Test of a Turbo-Encoder/Decoder

Michel Jezequel, Claude Berrou, Inizan Jean-René, Sichez Yves

Test of a Turbo-Encoder/Decoder

Michel Jézéquel, Claude Berrou, Jean René Inisan and Yves Sichez.

École Nationale Supérieure des télécommunications de Bretagne
Département Électronique
BP 832 29285 Brest Cedex, France

Phone (33) 98 00 11 61, Fax (33) 98 00 13 43
e-mail: Michel.Jezequel@enst-bretagne.fr

Abstract

This paper presents tests on the integrated circuit called "turbo4" which can be used as a turbo-encoder or as a turbo-decoder. The turbo-encoder is built using a parallel concatenation of two recursive systematic convolutional codes with constraint length k=5. The turbo decoder is cascadable, each circuit processing one iteration of the turbo-decoding algorithm. The decoder is built around 2 sixteen-state modified Viterbi decoders and 2 matrices of 64 x 32 bits for interleaving and deinterleaving.

The tests on the circuit have been using a digital channel simulator. This kind of simulator makes tests with the lowest bit error rates possible. Some measures for Gaussian and Rayleigh channels and for different coding rates are presented. Results are in accordance with the simulations. Finally, some conclusions for helping to improve the performances of future circuits are given.

1. Introduction

Turbo-codes are a new family of codes dedicated to channel coding invented by C. Berrou and al [1,2,3]. They implement a parallel concatenation of recursive and systematic convolutional codes eventually punctured. The decoding process is iterative and each iteration is associated with one module. These modules are assembled in a serial pipeline structure, each additional module giving an additional gain. Turbo-codes show results which are very close to the theoretical channel limit.

Turbo-codes have been implemented in two integrated circuits. The first one called "CAS5093" distributed by COMATLAS [4] is built around 5 eight state modified Viterbi decoders [5] and 4 matrices of 32 x 32 bits for interleaving and deinterleaving. This circuit contains 2.5 modules. The second one called *turbo4* includes one module and is cascadable, so the user can create a decoder consisting of several modules.

To test these circuits, digital channel simulators were designed at Telecom Bretagne. Results of the tests on *turbo4* are presented in this paper.

The paper is organised as follows : section 2 presents the *turbo4* integrated circuit. Section 3 is dedicated to the channel simulator. The following section gives the results of the tests. Finally, we conclude by indicating some solutions for helping to improve the performances of future circuits.

2. *Turbo4* turbo-Encoder/Decoder

Turbo4 includes a turbo-encoder and a turbo-decoder. The circuit can be used as an encoder or as a decoder.

The synchronisation of *turbo4* is well adapted to the coding rate R=1/2. However

the circuits can work with a coding rate between 1/3 and 5/6.

2.1. Turbo4 Encoder

The turbo-encoder is built using a parallel concatenation of two recursive systematic convolutional codes (figure 1). A synchronisation block is added in order to make synchronisation between the encoder and the decoder possible.

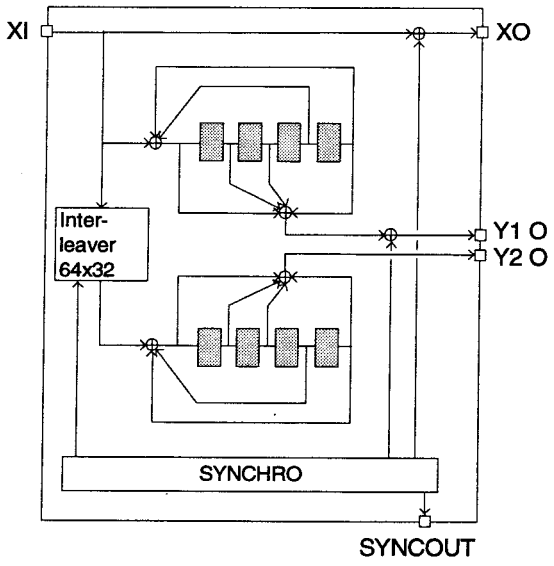

figure 1 : Turbo-encoder

2.2. Turbo4 Decoder

The decoder processes one turbo-decoding iteration. It is designed (figure 2) around 2 sixteen-state SOVAs (Soft Output Viterbi Algorithm, an acronym proposed by J. Hagenauer [6]) and 2 matrices of 64 x 32 bits for interleaving and deinterleaving. Moreover, the synchro/supervision block ensures the synchronisation between the encoder and the decoder.

