downarrow$). The access rate to the path metric memory decreases likewise ($f_{\text{access}} \downarrow$), especially if the recalculated metrics are stored in a register file. On the other hand, recalculation increases computational complexity by principle, and it results in a higher access rate to the branch metric memory ($f_{\text{access}} \uparrow$) in this case. Additionally, a register file has to be allocated.

4.4 Other algorithmic transformations

Franz and Anderson (1998) proposed two *reduced-search MAP algorithms.* These rely on trellis pruning during forward recursion which therefore simplifies backward recursion and soft-output calculation: The M-MAP algorithm keeps only the best M nodes at each trellis stage ($C_{\text{memory}} \downarrow$, $f_{\text{access}} \downarrow$), whereas the T-MAP algorithm keeps trellis nodes with values above a threshold ($f_{\text{access}} \downarrow$). The T-MAP algorithm does not sacrifice the decoding performance as much as the M-MAP algorithm.

Early detection & trellis splicing (Frey and Kschischang, 1998) analyzes the soft-outputs and tries to build a shortened trellis for the succeeding Turbo-iteration by splicing trellis stages. This reduces memory size and access rate ($C_{\text{memory}} \downarrow$, $f_{\text{access}} \downarrow$) for iterations with spliced trellises, but on the other hand requires a state transition array due to irregular trellis structures, and it can decrease the decoding performance.

4.5 Other techniques for power reduction

Dynamically applying voltage scaling to sub-systems with variable throughput requirements is referred to as *voltage scheduling* (Ishihara and Yasuura, 1998; Pering et al., 1998). Voltage scheduling has been shown to be superior to switching off sub-systems (e. g., by means of clock gating) during idle-times. Voltage scheduling proves to be most beneficial on systems where the average throughput is well below the maximum throughput. The power savings achieved by reducing the supply voltage and slowing down the clock frequency have to offset an "implementation loss" which is mainly due to two reasons: First, the DC-DC converters themselves consume power. Second, input data buffering is required for establishing a schedule. Furtheron, some practical problems (like speed of voltage switching) have to be solved. Nevertheless, voltage scheduling (also termed *dynamic voltage scaling*) is regarded as "possibly the most revolutionary low power technique to date" (Ackland and Nicol, 1998). Turbo-decoding is quite amenable for voltage scheduling, as seldomly the maximum number of iterations for decoding a data block has to be carried out.

Quantization is not a technique for power reduction in the first place, but it heavily influences system power consumption: The widths of the data-paths, the memory bit widths (W), the width of the buses (i. e., on-chip interconnect), and the number of I/O pads (i. e., off-chip interconnect) depend on the number of bits chosen to represent the respective numeric values. Proper quantization therefore aids memory size minimization ($C_{\text{memory}} \downarrow$) (Jeong and Hsia, 1999; Berens et al., 1999; Michel et al., 2000).

5 Conclusion

Turbo-decoding is a computationally challenging task that will be employed in 3rd generation mobile radio systems. The data-transfer related power consumption of Turbo-decoders can be significantly reduced by system level optimizations. The potential to minimize power consumption by rearranging (and eventually avoiding) memory accesses has therefore to be exploited before addressing, e. g., data-path optimizations. Table 2 summarizes the effects of the discussed optimization techniques on the main parameters of power consumption.

	C_{memory}	f_{access}	V_{DD}
Serial MAP	↓	↑	
Parallel MAP (voltage scaling)			↓
Parallel sub-tasks, solution (b)	↓	↓	
Recomputation	↓	↓ (path metric memory) ↑ (branch metric memory)	
Reduced-search MAP	↓	↓	
Early detection & trellis splicing	↓	↓	
Voltage scheduling			↓
Power-conscious quantization	↓		

Table 2. Effects of the discussed optimization techniques on the main parameters of power consumption.

References

3GPP, Third Generation Partnership Project, `http://www.3gpp.org`, 1999.

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

Bahl, L., Cocke, J., Jelinek, F., and Raviv, J., Optimal Decoding of Linear Codes for Minimizing Symbol Error Rate, *IEEE Transactions on Information Theory*, *IT-20*, 284–287, 1974.

Berens, F., Worm, A., Michel, H., and Wehn, N., Implementation Aspects of Turbo-Decoders for Future Radio Applications, in *Proc. VTC '99 Fall*, pp. 2601–2605, Amsterdam, The Netherlands, 1999.

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

Dawid, H., *Algorithmen und Schaltungsarchitekturen zur Maximum a Posteriori Faltungsdecodierung*, Ph.D. thesis, RWTH Aachen, Shaker Verlag, Aachen, Germany, 1996.

Franz, V. and Anderson, J. B., Concatenated Decoding with a Reduced-Search BCJR Algorithm, *IEEE Journal on Selected Areas in Communications*, *16*, 186–195, 1998.

Frey, B. J. and Kschischang, F. R., Early Detection and Trellis Splicing: Reduced-Complexity Iterative Decoding, *IEEE Journal on Selected Areas in Communications*, *16*, 153–159, 1998.

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

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

Jeong, G. and Hsia, D., Optimal Quantization for Soft-Decision Turbo Decoder, in *Proc. VTC '99 Fall*, pp. 1620–1624, Amsterdam, The Netherlands, 1999.

Landman, P., *Low-Power Architectural Design Methodologies*, Ph.D. thesis, Electronics Research Laboratory, College of Engineering, University of California, Berkeley, CA, 1994.

Michel, H., Worm, A., and Wehn, N., Influence of Quantization on the Bit-Error Performance of Turbo-Decoders, submitted to *VTC 2000 Spring*, 2000.

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

Robertson, P., Illuminating the Structure of Code and Decoder of Parallel Concatenated Recursive Systematic (Turbo) Codes, in *Proc. Globecom '94*, pp. 1298–1303, 1994.

Robertson, P., Hoeher, P., and Villebrun, E., Optimal and Sub-Optimal Maximum a Posteriori Algorithms Suitable for Turbo Decoding, *ETT*, *8*, 119–125, 1997.

Schurgers, C., Engels, M., and Catthoor, F., Energy Efficient Data Transfer and Storage Organization for a MAP Turbo Decoder Module, in *Proc. ISLPED '99*, 1999.

Shannon, C. E., A Mathematical Theory of Communication, *Bell Syst. Tech. J.*, *27*, 379–423, 1948a.

Shannon, C. E., A Mathematical Theory of Communication, *Bell Syst. Tech. J.*, *27*, 623–656, 1948b.

Wehn, N. and Münch, M., Minimizing power consumption in digital circuits and systems: an overview, in *Kleinheubacher Berichte*, 1999.

Wuytack, S., *System-Level Power Optimization of Data Storage and Transfer*, Ph.D. thesis, Katholieke Universiteit Leuven, 1998.