The Turbo Code Standard for DVB-RCS

C. Douillard[*], M. Jézéquel[*], C. Berrou[*],
N. Brengarth[**], J. Tousch[**] and N. Pham[***]
[*]ENST Bretagne, BP 832, 29285 Brest Cedex, France
Catherine.Douillard@enst-bretagne.fr
[**]TurboConcept, 1 av. du technopôle, 29280 Plouzané, France
Nathalie.Brengarth@enst-bretagne.fr
[***]Eutelsat, 70 rue Balard, 75015 Paris, France
npham@eutelsat.fr

Abstract*: Convolutional turbo codes are very flexible codes, easily adaptable to a large range of data block sizes and coding rates. This is the main reason for their being adopted in the DVB standard for Return Channel via Satellite (DVB-RCS). The paper presents the turbo coding/decoding scheme specified in this standard, for twelve block sizes and seven coding rates. Simulation results show the performance of the coding scheme chosen, in particular for the transmission of ATM cells and MPEG transport stream packets. The company TurboConcept has developed a hardware prototype for the double purpose of very low error rate performance measurement and demonstrates the hardware feasibility of this new turbo code. Some hardware measurement results are also discussed.*
Keywords: turbo codes, circular RSC codes, double-binary codes, two-level interleaver.

1. INTRODUCTION : DVB-RCS

1.1. System considerations

In a race involving ADSL and Cable modem to provide "broadband access", the DVB Committee recently approved a standard - known as DVB-RCS, for Return Channel via Satellite, or EN 301 790 in ETSI - to provide two-way, full-IP, asymmetric communications via satellite. One advantage that satellite offers is that the service can be deployed quickly, all over a large area, once a single "hub" infrastructure is in place. Another advantage is that service quality and the cost per susbscriber is independent of the distance between the terminal and the access point. This places satellite in a favourable position and especially as a necessary complement in countries where ADSL and Cable modem cannot economically cover more than 75% of the population.

This standard specifies an air interface allowing a large number of small terminals to send "return" signals to a central gateway, also called a "hub" and at the same time receive IP data from that hub on the "forward" link in the usual DVB/MPEG2 broadcast format, thus leveraging the ubiquity of both DVB and IP technologies, while avoiding a return connection via terrestrial means (such as a dial-up telephone line).

The return channel speed (terminals-to-hub) can range from 144 Kbps to 2 Mbps and the satellite resource on this return link is shared among the terminals transmitting small packets and using MF-TDMA (Multi-Frequency TDMA)/DAMA (Demand-Assigned Multiple Access) techniques. Since this access is packet-based and on-demand, once the session is established with the hub, the link is permanently open, thus providing an "always-on" IP connection, with efficient use of satellite resources (provided the duty-cycle of the terminals is low on the return link).

The new standard includes such advanced features as a software-radio system (the terminal acquires the characteristics of the transmission parameters it has to use from signals broadcast by the hub), network synchronisation in the digital domain (all terminals are synchronised to the same precise clock transmitted digitally by the hub), sophisticated bandwidth-on-demand protocols which can mix constant-rate, dynamic-rate, volume-based and best-effort bandwidth allocation and advanced forward-error-correction turbo coding (as will be used in the new On-Board satellite processing system, Skyplex [1]). This paper presents the rationale behind the selection of this FEC and the performance achieved on a hardware implementation of the decoder.

1.2. Coding requirements

Since DVB-RCS applications involve the transmission of data using various block sizes and coding rates, the coding scheme has to be very flexible, with better performance than the classical concatenation of a convolutional code and a Reed-Solomon code. Finally, it has to be able to process data so as to allow the transmission of data bit rates up to 2Mbps, as indicated in the introduction.

Convolutional turbo codes seemed to be a good candidate :

- A simple puncturing device is sufficient to adapt coding rate,
- The use of double-binary circular recursive systematic convolutional (CRSC) component codes makes turbo codes very efficient for block coding [2][3],

– The different permutations (interleavers) may be achieved using generic equations with only a restricted number of parameters.
– The same decoding hardware can be used to manage every block size / coding rate combination.

2. DETAILED FEATURES OF THE CODING SCHEME PROPOSED

Small component codes have been chosen for two main reasons. On the one hand, efficient turbo coding requires component codes with small minimum distances (i.e. with small constraint lengths for convolutional codes) in order to ensure convergence at very low signal to noise ratios and to minimize the correlation effects [3]. On the other hand, the material complexity of the decoder grows exponentially with code memory, so a hardware implementation on a single integrated circuit is only conceivable for reasonable constraint lengths. The solution chosen uses memory $\nu = 3$ component codes. This complexity/performance compromise allows the implementation of the decoder on a single FPGA (Field Programmable Gate Array) circuit. The same code size was also chosen for the UMTS standard.