The decoder is cascadable, so the user can choose the number of iterations, each circuit corresponding to one iteration. Its programmability makes it possible to adapt certain coefficients to the number of the iteration.

2.3. Main Characteristics

- Turbo-code with constraint length k=5 (polynomials 23,35)
- Non uniform interleaving (size 64 x 32 bits)
- Cascadable decoder
- Soft output
- Coding gain for a Gaussian channel with 4 iterations for the decoding process (4 circuits for the decoder) :
  - * 9 dB @ BER 10^{-7}, R=1/2
  - * 8 dB @ BER 10^{-7}, R=2/3
- Encoding latency : 4

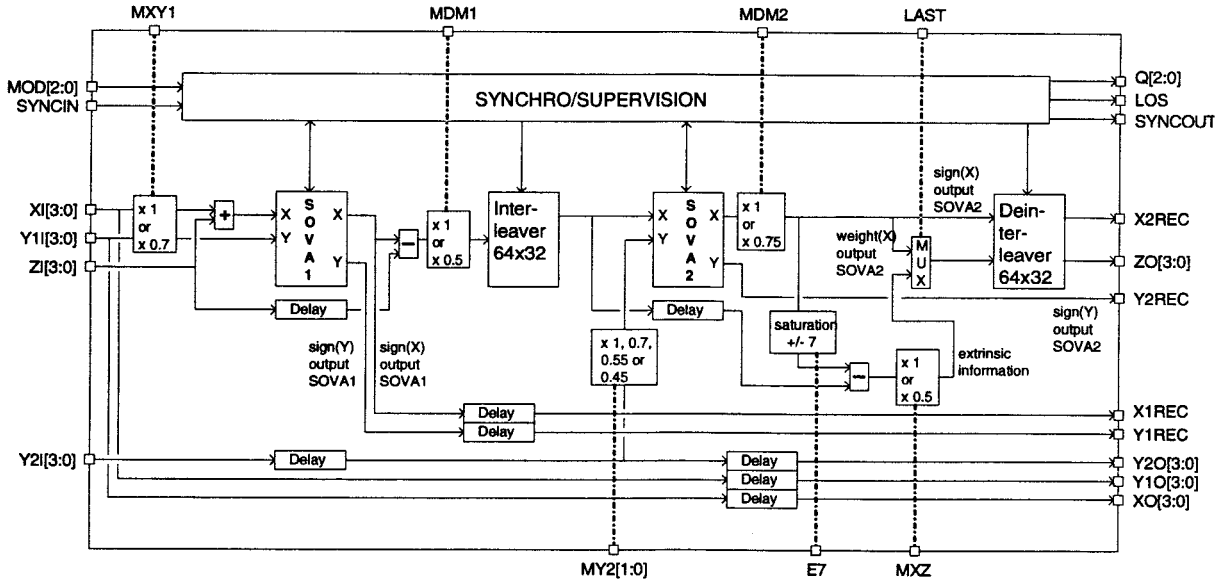

figure 2 : Turbo-code Decoder

- Decoding latency : 2178 per iteration
- Intrinsic auto-synchronisation (R=1/2), help for synchronisation (R≠1/2) or external synchronisation.
- Output giving an estimation of the channel quality

3. Channel Simulator

To test digital communication systems, it is necessary to use a noise generator to model the transmission channel. This kind of circuit is generally analogue. So, it is indispensable to have an ADC (Analogue to Digital Converter) to allow digital analysis of data downstream from the system. Moreover, it is often very complicated to fix and to reproduce the noise characteristics of the considered channel type precisely; in particular, the noise power measure requires very expensive devices (filters, voltmeters, ...) and many precautions.

