Journal of Wireless Networking and Communications 2012, 2(4): 35-42
DOI: 10.5923/j.jwnc.20120204.02

VHDL Design and FPGA Implementation of a Fully Parallel Architecture for Iterative Decoder of Majority Logic Codes for High Data Rate Applications

M. El Haroussi*, M. Belkasmi

SIME Laboratory, ENSIAS, Mohamed V Souissi Univ., Rabat, Morocco
belkasmi@ensias.ma

Abstract In this work, we propose a design and FPGA (Field Programmable Gate Arrays) implementation of three parallel architectures for majority logic decoder of low complexity for high data rate applications. These architectures are hard decision decoder architecture (Hard in - Hard out (HIHO)), the SIHO threshold decoding (Soft in – Hard out) and the SISO threshold decoding (Soft in – Soft out). The chosen code is the Difference Set Cyclic DSC code. The VHDL (Very high speed integrated circuit Hardware Description Language) design and the synthesis of such architectures show that such decoders can achieve high data rate with low complexity. In our case, the iterative decoder associated to the fully parallel SISO threshold decoders allows achieving high data rates, 2 clock cycles for two iterations with a complexity of 7350LEs.

Keywords Error Correcting Codes, Iterative Decoding, Majority Logic Decoding, Threshold Decoding, VHDL Language, FPGA

1. Introduction

The complexity and the latency, which are often higher during the decoding process, make up the major decoding drawbacks in the current digital systems. The decoding data rate depends of the parallelism of the chosen architecture, i.e., its ability to process simultaneously multiple data. The architecture presents a high level of parallelism more it is able to achieve a very high data rate. In electronic design, the complexity of the circuit (i.e., its surface) is often considered as a critical parameter. In this context of very high data rate, operating speed must be maximized with limitation of complexity induced by the parallelism of calculations. We will specially study the decoding of the codes belonging to the OSMLD (one step majority logic decoding) codes family [3-5] and used in an iterative decoding process. The majority logic decoding uses a linear combination of a reduced set of syndromes represented by orthogonal equations. Hence it allows, by choosing a pipeline and / or parallelized architecture, to have a less complex decoding circuit with a high data rate.

The principle of majority logic decoding and the proposed VHDL design and FPGA implementation of majority decoder with a hard decision is described in Section 1, the Section 2 present the VHDL design and FPGA implementation of architecture of the Massey's APP (A Posteriori Probability) algorithm[3]. The Section 3 present the VHDL design and FPGA implementation of a fully parallel architecture of SISO threshold decoding, using the joint exploitation the parallelism of symbol and the parallelism of sub-blocks so as to achieve high data rate. In Section 4, we will describe the iterative algorithm used to decode the OSMLD codes. Before concluding we will present the results of our simulations.

2. Decoding Majority Logic Codes

A. Simple majority logic code (MLC)

Consider a linear cyclic code C (n, k) with H parity-check matrix. The space generated by H is the cyclic code (n, n-k), denoted by $C^{\perp}$, which is the dual code of C, or the null space of C. For any vector v in C and any vector w in $C^{\perp}$, the inner product of v and w is zero.

Now suppose that a code vector in C is transmitted over a binary symmetric channel. Let e (e_1, e_2, …., e_n) and R (r_1, r_2, …, r_n) be the error vector and the received binary vector respectively. Then R = v + e. For any vector w in the dual code $C^{\perp}$, we can form the following linear sum of the received digits:

$$A = \sum_{p=1}^{n} r_p w_p$$

This is called a parity-check sum. Using the fact that <w,v>=0, we obtain the following relationship between the check sum A and error digits in e:

* Corresponding author:
m.elharoussi@yahoo.fr (M. El Haroussi)
Published online at http://journal.sapub.org/jwnc

$$A = \sum_{p=1}^{n} e_p w_p \qquad (1)$$

Suppose that there exist J vectors in the dual code $C^{\perp}$, which have the following properties:

1. The j^{th} component of each vector w_i is a 1
2. For i≠j there is at most one vector whose i^{th} component is 1

The application of the parity equation (1) on the set of J orthogonal vectors gives J equations on the j^{th} position

$$\forall i \in \{1,\ldots,J\}$$

$$A_i = \sum_{p \neq j}^{n} e_p + e_j$$

Example:

Let C (7, 3, 4) be the cyclic majority logic and DSC code having three orthogonal equations.

$$A_1 = e_4 + e_5 + e_7$$
$$A_2 = e_2 + e_6 + e_7$$
$$A_3 = e_1 + e_3 + e_7$$

B. hard-input hard-output Majority logic decoding (MLD)

The parity check equations orthogonal on e_n can be used to estimate or to decode the received bit r_n,

The value can be determined by the following:

- If at least J-(J / 2) +1 values of A_j are equal to 1 then the decision is e_n= 1,
- Otherwise, the decision is e_n=0.

