Performance of a Low-Complexity Turbo Decoder and its Implementation on a Low-Cost, 16-Bit Fixed-Point DSP

Ken Gracie, Stewart Crozier, Andrew Hunt, John Lodge
Communications Research Centre
3701 Carling Avenue, P.O. Box 11490
Station H, Ottawa, Ontario, K2H 8S2
Phone : (613) 990-5846, FAX : (613) 990-6339
E-mail : ken.gracie@crc.ca

Abstract

This paper presents the bit error rate, packet error rate, and throughput performance of a turbo decoder implemented on the Analog Devices ADSP-2181, a 16-bit fixed-point digital signal processing (DSP) chip. A simplified decoding algorithm is described, and example performance is given for block sizes between 64 and 512 information bits with a number of different code rates. Some implementation issues are also discussed.

1.0 Introduction

Recent years have seen considerable interest in turbo codes as an effective method of performing error correction in communications systems [1,2,3]. While nominally very computationally complex, key optimizations can be exploited to dramatically reduce the amount of processing required. This allows for the use of low-cost DSP chips as decoding engines for this powerful class of error-correcting codes. A very efficient turbo decoder using the Analog Devices ADSP-2181, a 16-bit fixed-point processor, has been developed to demonstrate this fact.

The decoding algorithm used is a form of iterative *a posteriori* probability (APP) decoding, also referred to as maximum *a posteriori* (MAP) decoding in the literature, implemented in the log domain [3,4,5,6,7]. The APP algorithm finds an estimate of the probability that the information bit is a 0 (or equivalently a 1) at each bit time given the entire received signal [1], in contrast to the Viterbi algorithm which performs maximum likelihood sequence estimation (MLSE) [8]. The APP or MAP decoding method naturally lends itself to providing the soft estimates needed for iterative decoding.

The decoder is implemented on its own separate processor and transfers data packets via a serial port. The received data is double-buffered to ensure that maximum throughput can be achieved. For 4 full decoding iterations, throughput on a 40 MIPS version of the ADSP-2181 was found to be approximately 16.8 kbps for all of the block sizes that were considered.

The paper presents bit and packet error rate results for block sizes between 64 and 512 information bits. The effect on performance of block size, code rate, and number of

iterations is illustrated. These findings are compared to the performance of K=7 and K=9 Viterbi decoders, which have also been implemented on the same platform. The complexity of the K=9 Viterbi decoder is comparable to that of the turbo decoder implementation performing 4 decoding iterations.

Using a 16-bit fixed-point processor means that normalization and precision become important issues. It has been found that performance is essentially identical to that of a 32-bit C simulation when as few as 9 bits are used to represent the channel samples.

Section 2 contains a brief description of the decoding algorithm, including the structure of both the encoder and decoder. The description of the decoder includes a summary of the max-log-MAP algorithm used in each constituent decoder as well as a discussion of implementation issues. Section 3 presents bit and packet error performance as well as throughput results. Section 4 gives the conclusions.

2.0 The Turbo Codec

The structure of the turbo encoder is shown in Figure 1. Two K=5, rate 1/2 recursive systematic convolutional (RSC) encoders are used in parallel, one encoding the information bits directly and the other encoding an interleaved version of the information bits. The parity produced by these two encoders is punctured to achieve the desired overall code rate. A set of K-1 termination bits is used to return RSC1 to the all-zeroes state at the end of each data block. These termination bits are also interleaved and encoded by RSC2. Note that because of the interleaver, RSC2 terminates in an arbitrary state.

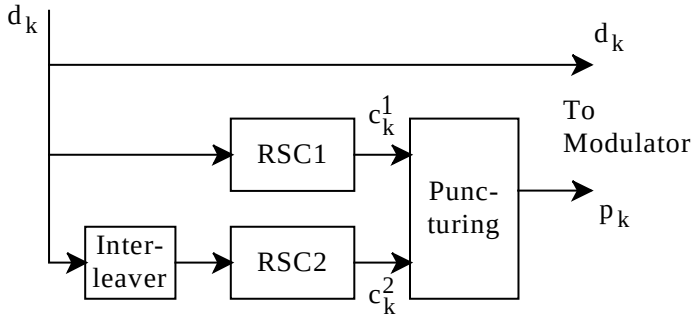

Figure 1: The turbo encoder with punctured parity.

