A Low Rate Turbo Product Codec Implemented via Combinational Logic Circuitry♦

Ivan Simões Gaspar & Dayan Adionel Guimarães

***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 that contains only about 6,000 logical elements.**

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

I. Introduction

New approaches for mobile wireless applications will require greater data rates at lower signal-to-noise ratios (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] it was proposed a coding/decoding scheme for the orthogonal MC-CDMA system suggested in [2]. This scheme was implemented by substituting the original repetition code in [2] 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, high parallelized combinatorial approaches are desired. In the next sections it is proposed and described one of these approaches. It is able to offer a good trade-off between performance, complexity and time propagation in the FPGA circuitry.

♦ A shorter version of this paper was presented during the 2nd International Workshop on Telecommunications, IWT'07, Sta. Rita do Sapucaí, MG, Frebuary, 12-15 2007, and published in its proceedings. That paper was classified as one of the best papers presented during IWT'07, and an extended version of it was invited to be published here.

I. S. Gaspar (igaspar@linear.com.br) is with *Linear Equipamentos Eletrônicos S.A* (Praça Linear, 100 - Santa Rita do Sapucaí - MG - Brasil - 37540-000). D. A. Guimarães (dayan@inatel.br) is with *Instituto Nacional de Telecomunicações - Inatel*. (Av. João de Camargo, 510 - Santa Rita do Sapucaí - MG - Brasil - 37540-000).

II. General Codec Description

The class of multidimensional product 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 $4D$, 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, as illustrated in Figure 1. In this figure, the sessions are related to the trellis sessions shown in Figure 2.

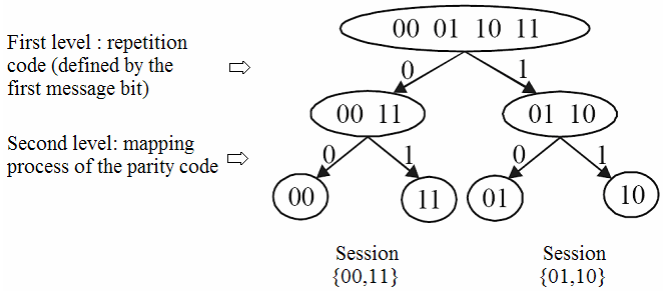


Fig. 1. Set partitioning used to form the code $(n, n/2, 4)$.

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].

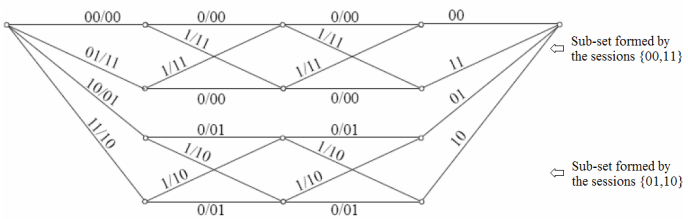


Fig. 2. Trellis for the code $(n, n/2, 4)$ [1].

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 [5]. 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 codewords is necessary. 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 codewords could not be found by the Chase algorithm.

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 $\mathbf{R}$ represents all n^D received noisy symbols. The expression "decoding in one dimension" in this figure means decoding n^{D-2}, $n \times n$ 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 means applying the SISO decoding algorithm n^{D-1} times or, as adopted in this paper, using n^{D-1} parallelized decoding structures [7].

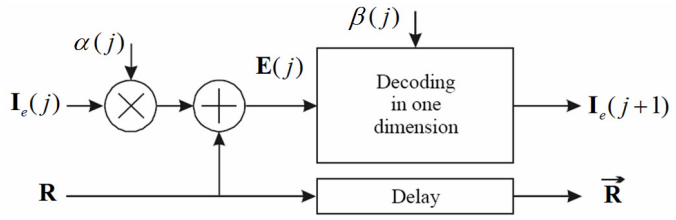


Fig. 3. Turbo decoding process proposed by Pyndiah in [6].