C. Implementation of the parallel Majority logic decoder

To implement this decoder, we have transcribed the decoding steps on simple blocks described in VHDL and assembled as shown in Fig 1.

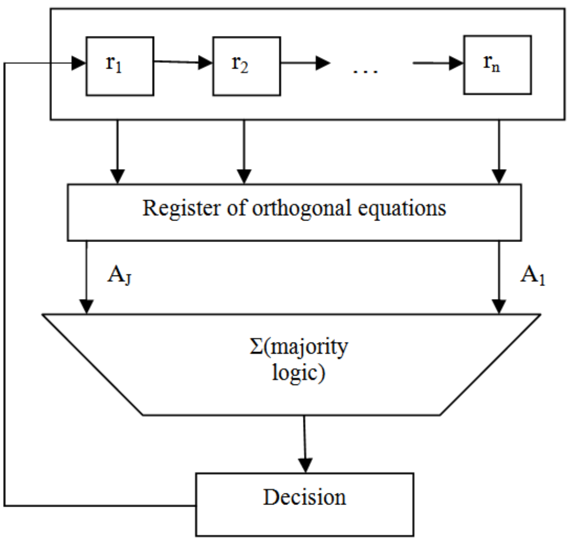

Figure 1. Parallel implementation of the proposed architecture for DSC code

By exploiting the parallelism of the symbols and by choosing purely combinatorial architecture, all the operations realized by the decoder:

- Calculation of A i
- Symbols rotation

Are obtained on the first rising edge of the clock (see Figure 2). The proposed architecture is described in VHDL and implanted on FPGA using the Altera's Quartus II tool. The FPGA used circuit is the Altera's EP1S10F484C5. This circuit is characterized by 10570 LEs and 336 input / output pins.

The complexity of our decoder is 174 LEs with 43 inputs / outputs. The Figure 3 shows the inputs/outputs of the VHDL circuit of our decoder. The Figure 4 shows the simulations in the C language and the hardware simulation.

