

# **TURBO CODES: a tutorial on a new class of powerful error correcting coding schemes**

## **Part II: Decoder Design and Performance**

S. Adrian Barbulescu (adrian@spri.levels.unisa.edu.au)

Institute for Telecommunications Research  
University of South Australia

Steven S. Pietrobon (steven@sworld.com.au)

Small World Communications

Revised: 28 October 1998

### ***Abstract***

This is a tutorial paper meant to introduce the reader to the new concept of turbo codes. This is a new and very powerful error correction technique which outperforms all previous known coding schemes. It can be used in any communication system where a significant power saving is required or the operating signal-to-noise ratio (SNR) is very low. Deep space communications, mobile satellite/cellular communications, microwave links, paging, etc., are some of the possible applications of this revolutionary coding technique.

Part I of the paper discussed the history of turbo codes, why they are different from traditional convolutional/block codes, the turbo encoder structures and issues related to the interleaver design.

Part II addresses the decoder architecture, the achievable performance for turbo codes for a wide range of coding rates and modulation techniques and discusses delay and implementation issues.

# 1 Introduction

The optimum decoding of turbo codes is the maximum likelihood decoding algorithm applied to the turbo code trellis structure. However, due to the interleaver embedded in the turbo encoder, the turbo code trellis will have an extremely large number of states. This fact makes the maximum likelihood decoding process almost impossible to implement in practice for large interleaver sizes.

A more practical approach is an iterative decoding approach where the maximum likelihood decoding algorithm is applied to the elementary convolutional/block codes of the turbo code.

This iterative process raises the question of how close its performance is to the performance of the optimum decoding process. In [1], it was shown that turbo codes are near-optimal codes. Simulation results based on iterative decoding approach within 0.7 dB of the sphere-packing lower bound for various code rates and information block size from 100 to 10,000 bits. Therefore, this iterative technique is a very efficient way to decode turbo codes and to achieve performance close to the theoretical limits.

The second question related to the iterative process is how to pass the information from one decoder to the other. We answer this in the following section. However, it is worth mentioning here that due to the iterative process, some parameters can grow to infinity unless special care is taken in the decoding algorithm. Limitations imposed on these parameters can have negative effects when very low BERs are simulated. Another cause for the limitation in BER performance is a poor interleaver design. Due to highly correlated sequences, the BER decreases to a certain level from where there is no further improvement in the decoding process. This is called the “error floor” effect of the turbo code BER curve.

Figure 1.1. shows three curves. The “error floor” curve is typical of bad interleaver design or truncations in the decoding algorithm. The simulation result curve also presents a change in slope but at very low BERs, when the theoretical bound for the turbo code is reached.

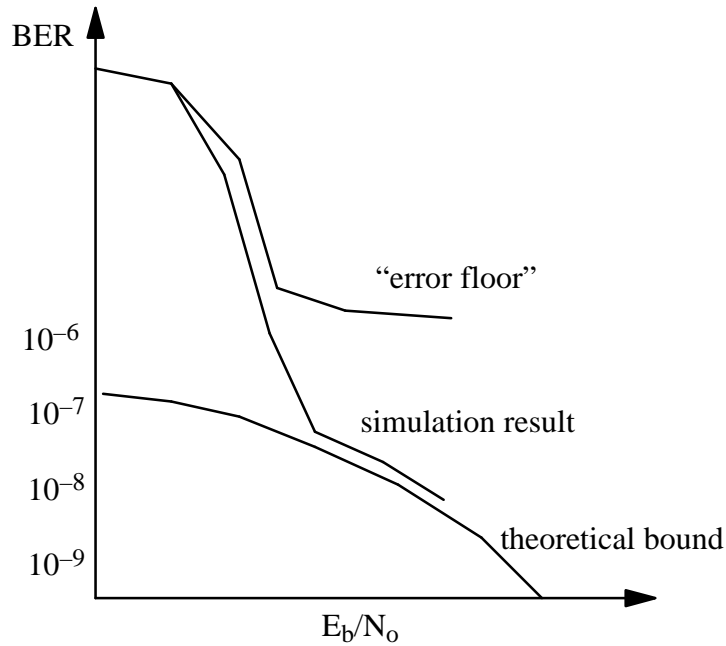


Figure 1.1: Typical BER curve for turbo codes.

## 2 Turbo Decoding

Consider the generic turbo encoder in Figure 2.1. The outputs of the turbo encoder are the information sequence  $\mathbf{D}_k$ , renamed as  $\mathbf{X}_k^-$  together with the corresponding parity sequence  $\mathbf{Y}_k^-$  produced by the encoder block  $\mathbf{ENC}^-$  and the parity sequence  $\mathbf{Y}_k^l$  produced by the encoder block  $\mathbf{ENC}^l$  at time  $\mathbf{k}$ . These sequences are modulated and sent through the channel. The interleaved data sequence  $\mathbf{X}_k^l$  is not sent because it can be regenerated at the receiver by interleaving the received sequence corresponding to  $\mathbf{X}_k^-$ . Upper case is used for the binary turbo encoder outputs. Lower case is used for the noisy received signals at the turbo decoder.

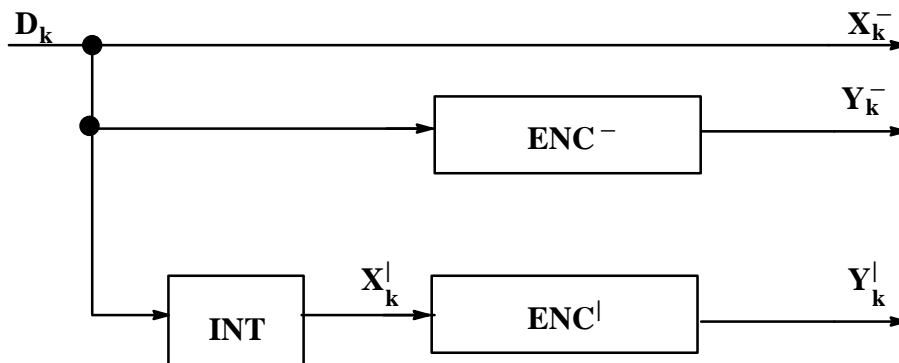


Figure 2.1: Generic turbo encoder.

At the receiver, decoding is performed in an iterative process as shown in Figure 2.2.

