A Low Rate Turbo Product Codec Implemented via Combinational Logic Circuitry

Ivan Simões Gaspar
National Institute of Telecommunications - Inatel
P.O. Box 05 - 37540-000
Santa Rita do Sapucaí - MG - Brazil
ivans@mtel.inatel.br

Dayan Adionel Guimarães
National Institute of Telecommunications - Inatel
P.O. Box 05 - 37540-000
Santa Rita do Sapucaí - MG - Brazil
dayan@inatel.br

***Abstract*—This paper presents results concerning the implementation of the two-dimensional product code $(8,4,4)^2$ and its turbo decoding, using a combinational circuitry in a totally parallelized structure. This allows for a partial iteration to be completed in only one clock cycle. The project was built in a low cost FPGA, the Altera EP1C6T144, a device which contains only about 6,000 logical elements.**

***Index Terms*—Combinational logic circuitry, low-rate turbo product codec.**

I. Introduction

New approaches for mobile wireless applications will require greater data rates at lower channel SNR than ever before. Multi-carrier (MC) systems, specially those combined with the code-division multiple access (CDMA) technique, have being considered as an adequate solution against multi-path fading, combining time and frequency diversity, simple one-tap equalization and flexible spectrum shaping. To improve reliable data transmission for these applications, more advanced error correcting techniques are required. In this context, in [1] was proposed a coding/decoding scheme for the orthogonal MC-CDMA system suggested in [2]. This was achieved by substituting the original repetition code by a low-rate multidimensional product code with iterative (turbo) decoding. Results reported in [1] demonstrate good code gains, without penalties in the original data rate and bandwidth.

From the implementation point of view, to ensure a maximum throughput in the turbo decoding process, a high parallel combinatorial approach is needed. In the next sections it is described such approach. It is able to offer a good trade-off between performance and time propagation in the FPGA circuitry.

II. General Codec Description

The class of multidimensional codes proposed in [1] is formed by using the same nonsystematic $(n, k, d_{min}) = (n, n/2, 4)$ component code in each dimension, leading to a product code $(n, k, d_{min})^D$ with codeword length n^D, rate $(1/2)^D$ and minimum distance 4^D, where D is the code dimension. The component code (8,4,4) is formed by mapping a single-parity check code in accordance to the set-partitioning rule defined by a repetition code, according to Figure 1. In this figure, the sessions are related to the trellis sessions in Figure 2.

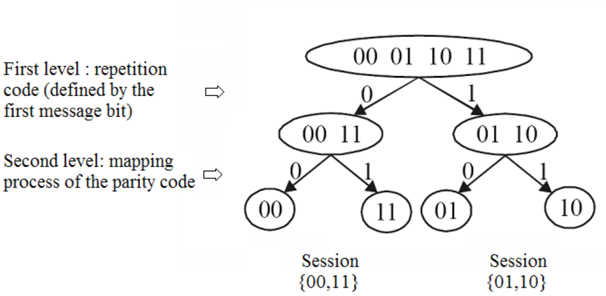

First level : repetition code (defined by the first message bit)
Second level: mapping process of the parity code
00 01 10 11
0
1
00 11
01 10
0
1
0
1
00
11
01
10
Session {00,11}
Session {01,10}

Fig. 1. Partition set used to compose the code (*n*,*n*/2,4).

To decode the component code, a very simple minimum-distance (MD) decoding algorithm is used and applies the Wagner decoding rule [4] twice over the trellis diagram shown in Figure 2. The decoding complexity for the component code is similar to that of the single-parity check multidimensional product codes described in [5].

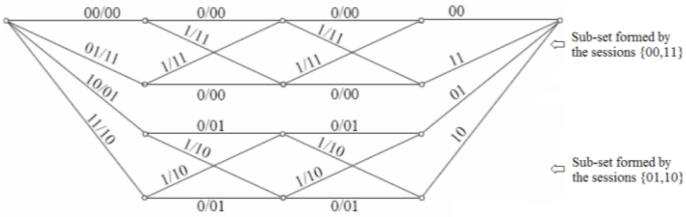


Fig. 2. Trellis for the code (*n*,*n*/2,4) [1].