A well-known approach to turbo decoding which makes use of APP or MAP decoding is shown in Figure 2. $\mathbf{X}$ is the set of systematic channel samples, $\mathbf{Y^1}$ is the set of unpunctured parity samples corresponding to RSC1, and $\mathbf{Y^2}$ is the set of unpunctured parity samples corresponding to RSC2. The constituent decoders are implemented in the log domain and utilize the log-MAP algorithm [3,4]. The first decoder attempts to improve the systematic bit estimates with the additional information contained in $\mathbf{Y^1}$, while the second decoder attempts the same task with the additional information contained in $\mathbf{Y^2}$. The improved estimates produced by each decoder are the *log-*

likelihood ratios (LLR's) and may be thought of as the sum of the systematic input ($\mathbf{L_{in}}$) and the so-called extrinsic information ($\mathbf{L_{ex}}$) obtained from the parity samples [1,2,3]. With this approach, the input channel samples must be scaled by the channel reliability factor L_c, which is a function of the channel signal-to-noise ratio.

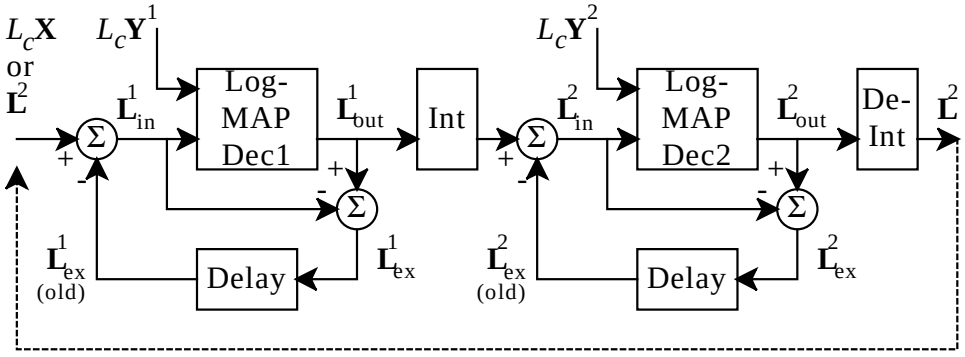

Figure 2: Turbo decoder using two log-MAP component decoders.

Figure 3 shows a modified turbo decoder structure used in this implementation. The max-log-MAP algorithm, described in the next section, is used as the constituent decoder and a correction operation is performed on both the extrinsic information and the constituent decoder output. Note that channel reliability factor L_c no longer needs to be estimated or applied, since the max-log-MAP decoder is not sensitive to scale factors. With appropriate correction, this modified decoder structure has been found to give performance within approximately 0.1 dB of the structure shown in Figure 2, for the same number of iterations. An additional half-iteration can often more than compensate for this difference in performance.

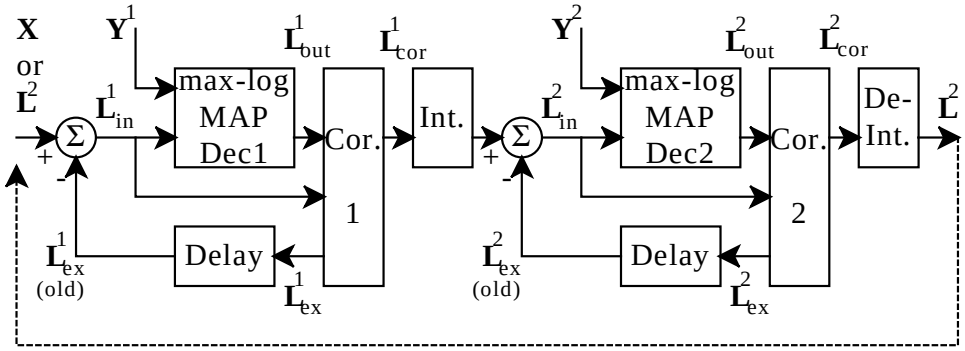

Figure 3: Modified turbo decoder using two max-log-MAP component decoders with corrections.

