TRACEBACK-ENHANCED MAP DECODING ALGORITHM

Curt Schurgers
ECE Dept., UCSD
curts@ece.ucsd.edu

Anantha Chandrakasan
MTL, Massachusetts Institute of Technology
anantha@mtl.mit.edu

ABSTRACT

Soft-input soft-output algorithms are the principal component of the iterative decoding used in turbo codes and other 'turbo' feedback schemes. To enable efficient implementation, especially on energy constrained platforms such as portable devices, it is crucial to reduce the computational complexity to a minimum. We propose an enhancement to the MAX-LOG-MAP algorithm by adding a traceback operation similar to that used in the Viterbi algorithm, and devise a new efficient way to initialize the start state of the traceback. This enhancement is effective for each decoding iteration, and provides saving on top of existing techniques such as early termination and memory optimizations. It reduces the computational complexity by an additional 15%, without incurring any performance penalty.

1. INTRODUCTION

As turbo coding is being considered for use in portable and battery operated systems such as 3G cellular [1][2], energy efficiency is emerging as one of the chief design considerations. To reduce the energy consumption of the soft-output decoding, we propose a set of algorithmic optimizations that lower the computational complexity, without sacrificing decoding performance. Furthermore, our improvements are interoperable with existing low-power techniques.

One of most powerful exiting ways to reduce the energy consumption is limiting the number of decoding iterations by means of a stop criterion [3][4]. Thul et al. [3] and Garrett et al. [4] observed that only static algorithmic optimizations, which are effective at each decoding step, can yield additional savings. Techniques that kick in at later iterations, have limited effectiveness as they are executed very infrequently due to the early termination. Our optimization is static, and thus yields savings above and beyond those of early termination.

2. DECODING ALGORITHM PRINCIPLE

2.1. MAP versus SOVA

Our new scheme is essentially a hybrid between the two classes of existing decoding algorithms: Maximum A Posteriori (MAP) and Soft-Output Viterbi Algorithm (SOVA). Fossorier et al. have proven a basic relationship between these two classes, namely that the MAX-LOG-MAP variant is equivalent to an improved version of SOVA, despite the fact that the algorithms themselves are very dissimilar [5]. Although the SOVA class is believed to be less complex than MAP, the improvements to SOVA have a negative effect on the implementation efficiency. On the other hand, without the optimizations, SOVA sacrifices decoding performance.

Furthermore, Vogt and Finger have shown that a simple scaling factor brings the performance of MAX-LOG-MAP to within 0.1 dB of that of optimal MAP for AWGN channels and the settings of the 3G standard [6]. We therefore select the MAX-LOG-MAP algorithm as our basis to build upon, as its performance is to the optimal one of MAP, but at lower implementation complexity. The equivalence reported by Fossorier et al. [5], served as inspiration to incorporate principles from SOVA in MAX-LOG-MAP. Our new algorithm that is described in this paper reduces the complexity even further, while maintaining the same level of decoding performance.

2.2. MAX-LOG-MAP

Before we can introduce our enhancement, we first briefly review MAX-LOG-MAP. For more details, we refer to Benedetto et al. [7]. For each position k in the input block of size N, the log-likelihood of the decision bits u_k is calculated as:

$$L(u_k) = Q_k^1 - Q_k^0 \tag{1}$$

Q_k^i (with i = 0, 1) is given by (2). It is the logarithm of the probability of the most likely codeword c that corresponds to $u_k = i$, given the entire received block. In general, R_x^y denotes the received information from position x to y.

$$Q_k^i = \max_{c;\ u_k=i} \left[\log \Pr(c \mid R_1^N) \right] \tag{2}$$

This can be further decomposed into:

$$\log_{u_k=i} \Pr(c \mid R_1^N) = \left[\alpha_k(s') + \gamma_k(s',s) + \beta_{k+1}(s) + Const \right]_{u_k=i} \tag{3}$$