III. CODEC IMPLEMENTATION

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

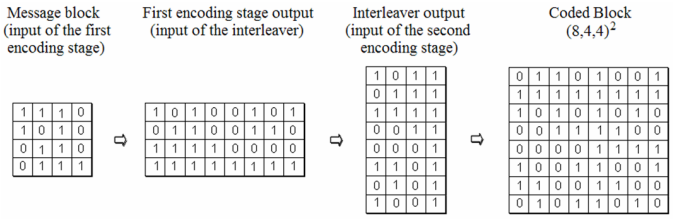


Fig. 4. $(8,4,4)^2$ encoding process.

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 chunks 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 the classic quadratic Euclidian distance between the received signal-vector and its expected value, metrics are calculated by comparing the received signal-vector to its 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, Figures 5a and 5b show a hypothetical code vector $\mathbf{r}$ being decoded according to the trellis structure showed in Figure 2. The matrix $\boldsymbol{\Delta}$ contains all values operated in order to obtain the metrics for the trellis sessions {00,11} and {01,10}. The maximum expected values for the quantized signal are +15 and −16 (5 bits of quantization).

$$\mathbf{r} = [-11 \quad -9 \quad +7 \quad +8 \quad +9 \quad +5 \quad -3 \quad -7]$$

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

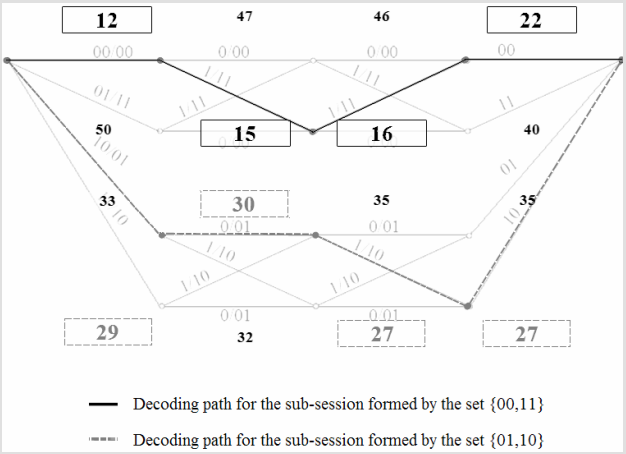
$$\Delta = \begin{bmatrix} -(-16-(-11))+(-16-(-9)) & -(-16-7)+(-16-8) & -(-16-9)+(-16-5) & -(-16-(-3))-(-16-(-7)) \\ (15-(-11))+(15-(-9)) & (15-7)+(15-8) & (15-9)+(15-5) & (15-(-3))+(15-(-7)) \\ -(-16-(-11))+(15-(-9)) & -(-16-7)+(15-8) & -(-16-9)+(15-5) & -(-16-(-3))+(15-(-7)) \\ (15-(-11))-(-16-(-9))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

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

Fig. 5a. Hypothetical received code vector $\mathbf{r}$ and its double decoding process in the trellis structure of the (8,4,4) code.

The codewords $\hat{\mathbf{c}}$ e $\hat{\mathbf{c}}$' and their respective accumulated metrics are also shown in Figure 5 and results in a final decision in favor of $\hat{\mathbf{c}}$. The soft-output, $\hat{\mathbf{c}}_{SISO}$, are then calculated from the Wagner hard-output $\hat{\mathbf{c}}$ by the simple replacement of the '1' and '0' elements by $+\beta$ and $-\beta$, respectively. This function was built using a *mux* structure for each element in each SIHO outputs and results in:

$$\hat{\mathbf{c}}_{SISO} = (-\beta \quad -\beta \quad +\beta \quad +\beta \quad +\beta \quad +\beta \quad -\beta \quad -\beta)$$



$$\hat{\mathbf{c}} = (0 \quad 0 \quad 1 \quad 1 \quad 1 \quad 1 \quad 0 \quad 0)$$