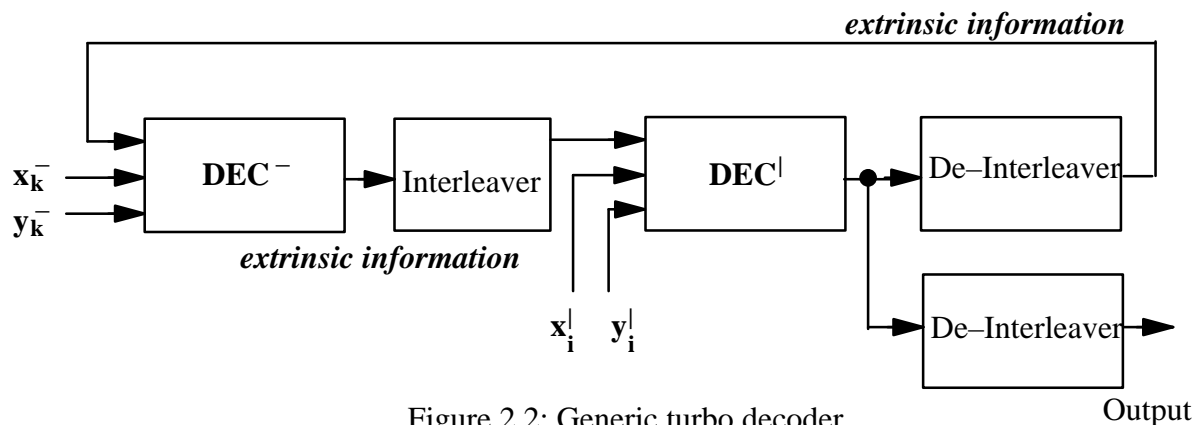


Figure 2.2: Generic turbo decoder.

Consider for example the case where a single iteration is employed using two soft decision decoders. Decoder  $DEC^-$  provides a soft output which is a measure of the reliability of each decoded bit. From this reliability information, the extrinsic information is produced, which does not depend on the current inputs to the decoder. This extrinsic information, after interleaving, is passed on to  $DEC^I$  which uses this information to decode the interleaved bit sequence. From the soft outputs of  $DEC^I$ , the new extrinsic information is fed back to the input of  $DEC^-$  and so on.

If an error occurs at the output of the first decoder due to a very noisy input, it might very well be corrected by the second decoder.

A soft decision decoder outputs a real number which is a measure of the probability of a correct decision. This real number is called the *a posteriori* probability (APP). There are two types of soft decision decoding algorithms which are typically used, the first being a modified Viterbi algorithm which produces soft outputs and hence is called a soft output Viterbi algorithm (SOVA) [2]. A second is the maximum *a posteriori* (MAP) algorithm [3, 4]. Estimates put the complexity of the MAP algorithm at two times that of the Viterbi algorithm. However, the MAP algorithm results in better performance at low SNR due to a more accurate evaluation of the APP.

The performance of a turbo coding scheme improves as the number of decoder iterations is increased. However, the coding gain from one iteration to another, decreases with the number of iterations. Each iteration involves two decoding stages. Therefore, the overall complexity of a turbo decoder depends on how efficient the decoding algorithm is implemented.

There are other algorithms that might be used in a turbo decoder. The soft input soft output (SISO) algorithm described in [5] is very similar to the MAP algorithm but easier to use in the case where there are parallel paths between two states in a trellis diagram.

Recently, a new algorithm based on the list output Viterbi algorithm (LOVA) [6] was applied to turbo codes [7]. This algorithm is proposed as an alternative technique to reduce the “error floor” of turbo codes. The overall receiver structure combines the conventional turbo decoder algorithm with a list decoder. After each iteration, the turbo decoder output is checked for errors using the CRC check. If the test is passed in less than a pre-defined number of iterations, the block is accepted, otherwise a list decoder is invoked and a list of  $L$  probable paths is produced. The algorithm used is similar to the soft list Viterbi algorithm (SLVA) presented in [8].

In order to reduce the computational complexity of the turbo decoder, an early detection method was presented in [9]. Based on a confidence criteria, some information symbols, state variables and parity symbols are detected early on during decoding. This early detection allows sections to be removed from the constituent trellises, resulting in spliced trellises that can be decoded faster and with a lower computational effort.

A reduced-search MAP algorithm was used in [10] where a reduction in computational complexity is achieved based on a threshold scheme. This scheme uses only the state metrics that are above a certain threshold in computing the bit estimates. However, the variable computational effort required might present difficulties for hardware implementation.

Given the principles of iterative decoding presented in [4, 11, 12], the detailed implementation of an iterative rate 1/2 decoder is described in Figure 2.3. This represents only one iteration of the decoding process. The whole block can be repeated for the number of desired iterations (less than 10 usually).

For an AWGN channel  $x_k$  and  $y_k$  represent the received noisy values of  $X_k$  and  $Y_k$ , respectively, which are scaled by the reliability value of the channel,  $L_c = 4 \frac{E_s}{N_0}$  where  $E_s/N_0$  is the SNR. For example, if the binary  $X_k$  is transmitted, the  $x_k$  value used by the decoder is:

$$x_k = 4 \frac{E_s}{N_0} (1 - 2X_k + n_k) \quad (1)$$

where  $n_k$  is the noise sample at time  $k$ .

A turbo decoder starts by first decoding the data transmitted by  $X_k$  and  $Y_k^-$ . The input to the first decoder must then be  $x_k$  and  $y_k^-$ , the punctured version of  $y_k$ .

Figure 2.3 shows the inputs to the first decoder ( $\text{DEC}^-$ ) as  $x_k + L_a^-(d_k)$  and  $y_k^-$ . The input  $L_a^-(d_k)$  represents the *a priori* information about the transmitted information bit  $d_k$ . In the first decoding stage, we know nothing about  $d_k$ , and so  $L_a^-(d_k)$  is set to zero. A positive value of  $L_a^-(d_k)$  would indicate that  $d_k = 0$  was likely to have been transmitted, while a negative value of  $L_a^-(d_k)$  would indicate that  $d_k = 1$  was likely to have been transmitted.

It can be shown that the output of  $\text{DEC}^-$  is equal to

$$L(d_j) = x_j + L_a^-(d_j) + L_e^-(d_j) \quad (2)$$

where the subscript  $j = k - D$ ,  $D$  is the delay of the decoder and  $L_e^-(d_j)$  is the extrinsic information of  $d_j$ . We can think of  $L_e^-(d_j)$  being a correction term produced by the decoder to correct any errors on the channel.

In Figure 2.3 we see that  $L_a^-(d_j)$  is subtracted from  $L(d_j)$  to give  $x_j + L_e^-(d_j)$ . Since  $L_a^-(d_j) = 0$  for the first iteration, this has no effect on later decodings, but as will be explained later, this is a very important computation.