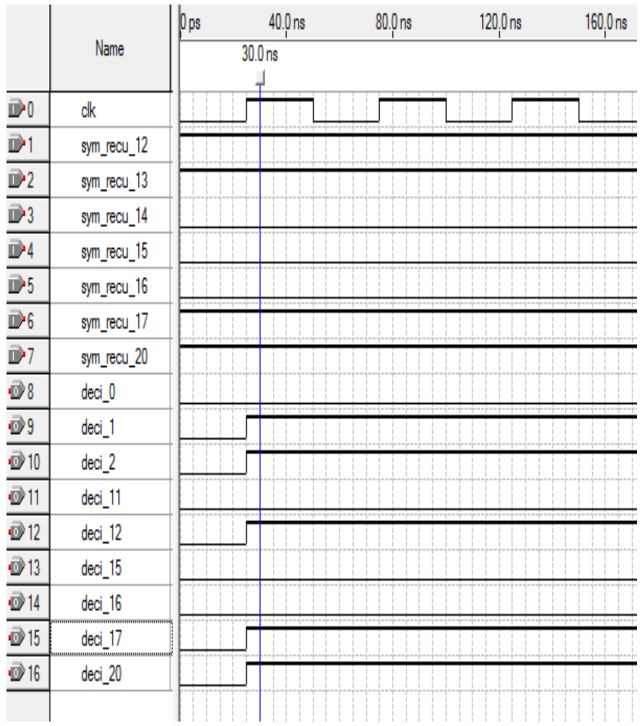

Figure 2. Example of functional simulation of Majority Decoding

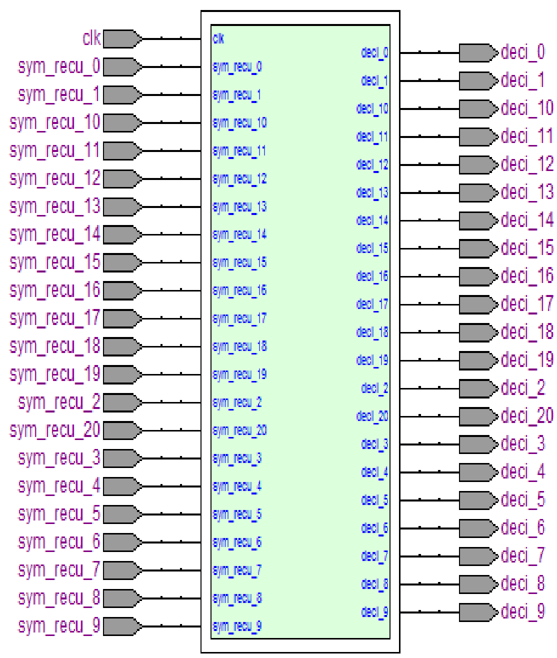

Figure 3. VHDL model (fully parallel) of hard decoding of DSC (21,11) code

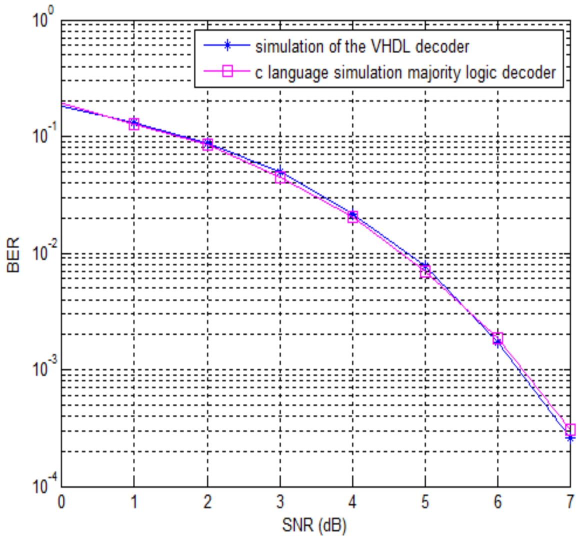

Figure 4. Comparison of the performances of two decoders for the DSC (21, 11)

3. Design of the Threshold Decoding

A. threshold decoding

In this section, we present the algorithm of threshold decoding [3]. We keep the same notations as those adopted in the previous section, assume the existence of J equations orthogonal on each position.

We consider a transmission of a codeword c ($c_1, c_2, \ldots c_n$), using BPSK modulation over a channel with additive white Gaussian noise (AWGN). Suppose that the word y ($y_1, y_2, \ldots, y_n$) is the received word. The decoder calculates for each bit y_k the Log Likelihood Ratio LLR defined as follows:

$$LLR(y_k) = \ln\left[\frac{p(c_k = 1/y)}{p(c_k = 0/y)}\right] \quad (2)$$

Where c_k is the k^{th} bit of the transmitted code word. For a One-Step Majority Logic Decodable (OSMLD) code C, there is J (A_j j in {1, ..., J}) orthogonal equations for each position k, The equation (2) becomes:

$$LLR(y_k) = \ln\left[\frac{p(c_k = 1/\{B_i\})}{p(c_k = 0/\{B_i\})}\right] \quad (3)$$

Where B_i, for i in {1, ..., J}, are obtained from the orthogonal equations on c_k bit, as follows:

B0 = y_k and each B_i with i in {1, ..., J}, is calculated by eliminating the term y_k from the i^{th} orthogonal equation.

By applying BAYES rule, (3) becomes:

$$LLR(y_k) = \ln\left[\frac{p(\{Bi\}/c_k = 1)}{p(\{Bi\}/c_k = 0)} \times \frac{p(c_k = 1)}{p(c_k = 0)}\right] \quad (4)$$

Since the parity check equations are orthogonal on the position j, so the probabilities P (Bi = 1 or 0) are independent and (4) can be written as:

$$LLR(y_k) = \sum_{i=0}^{i=J} \ln\left[\frac{p(B_i/c_k = 1)}{p(B_i/c_k = 0)}\right] \times \ln\left[\frac{p(c_k = 1)}{p(c_k = 0)}\right] \quad (5)$$