In equation, s is the trellis state that is reached from state s' with a decision bit $u_k = i$. The branch metrics γ_k capture the information from the channel. The forward state metrics α_k and backward state metrics β_k are obtained through a forward and backward recursion respectively as given by (5) and (6). Here, S_k is the trellis state at position k. Note that the α and β metrics relate the probability of being in a state at position k to all the received information before or after k respectively.

$$\gamma_k(s',s) = \log\left[\Pr\left(S_k = s', S_k = s, R_k^k\right)\right] \tag{4}$$

$$\alpha_k(s) = \log\left[\Pr\left(R_1^{k-1} \middle| S_k = s\right)\right] = \max_{s' \in S}\left(\alpha_{k-1}(s') + \gamma_{k-1}(s',s)\right) \tag{5}$$

$$\beta_k(s) = \log\left[\Pr\left(R_k^N \middle| S_k = s\right)\right] = \max_{s' \in S}\left(\beta_{k+1}(s') + \gamma_k(s,s')\right) \tag{6}$$

2.2. Traceback Enhancement

Maximum Likelihood Decoding (MLD) gives the most probable transmitted codeword given the received sequence, and marks the most likely path through the trellis. From (2), it can be observed that the **max** operation of either Q_k^0 or Q_k^1 selects the branch from this maximum likelihood path at point k, depending whether this branch corresponds to i = 0 or i = 1. This was also observed in the equivalence proof between the MAX-LOG-MAP algorithm and SOVA [5].

In other words, assuming the maximum likelihood path at point k runs from state s_1 to state s_2 and arbitrarily choosing the constant in (3) equal to zero, the calculation of Q_k^0 or Q_k^1 is simplified considerably as follows:

If (maximum likelihood path) $u_k = 0$, $s_1 \rightarrow s_2$, *then*

$$Q_k^0 = \alpha_k(s_1) + \gamma_k(s_1,s_2) + \beta_{k+1}(s_2)$$

$$Q_k^1 = \max_{u_k = 1}\left(\alpha_k(s') + \gamma_k(s',s) + \beta_{k+1}(s)\right)$$

else (maximum likelihood path) $u_k = 1$, $s_1 \rightarrow s_2$, *then*

$$Q_k^0 = \max_{u_k = 0}\left(\alpha_k(s') + \gamma_k(s',s) + \beta_{k+1}(s)\right)$$

$$Q_k^1 = \alpha_k(s_1) + \gamma_k(s_1,s_2) + \beta_{k+1}(s_2) \tag{7}$$

The maximum likelihood path can calculated with the traditional Viterbi algorithm, a technique that also forms the basis of SOVA. More specifically, it is obtained by a forward recursion of state metrics as in (5), followed by a trace back procedure. Ideally, this traceback starts at the end of the data block. However, the Viterbi algorithm can be simplified with minimum performance loss by initializing the traceback at an intermediate point, followed by an initial traceback of sufficient depth to provide convergence [8]. Knowledge of the maximum likelihood path enables us to use the simplified expressions of (7).