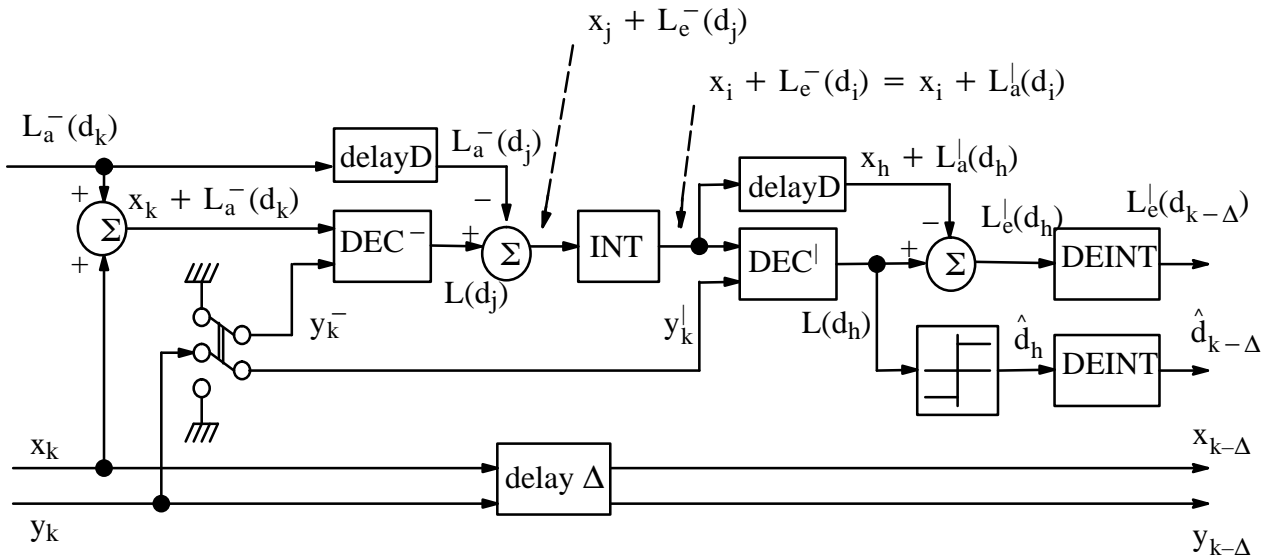


Figure 2.3: Rate 1/2 turbo decoder.

We now come to a very important block of a turbo decoder, the interleaver. A normal decoder will try to find the closest code sequence to the received noisy sequence that minimises some probability of error criteria. A good code has its code sequences separated by a large number of bits. Thus, it takes a large amount of noise in a short period of time to cause a decoder to make an error. Therefore, a decoder will produce its errors in bursts, instead of randomly as in an AWGN channel. The interleaver is used to randomise the burst errors from  $\text{DEC}^-$ . The larger

the interleaver, the better that the error bursts can be de-correlated. After interleaving, the data becomes  $x_i + L_e^-(d_i)$  which now corresponds to the data that is input to  $DEC^{\dagger}$ .

The extrinsic information from the first decoder can be thought of *a priori* information for the second decoder. So we let  $L_e^-(d_i) = L_a^{\dagger}(d_i)$  and feed in  $x_i + L_a^{\dagger}(d_i)$  and  $y_k^{\dagger}$ , the punctured version of  $y_k$  corresponding to  $Y_k^{\dagger}$ , in  $DEC^{\dagger}$ . The output of  $DEC^{\dagger}$  is

$$L(d_h) = x_h + L_a^{\dagger}(d_h) + L_e^{\dagger}(d_h) \quad (3)$$

where  $h = i - D$  and  $L_e^{\dagger}(d_h)$  is the new extrinsic information provided by  $DEC^{\dagger}$ . We can make a hard decision on  $L(d_h)$  and output the decoded data. Since  $d_h$  has been interleaved, a deinterleaver is required to provide the corrected data in the correct order. Since  $x_i + L_a^{\dagger}(d_i)$  has fewer errors than  $x_i$  alone,  $DEC^{\dagger}$  is able to correct more errors using the information from  $y_k^{\dagger}$ .

We can now use the new information from  $DEC^{\dagger}$  and feed it into the next  $DEC^-$ . To do this we want to obtain the *a priori* information from  $L(d_h)$ . We see from Figure 2.3 that we subtract  $x_h + L_a^{\dagger}(d_h)$  from  $L(d_h)$  to leave only  $L_e^{\dagger}(d_h)$ . We then deinterleave  $L_e^{\dagger}(d_h)$  to form  $L_e^{\dagger}(d_{k-\Delta}) = L_a^-(d_{k-\Delta})$  where  $\Delta$  is the total delay of  $DEC^-$ ,  $DEC^{\dagger}$ , INT and DEINT.  $L_a^-(d_{k-\Delta})$  then becomes the *a priori* information for the next  $DEC^-$  and the decoding process starts over again.

Note that we do not include  $L_a^{\dagger}(d_h) = L_e^-(d_{i-D})$  in  $L_a^-(d_{k-\Delta})$ . This is because after deinterleaving,  $L_e^-(d_{i-D})$  becomes  $L_e^-(d_{k-\Delta})$ . Thus any error bursts in  $L_e^-(d_{k-\Delta})$  would now reappear which we should avoid feeding into  $DEC^-$ . However, any error bursts in  $L_e^{\dagger}(d_h)$  will be de-correlated by DEINT. This is also the reason why we subtract  $L_a^-(d_j)$  from  $L(d_j)$  (since INT will re-correlate the error bursts in  $L_a^-(d_j)$ ).

The above process is iterated many times until eventually all the errors are corrected, or there remains an error pattern that can not be corrected, despite the interleaving and deinterleaving processes.

### 3 Performance of turbo codes

The performance of turbo codes depends on the interleaver size, the interleaver design, the constituent codes, and the number of decoder iterations. The larger the interleaver size, the lower the bit error ratio which can be achieved. Since increasing the interleaver size increases the encoding and decoding delay, to achieve the best performance from a turbo code requires a large corresponding delay. However, even for small delays which imply small interleaver sizes, their performance is still better than any other conventional codes of similar complexity. We compare the performance of turbo codes for different interleaver sizes and different number of iterations with the performance of the industry standard rate 1/2, constraint length (K) seven, convolutional code [13], labelled as “cc” in Figures 3.1 and 3.2. In all simulations, the *maximum a posteriori* (MAP) algorithm was used [3]. We also used S-type interleavers as described in Section 4.7 of Part I. Because PCCC schemes (Figure 3.2 of Part I) achieve better BER at low SNR than other code structures, we show mainly results for these type of codes.