Furthermore, the code proposed calls for two recent techniques in turbo coding:

– Parallel concatenation of circular recursive systematic convolutional (CRSC) codes [4] makes convolutional turbo codes efficient for block coding,
– Double-binary elementary codes provide better error-correcting performance than binary codes for equivalent implementation complexity [2].

2.1. Circular recursive systematic (CRSC) convolutional codes

Adopting circular coding avoids the degradation of the spectral efficiency of the transmission when forcing the value of the encoder state at the end of the encoding stage by the addition of tail bits.

Circular coding is an adaptation of the so – called "tail-biting" technique to recursive convolutional codes. It ensures that, at the end of the encoding operation, the encoder retrieves the initial state, so that data encoding may be represented by a circular trellis. The existence of such a state, called *circulation state* $\mathbf{S}_c$, is ensured when the size of the encoded data block, *N*, is not a multiple of the period of the encoding recursive generator. The value of the circulation state depends on the contents of the sequence to encode and determining $\mathbf{S}_c$ requires a pre-encoding operation: first, the encoder is initialised in the "all zero" state. The data sequence is encoded once, leading to a final state $\mathbf{S}_N^0$. $\mathbf{S}_c$ value is then calculated from expression $\mathbf{S}_c = \left\langle \mathbf{I} + \mathbf{G}^N \right\rangle^{-1} \mathbf{S}_N^0$. In practice, the relation between $\mathbf{S_c}$ and $\mathbf{S}_N^0$ is provided by a small combinational operator with $\nu = 3$ input and output bits. Finally, to perform a complete encoding operation of the data sequence, two circulation states have to be determined, one for each component encoder, and the sequence has to be encoded four times instead of twice. This is not a real problem, as the encoding operation can be performed at a frequency much higher than the data rate.

From the decoder point of view, a simplified version of the APP algorithm [5] was implemented. Decoding a sequence then consists of going around the circular trellis anticlockwise for the backward process and clockwise for the forward process, during which the data are decoded and extrinsic information is generated. For both processes, probabilities computed at the end of a turn are used as initial values for the next turn. The number of turns performed around the circular trellises is equal to the number of iterations required by the iterative process. In practice, the iterative process is preceded by a "prologue" decoding step, performed on a part of the circle for a few ν. This is intended to guide the process towards an initial state which is a good estimation of the circulation state.

2.2. Double-binary codes

Using double-binary codes as component codes represents a simple means to reduce the correlation effects, as explained in [2]. The substitution of binary codes by double-binary codes has a direct incidence on the erroneous paths in the trellises, which leads to a lowered path error density and reduces correlation effects in the decoding process. This leads to better performance, even in comparison with 16-state binary turbo codes.

Furthermore, using double-binary codes leads to extra advantages:

– The influence of puncturing is less significant than with binary codes (the natural rate of the double-binary turbo code being 1/2 instead of 1/3 for a binary turbo code).
– It has also been observed that the performance degradation due to the application of a simplified version of the APP algorithm is less significant in the case of double-binary codes (less than 0.1 dB) than in the case of binary codes (0.3 to 0.4 dB).
– From a material implementation point of view, the bit rate at the decoder output is twice that of a binary decoder processing the same number of iterations, with the same circuit clock frequency and with an equivalent complexity per decoded bit (the material complexity of the decoder, for a given memory length, is about twice the complexity of a binary decoder, but two bits are decoded at the same time). Moreover, given the data block size, the latency of the decoder is divided by 2 compared with the binary case because the size of the permutation matrix is halved.

– The use of double-binary codes also makes the introduction of two-level permutations possible, as explained in section 2.3.

2.3. Interleaving with local disorder

The performance of parallel concatenation of convolutional codes (PCCC) at low error rates, is essentially governed by the permutation that links the component codes. The simplest way to achieve interleaving in a block is to adopt uniform or regular interleaving: data are written linewise and read columnwise in a rectangular matrix. This kind of permutation behaves very well towards error patterns with weight 2 or 3, but is very sensitive to square or rectangular error patterns, as explained in [6]. Classically, in order to increase distances given by rectangular error patterns, non-uniformity is introduced in the permutation relations. Many proposals have been made in this direction, especially for the UMTS application. The CCSDS turbo code standard may also be cited as an example of non-uniform permutation. However the disorder that is introduced with non-uniformity can affect the scattering properties concerning weight 2 or 3 error patterns.