3. ARCHITECTURAL ALTERNATIVES

3.1. Traditional Sliding Window

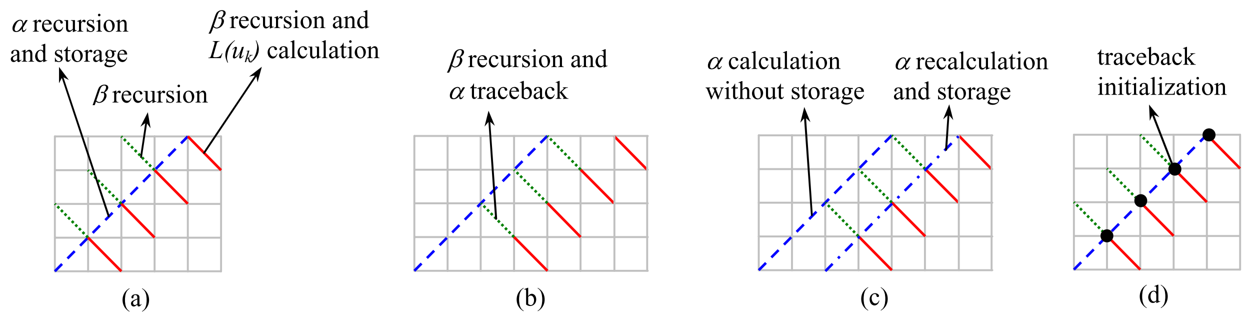
We use the graphical representation of [9] to illustrate the different architectural alternatives of our traceback enhanced MAX-LOG-MAP algorithm. Figure 1a shows the sequence of operations for the traditional MAX-LOG-MAP algorithm. The sliding window approach is used to limit the state metric storage [10]. The x-axis represents the decoding time, while the y-axis denotes the bit position k. The α state metrics are calculated via the forward recursion and stored in memory. After an initial backward β recursion to ensure convergence of the sliding window approach, valid β metrics are calculated and combined with the stored α metrics to generate the output likelihoods $L(u_k)$ based on equations (1)-(3).

3.2. Traceback with Forward Metric Storage

The most straightforward way to utilize knowledge of the maximum likelihood path is illustrated in figure 1b. While calculating the forward α metrics, the path decisions are stored in a separate traceback memory. The initial β recursion is accompanied by an initial traceback. When both have converged, the outputs can now be calculated using (5) where the maximum likelihood path is obtained from the continued traceback. However, due to the initial traceback, more α metrics need to be calculated before the backward process can start. This results in a tripling of the required metric storage, as can be observed by comparing the production-consumption distance of these metrics in figures 1a and 1b.

3.3. Traceback with Forward Metric Recalculation

A way to reduce the memory requirements is illustrated in figure 1c. Instead of storing the α metrics during the



α recursion and storage
β recursion and $L(u_k)$ calculation
β recursion
(a)
β recursion and α traceback
(b)
α calculation without storage
α recalculation and storage
(c)
traceback initialization
(d)

Figure 1 – Graphic representation of the decoding algorithm alternatives

initial forward recursion, only the path decisions are saved. After a time offset, the α metrics are recalculated and stored, leading to state metric storage requirements identical to those of the original algorithm of figure 1a. These recalculations can be heavily simplified be utilizing the stored path decisions. Only one addition of equation (5) needs to be performed for each state, and no max operation is required as the surviving branch is already known.

3.4. Traceback with Initialization

The inefficiencies still present in the two previous solutions arise from the fact that extra forward metrics need to be calculated compared to the traditional sliding window algorithm in order to support the initial traceback. This initial traceback is required for convergence reasons.

However, it is possible to initialize the traceback directly when α and β metrics are available. For each state *s* and each bit position *k*, the sum of these two metrics can be interpreted from (5) and (6) as:

$$\alpha_k(s)+\beta_k(s)=\log\left[\Pr\left(R_1^{k-1}\middle|S_k=s\right)\right]+\log\left[\Pr\left(R_k^N\middle|S_k=s\right)\right]$$
$$=\log\left[\Pr\left(R_1^N\middle|S_k=s\right)\right] \qquad (8)$$

Using Bayes' rule, this can be rewritten as (9). As there is no a priori information on the probability of each state, the sum of the forward and backward metrics is thus an indication of the log likelihood of being in state *s* at index *k*, given the entire received sequence. The maximum likelihood path thus runs through the state with the highest value for this metric. As a result, this state can be used directly to initialize the traceback, without the need for extra convergence steps.

$$\alpha_k(s)+\beta_k(s)=\log\left[\frac{\Pr\left(S_k=s\middle|R_1^N\right)\cdot\Pr\left(R_1^N\right)}{\Pr\left(S_k=s\right)}\right] \qquad (9)$$

Strictly speaking, expression (9) is only correct if the forward and backward metrics are calculated from the start and end of the block respectively. However, the sliding window approach is expected to yield a sufficiently accurate approximation. The simulations in section 4 will validate this assumption. The resulting architecture is depicted in figure 1d. While performing the α recursion, the path decisions are stored in a traceback memory. After the initial β metrics have converged, which is indicated by the black circles in figure 1d, the start state of the traceback is set to the state *s* that maximizes (9). The output likelihoods are calculated via (7) by using the maximum likelihood path obtained from the traceback.

