Block Turbo-Codes Decoding using the Walsh-Hadamard Transform applied to Hartmann Nazarov algorithm. Real Time Implementation on the DSP Texas TMS320C6201 platform

Texas Instrument DSP Solution Challenge

Ecole Nationale Supérieure de Télécommunication de Bretagne

Project Team: William TAPSOBA

Advising Professor: André GOALIC
Abstract

From the need to transmit more efficiently and more reliably data through a channel, has arisen the coding. A lot of methods have been proposed to improve data transmission. One of them is the Block Turbo Coding (BTC), introduced by Ramesh Pyndiah, in 1994, one year after the introduction of Turbo Code by Claude Berrou. All of them are part of the family of concatenated codes introduced by P Elias (1954) and G.D. Fornet (1966). The performance of the Block Turbo Coding seems to be similar to the Turbo Coding with identical parameters.

Coding goes with Decoding. The problem is more often to find a right decoding algorithm for the coding, allowing to keep the same coding gain, in term of computation and implementation. Iterative decoding algorithms have shown better performance than algebraic decoders or hard decoders one.

A new algorithm based on Walsh-Hadamard Fast Transformation, gives in theory, the best performance in Block Turbo-Code decoding. It uses a symbol-by-symbol decoding rather than the whole word code decoding.

The purpose of this project is to implement this Walsh-Hadamard Fast Transformation Decoding algorithm on the DSP Texas TMS320C6201 platform, and get practical results to match them with the theoretical ones. For that, we had to create a simulation environment.
I. INTRODUCTION

To have a practical test of the Block Turbo-Code decoding using Walsh-Hadamard Fast Transformation, we have created a simulation environment. It consists of one PC in which the coding program is stored and a DSP TMS320C6201 which has the decoding program on it. The BTC Coder has in charge to generate a random data and code them. Then it adds a noise to the data and send them to the DSP. There, data have to be decoded and sent back to the host. The received data is compared to the original one; thus we get a value of the Bit Error Rate (BER) corresponding to this noise. This process is iterated for each predefined value of Signal to Noise Ratio (SNR). We get couples of values (BER; SNR) and a graphic is created on the host to display the variation of the BER with SNR. From this graphic, the decoding performance is deducted. The data exchange between the BTC Coder on the host (PC) and the BTC Decoder target (DSP) is ensured by RTDX link (Real Time Data eXchange).

Code Composer Studio and Visual C++ are the tools used to realise this project. The first has permitted to create the decoder program and to load it on the DSP chip whereas the second is used to implement the coder program and for the graphic display.

We will start with an overview of the Block Turbo-Code in the section II. The decoding process using Walsh-Hadamard Fast Transformation is explained in section III. The section IV details the implementation on the TI DSP chip.
II. OVERVIEW OF BLOCK TURBO-CODES

Various factors (noise, interference, fading…) can corrupt an emitted data during its transmission. To overcome that, the data to transmit have to be protected. This is relevant to the channel coding. It only consists in adding redundancy bits. There are several methods of channel coding. One of them is the Block Turbo-Code introduced by Ramesh Pyndiah. Its principles are derived from Product Code one.

1. The product Code

A linear block code is defined by 3 parameters \((n,k,\delta)\) where \(n,k,\delta\) respectively stand for code word length, number of information bits and minimum Hamming distance. Thus the number of redundancy bits is \(nk\). The Hamming distance between two code words is the number of position in which their bits differ. The code rate is defined by the ratio \(r = k/n\).

The product code \(P = C \otimes C\) is then obtained by :
- placing \((k \times k)\) information bits in an array of \(k\) rows and \(k\) columns.
- coding \(k\) rows and \(n\) columns using code \(C\).

The parameters of the resulting product code \(P\) are given by :
\[ N = n \times n, \quad K = k \times k, \quad \Delta = \delta \times \delta, \quad \text{and the code rate} : R = r \times r. \]
The \((n-k)\) last rows and the \((n-k)\) last columns also are code words of \(C\). A soft decision algorithm derived from the theoretical Log-Likelihood-Ratio (LLR), has been proposed by R. Pyndiah in 1993 to compute the extrinsic information.

2. Block Turbo Codes

Turbo Codes consist in a concatenation of several codes. Block Turbo Codes used two concatenated block codes, yielding two dimensional turbo codes. The principle of serial encoder consists in encoding the data flow by the first encoder, the output is then interleaved before encoding it by the second encoder. Block Turbo Codes use BCH codes as component codes because they are simple to decode they have a large minimum distance which allows a low-complexity decoding. And therefore a good performance is obtained.
III. BLOCK TURBO-CODES DECODING using Walsh-Hadamard Transform

The turbo decoding uses soft input and soft output: the received data and the decoded one are sequences of real numbers.