The iterative decoding algorithm uses a modified form of the Pyndiah's SISO (Soft-Input, Soft-Output) decoding algorithm [6], and conserves the three main steps of the algorithm for single-parity check product codes: initialization, decoding of each dimension, and repetition. The original Chase algorithm used in [6] is substituted here and in [1] by the Wagner decoding rule. The output of the Wagner algorithm is a unique decision, so no list of concurrent code-words is expected. This is why, in this approach, the soft-output is generated by the influence of the simple semi-empiric

β weigh factor, proposed by Pyndiah to be used when concurrent code-words could not be found by the Chase algorithm [6]. The repetition phase consists of repeating decoding iterations as long as required. Figure 3 shows a block diagram representing operations for the *j*-th decoding step, where the maximum value of *j* is the total number of iterations multiplied by *D*. The vector **R** represents all n^D received noisy symbols. The expression “decoding in one dimension” in this figure means decoding n^{D-2}, $n \times n$ two-dimensional arrays in the “direction” of one of the dimensions. Decoding an array consists of decoding *n* rows (or *n* columns). Hence, “decoding in one dimension” implies applying the SISO decoding algorithm n^{D-1} times or, as adopted in this paper, using n^{D-1} parallel structures [7].

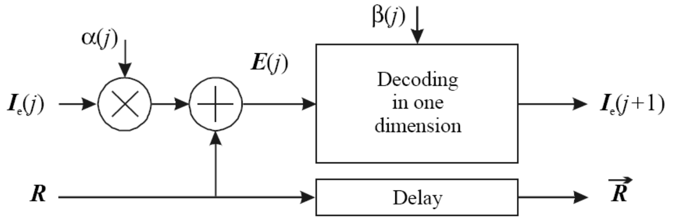
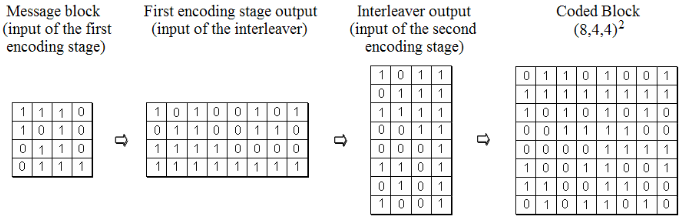


Fig. 3. Turbo decoding process proposed by Pyndiah [6].

III. Codec Implementation

The first part of the circuit, that is, the two-dimensional encoder, was implemented in a sequential VHDL process which requires less than 1% of the total available FPGA logic. The two-dimensional $(8,4,4)^2$ encoding process was achieved by interconnecting two component codes (8,4,4) through an interleaver process constructed with the internal FPGA RAM. This process is summarized in Figure 4.



Message block (input of the first encoding stage)
First encoding stage output (input of the interleaver)
Interleaver output (input of the second encoding stage)
Coded Block $(8,4,4)^2$
⇨
⇨
⇨

1	1	1	0
1	0	1	0
0	1	1	0
0	1	1	1

1	0	1	0	0	1	0	1
0	1	1	0	0	1	1	0
1	1	1	1	0	0	0	0
1	1	1	1	1	1	1	1

1	0	1	1
0	1	1	1
1	1	1	1
0	0	1	1
0	0	0	1
1	1	0	1
0	1	0	1
1	0	0	1

0	1	1	0	1	0	0	1
1	1	1	1	1	1	1	1
1	0	1	0	1	0	1	0
0	0	1	1	1	1	0	0
0	0	0	0	1	1	1	1
1	0	0	1	1	0	0	1
1	1	0	0	1	1	0	0
0	1	0	1	1	0	1	0

Fig. 4. $(8,4,4)^2$ encoding process.

After adding channel noise, the matrix containing the product code received symbols, quantized in 5 bits, is applied to an iterative decoding circuit composed by eight SISO decoders operating in parallel. A Wagner decoder and an apparatus for generating soft-outputs compose each SISO decoder. In the combinational Wagner decoder circuitry, the component codes are decoded simultaneously according to each possible set-partition. The final decision is made based on the accumulated metrics.