Note that the extra calculations to determine the most likely state (*i.e.* adding the α and β for each state, followed by determining the maximum) only need to be performed once for each sliding window. A detailed comparison between our traceback enhanced algorithm and the traditional setup is given in section 4.

4. EVALUATION

4.1. Decoding Performance

First, we validate that our traceback enhanced MAX-LOG-MAP algorithm indeed does not suffer a loss in decoding performance. Figure 2 shows the simulation results for a 4-state parallel concatenated turbo code with generator polynomial (7,5) and a block length of 400 bits. There are two component encoders that are both terminated. The sliding window size is chosen equal to 20, and a scaling factor of 0.8 is applied [6]. The bit error rate (BER) after each of the six decoding iterations is plotted. Results are averaged over a large number of simulation runs where for each data point at least 100 block errors are recorded.

The solid lines correspond to the traditional sliding window MAX-LOG-MAP algorithm shown in figure 1a. The circles mark the performance of our new traceback enhanced MAX-LOG-MAP algorithm with the traceback initialization of section 3.4 and figure 1d. We can verify that virtually no performance degradation occurs. The other two options which were discussed in sections 3.2 and 3.3 are not shown in figure 2 to avoid cluttering. We have verified that their performance is the same as the one shown for the traceback enhanced scheme with traceback initialization (section 3.4). Although not shown here due to space limitations, these experiments were repeated for other codes and interleaver sizes, resulting in similar fits.

4.2. Computational Efficiency

The main benefit of our new schemes is a reduction in computational complexity, which lowers the decoder energy consumption. Table I gives a high-level summary

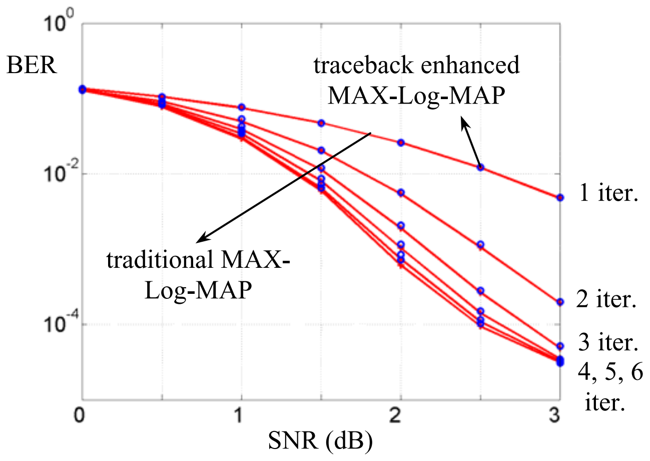


Figure 2 – Decoding performance comparison

of the computational cost of the MAX-LOG-MAP algorithm in terms of additions (ADD) and maximum operations (MAX) for each index *k.* In this table, *S* is the number of states of the component code. An extra 3 additions is required to generate the branch metrics γ [7], which are calculated once and retrieved from a cache. There are three metric recursions and one $L(u_k)$ calculation, resulting in the rough totals listed in the last row of Table I.

Table I – Computational cost of MAX-LOG-MAP

	ADD	MAX
α and β recursion (each)	2·S	S
$L(u_k)$ calculation	4·S+1	2·(S-1)
total	10·S+4	5·S-2

The first row of Table II summarizes the savings in computational complexity that result from our traceback enhancement technique and which are derived from equation (5). There is, however, an additional cost involved, which is different for the three alternatives discussed in sections 3.2, 3.3 and 3.4. These are listed in the lower three rows of the table, and include the average number of traceback operations per bit position and the extra memory cost. Here, *L* denotes the sliding window length and *w* is the bit width of the state metrics.