2.1 The Max-Log-MAP Algorithm

The max-log-MAP algorithm [3,4,7,9] calculates the LLR's according to

$$L_k = \max_m [A_k^m + D_k^{0,m} + B_{k+1}^{f(0,m)}] - \max_m [A_k^m + D_k^{1,m} + B_{k+1}^{f(1,m)}] \qquad \textbf{(1)}$$

where k is the time index, m is the present state (m = 0,…, M-1), $f(d,m)$ is the forward or next state given present state m and input bit $d=\{0,1\}$, A_k^m is the *forward state metric* for state m, B_k^m is the reverse or backward state metric for state m, and $D_k^{d,m}$ is the branch metric given present state m and input bit $d=\{0,1\}$. The metrics are calculated according to :

$$A_k^m = \max[A_{k-1}^{b(0,m)} + D_{k-1}^{0,b(0,m)}, A_{k-1}^{b(1,m)} + D_{k-1}^{1,b(1,m)}] \quad \textbf{(2)}$$

$$B_k^m = \max[D_k^{0,m} + B_{k+1}^{f(0,m)}, D_k^{1,m} + B_{k+1}^{f(1,m)}] \quad \textbf{(3)}$$

$$D_k^{d,m} = \frac{1}{2}(x_k d' + y_k c'^{d,m}) \quad \textbf{(4)}$$

where $b(d,m)$ is the previous or backward state given present state m and previous input bit $d=\{0,1\}$, x_k is the k^{th} systematic sample, y_k is the k^{th} parity sample, d is a systematic bit, $c^{d,m}$ is the corresponding coded bit given state m and information bit d. For binary antipodal signalling, the corresponding transmit symbols are given by $d' = 1 - 2d$ and $c'^{d,m} = 1 - 2c^{d,m}$. The state metrics give a measure of the likelihood that state m was the correct encoder state at time k, while the branch metrics measure the likelihood of the transmitted bits at time k given the received signal samples. The forward state metrics are calculated starting at the beginning of the block (time 0) and working towards the end of the block (Equation (2)). The backward state metrics are calculated in the opposite direction, starting with the samples at the end the data block and working back towards the beginning (Equation (3)).

A number of observations may be made about the algorithm. First, it is clear that the use of the max approximation means that this method is not optimum with regard to MAP decoding. Second, as mentioned above, this particular method does not require an estimate of the channel signal-to-noise ratio. Third, it is significant that this method is very similar to a standard Viterbi algorithm without history [3,4,7,9]. In fact, this algorithm will find the same maximum likelihood path through the trellis as the Viterbi algorithm while producing soft outputs that can be used in successive decoding stages.

2.2 Implementation Issues

The codec is implemented on a pair of Analog Devices EZ-KIT development systems, each featuring an ADSP-2181 DSP chip. One acts as a general purpose channel simulator: random information bits are generated and encoded, an AWGN channel is simulated, the resulting channel samples are sent to the decoder via a serial port, and decisions from the decoder are compared with the original bits in order to compile error rate statistics. The second board acts as the decoder, taking in noisy samples and producing bit estimates. This test bed is flexible, low-cost, and illustrates the practicality of turbo codes.

The fact that the ADSP-2181 is a 16-bit fixed-point processor combined with the fact that the turbo decoder is iterative means that precision and normalization are important issues.

In particular, both the systematic samples themselves and the state metrics must be regularly normalized in order to prevent overflow. Note that the two requirements are related, since the magnitude of the state metrics is a function of the input signal strength. Block normalization is used to prevent overflow of the systematic LLR values. That is, the current block of systematic samples are periodically scaled down such that the largest sample in the block does not exceed some predefined level. In the current decoder implementation, block normalization is performed after the extrinsic information has been subtracted, just before the samples are passed to the max-log-MAP decoder. This guarantees the desired signal level at the input to the max-log-MAP decoder. It is assumed that these samples have not overflowed since the last block normalization. The maximum tolerable signal level was determined empirically, and involved a tradeoff between precision and probability of overflow. It was found that 9 bits of precision (8 magnitude, 1 sign) satisfied these requirements, as witnessed to by the fact that the fixed-point implementation was able to match C simulation results. The state metrics themselves are periodically normalized by subtracting an arbitrary metric from the current set of M state metrics. It was found that with the input signal bounded by 9 bits, a normalization period of once every 32 input samples was sufficient to prevent the state metrics from overflowing.