An alternative to the noise characterisation complexity is the programmed generation of the noise, in a material form close to what is undertaken by software during simulations on a computer. The noise generator system, built with a digital algorithm, takes the place of the analogue characterisation bench of the channel [7].

Figure 3 presents a diagram of the system used for tests of *turbo4*. It is made up of :

- –An encoder and a decoder (consisting of 1 to 5 modules) which are on test. The decoder can be synchronised by its intrinsic auto-synchronisation or by the test system.
- –A digital transmission analyser which sends pseudo-random data to the encoder then compares these data to the data received from the decoder after 1, 2, 3, 4 or 5 iterations (modules). This comparison gives the BER (Bit Error Rate).
- –A puncturing block which makes adjustments to the coding rate possible.
- –A digital channel emulator which transforms all data (X, Y1 and Y2) into a noisy form quantified on 4 bits.
- –A PC (Personal Computer) which prepares samples of noisy information and transmits it to the memories of the digital channel emulator.

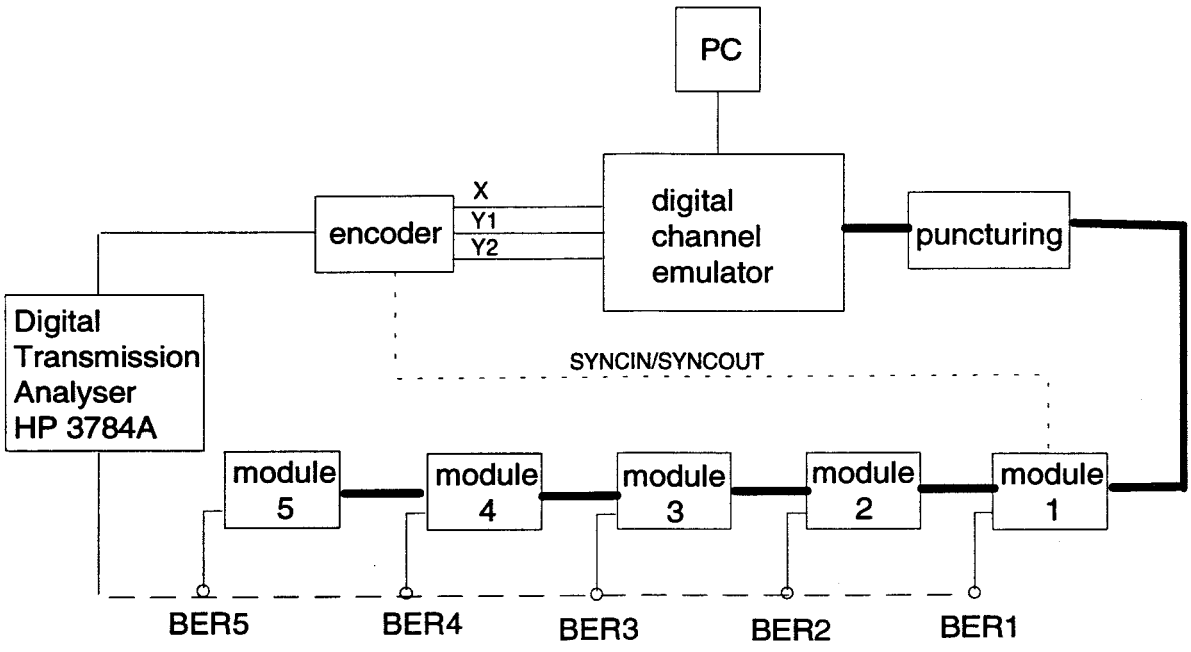

figure 3 : Functional diagram of the simulator

The transformation of data coming from the encoder into a noisy form could be directly made by the PC. But, in this case, the working speed of the test system would be dramatically low, so low BER could not be measured. The idea is to prepare files of noisy samples with the PC then to transmit them to the digital channel emulator. Before the test the channel emulator stores this information in a RAM. Then during the test it associates each piece of data to a noisy sample in a pseudo-random manner.