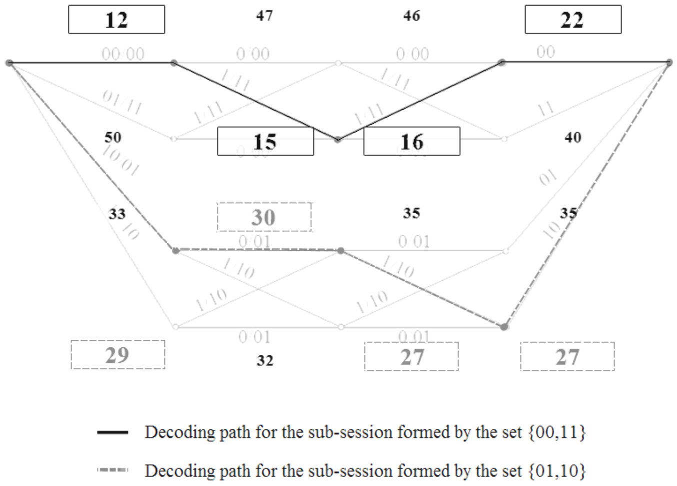
The key feature of this parallel decoding structure is that it can be constructed by grouping a few numbers of basic components that use small peaces of the FPGAs combinational logic. Furthermore, it does not require combinational logic loops. The logic requirements in the Wagner decoding circuit are guaranteed by using a modified metric calculation: instead of using classic quadratic Euclidian distance between the received signal and an expected value, the metrics are calculated in comparison to a maximum expected value. Our approach eliminates the need of estimating signal averages and quadratic terms, and was successfully tested in practice [7]. Bit error rate (BER) results of decoding the component codes were the same as the one using a Maximum Likelihood method. A total of 285 logical elements were employed in the Wagner algorithm constructed with combinational circuit, and the propagation delay was less then 19 ns. To better clarify this idea, the Figure 5 presents a hypothetical code vector **r** being decoded in the trellis structure showed in Figure 2. Matrix **Δ** contains all values operated in order to obtain the metrics for the sessions {00,11} and {01,10}. The maximum expected values for the quantized signal is +15 and –16.

$$\mathbf{r} = \begin{bmatrix} -11 & -9 & 7 & 8 & 9 & 5 & -3 & 7 \end{bmatrix}$$

$$\Delta = \begin{bmatrix} -(-16-r_0)-(-16-r_1) & -(-16-r_2)-(-16-r_3) & -(-16-r_4)-(-16-r_5) & -(-16-r_6)-(-16-r_7) \\ (15-r_0)+(15-r_1) & (15-r_2)+(15-r_3) & (15-r_4)+(15-r_5) & (15-r_6)+(15-r_7) \\ -(-16-r_0)+(15-r_1) & -(-16-r_2)+(15-r_3) & -(-16-r_4)+(15-r_5) & -(-16-r_6)+(15-r_7) \\ (15-r_0)-(-16-r_1) & (15-r_2)-(-16-r_3) & (15-r_4)-(-16-r_5) & (15-r_6)-(-16-r_7) \end{bmatrix}$$

$$\Delta = \begin{bmatrix} -(-16-(-11))+(-16-(-9)) & -(-16-7)+(-16-8) & -(-16-9)+(-16-5) & -(-16-(-3))+(-16-7) \\ (15-(-11))+(15-(-9)) & (15-7)+(15-8) & (15-9)+(15-5) & (15-(-3))+(15-7) \\ -(-16-(-11))+(15-(-9)) & -(-16-7)+(15-8) & -(-16-9)+(15-5) & -(-16-(-3))+(15-7) \\ (15-(-11))-(-16-(-9)) & (15-7)-(-16-8) & (15-9)-(-16-5) & (15-(-3))-(-16-7) \end{bmatrix}$$

$$\Delta = \begin{bmatrix} 12 & 47 & 46 & 22 \\ 50 & 15 & 16 & 40 \\ 29 & 30 & 35 & 35 \\ 33 & 32 & 27 & 27 \end{bmatrix}$$



12
47
46
22
00/00
0/00
0/00
00
01/11
1/11
1/11
1/11
1/11
11
50
15
16
40
10/01
01
33
30
35
35
1/10
0/01
0/01
10
1/10
1/10
1/10
1/10
0/01
0/01
29
32
27
27
Decoding path for the sub-session formed by the set {00,11}
Decoding path for the sub-session formed by the set {01,10}

Fig. 5. Hypothetical received code vector **r** and its double decoding process in the trellis structure of the code (8,4,4).