Since probability to transmit the symbol 1 is equal to that of 0. The equation (5) becomes:

$$LLR(y_k) = \sum_{i=1}^{i=J} \ln\left[\frac{p(B_i/c_k = 1)}{p(B_i/c_k = 0)}\right] + \ln\left[\frac{p(B_0/c_k = 1)}{p(B_0/c_k = 0)}\right] \quad (6)$$

This likelihood ratio becomes:

$$LLR(y_k) \approx (1 - 2B_0)w_0 + \sum_{i=1}^{J}(1 - 2B_i)w_i \quad (7)$$

Where the value of ($1-2B_i$) is equal +1 or -1.

w_i is an information proportional to the reliability of the i^{th} parity check equation. We can then show that:

$$(1-2B_0)w_0 = 4\frac{E_S}{N_0}y_k \quad et \quad w_j = \ln\left[\frac{1 + \prod_{k=1, j\neq k}^{k=n_j} \tanh(\frac{L_{jk}}{2})}{1 - \prod_{k=1, j\neq k}^{k=n_j} \tanh(\frac{L_{jk}}{2})}\right] \quad (8)$$

Where n_k is the total number of terms in the k^{th} orthogonal parity equation and with.

$$L_{jk} = 4\frac{E_S}{N_0}\left|y_{jk}\right|.$$

The term $W(y_k) = \sum_{i=1}^{i=J}(1-2B_i)w_i$ is extrinsic information representing the estimation of the orthogonal equations on the symbol y_k. Thus (8) becomes:

$$LLR(y_k) = 4\frac{E_S}{N_0}y_k + W(y_k) \quad (9)$$

The algorithmic structure of the threshold decoding is summarized as follows:

For each k=n,…,1

- Calculate the terms B_i and w_i ,i in *{1,..,J}*
- Calculate B_0 and w_0
- Calculate the extrinsic information w(y_k)
- the decoder output is obtained by

$$LLR(y_k) = 4\frac{E_S}{N_0}y_k + W(y_k)$$

B. Architecture of the threshold decoder

The entry consists of analogy samples (called symbols) encoded on q=4 or 5 bits: 3 or 4 bits for reliability and 1 for sign. The sign bit represents the binary value of the symbol ('0 'or '1'), the absolute value of a received bit represent its reliability (high reliability means that this bit has more chance to be correct).

We can simplify the implementation of the log likelihood ratio using an operator ADD_MIN [6] noted ◊. This operator searches the minimal value in absolute among n inputs and computes the sign according to the product of inputs signs.

$$w_i = \sum_{k=1}^{n_i} \Diamond L_{ik} = L_{i1} \Diamond L_{i2} \Diamond \cdots \Diamond L_{in_i} = \min_{k=1}^{n_i}\left|L_{ik}\right| \times \prod_{k=1}^{n} sig(L_{ik})$$

The simplified expression of the decoder output becomes:

$$LLR(y_k) = y_k + \sum_{i=1}^{i=J} (1 - 2B_i) w_i$$

The figure 5 presents the description, after simplification, the steps decoding from simples blocks described in VHDL and assembled. The architecture of the threshold decoding has three fundamental units which are:

- The shift register contains the quantified symbols received in parallel, the loading of the register is controlled by Ld input on the first rising front of the clock (see figure 6) and with each new rising front of the clock the shifted symbols are available at the register output.
- In the implementation of the add-min operator and the adder, the number of input ports must be configurable as well as the number of bits per port to avoid poor performance of the correction errors caused by deficient resolution of different operators within the architecture.
- The results of the add-min operation, the addition and the decision are available at the output at each rising edge of the clock.

Table 1. Surface comparison between 4 and 5-bit quantization

	Complexity (LE_S)	Number of I/O
q=4bits	259 LE_S	87 E/S
q=5bits	331 LE_S	108 E/S

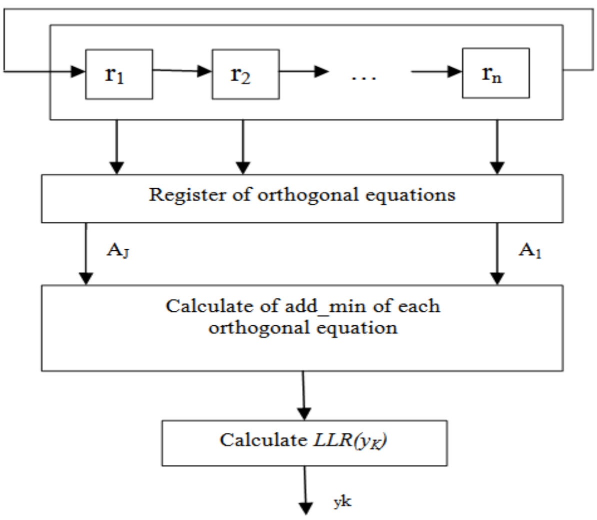

Figure 5. Parallel implantation of proposed architecture of the threshold decoding of MLC code