Figure 3.1 shows the performance of a 16 state rate 1/2 turbo codes for up to 8 iterations that we simulated for an interleaver of 200 bits.

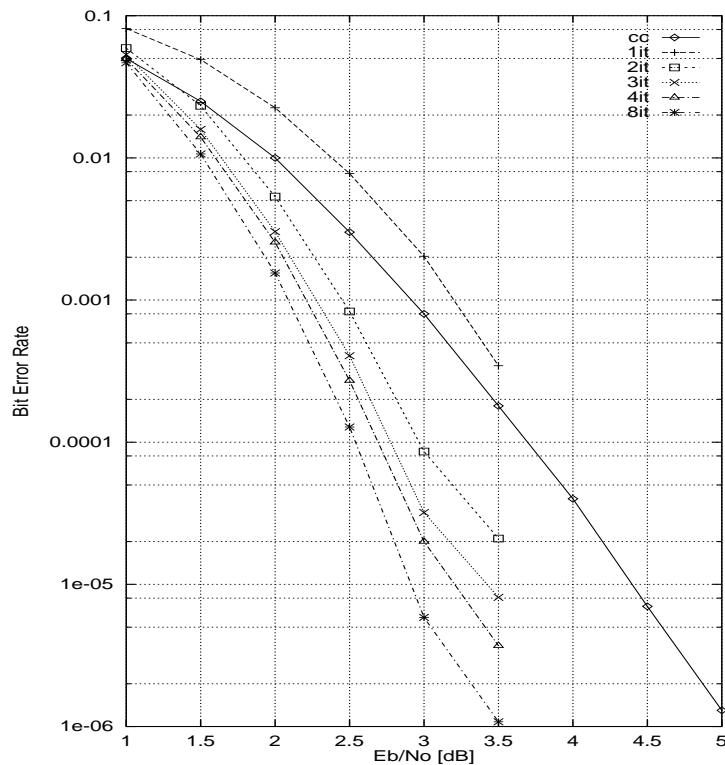


Figure 3.1: BER for rate 1/2 turbo codes, 200 bit interleaver, QPSK modulation, AWGN channel.

This size is close to the short frame sizes required in GSM (189 bits) or Digital Cellular System, DCS1800, (192 bits) and is limited by the maximum acceptable delay in the system. The two bit error rates (BER) of interest are  $4 \times 10^{-2}$  for speech and  $10^{-5}$  for data transmissions. We can see that turbo codes have a significant coding gain for the above BERs when compared with the performance of the industry standard convolutional code [14].

Figure 3.2 shows the performance of a 16 state rate 1/2 turbo codes that we simulated for an interleaver of 2000 bits and QPSK modulation. This size is suitable for applications which require a delay of less than 100ms and bit rates of 64kbit/s.

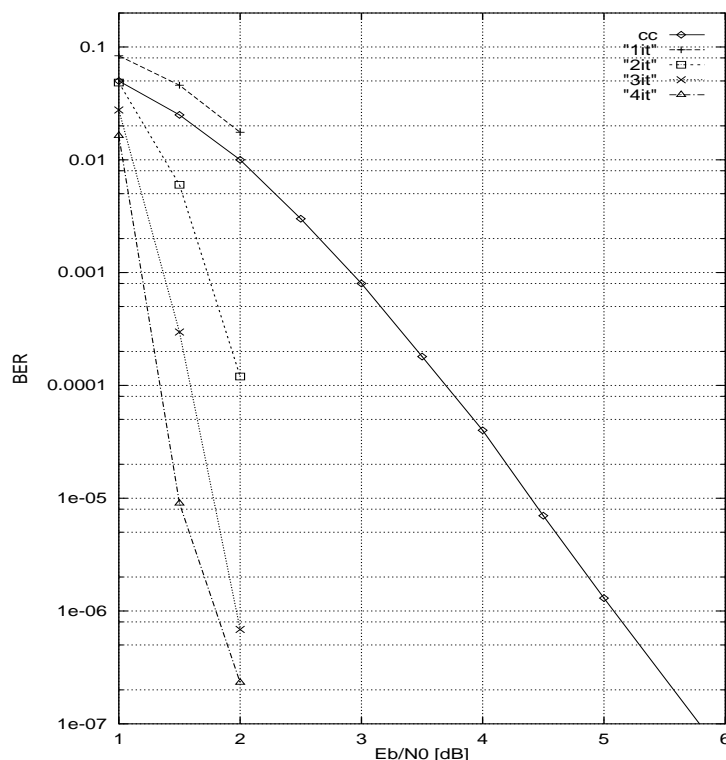


Figure 3.2: BER for rate 1/2 turbo code, 2000 bit interleaver, QPSK modulation, AWGN channel.

For higher interleaver sizes the performance of turbo codes improves significantly. Also, the time to run the simulations increases to days and weeks. Figure 3.3 shows the performance of turbo codes for interleaver sizes of 16,384 and 65,536 bits [15, 16]. The BER curves are as follows:

- R1: two 16 state rate 2/3 constituent codes used to construct a rate 1/2 turbo code, 12 iterations, 16,384 bits interleaver.
- R2: two 32 state rate 1/2 constituent codes used to construct a rate 1/2 turbo code, differential encoding, 18 iterations, 16,384 bits interleaver.

– R3: two 16 state rate 1/2 punctured constituent codes used to construct a rate 1/2 turbo code, 18 iterations, 256x256 bits interleaver.

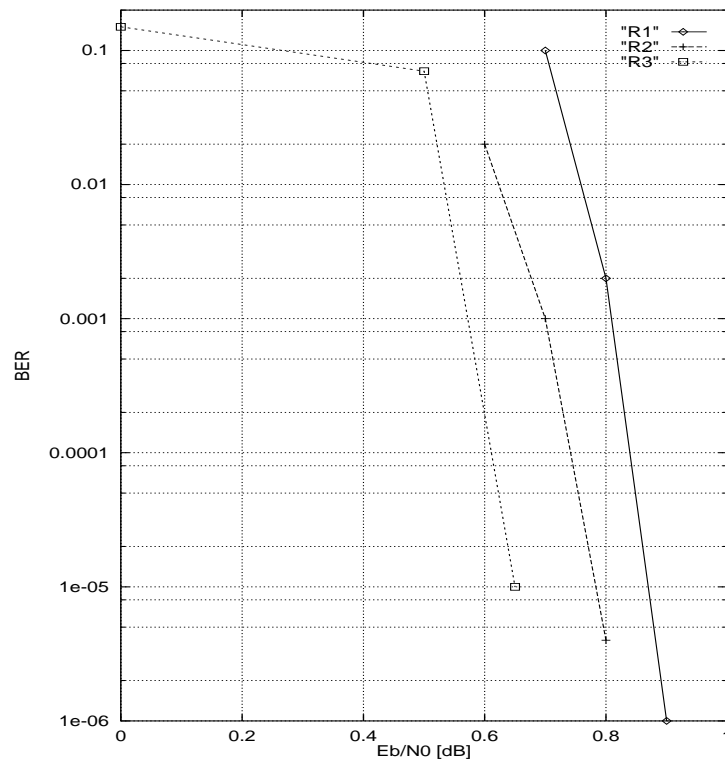


Figure 3.3: BER for rate 1/2 turbo code, 16,348 (R1, R2) and 65,536 (R3) bit interleavers, QPSK modulation, AWGN channel.