With double-binary codes, non-uniformity can be introduced without any repercussion on the good scattering properties of regular interleaving [3]. The principle involves introducing local disorder into the data couples, for example (A,B) becoming (B,A) – or $(B,A+B)$, or others – periodically before the second encoding. This helps to avoid many error patterns that would appear without applying it. Therefore, this appears to be a significant gain in the search for large minimum distances.

2.4. Summary

This section sums up the technical features of the turbo code adopted for DVB-RCS standard.

Encoder structure: the encoder consists of a parallel concatenation of two identical component codes with generators (in octal notation) 15 (recursion), 13 (redundancy Y_1), 11 (redundancy Y_2) as depicted in figure 1. Redundancy Y_2 is only used for coding rates less than 1/2, i.e. 1/3 and 2/5.

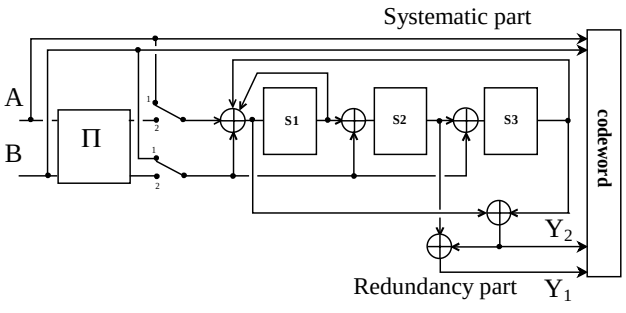

Figure 1: DVB-RCS encoder

Block sizes: 12, 16, 53, 55, 57, 106, 108, 110, 188, 212, 214 and 216 bytes.

Coding rates: 1/3, 2/5, 1/2, 2/3, 3/4, 4/5, and 6/7.

Permutation equations $i = \Pi(j)$:

Let N be the number of data couples in each block at the encoder input (each block contains $2N$ data bits)

For $j = 0, \cdots, N-1$

- 1st level: Inversion of A_j and B_j in the data couple, if $j \bmod 2 = 0$,
- 2nd level: $i = (P_0 \times j + P) \bmod N$, with

$P = 0$ if $j \bmod 4 = 0$

$P = N/2 + P_1$ if $j \bmod 4 = 1$

$P = P_2$ if $j \bmod 4 = 2$

$P = N/2 + P_3$ if $j \bmod 4 = 3$

where the values of parameters P_0, P_1, P_2, and P_3 depend on N (table 1).

Block size (bytes)	P_0	P_1	P_2	P_3
12	11	24	0	24
16	7	34	32	2
53	13	106	108	2
55	23	112	4	116
57	17	116	72	188
106	11	6	8	2
108	13	0	4	8
110	13	10	4	2
188	19	376	224	600
212	19	2	16	6
214	19	428	224	652
216	19	2	16	6

Table 1: Values of parameters P_0, P_1, P_2, and P_3.

2.5. Simulation results

Table 2 gives some examples of the DVB-RCS turbo code performance observed over a Gaussian channel at Frame Error Rate (FER) $= 10^{-4}$, compared to the theoretical limits [7]. The component decoding algorithm applied is the Max-Log-APP, with samples quantized on 4 bits. Good convergence, close to the theoretical limits – from 1.0 dB to 1.5 dB, depending on the coding rate – can be observed, thanks to the double-binary component code.

Block size / Coding rate	ATM 53 bytes	MPEG 188 bytes
1/2	2.3 (1.3)	1.8 (0.8)
2/3	3.3 (2.2)	2.6 (1.7)
3/4	3.9 (2.6)	3.2 (2.1)
4/5	4.6 (3.1)	3.8 (2.6)
6/7	5.2 (3.8)	4.4 (3.3)

Table 2: E_b / N_0 (dB) @ FER $= 10^{-4}$, 8 iterations, 4-bit input quantization, simulation over Gaussian channel, (theoretical limits in brackets).

3. IMPLEMENTATION

3.1. Description

TurboConcept has developed a turbo decoder that has been used to measure low error rate performances and also to demonstrate to system builders that these efficient turbo codes are affordable with reasonable complexity.

The main features of this implementation (TurboConcept TC1000 product [8]) are :

- Single-chip FPGA solution, e.g. Altera APEX 200 or Xilinx Virtex 400. This leads to fast and easy system integration.
- User bit rate up to 4 Mbit/s with 6 iterations.
- Every block size / coding rate combination is supported by the same hardware.

Additionally, the implementation takes advantage of the high flexibility of this code to allow the user to configure "on the fly" the block size and coding rate, with no guard time needed for interleaver pre-calculation, for example.