C. Implantation of the threshold decoding of DSC(21, 11) code.

The FPGA have Known a great improvement in size and speed. Also, the FPGAs provide the most suitable platform for implementing applications of error correcting and detecting codes. Several studies on the "threshold decoding" have already been carried out [7][8]. The VHDL description of the threshold decoding is made so that each unit of the proposed architecture is described in an independent entity. The final architecture of the parallel threshold decoding is given in Figure 5. The figure 6 shows the VHDL circuit input/output in our decoder. The proposed architecture has been described in VHDL and embedded on FPGA using the Altera's Quartus II tool. The FPGA used circuit is the Altera's EP1S10F484C5 type, this circuit is characterized by 10570 LEs, and 336 input / output pins.

The proposed architecture for the threshold decoding DSC (21, 11), the operations require a latency of 22cycles clock. The Table 1 shows a comparison of complexity (LEs) of our architecture with quantification q = 4 and q = 5 bits. The figure 7 shows an example of functional simulation on which the information presented to the entry of the decoder circuit will be decoded in 22 clock cycles.

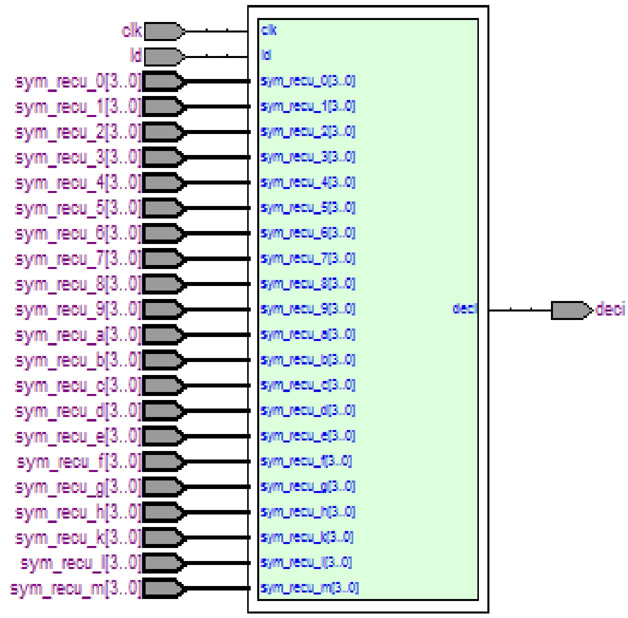

Figure 6. The VHDL model of threshold decoding of DSC (21, 11) code.

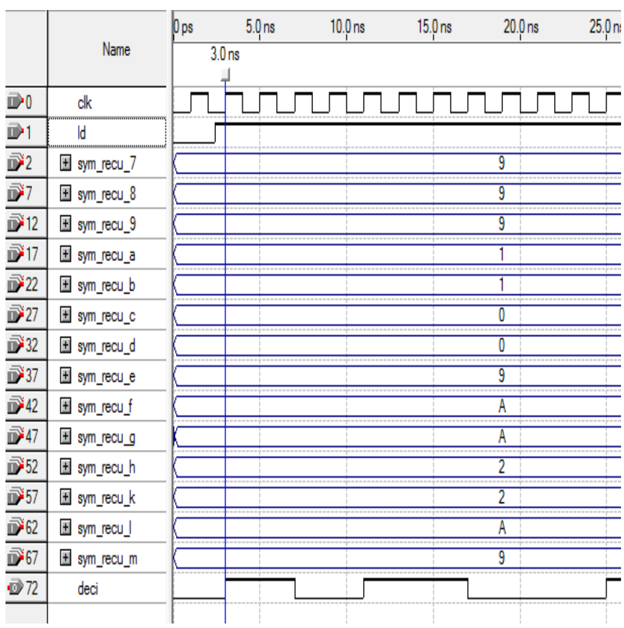

Figure 7. Example of functional simulation of our decoder

For validation of our decoder circuit, we were able to simulate and verify the decoding of 20000 words used for each SNR (Signal to Noise Ratio). The figure 8 shows the comparison of two decoders using threshold decoding algorithm for DSC (21, 11). The red curve shows the performance of threshold decoding by using simulation in C language (real data). The black curve present performance of the decoder with functional simulation to the material level for quantification respectively of 4bits and 5bits.