The best performance achieved with turbo codes so far is a BER lower than  $10^{-5}$  at 0.35 dB above the Shannon limit [17]. This is the highest coding gain ever achieved.

For 2 bit/symbol bandwidth efficiency, Figure 3.4 shows the performance of rate 1/2 turbo codes we simulated (labelled “TC”) for an interleaver size of 4096 bits and 16QAM [18]. In this case we compare the turbo code performance with the performance of trellis-coded modulation (TCM) schemes as follow:

R1 = a rate 1/2 turbo code, 4096 bits interleaver, with 16QAM [19]

R2 = a rate 2/3 TCM 64 state with 8PSK [20]

R3 = a rate 1/2 TCM 64 state code with 4AM.

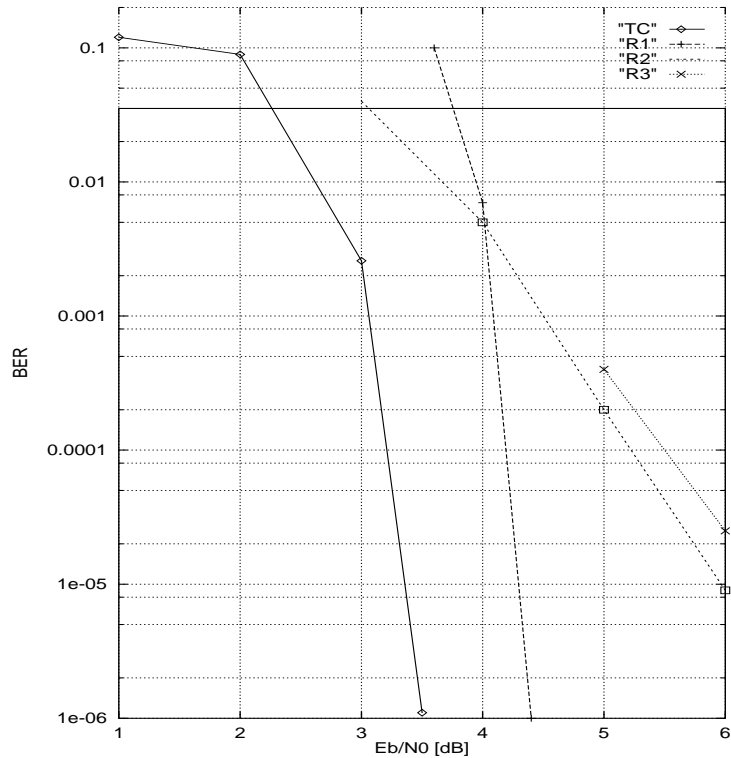


Figure 3.4: BER for rate 1/2 turbo codes and TCM, AWGN channel, 2 information bit/symbol.

Figure 3.5 shows results for 16,384 bits interleaver size. We used the same code and mapping as described in [18] for the curve labelled “TC”. This code is compared against the performance of turbo trellis-coded modulation (TTCM) schemes as described in [15]. The curves are given for the following schemes:

- R1: TTCM, two 16 state rate 4/5 constituent codes used to construct a rate 2/3 turbo code, 8 iterations, 16,384 bits interleaver, 8PSK.
- R2: TTCM, two 16 state rate 2/3 constituent codes used to construct a rate 1/2 turbo code, 9 iterations, 16,384 bits interleaver, 16QAM.

Figure 3.5 clearly shows that turbo codes give excellent performance not only for BPSK/QPSK modulations, but even for higher order modulations in order to achieve higher bandwidth efficiencies. The curve labelled “R3” is for a TTCM scheme using 64QAM modulation. Two 16 state rate 4/5 constituent codes were used to construct a rate 2/3 turbo code. The results are for 10 iterations and 16,384 bits interleaver size.

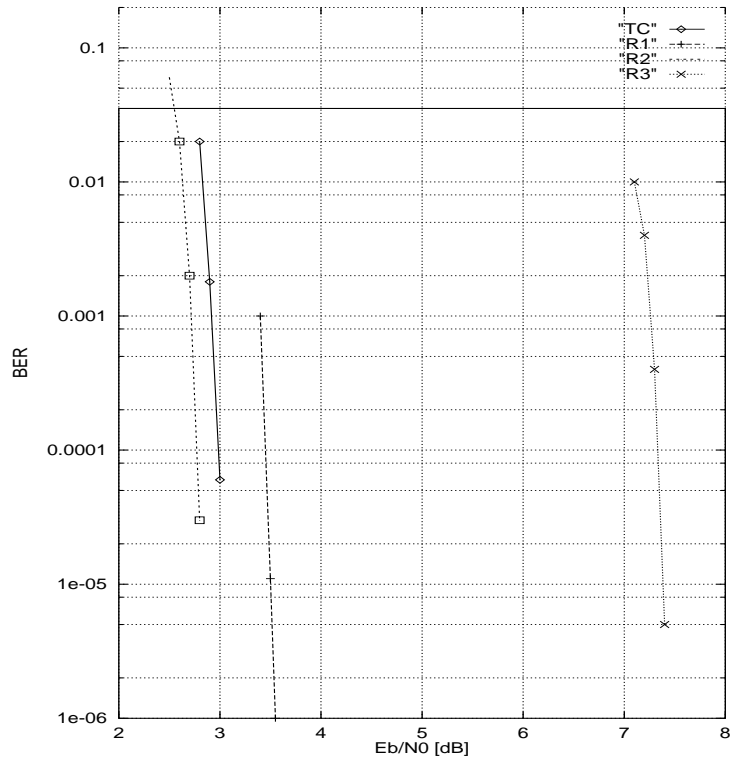


Figure 3.5: BER for higher order modulation schemes, 8PSK/16QAM/64QAM, 2/2/4 information bit/symbol, AWGN channel.