If the size of the RAM is big enough and if the quality of the random access is good, then the digital channel emulator simulates the desired channel.

The digital channel simulator used for the test of *turbo4* can simulate Gaussian and Rayleigh channels. It produces a noisy signal quantified on 4 bits. The noisy signal corresponds to an initial signal (coming from the encoder) transformed by the channel then multiplied by a coefficient Q. Q is needed to adapt the saturation quantization level to the level required by the decoder. Q is the mean absolute value of the signal, without noise, when saturation value equals 1.

The digital channel emulator used for the test of *turbo4* works with 3 RAM of 256 Kwords (one word = 4 bits). One RAM is used for the transformation of X; the others are used for the transformation of Y1 and Y2. Each RAM is addressed by a pseudo-random generator. $1+X^3+X^{28}$, $1+X^3+X^{25}$ and $1+X^3+X^{17}$ are the generator polynomials used.

4. Tests of the circuit

First tests were made in order to compare the performances of the circuit with simulation results. These tests are presented in figures 4 and 5.

Figure 4 shows the comparison between simulation results and measures for a Gaussian channel. Measures are made

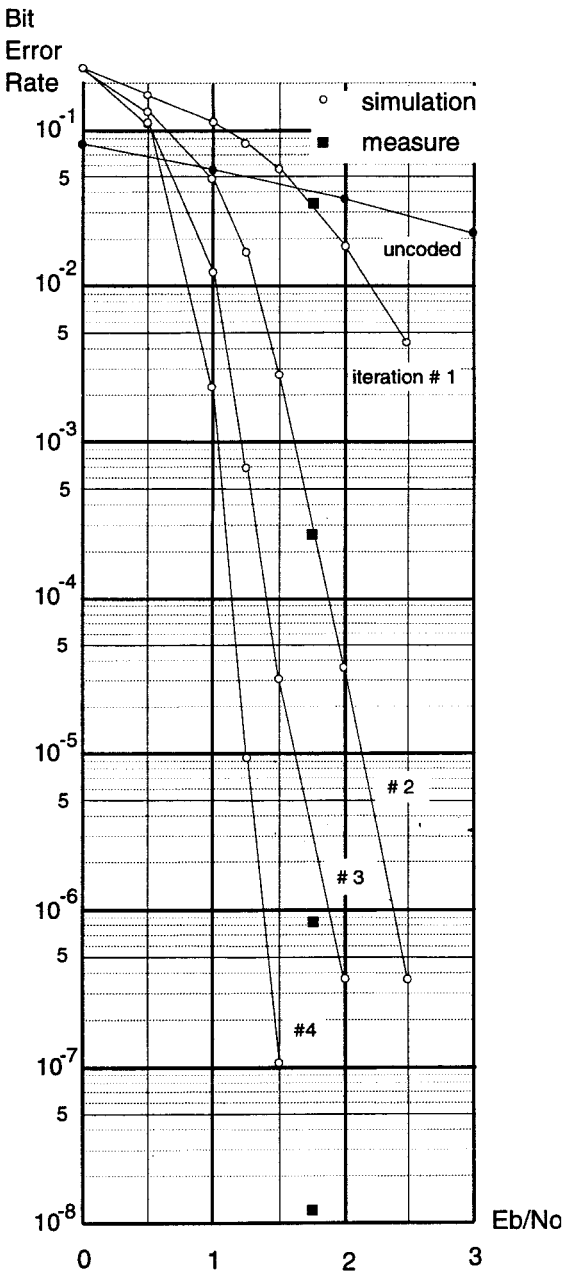

figure 4 : Typical BER curves, Gaussian channel (R=1/3)

with 1, 2, 3 and 4 modules for a global coding rate R = 1/3, each code working with a coding rate equal to half.