A Max-log-APP algorithm is implemented, with input quantization on 4 bits, which is a good trade-off between complexity and performance.

To reduce the decoding latency, two buffers are used at the input enabling bank-swapping. This is possible thanks to the high density of embedded memory blocks in recent FPGA device families.

3.2. Measurement results

The following figures exhibit TurboConcept's hardware turbo decoder FER performances, for block length of 53 and 188 bytes. We can observe that:

- Very close matching between simulation and hardware is obtained.
- FER down to 10^{-8} (equivalent to BER=10^{-10} / 10^{-11}) measurements show the absence of error floor.
- Regular behaviour regarding puncturing patterns (between different coding rates).

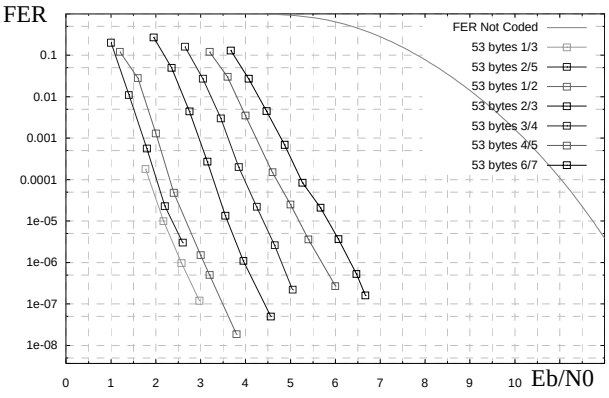

Figure 2: Frame Error Rate for ATM blocks, 8 iterations (TC1000 hardware measurements)

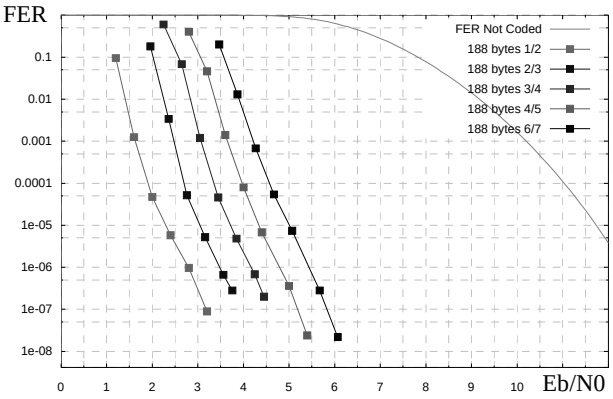

Figure 3: Frame Error Rate for MPEG blocks, 8 iterations (TC1000 hardware measurements)

4. CONCLUSION

The turbo code that was proposed for DVB-RCS applications is powerful, very flexible and can be implemented with reasonable complexity.

This code could also be easily adjusted to many other applications, for various configurations of block sizes and code rates while retaining excellent coding gains.

REFERENCES

[1] N. Brengarth, R. Novello, N. Pham, V. Piloni and J. Tousch, "DVB-RCS turbo code on a commercial OPB statellite payload: Skyplex", *Proc. of the 2nd Int'l Symp. on Turbo Codes"*, Brest, France, Sept. 2000.

[2] C. Berrou and M. Jézéquel, "Non binary convolutional codes for turbo coding", *Electronics Letters*, Vol. 35, N°1, pp. 39-40, Jan. 1999.

[3] C. Berrou, C. Douillard, and M. Jézéquel, "Designing turbo codes for low error rates", *Digest of IEE Colloq. on "Turbo codes in digital brodcasting – could it double capacity?"*, Vol. 165, Nov. 1999.

[4] C. Berrou, C. Douillard, and M. Jézéquel, "Multiple parallel concatenation of circular recursive convolutional (CRSC) codes", *Annals of Telecommunications*, Vol. 54, N°3-4, pp. 166-172, March-April 1999.

[5] P. Robertson, P. Hoeher and E. Villebrun, "Optimal and suboptimal maximum a posteriori algorithms suitable for turbo decoding", *ETT*, Vol. 8, pp. 119-125, March-Apr. 1997.

[6] C. Berrou and A. Glavieux, "Near optimum error correcting coding and decoding: turbo codes", *IEEE Trans. Com.*, Vol. 44, N°10, pp. 1261-1271, Oct. 1996.

[7] S. Dolinar, D. Divsalar and F. Pollara, "Code performance as a function of block size", TMO progress report 42-133, JPL, NASA.

[8] "TC1000 DVB-RCS turbo decoder data sheet", TurboConcept (http://www.turboconcept.com)