The code-words $\hat{\mathbf{c}}$ e $\hat{\mathbf{c}}$’ and their respective accumulated metrics are also shown in Figure 5. The final decision are.

$$\hat{\mathbf{c}} = \begin{pmatrix} 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 \end{pmatrix}$$
$$\sum \Delta = 12 + 15 + 16 + 22 = 65$$

and

$$\hat{\mathbf{c}}' = \begin{pmatrix} 0 & 1 & 0 & 1 & 1 & 0 & 1 & 0 \end{pmatrix}$$
$$\sum \Delta' = 29 + 30 + 27 + 27 = 113$$

The original Pyndiah's turbo-decoding algorithm was modified to a simplified version, where soft-outputs are calculated from the Wagner hard-outputs by multiplying them by β, quantized in 4 bits. A parameter α, as in [6], is used here to scale the extrinsic information. However, instead of describing a logarithmic variation, as proposed in [1], it is now kept constant and equal to 0.25. With these considerations at hand, it was possible to complete a partial iteration in only one clock cycle. A complete decoding has used only 60% of the available FPGA logic resources, and experimented a propagation delay in the combinational circuit less than 30 ns.

Figure 6 shows the performance result expected for the $(8,4,4)^2$ turbo product code. This result was obtained by software simulation. Figure 7 shows performance results measured with the turbo decoder implemented in FPGA. A very close agreement can be observed between the simulation and the implementation results.

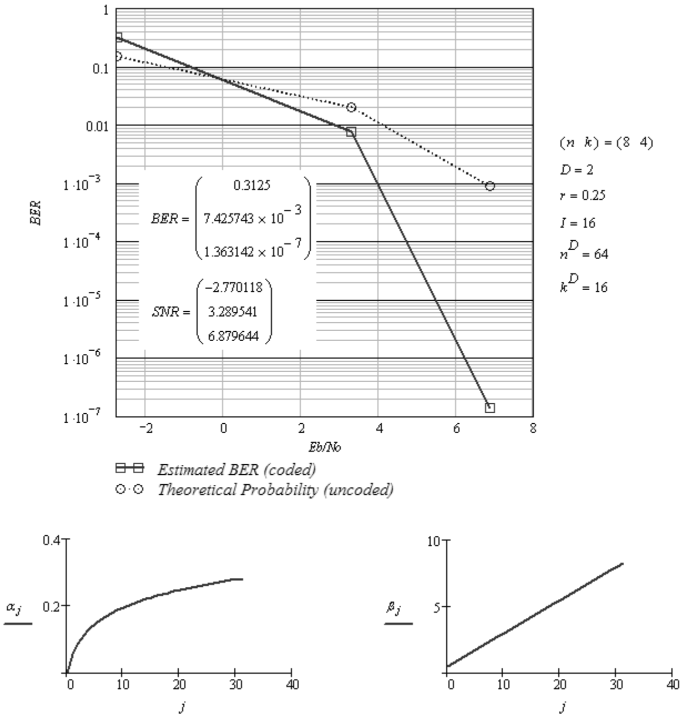
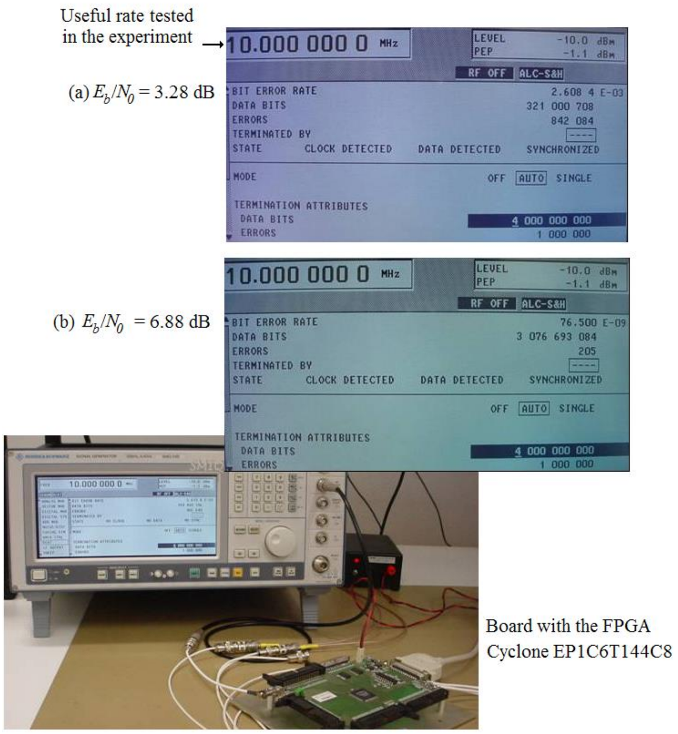