Figure 5 shows the comparison between simulation results and measures for a Gaussian channel. Measures are made with 3 modules (3 iterations) for BPSK or QPSK modulations. These curves are given for different coding rates. If R is the global coding rate, R_1 the coding rate associated with the first code and R_2 the

coding rate associated with the second code we have :

–R= 1/2 :R_1 = 4/7, R_2 = 4/5;
–R= 2/3 :R_1 = 4/5, R_2 = 4/5;
–R= 3/4 :R_1 = 6/7, R_2 = 6/7;
–R= 4/5 :R_1 = 8/9, R_2 = 8/9;

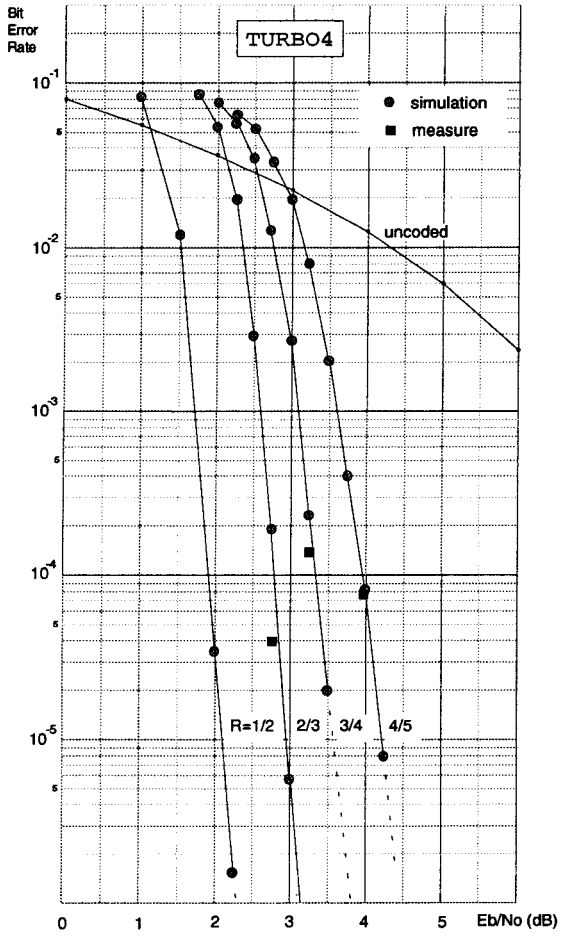

figure 5 : Typical BER curves,
Gaussian channel
(3 iterations)

Figures 4 and 5 show that measure results are very closed to the simulation results. Of course, for a given BER measures are faster than simulations (emulator works at 4 MHz). So it is easier to find accurate parameters. That is the reason why some measure results are better than simulation results.

Figure 6 shows measured results for a Gaussian channel. Measures are made with 1, 2, 3, 4 and 5 modules for a global coding rate R = 1/2 (R_1 = 4/7, R_2 = 4/5) and for a coefficient Q = 0.6.

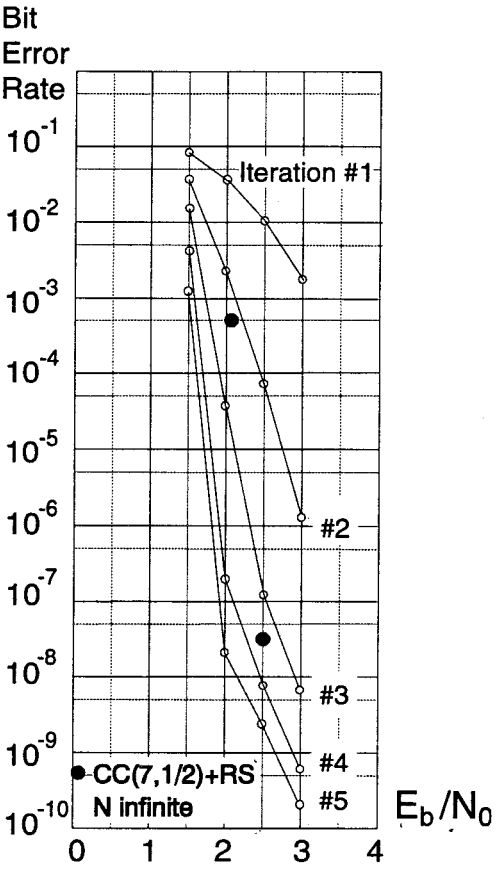

figure 6 : Measured BER curves,
Gaussian channel
(R=1/2, Q=0.6)