For two dimensional turbo codes (two sets of parity data without puncturing), the BER is proportional with  $N^{-1}$  where  $N$  is the interleaver size. We can add a second interleaver and a third encoder to produce a rate quarter code which is a three dimensional turbo code. Generalising, for a  $D$  dimensional turbo code, the BER is proportional with  $N^{(1-D)}$ . This formula is accurate for large interleavers and high SNR values.

The simulation results for 16 state rate 1/3 and 1/4 turbo codes for 200 and 2000 bit interleaver sizes in an AWGN channel are given in Figure 3.6.

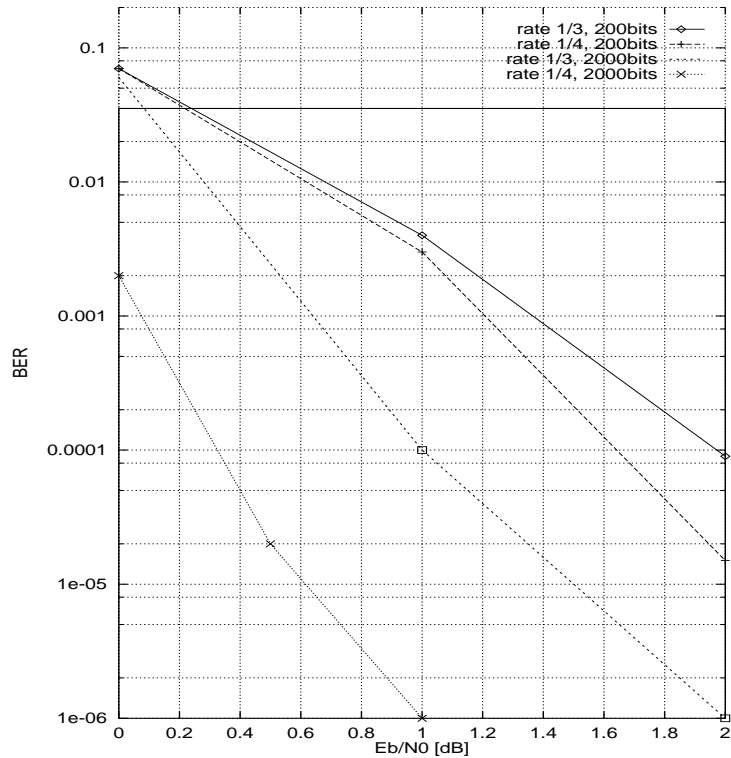


Figure 3.6: BER for rate 1/3 and rate 1/4 turbo codes for 200 and 2000 bit interleaver, AWGN channel.

For lower bandwidth efficiencies, as in the case of deep-space applications, the delay is not important and therefore huge interleavers of 16,384 bits can be used with very small coding rates. Figure 3.7 shows BER curves for rate 1/3 and 11 iterations, rate 1/4 and 13 iterations, rate 1/15 and 12 iterations [15].

Turbo codes also perform very well in a fading environment. The interleaver embedded in the turbo encoder helps to recover data from small fades better than traditional convolutional codes. Turbo codes are currently being tested for mobile satellite/cellular channels. The *turbo diversity combining scheme* we described in [21] can be used to give better results than maximum ratio combining of convolutional codes. This scheme can transform a transmitted rate half turbo code into a received rate third turbo code if both channels are received.

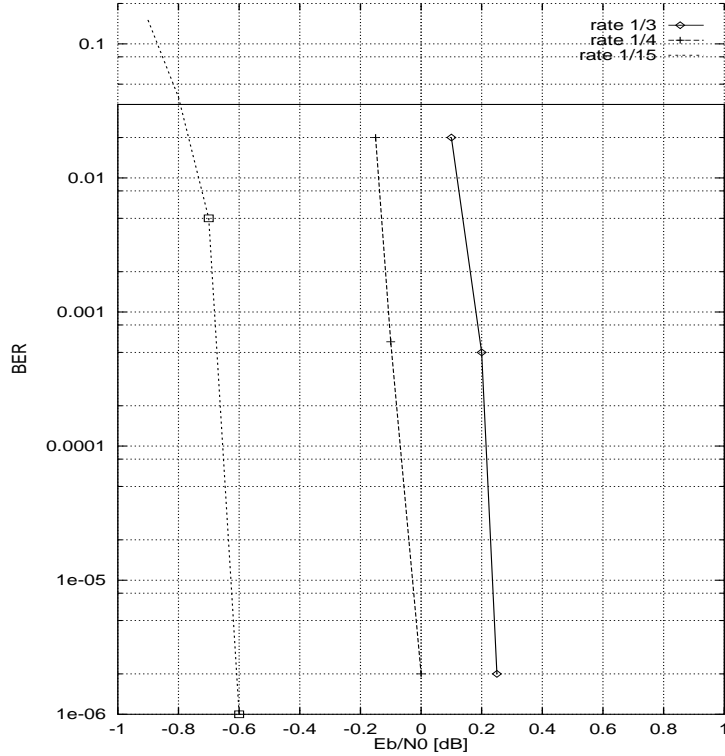


Figure 3.7: BER for low rate turbo codes and 16,384 bits interleaver, BPSK modulation, AWGN channel.

## 4 Implementation complexity

The turbo encoder is equivalent to two convolutional encoders plus, a look-up table (LUT) to store the interleaver addresses. The turbo decoder includes two soft output decoders plus a random access memory (RAM) for the interleaver/deinterleaver functions. The soft output decoders can be either soft output Viterbi decoders or MAP decoders. MAP decoders give 0.5 to 1.0 dB more coding gain than soft output Viterbi decoders and are approximately two times more complex in number of equivalent computations. All previous results were obtained using MAP decoders.

For a DSP implementation we need to estimate the number of instructions needed per decoded bit. Considering the encoder memory  $v$  for each recursive systematic code, the total number of instructions for the MAP algorithm can be estimated at  $3 \times 2^{v+1}$  per decoded bit. Since the MAP algorithm is used twice in each iteration of the turbo decoding algorithm, the required DSP speed (in instructions per second) is

$$f_i = 3k \times 2^{v+2} \times f_b, \quad (4)$$

where  $k$  is the number of iterations and  $f_b$  is the required bit rate (in bit/s).

The SOVA algorithm is considerably simpler than the MAP algorithm. It is basically the Viterbi algorithm with some small changes. The largest change is in the add–compare–select (ACS) circuit where the difference between path metrics into a state is calculated. The difference for each state is stored along with the path decision. Traceback occurs as normal with the absolute difference used to indicate the reliability of a decoded bit. A difference of zero indicates that the bit is very unreliable, while a large value indicates a reliable value.