The algorithm based on Walsh-Hadamard Transformed (WHT) gives an optimal symbol-by-symbol decoding of binary block code. Let’s take the block code $C(n,k)$ and let the $m=(m_i; \ 0 \leq i < k)$ denotes information symbols for codewords $B=(m_i; \ 0 \leq i < n)$ of elementary binary block. Let the received word $R= (m_i; \ 0 \leq i < n)$

For memoryless channel, the optimum symbol-by-symbol decoding algorithm can be written as: $C(l) = \Pr\{b_l = 0/R\} - \Pr\{b_l = 1/R\}$

Nazarov and Smalyaninov, through tedious calculation, rewrote $C(l)$ and expressed the soft output LLRs as:

$$LLR_{S_l} = \log \frac{1+C(l)}{1-C(l)}$$

$$C(l) = \frac{1}{2} \left[ \rho_l \left( 1 + \frac{F_D(h_l)}{F_D(0)} \right) \right] + \frac{1}{\rho_l} \left[ \rho_l \left( 1 + \frac{F_D(h_l)}{F_D(0)} \right) \right]$$

Where $F_D$ is the Walsh-Hadamard Transformed of $D_\gamma$

$$F_D(h_l) = \sum_{\gamma=0}^{2^{n-1}-1} D_\gamma \ast \exp \left( j \pi \sum_{m=0}^{2^{n-1}-1} \gamma_m \ast h_m \right)$$

with

$$D_\gamma = \exp \left[ 1/h \left( F_p(0) \ast F_p(\gamma) \right) \right] \ast \exp \left[ 1/2 \pi \left( F_p(0) \ast F_p(\gamma) \right) \right]$$

Thus the soft output LLRs$_l$ has been calculated and its symbol can be decomposed into a sum of two independent symbols:

$$LLR_S = LLRe_l + w_l$$

Where $LLRe_l$ is the soft input and LLRs$_S$ is the soft output of the decoder for a sample at position $l$. $w_l$ is a correction term on the soft input LLRe$_l$ called extrinsic information. This extrinsic information calculated for each bit and at each iteration is used as a a-priori information for each next iteration improving thus the decoding.

$$LLR_S = LLRe_l + w_l \text{ (from the previous iteration) } + w_l$$
For the Product Codes, each iteration consist in two half-iterations: one to decode the rows and the other for the columns. The extrinsic information is used by the following elementary decoder to update its input information. Thus from one decoder to the next one, the value of the decoded bit becomes more and more reliable; therefore the performances are increased at each iteration.

\[
[W]\_p = [LLR]\_p - [LLRe] - [W]\_{p-1}
\]

Figure 1: Presentation of a SISO Decoder

Figure 2: Iterative Decoding schema

At the \(p^{th}\) iteration, the soft input matrix \([LLRe]\) and the extrinsic information matrix \([W]\), can be expressed as follows:

\[
[W]\_p = [LLRs]\_p - [LLRe] - [W]\_p-1
\]
IV. IMPLEMENTATION

The demonstration platform uses hardware and software environment and includes two parts:

- the first part is implemented in the host computer which main tasks are: generation of the matrix of information bits, coding rows and columns and sending the noisy samples to the decoder. This part also includes BER computation before drawing curves.

- the second part is implemented on the DSP, it is the decoding process. Each noisy frame is decoded before sending back the matrix to the host, then statistics can be made.

Data exchange between the Turbo-Coder (on the host) and the Turbo-Decoder (on the DSP) is made via the RTDX link. This data exchange is based on the COM (Common Object Model) technology. Any COM client can access to the data: Visual C++ have been selected.

![Figure 3: Data Exchange Architecture](image)

The Turbo Decoder (TD) is fed by a block of 1024 samples quantized on 8 bits. The main tasks of the TD are:

- processing of each row and each column;
- calculating an intrinsic information to provide to the next module.

We have limited the number of our modules to four (4).

The decoded block is sent back to the host where it is compared to the original one. Another block of data is sent to the TD for decoding.
To respect Montecarlo criteria this process is repeated until the number of wrong decoded block reaches 100. The Bit Error Rate (BER) is then calculated. Therefore, we get a couple of value: the BER and the RSB put to the corresponding block data. This couple of value constitute a point in the graphic to display.

To have the others points of the graphic, the whole process above is recalled.

One way to reduce the number of instructions is to keep when it’s possible the parameters in the CPU. Thus, local parameters can stay in the CPU registers decreasing the number of access memory. In this implementation, the variables that are used during the four tasks are declared as global variables and the ones that are needed only for one task are defined as local variables.

With the results we get, the following graphic can be displayed in our Visual C++ programme:

![Figure 4: Comparison between floating point implementation in C language and fixed point implementation using quantized parameters in C language](image-url)
V. **CONCLUSION**

The Hartmann Nazarov algorithm uses mathematical functions such as tanh, log, exp. For the implementation on the DSP, the parameters are quantized in 8 bits and their tanh, log, and exp values are stored in arrays.

The implementation of Hartmann Nazarov algorithm to decode Block Turbo-Codes shows us good performances with quantized parameters (0.2 dB to the values we get in floating points).

To maintain the performances in term of BER, the decoder uses formats adapted to the dynamic of the different parameters.

Due to the fact that the DSP uses a specific format not directly dealt in C, only the decoder assembler version, which is still in development, can allow to ensure an acceptable data flow. We continue to develop the assembly version of the decoder. That will permit to improve and to have better performances.

As Hartmann Nazarov algorithm is able to give extrinsic reliably per bit, it gives better performances on some real channels.