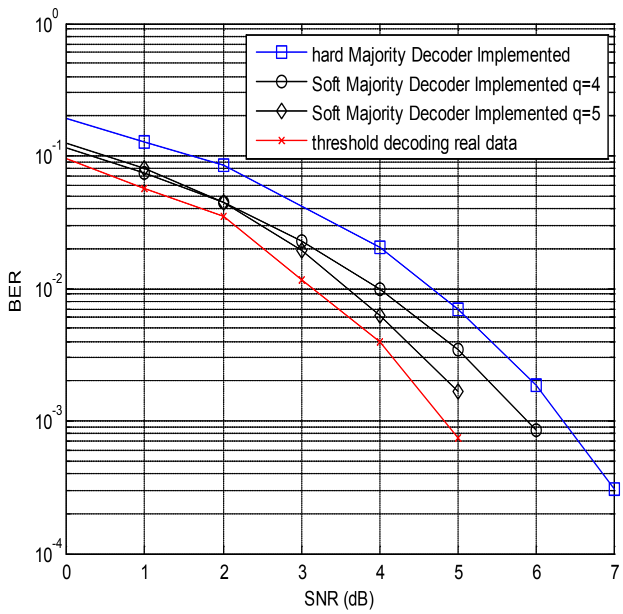

Figure 8. Comparison of performance of the Threshold decoder of DSC (21, 11) code in C language simulation and hardware level simulation

4. Design of Parallel Architecture of the SISO Threshold Decoding

A. Designe of SISO threshold decoding

The basic idea is to modify the conventional threshold algorithm by associating a reliability to each decision to be exploited in iterative decoding [1][2][11-13].

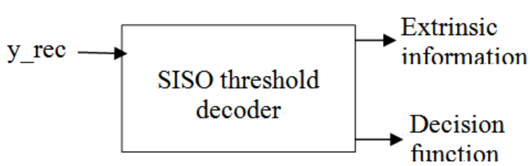

Figure 9. Implementation of soft-output decoder

B. Implementation of the parallel SISO threshold decoder of DSC code

Several studies on the "SISO decoders" have already been carried out[1][2][4]][5][13]. In this paper the VHDL description of the SISO threshold decoding is made so that each unit of the proposed architecture is described in an independent entity. The final architecture of the parallel SISO threshold decoding is given in figure 10. The figure 11 shows the VHDL circuit of our decoder and its input/output. The proposed architecture has been described in VHDL and embedded on FPGA using the Altera's Quartus II tool. The FPGA circuit used is similar to the one used to implement the threshold decoder.

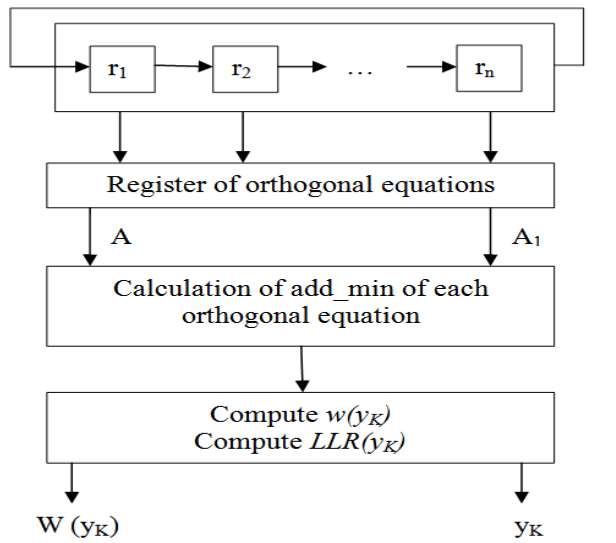

Figure 10. Parallel Implementation of proposed architecture of the SISO threshold decoder of an OSMLD code