It is assumed that each instruction on the DSP chip can perform an addition or a subtraction. Thus  $2^{v+1}$  instructions are required for the path metric computations and  $2^v$  instructions for the subtraction. The traceback involves reading the memory and logic shifting a register for each decoded bit. It is assumed that this requires 2 instructions. Each decoded bit will require this operation to be performed twice, giving a total of 4 instructions. The total number of instructions per decoded bit is thus  $3 \times 2^v + 4$ .

## 5 Decoder Delay

An important consideration is the decoder delay. The dominant factor in this delay is the interleaver. Let  $f(i)$  be the interleaved address of the input  $i$ . Then, it can be shown that the minimum interleaver delay  $D_I$ , can be made equal to

$$D_I = \max |i - f(i)| \quad \text{from } i = 1 \text{ to } N. \quad (5)$$

It is usually possible to design a good interleaver such that  $D_I = N/2$ . For an encoder, the delay will therefore be equal to  $N/2$ . For low data rates, we let the receiver wait until a whole block of data is received. This causes an additional delay equal to  $N$ . Then, the decoder can iteratively decode the received data while the next data block is arriving. For an infinitely fast processor, we could do this in zero time, implying a total delay equal to  $1.5N$ . More realistically we would spread the processing load while receiving the next block so that the delay would be equal to  $2.5N$ .

For higher data rates, decoding must be done in pipeline. The total delay would then be equal to

$$D_P = 2N_i(D_D + D_I) \quad (6)$$

where  $N_i$  is the number of iterations and  $D_D$  is the decoder delay. The decoder delay can be significantly reduced to a few hundred bits by using a continuous decoding algorithm [4, 11, 22].

As can be seen from the above equation, the total bit delay can be quite large. However, since the data rate is high, the corresponding time delay can be made very small.

## 6 Examples of turbo decoders

The first commercially available turbo decoder chip was produced in association with the inventor of turbo codes, Claude Berrou [23]. The rate 1/2 decoder implements five eight state SOVA decoders in parallel (2.5 iterations), and four  $32 \times 32 = 1024$  bit interleavers/deinterleavers to achieve 40 Mbit/s and a delay of 2318 bits. A BER of  $10^{-5}$  is achieved at an  $E_b/N_0 = 2.7$  dB.

A single iteration of two state SOVA decoders in parallel with  $64 \times 32 = 2048$  bit interleaver/deinterleaver was later implemented in the “turbo4” chip [24]. This single iteration has a delay of 2178 bits and achieves a BER of  $10^{-5}$  at an  $E_b/N_0 = 2.1$  dB for three iterations and rate 1/2.

A flexible turbo decoder that is programmable down to rate 1/7, from 4 to 512 states, and interleaver sizes up to 65,536 bits was developed at ITR using Xilinx XC3000 field programmable gate arrays (FPGA) [25]. For 16 state codes, a decoder speed up to 356 kbit/s was achieved. With seven iterations in parallel the decoder obtained an  $E_b/N_0$  of 0.32 dB and  $-0.30$  dB at rates 1/3 and 1/7, respectively. Each decoder iteration was implemented on a 6U size card.

Another turbo decoder using Actel FPGAs has been implemented by Jet Propulsion Laboratory [26]. This decoder operates down to rate 1/10, up to 64 states, and interleaver sizes to 65,536 bits. Like the decoder in [25], the log-MAP [4, 27] algorithm is used.

Although not a turbo decoder, a 16 state MAP decoder implemented in a single Xilinx XC4000 FPGA is available from [28]. This is a continuous log-MAP decoder with a maximum delay of 257 bits and decoding speeds up to 1.6 Mbit/s. The decoder provides soft-inputs and outputs that allow it to be used in a turbo decoder if desired.

Recently, ITR designed a flexible turbo decoder in VHDL using a single Altera FPGA [18]. VHDL is an acronym that stands for VHSIC Hardware Description Language. VHSIC is another acronym which stands for Very High Speed Integrated Circuits. The whole turbo decoder fits on a mezzanine card as shown in Figure 6.1.

The card can fit on any DSP board which has a PMC slot and two fast 4 bit serial ports for data transfer. One single card can run all required number of iterations, or can run a single MAP

decoder (half of an iteration) and be connected in a daisy chain with other cards to increase the decoding speed. A typical MAP decoding speed is 800 ns/bit, that is an information data rate of 1.25 Mbit/s.



Figure 6.1: Turbo decoder card

The turbo decoder card can be mounted on a DSP card as shown in Figure 6.2.

The DSP card can implement all modulator and demodulator functions at baseband. An IF card can be used to translate the signal from baseband to the required intermediate frequency and

vice-versa. Both the DSP and IF cards can be controlled via the PCI bus of a PC. A user-friendly graphical interface is used to set up the modem parameters and monitor its performance.

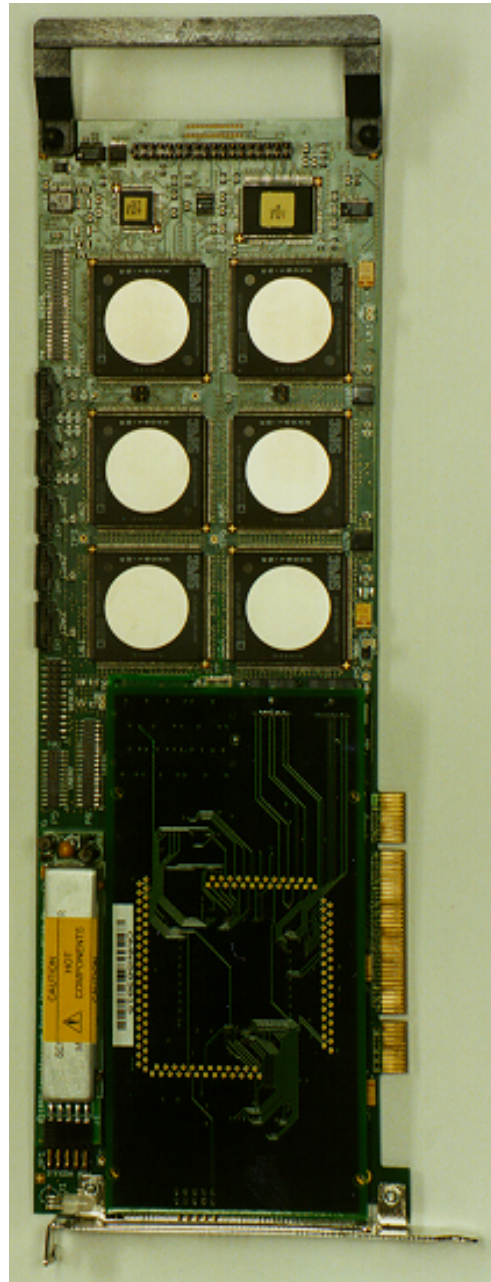


Figure 6.2: DSP Card with mezzanine turbo decoder card

## 7 Conclusions

The recent developments in the area of turbo codes for AWGN and fading channels showed the great potential of this new coding technique in achieving communications at very low values of  $E_b/N_0$ . The performance of turbo codes depends on the interleaver design which is embedded in the turbo encoder. A larger interleaver size, which means a longer decoding delay, gives a

lower bit error rate. Due to their excellent performance, it is expected that turbo codes will become the standard coding technique by the end of this millennium.