In addition, figure 6 presents a comparison between *turbo4* performances and a classical concatenation of a convolutional code K=7 and a 8-bit(255,223) Reed-Solomon with an infinite interleaver. This comparison shows that the coding gain given by 4 or 5 modules is better than that given by the classical concatenation for a BER higher than 10^{-8}. This comparison is made without taking into account the interleaver size which would decrease classical concatenation performances. Moreover the coding rate is more efficient for *turbo4* (R=1/2) than for the classical concatenation (R=0.437).

Figure 7 presents measured results for a Gaussian channel. Measures are made with 1, 2, 3, 4 and 5 modules for a global coding rate R = 2/3 (R_1 = 4/5, R_2 = 4/5) and for a coefficient Q = 0.7.

Figures 6 and 7 show a flattening degradation for low BERs. We think that this degradation is not due to turbo-codes

but to the internal accuracy of *turbo4*, which works on 4 bits.

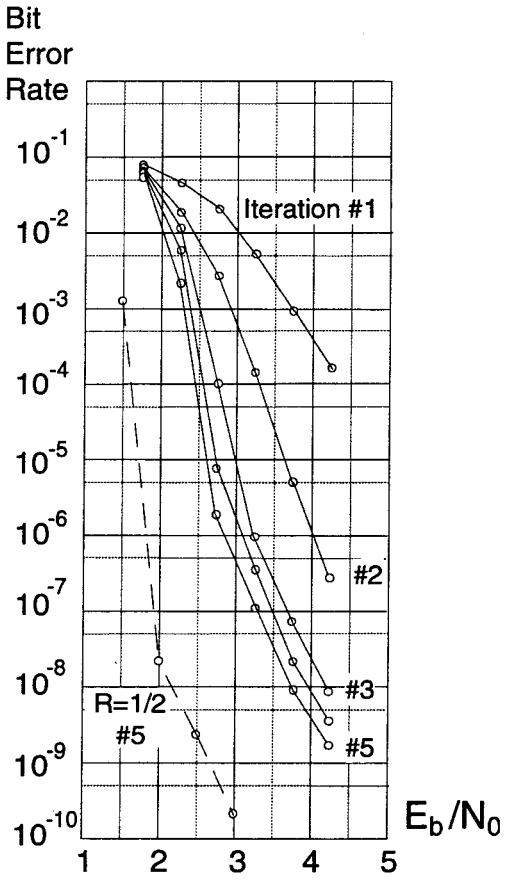

figure 7 : Measured BER curves, Gaussian channel (R=2/3, Q=0.7)

The SOVA2 works with 2 inputs X and Y. Y receives information Y2 multiplied by a coefficient β. Y2 is a Gaussian variable, and X can be also assimilated to a Gaussian variable, so β has to verify the relation: $\beta = \frac{\sigma_x^2}{\sigma_y^2}$ where σ_x^2 is the variance of X and σ_y^2 the variance of Y2. σ_y^2 is independent on the iteration however, due of the decoding process, σ_x^2 decreases with the iteration number. For the first iteration, σ_x^2 is nearly equal to σ_y^2 so β has to be equal to 1 roughly, but for the fourth iteration, with a coding rate equal to 1/2, the theoretical value of β would be around 0.1. It is impossible to program β around 0.1 because only 4 values of β can be programmed (1, 0.7, 0.55, 0.45). But even if β could be programmed to 0.1 the accuracy of the SOVA would not be sufficient. To avoid this kind of flattening in the future circuits, samples will be quantified on 5 bits