A final point regarding normalization is that all of the sets of data in the turbo decoder must be adjusted in the same manner as the systematic samples. That is, the same scaling that is applied to the systematic data must also be applied to the parity samples and to the appropriate set of extrinsic samples. The relative confidence that is placed in the various data sets is therefore maintained as the LLR's grow.

3.0 Performance

This section discusses the performance of the turbo decoder. The encoder used the structure shown in Figure 1, and each constituent RSC encoder used the TURBO4 polynomials given in [10], namely (23_8, 35_8). The performance results given here were gathered from a fixed-point C simulation, but the performance of the ADSP-2181 implementation was found to be virtually identical for all of the block sizes that were compared.

Figure 4 and Figure 5 show bit error rate (BER) and packet error rate (PER) performance for nominal rate 1/2 coding and block sizes of 64, 128, 256, and 512 information bits. Note that the code rate is not exactly 1/2 due to the presence of 4 termination bits; the actual code rates for each block size are 0.471, 0.485, 0.492, and 0.496, respectively. As expected, increasing the number of iterations of the turbo decoder from 4 to 8 leads to an improvement in performance, approximately 0.1dB at a BER of 10^{-4} and approximately 0.2dB at a PER of 10^{-3}. The effect of block size is apparent, and shows the advantage of using turbo codes with larger blocks.

Figure 6 and Figure 7 show the BER and PER performance for a block size of 512 information bits and nominal code rates of 1/3, 1/2, 2/3, 3/4. Again, the different code

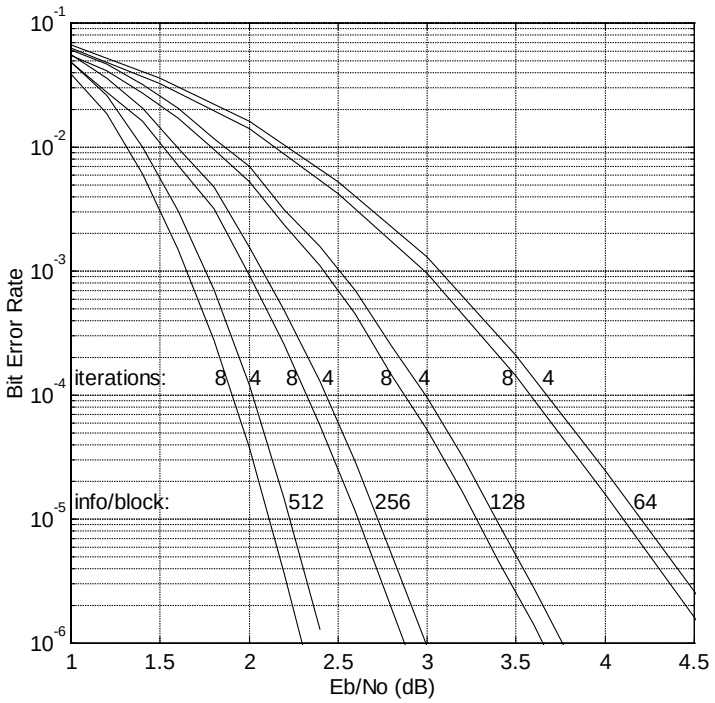

Figure 4: BER performance for several block sizes and rate 1/2 coding.