## 8 Acknowledgements

We want to give special thanks to Professor Ken Lever for his useful suggestions in reviewing this paper.

## 9 References

1. Dolinar S., Divsalar D. and Pollara F., “Code performance as a function of block size,” TDA Progress Report 42–133, May 1998.
2. J. Hagenauer and P. Hoher, “A Viterbi Algorithm with Soft–Decision Outputs and its Applications”, In *Proc. Globecom '89*, (Dallas, USA), pp 1680–1686, 1989
3. L. Bahl, J. Jelinek, J. Raviv, and F. Raviv, “Optimal Decoding of Linear Codes for minimising symbol error rate,” *IEEE Transactions on Information Theory*, vol. IT–20, pp. 284–287, March 1974.
4. S. Pietrobon and S. A. Barbulescu, “A simplification of the modified Bahl decoding Algorithm for systematic convolutional codes, “ *Int. Symp. on Inform. Theory and its Applications*, (Sydney, Australia), pp. 1073–1077, Nov. 1994.
5. S. Benedetto, G. Montorsi, D. Divsalar and F. Pollara, “Serial Concatenation of Interleaved Codes: Performance Analysis, Design, and Iterative Decoding”, *JPL TDA Progress Report 42–126*, August 15, 1996.
6. N. Seshadri and C–E. W. Sundberg, “List Viterbi Decoding Algorithms with Applications”, *IEEE Trans. Comm.*, vol. 42, pp. 313–323, Feb/Mar/Apr 1994.
7. K. R. Narayanan and G. L. Stueber, “List decoding of Turbo codes” submitted to *ICC'98*.
8. C. Nill and C–E. W. Sundberg, “List and Soft Symbol Output Viterbi Algorithms: Extensions and Comparisons”, *IEEE Trans. Comm.*, vol. 43, pp. 277–287, Feb/Mar/Apr 1995.
9. B. J. Frey and F. R. Kschischang, “Early Detection and Trellis Splicing: Reduced–Complexity Iterative Decoding”, *IEEE Journal on Selected Areas in Communications*, vol.16, no. 2, pp. 153–159, Feb. 1998

10. V. Franz and J.B. Anderson, "Concatenated Deoding with a Reduced-Search BCJR Algorithm", *IEEE Journal on Selected Areas in Communications*, vol.16, no. 2, pp. 186-195, Feb. 1998
11. Barbulescu, S. A., "Iterative decoding of turbo codes and other concatenated codes," Ph.D. dissertation, Feb. 1996.
12. Sklar B. "A Primer on Turbo Codes", *IEEE Communications Magazine*, vol 35, no. 12, pp94-102, Dec 1997.
13. Clarke, G. C. and Cain, J. B., "Error-correction coding for digital communications," Plenum Press, New York, 1983.
14. Jung, P. and Nasshan, M., "Designing turbo codes for speech transmission in digital mobile radio systems," *IEEE Int. Conf. on Telecom. 1995*, Bali, Indonesia, Apr. 1995.
15. Divsalar, D. and Pollara, F., "On the design of turbo codes," *TDA Progress Report 42-123*, Nov. 1995.
16. Berrou, C., Glavieux, A. and Thitimajshima, P., "Near Shannon limit error-correcting coding and decoding: turbo-codes," *ICC 1993*, Geneva, Switzerland, pp. 1064-1070, May 1993.
17. C. Berrou, "Some clinical aspects of turbo codes," *Proc. Int. Symp. on Turbo Codes and Related Topics*, (Brest, France), pp 26-31, Sep. 1997.
18. S. A. Barbulescu, W. Farrell, P. Gray, and M. Rice, "Bandwidth Efficient Turbo Coding for High Speed Mobile Satellite Communications," in *Proc. Int. Symp. on Turbo Codes and Related Topics*, (Brest, France), pp 119-126, Sep. 1997.
19. Le Goff, S., Glavieux, A. and Berrou, C., "Turbo-Codes and high spectral efficiency modulation", *Proceedings of ICC '94*, New Orleans, Louisiana, p. 648, May 1994.
20. Pietrobon, S. S. *et al.*, "Rotationally Invariant Nonlinear Trellis Codes for two-dimensional modulation", *IEEE Transactions on Information Theory*, vol 40, no 6, p. 1790, Nov. 1994.
21. Barbulescu, S. A., Rice, M. and Pietrobon, S. S. "Turbo diversity combining scheme," *Proceedings of the Seminar on Turbo Coding*, Lund, Sweden, pp. 51-57, Aug. 1996.

22. Pietrobon, S. S., "Implementation and performance of a serial MAP decoder for use in an iterative turbo decoder," *1995 IEEE Int. Symp. Inform. Theory*, Whistler, Canada, pp. 471, Sep. 1995.
23. Comatlas, "CAS 5093 40 Mbit/s turbo code codec." rev 4.1, May 1995.
24. M. Jezequel, C Berrou, J. R. Inisan and Y. Sichez, "Test of a turbo–encoder/decoder," Turbo Coding Seminar, Lund, Sweden, pp.35–41, Aug. 1996.
25. S. S. Pietrobon, "Implementation and performance of a Turbo/MAP decoder", *International Journal of Satellite Communications*, vol. 16, pp.23–46, 1998.
26. D. Watola, "JPL turbo decoder," <http://www331.jpl.nasa.gov/public/TurboDecoder.html>.
27. P. Robertson, E. Villebtun and P. Hoehner, "A comparison of optimal and suboptimal MAP decoding algorithms operating in the log domain," ICC'95, Seattle, WA, USA, pp.1009–1013, June 1995.
28. Small World Communications, "MAP04 16 State MAP Decoder", ver 1.0, Sep. 1998 <http://www.sworld.com.au/products>.