Table II – Savings and costs in computational from traceback enhancement

		ADD	MAX	Trace -back	Extra memory: traceback and α (bits)
savings		2·S-2	S-1		
extra cost	3.2			2	L·S + 2· L·S·w
	3.3	S		2	L·S
	3.4	S/L	(S-1)/L	1	L·S

Table III – Relative savings for different number of states

	S = 2	S = 4	S = 8
3.2	0.97	0.89	0.84
3.3	1.03	0.95	0.91
3.4	0.95	0.88	0.84

Table III compares the relative computational efficiency of our three schemes to the traditional MAX-LOG-MAP algorithm, for different number of states. It is assumed an addition, a MAX operation and a traceback operation are equally costly. Although this is a rough approximation, it gives us a yardstick for a first cut comparison of the different schemes. The traceback initialized scheme of section 3.4 is the most efficient, and is more beneficial for codes with a higher number of states (*i.e.* a larger constraint length).

5. CONCLUSIONS

In this paper, we introduced a novel modification to the MAX-LOG-MAP algorithm which borrows the traceback principle from the SOVA and Viterbi algorithm. By utilizing knowledge of the maximum likelihood path, the calculation of the output likelihood ratios can be simplified significantly. We presented three alternative approaches for obtaining a good start state of the traceback operation. The most promising approach is to directly initialize the traceback based on the forward and backward state metrics. This results in a computational complexity reduction of up to 15%, without any performance penalty.

6. ACKNOWLEDGMENTS

Effort sponsored by the Defense Advanced Research Projects Agency (DARPA) and Air Force Research Laboratory, under agreement number F33615-02-2-4005. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation thereon.

7. REFERENCES

[1] 3rd Generation Partnership Project (3GPP), "Technical Specification Group Radio Access Network; Multiplexing and channel coding (FDD)", *TS 25.212 v5.0.0*, http://www.3gpp.org.
[2] 3rd Generation Partnership Project 2 (3GPP2), "Physical layer standards for cdma2000 spread spectrum systems," *ARIB STD-T64-C.S20002-*A, http://www.3gpp2.org.
[3] M. Thul, T. Vogt, F. Gilbert, and N. Wehn, "Evaluation of algorithm optimizations for low-power turbo-decoder implementations," *IEEE ICASSP'02*, Orlando, FL, pp.III-3101-4, May 2002.
[4] D. Garrett, B. Xu, and C. Nicol, "Energy efficient turbo decoding for 3G mobile", *ACM/IEEE ISLPED'01*, Huntington Beach, CA, pp. 328-33, August 2001.
[5] M. Fossorier, F. Burkert, S. Lin, and J. Hagenauer, "On the equivalence between SOVA and Max-Log-MAP decodings," *IEEE Comm. Letters*, Vol.2, No.5, pp.137-139, May 1998.
[6] J. Vogt, and A. Finger, "Improving the max-log-MAP turbo decoder", *Electronics Letters*, Vol.36, No.23, pp.1937-39, Nov. 2000.
[7] S;. Benedetto, D. Divsalar, G. Montorsi, and F. Pollara, "A soft-input soft-output APP module for iterative decoding of concatenated codes," *IEEE Comm. Letters*, Vol.1, No.1, pp. 22-24, Jan. 1997.
[8] G. Forney, "The Viterbi algorithm," *Proceedings IEEE*, Vol.61, pp.268-278, 1973.
[9] C. Schurgers, F. Catthoor, M. Engels, "Memory optimization of MAP turbo decoder algorithms," *IEEE Trans. on VLSI Systems*, Vol.9, No.2, pp.305-312, April 2001.
[10] S. Benedetto, D. Divsalar, G. Montorsi, F. Pollara, "Algorithm for continuous decoding of turbo codes," *Electronics Letters*, Vol.32, No.4, pp. 314-315, Feb. 1996.