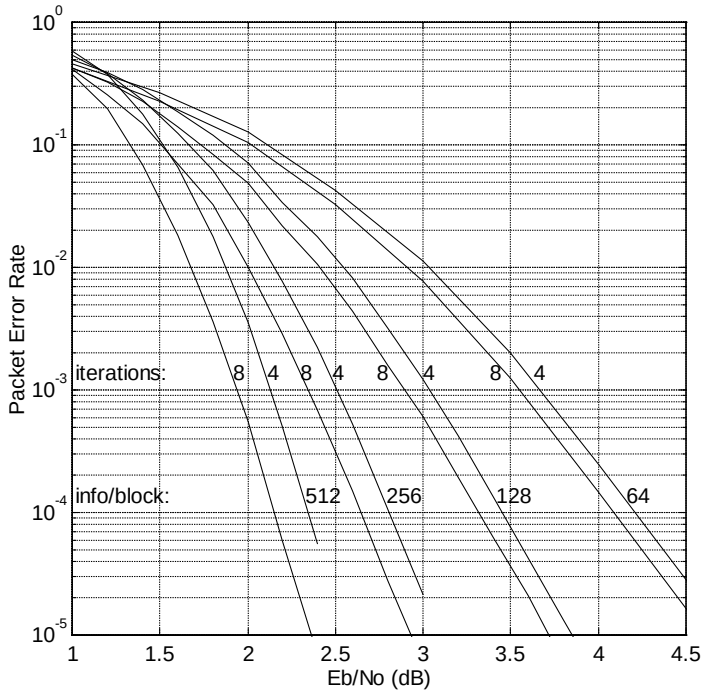

Figure 5: PER performance for several block sizes and rate 1/2 coding.

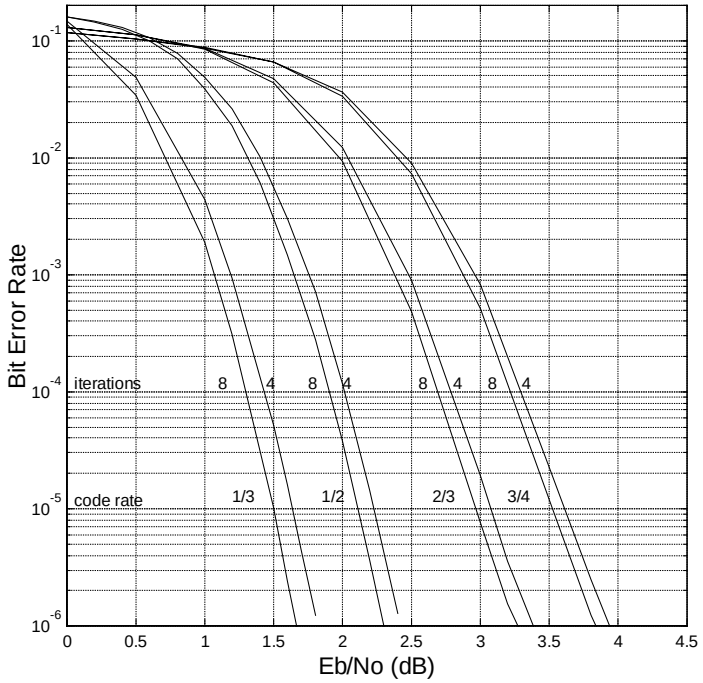

Figure 6: BER performance for a block size of 512 information bits and several code rates.

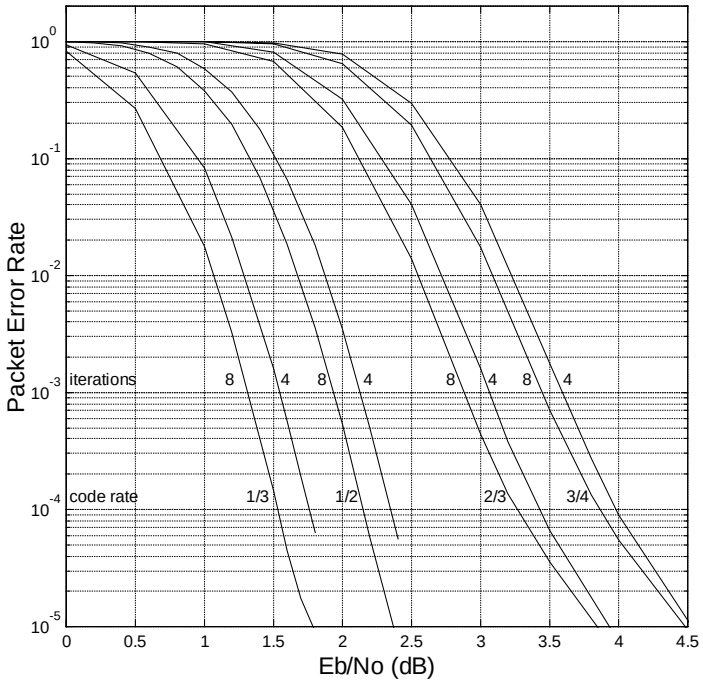

Figure 7: PER performance for a block size of 512 information bits and several code rates.

rates were achieved by puncturing the parity bits produced by the turbo encoder and are reduced slightly by the addition of 4 termination bits. The actual code rates are 0.331, 0.496, 0.661, and 0.744, respectively.