$$\sum \Delta = 12 + 15 + 16 + 22 = 65$$

$$\hat{\mathbf{c}}` = (0 \quad 1 \quad 0 \quad 1 \quad 1 \quad 0 \quad 1 \quad 0)$$

$$\sum \Delta' = 29 + 30 + 27 + 27 = 113$$

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

The absolute values of β increase linearly and proportionally to the j-th decoding step. To exemplify the influence of β over the extrinsic information during the turbo decoding process, consider a maximum value of j equal to 4, corresponding to 2 iterations. A linear progression of $\beta(j)$ could be [3, 7, 11, 15] and the matrix **R** below could represent the received codeword corresponding to the quantized output a matched filter or correlator.

$$\mathbf{R} = \begin{bmatrix} -11 & -9 & +7 & +8 & +9 & +5 & -3 & -7 \\ +3 & -6 & -3 & +3 & -5 & +5 & +0 & -4 \\ -5 & +4 & -5 & -7 & -2 & -4 & -4 & -2 \\ -3 & -6 & -3 & -3 & +5 & -6 & +7 & -5 \\ -4 & +8 & -4 & -4 & -6 & -8 & -6 & -5 \\ -6 & -4 & +8 & +7 & +6 & -7 & +4 & +0 \\ -5 & +9 & +2 & +7 & +2 & +3 & -5 & -2 \\ +6 & +1 & -4 & -3 & -3 & -6 & +4 & -7 \end{bmatrix}$$

The values in **R**, visualized sequentially in time, would appear as the ones shown in Figure 6.

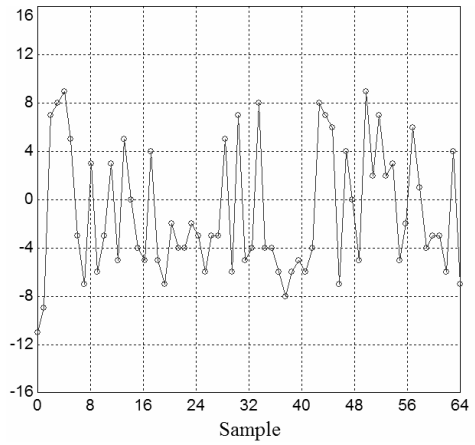


Fig. 6. Hypothetical received codeword samples in **R**

In the first decoding step j = 1, $\mathbf{I}_e(1) = 0$ and $\mathbf{E}(1) = \mathbf{R}$. As previously demonstrated, after decoding each row of $\mathbf{E}(1)$, using eight identical parallelized decoding structures, we obtain the SIHO (Soft-Input, Hard-Output) result:

$$\mathbf{SIHO}_{out}(1) = \begin{bmatrix} 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix}$$

The soft-output is generated by applying the first value of β over the SIHO result, leading to:

$$\mathbf{SISO}_{out}(1) = \begin{bmatrix} -3 & -3 & +3 & +3 & +3 & +3 & -3 & -3 \\ +3 & -3 & -3 & +3 & -3 & +3 & +3 & -3 \\ -3 & -3 & -3 & -3 & -3 & -3 & -3 & -3 \\ +3 & -3 & +3 & -3 & +3 & -3 & +3 & -3 \\ -3 & -3 & -3 & -3 & -3 & -3 & -3 & -3 \\ -3 & -3 & +3 & +3 & -3 & -3 & +3 & +3 \\ -3 & +3 & -3 & +3 & -3 & +3 & -3 & +3 \\ +3 & +3 & -3 & -3 & -3 & -3 & +3 & +3 \end{bmatrix}$$