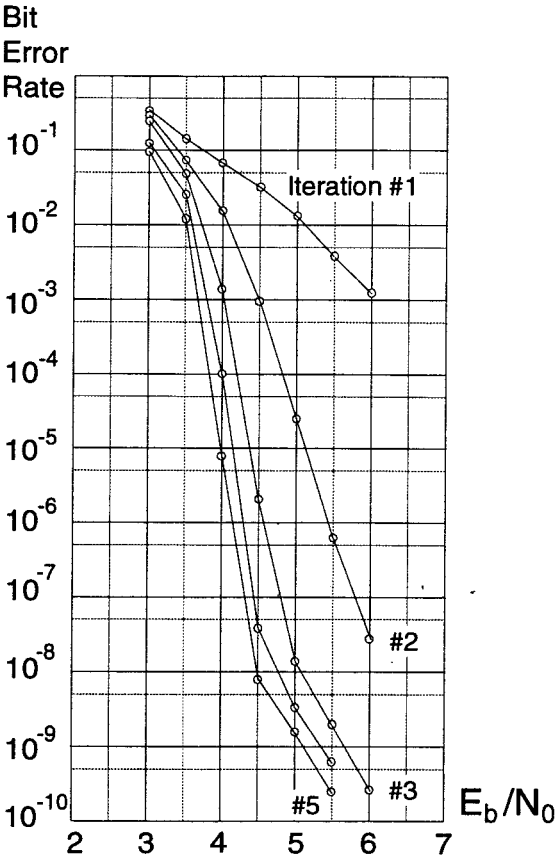

figure 8 : Measured BER curves Rayleigh channel, optimum interleaving and weighting (R=1/2, Q=1)

Figure 8 presents measured results for a Rayleigh channel with an optimum interleaving and weighting. Measures are made with 1, 2, 3, 4 and 5 modules for a global coding rate R = 1/2 (R_1 = 4/7, R_2 = 4/5) and for a coefficient Q=1. Results are very good, as the slope is the same as with the Gaussian channel with a gap of 2.5 dB.

5. Conclusion

Turbo4 tests show that the coding gain of turbo-codes is verified. At the moment *turbo4* is the circuit with the best coding gain, on both Gaussian and Rayleigh channels

Measures show a flattening degradation, we think that this degradation is not due to turbo-codes but to the limited set of β values and to a lack of internal accuracy of the circuit. Some measures are in process in order to prove this affirmation.

References

[1] C. Berrou, A. Glavieux and P. Thitimajshima, "Near Shannon limit error-correcting coding and decoding: turbo-codes", Proc. of IEEE ICC '93, Geneva, pp. 1064-1070, May 1993.

[2] C. Berrou and A. Glavieux, "Turbo-codes, general principles and applications", Proc. of the 6th Int. Tirrenia Workshop on Digital Communications, Pisa, Italy, pp. 215-226, September 1993.

[3] C. Berrou and A. Glavieux, "Near optimum error correcting coding and decoding : turbo-codes", to appear in IEEE Transactions on communications.

[4] C. Berrou and G. Lochon, "CAS5093 : Turbo encoder/decoder", Data sheet, COMATLAS, Chateaubourg, France, November 1993.

[5] C. Berrou, P. Adde, E. Angui and S. Faudeil, "A low complexity soft-output Viterbi decoder architecture", Proc. of IEEE ICC '93, Geneva, pp. 737-740, May 1993.

[6] J. Hagenauer and P. Hoeher, "A Viterbi algorithm with soft-decision outputs and its applications", Proc. IEEE Globecom'89,Dallas, Texas, Nov. 1989, p.47.1.1-47.1.7.

[7] D. Degrugillier, M. Jézéquel, G. Graton, "ASIC for gaussian channel simulation", EDAC-EUROASIC'93, Paris, pp. 256-259, February 1993.