Figure 8 and Figure 9 show the BER and PER performance of the turbo decoder against that of a flushed Viterbi decoder with K=7 and K=9 for a block size of 128 information bits. It can be seen that both the turbo decoder and the K=9 Viterbi decoder yield the same performance at a PER of approximately 10^{-2}. In general, the turbo decoder performs better for higher signal-to-noise ratios while either Viterbi decoder performs better for lower signal-to-noise ratios. Even for this relatively small block size, it is apparent that the turbo decoder gives very competitive error rate performance. It is also interesting to note that the turbo decoder with 4 iterations and the K=9 Viterbi decoder have comparable complexities, based on DSP implementations and measured throughputs.

Iterations	***Achieved Throughput on a 40 MIPS ADSP-2181 (kbps)***	***Projected Throughput on a 40 MIPS ADSP-2181 (kbps)***
4	16.8	20
8	8.5	10

Table 1: Approximate throughput values, achieved and projected.

Table 1 shows both achieved and projected throughput values for a 40 MIPS version of the ADSP-2181. The turbo decoder with 4 iterations achieved 16.8 kbps. More recent work with the ADSP-2106x SHARC, a 32-bit device, has resulted in a decoder able to perform 4 iterations at a speed of 48 kbps. The projected throughputs shown in Table 1 are based upon incorporating algorithmic enhancements already present in the SHARC implementation into the ADSP-2181 implementation. The current ADSP-2181 implementation is able to accommodate a maximum block size of 650 information bits. This limit is dictated by the fact that only 32K of on-chip memory is available (16K data and 16K program). It is expected that block sizes up to about 2000 information bits could be accommodated with overlapped sub-block processing, though this would lead to a slight reduction in throughput.

A search for suitable interleavers to use with these block sizes was also done. While this search was not exhaustive, many different interleavers were tested and the results shown above were gathered with those that gave the best performance.

4.0 Conclusions

The structure and performance of a modified, low-complexity turbo decoder was presented. Performance results showed the effectiveness of turbo codes for large data blocks. Comparisons were drawn between the turbo decoder and Viterbi decoders of comparable complexity, showing that turbo codes display competitive performance even

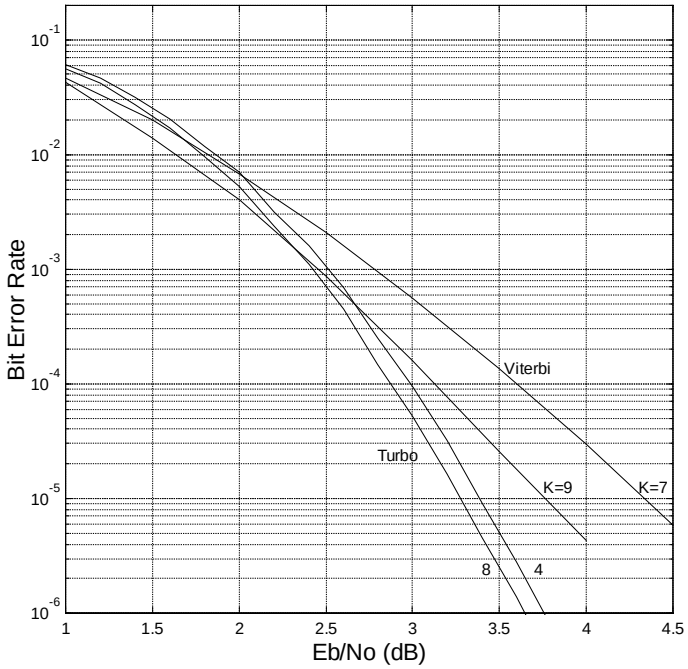

Figure 8: BER performance of the turbo decoder versus that of a standard zero-flushed Viterbi decoder with K=7 and K=9 (128 information bits).