Now, subtracting the soft-input $\mathbf{E}(1)$ from the above soft-output, we obtain $\mathbf{I}_e(2)$, the extrinsic information for the next step:

$$\mathbf{I}_e(2) = \mathbf{SISO}_{out}(1) - \mathbf{E}(1) \Rightarrow$$

$$\mathbf{I}_e(2) = \begin{bmatrix} +8 & +6 & -4 & -5 & -6 & -2 & +0 & +4 \\ +0 & +3 & +0 & +0 & +2 & -2 & +3 & +1 \\ +2 & -7 & +2 & +4 & -1 & +1 & +1 & -1 \\ +6 & +3 & +6 & +0 & -2 & +3 & -4 & +2 \\ +1 & -11 & +1 & +1 & +3 & +5 & +3 & +2 \\ +3 & +1 & -5 & -4 & -9 & +4 & -1 & +3 \\ +2 & -6 & -5 & -4 & -5 & +0 & +2 & +5 \\ -3 & +2 & +1 & +0 & +0 & +3 & -1 & +10 \end{bmatrix}$$

The parameter α, as in [6], is used here to scale the extrinsic information. However, instead of describing a logarithmic variation, as proposed in [1], this parameter is now kept constant and equal to 0.25. This is done discarding two least significant bits on each element of $\mathbf{I_e}(2)$. To save some logical resources, this is done before the transposition process required for the next decoding step, which results in:

$$\left[\alpha \times \mathbf{I}_e(2)\right]^T = \begin{bmatrix} +2 & +0 & +0 & +1 & +0 & +0 & +0 & +0 \\ +1 & +0 & -1 & +0 & -2 & +0 & -1 & +0 \\ -1 & +0 & +0 & +1 & +0 & -1 & -1 & +0 \\ -1 & +0 & +1 & +0 & +0 & -1 & -1 & +0 \\ -1 & +0 & +0 & +0 & +0 & -2 & -1 & +0 \\ +0 & +0 & +0 & +0 & +1 & +1 & +0 & +0 \\ +0 & +0 & +0 & -1 & +0 & +0 & +0 & +0 \\ +1 & +0 & +0 & +0 & +0 & +0 & +1 & +2 \end{bmatrix}$$

At this point the resultant soft-decoded samples for one dimension are stored in a buffer and the decoding process continues in a clock pulse basis, operating with all samples together in a sequential manner.

Figures 7 and 8 present the values of **E**(2) and **E**(4) (last partial decoding) overploted to **R** and its respective SIHO decoding results. As an example, the soft-input **E**(2) is formed according to:

$$\mathbf{E}(2) = \mathbf{R}^T + \left[\alpha \times \mathbf{I_e}(2)\right]^T \Rightarrow$$

$$\mathbf{E}(2) = \begin{bmatrix} -9 & +3 & -5 & -2 & -4 & -6 & -5 & +6 \\ -8 & -6 & +3 & -6 & +6 & -4 & +8 & +1 \\ +6 & -3 & -5 & -2 & -4 & +7 & +1 & -4 \\ +7 & +3 & -6 & -3 & -4 & +6 & +6 & -3 \\ +8 & -5 & -2 & +5 & -6 & +4 & +1 & -3 \\ +5 & +5 & -4 & -6 & -7 & -6 & +3 & -6 \\ -3 & +0 & -4 & +6 & -6 & +4 & -5 & +4 \\ -6 & -4 & -2 & -5 & -5 & +0 & -1 & -5 \end{bmatrix}$$

The soft-input **E**(4) is formed in the same manner.

From Figures 7 and 8 we can notice a tendency of changing the signs of some elements of **E**(2) and **E**(4) when compared with the original received codeword **R**. Specifically, the samples 17 and 24 in Figure 7 show that in this step the decoder made a wrong decision, since the signs of samples 17 and 24 are opposite to the decoded symbols.