Fig. 6. Simulated $(8,4,4)^2$ turbo product code performance (16 interactions) on AWGN channel. The parameters α e β are show as they were originally proposed in [1].


Useful rate tested in the experiment
(a) E_b/N_0 = 3.28 dB
(b) E_b/N_0 = 6.88 dB
Board with the FPGA Cyclone EP1C6T144C8

```
10.000 000 0 MHz          LEVEL   -10.0 dBm
                          PEP      -1.1 dBm
                   RF OFF  ALC-S&H
BIT ERROR RATE                          2.608 4 E-03
DATA BITS                             321 000 708
ERRORS                                    842 084
TERMINATED BY                                ----
STATE      CLOCK DETECTED   DATA DETECTED   SYNCHRONIZED
MODE                           OFF  AUTO  SINGLE
TERMINATION ATTRIBUTES
 DATA BITS                          4 000 000 000
 ERRORS                                1 000 000
```

```
10.000 000 0 MHz          LEVEL   -10.0 dBm
                          PEP      -1.1 dBm
                   RF OFF  ALC-S&H
BIT ERROR RATE                          76.500 E-09
DATA BITS                           3 076 693 084
ERRORS                                        205
TERMINATED BY                                ----
STATE      CLOCK DETECTED   DATA DETECTED   SYNCHRONIZED
MODE                           OFF  AUTO  SINGLE
TERMINATION ATTRIBUTES
 DATA BITS                          4 000 000 000
 ERRORS                                1 000 000
```

Fig. 7. Measured $(8,4,4)^2$ turbo product code performance (16 interactions) on AWGN channel. The parameter α was kept constant and equal to 0.25; the parameter β was quantized in 4 bits.

IV. Final Comments

With 16 iterations, code gains of about 4.5 dB were achieved. Considering that the product code has only 64 bits in length, this is a quite good result. With four iterations, rates up to 60 Mbps can be achieved using the lesser speed grade available for this FPGA family. By using more powerful FPGA devices, rates up to 200 Mbps can be easily obtained.

References

[1] D. A. Guimarães, *Uma Classe de Códigos Produto e sua Decodificação Turbo Aplicada em um Sistema CDFMA Multiportadora*, Ph.D. Thesis, State University of Campinas – Unicamp, SP, Brazil, 2003 (in Portuguese).

[2] E. Sourour and M. Nakagawa, *Performance of Orthogonal Multicarrier CDMA in a Multipath Fading Channel*, IEEE Transactions on Communications, vol. 44, no. 3, pp. 356-367, Mar. 1996.

[3] S. Kaiser, *Multi-Carrier CDMA Mobile Radio Systems - Analysis and Optimization of Detection, Decoding and Channel Estimation*, Ph.D. Thesis: VDI Verlag GmbH. Düsseldorf, 1998.

[4] R. A. Silverman and M. Balser, *Coding for Constant Data-Rate Systems*, IRE Trans. Inform. Theory, PGIT-4, pp. 50-63, 1954.

[5] D. M. Rankin and T. A. Gulliver, *Single Parity Check Product Codes*, IEEE Trans. Commun., vol. 49, no. 8, pp. 1354-1362, Aug. 1998.

[6] R. M. Pyndiah, *Near-Optimum Decoding of Product Codes: Block Turbo Codes*, IEEE Trans.Commun., vol. 46, no. 8, pp. 1003-1010, Aug. 1998.

[7] I. S. Gaspar, *Implementação de uma classe de códigos produto com decodificação turbo em FPGA*, Master's Dissertation, National Institute of Telecommunications – Inatel, MG, Brazil, 2006 (in Portuguese).