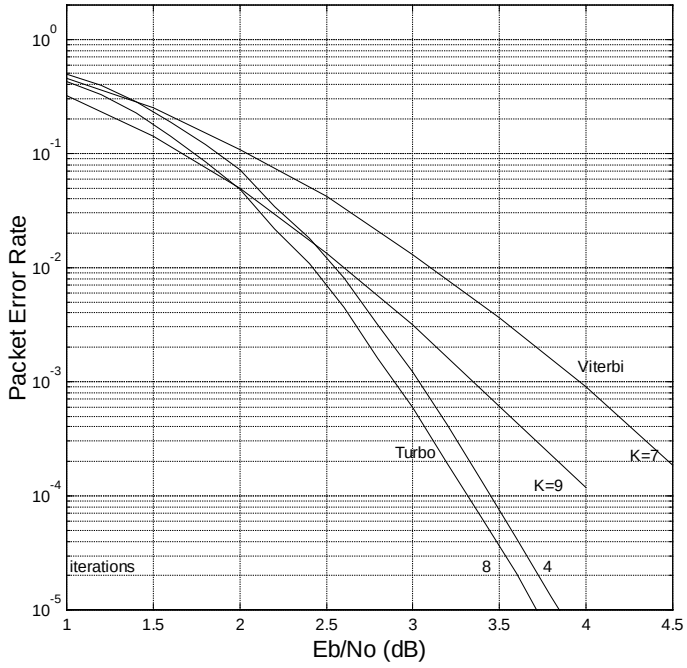

Figure 9: PER performance of the turbo decoder versus that of a standard zero-flushed Viterbi decoder with K=7 and K=9 (128 information bits).

for relatively short data blocks. A version of this decoder implemented on the Analog Devices ADSP-2181, a low-cost, 16-bit fixed-point DSP chip, was also described. Throughput for 4 iterations of the turbo decoder implemented on a 40 MIPS ADSP-2181 processor was found to be 16.8 kbps.

References

[1] C. Berrou and A. Glavieux, "*Near Optimum Error Correcting Coding and Decoding : Turbo-Codes*", IEEE Transactions on Communications, Vol.44, No.10, October 1996.

[2] C. Berrou, A. Glavieux, and P. Thitimajshima, "*Near Shannon Limit Error-Correcting Coding and Decoding: Turbo-Codes*", Proceedings of ICC'93, Geneva, Switzerland, pp. 1064-1070, May, 1993.

[3] P. Robertson, P. Hoeher, and E. Villebrun, "Optimal and Sub-Optimal Maximum a Posteriori Algorithms Suitable for Turbo Decoding", IEEE Communications Theory, Vol. 8, No. 2, March-April 1997.

[4] P. Robertson, E. Villebrun, and P. Hoeher, "A Comparison of Optimal and Sub-Optimal MAP Decoding Algorithms Operating in the Log Domain", Proceedings of ICC'95, Seattle, pp. 1009-1013, June 1995.

[5] L. Bahl, J. Cocke, F. Jelinek, and J. Raviv, "Optimal Decoding of Linear Codes for Minimizing Symbol Error Rate", IEEE Trans. on Inform. Theory, Vol. IT-20, pp. 284-287, March 1974.

[6] J. Hagenauer, E. Offer, and L. Papke, "Iterative Decoding of Binary Block and Convolutional Codes", IEEE Trans. on Inform Theory, Vol. 42, No. 2, pp. 429-445, March 1996.

[7] J. Erfanian, S. Pasupathy, G. Gulak, "Reduced Complexity Symbol Detectors with Parallel Structures for ISI Channels", IEEE Trans. on Communications, Vol. 42, No. 2/3/4, pp.1661-1671, February/March/April 1994.

[8] G. Forney, "*The Viterbi Algorithm*", Proceedings of the IEEE, Vol.61, No.3, pp. 268-278, March 1973.

[9] S. Pietrobon, "*Implementation and Performance of a Turbo/MAP Decoder*", Submitted to the International Journal of Satellite Communications, February 21, 1997.

[10] B. Talibart and C. Berrou, "*Notice Preliminaire du Circuit Turbo-Codeur/Decodeur TURBO4*", Version 0.0, June, 1995.