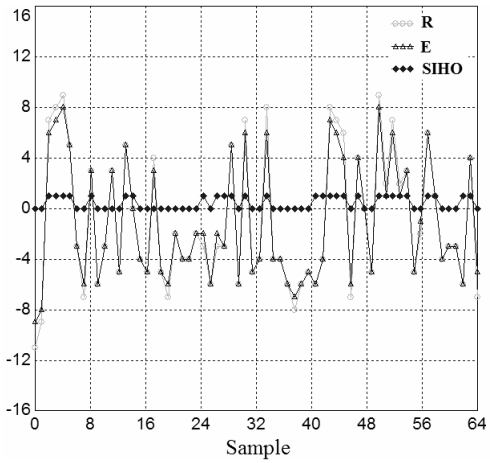


Fig. 7. Graphical representation of **E**(2) elements example over the received symbols **R** and its respective SIHO decoding results.

However, it can be noticed that the magnitude of the soft-input samples **E**(2) were changed slightly as compared to the original received codeword **R**. This behavior also happened from the second step to the fourth (second iteration), showing a tendency of changing to the opposite polarity and, then, to the right decision. In this case, **E**(4) were further changed as compared to **E**(2).

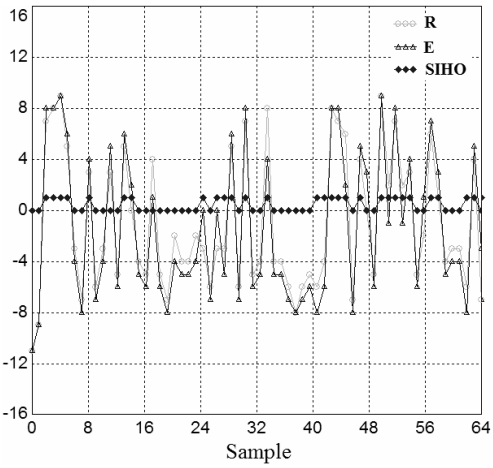


Fig. 8. Graphical representation of **E**(4) elements example over the received symbols **R** and its respective SIHO decoding results.

It is worthy mentioning that the tendency mentioned above can happen in the contrary and undesired way, as an example of a typical decoding divergence that leads to decoding errors.

Figure 9 shows the performance result expected for the $(8,4,4)^2$ turbo product code. This result was obtained by computer simulation using the software *Mathcad* [1].

Finally, Figure 10 shows performance results with the codec implemented in FPGA. A very close agreement was observed between these results and those obtained by simulation.

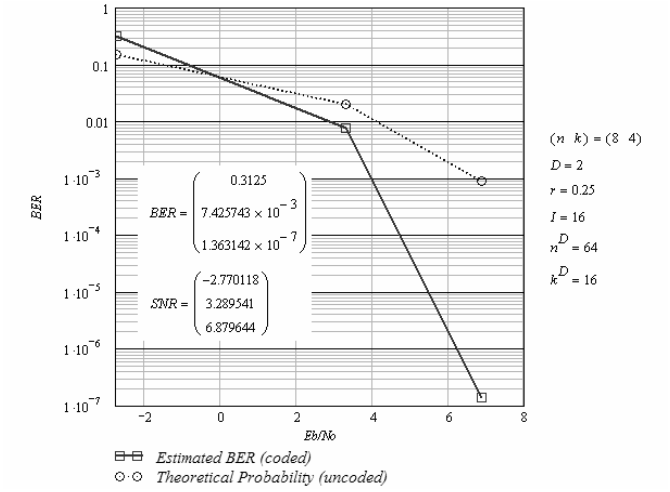


Fig. 9. Simulated $(8,4,4)^2$ turbo product code performance (16 interactions) on AWGN channel.

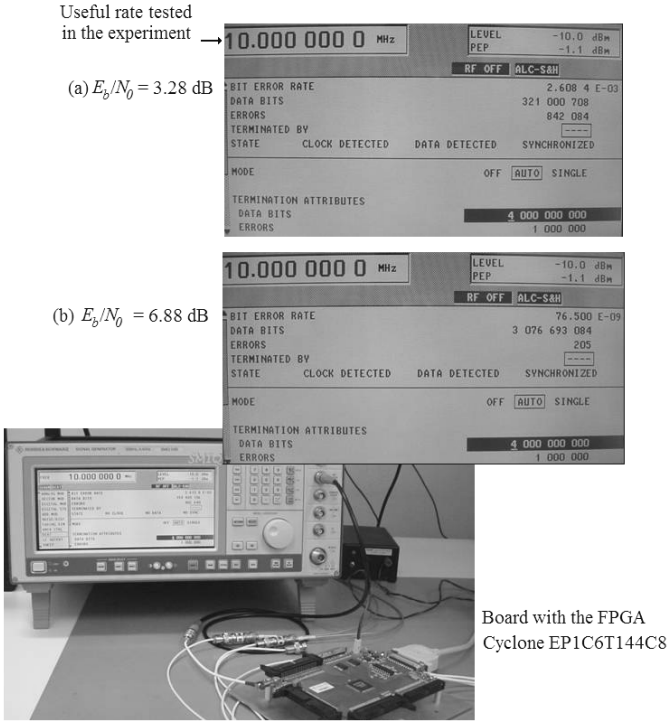


Fig. 10. Measured $(8,4,4)^2$ turbo product code performance (16 interactions) on AWGN channel. The parameter α was kept constant and equal to 0.25. The parameter β was quantized in 4 bits.

IV. Final Comments

With 16 iterations, the proposed scheme achieved code gains of about 4.5 dB. 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.

Using the approach presented here it was possible to complete a partial iteration, which means the decoding of one dimension, in only one clock cycle. A complete decoding has used only 60% of the available FPGA logic resources, and experienced a propagation delay in the combinational circuit less than 30 ns. It would be possible to have a complete iteration in just one clock pulse by combining two of the described circuit in a pipe-line structure. In this case, however, more logical elements will be needed and this would demand an FPGA with more resources.

References

[1] D. A. Guimarães, *Uma Classe de Códigos Produto e sua Decodificação Turbo Aplicada em um Sistema CDMA 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).

Ivan Simões Gaspar was born in Itajubá, MG, Brazil, on november 22, 1979. He holds the titles: Electronics Technician (ETE FMC, 1997), Electrical Engineer (Inatel, 2003) and M.Sc. in Electrical Engineering (Inatel, 2006). From 1999 to 2003 he was a teaching assistant at Inatel where he had the opportunity to work with development of didactical hardware and computational simulations to support the activities of Digital Communications Laboratory. Since February of 2003 he is with *Linear Equipamentos Eletrônicos S.A.* where he held a Technical Supervisor position and was involved with the development of digital communications systems like Digital Radio Links and Digital TV Transmitters. His interests include the general aspects of digital transmission, specially channel coding, digital modulation schemes, and hardware implementation using FPGA.

Dayan Adionel Guimarães was Born in Carrancas, MG, Brazil, on March 01, 1969. He holds the titles: Electronics Technician (ETE "FMC", 1987), Electrical Engineer (Inatel, 1994), Specialist in Data Communication Engineering (Inatel, 2003), Specialist in Human Resources Management (FAI, 1996), Master in Electrical Engineering (Unicamp, 1998) and Doctor in Electrical Engineering (Unicamp, 2003).

From 1988 to 1993 he developed equipment for Industrial Instrumentation and Control, and also occupied the positions of Manufacturing and Product Engineering Supervisor at *SENSE Sensores e Instrumentos*. Since January 1995 he is Professor at Inatel where, for eight years, he was responsible for the structure that supports practical teaching activities for the Electrical Engineering undergraduate course. His research includes the general aspects on Digital and Mobile Communications, specifically Multi-Carrier CDMA systems, and coding for fading channels, specifically Block Turbo Codes.

Dr. Dayan is member of the *Telecomunicações* magazine's Editorial Board, member of the Inatel's Master Degree Counseling Board and of the IEICE *(Institute of Electronics, Information and Communication Engineers*), Japan.