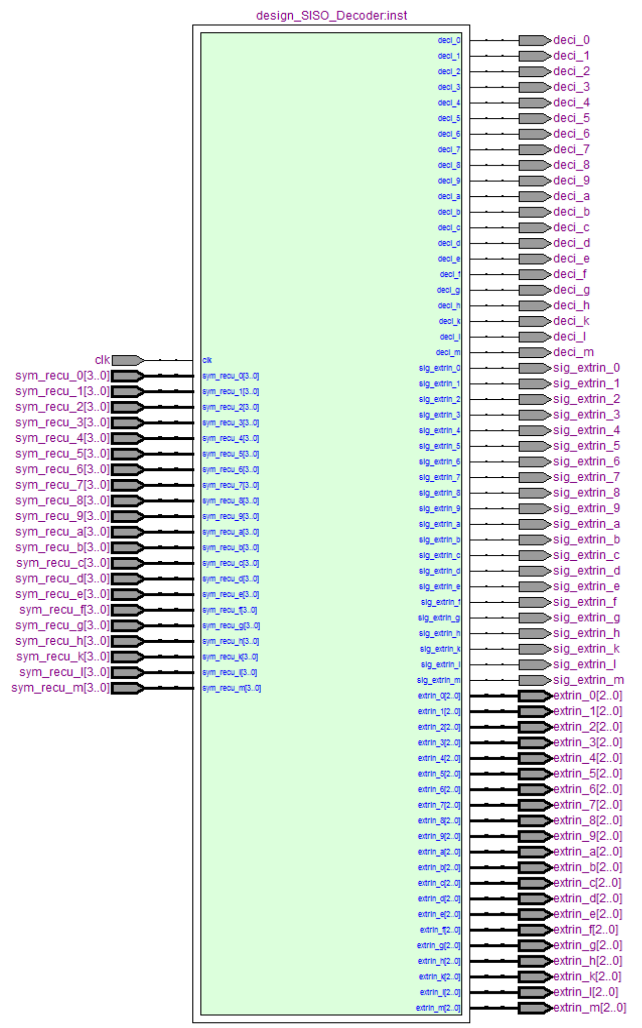

Figure 11. The VHDL model of the fully parallel SISO threshold decoder of DSC (21, 11) code

For the adopted architecture of the DSC SISO threshold decoder, the operations are obtained on the first rising edge of the clock and the occupied area for this decoder is 3672 LEs (logic elements) with 190 inputs/outputs for the DSC (21, 11) code. The figure 12 gives an example of functional simulation on which it is reported that the information presented at the input of the circuit decoder will be decoded on the first rising edge of the clock.

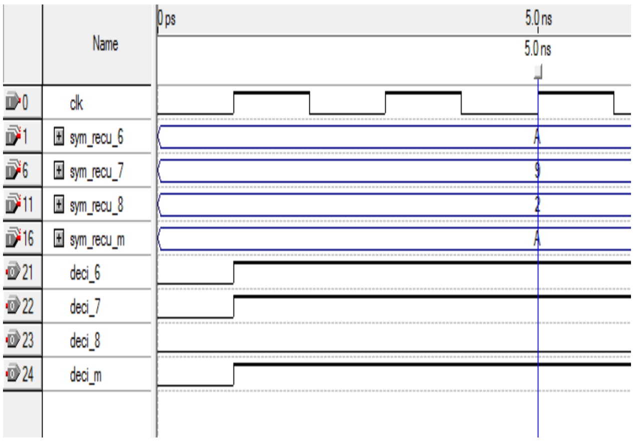

Figure 12. Example of functional simulation of our decoder

For validation of our decoder circuit, we were able to simulate and verify the decoding of 20000 words used for each SNR (Signal to Noise Ratio). We found the same results as Figure 8.

5. Iterative Threshold Decoding

A. Iterative threshold decoding

The iterative decoding of codes OSMLD was introduced for the first time by Lucas[14]. Two years after, the latter with Fossorier et al.[15] have proposed the belief propagation algorithm for decoding of OSMLD codes, in[16]. They showed that the iterative Hartmann-Rudolf-Lucas decoding, originally used for the decoding of product codes can be used to decode the codes OSMLD. In the process of iterative decoding, each decoder takes advantage of the extrinsic information produced by the other decoder in the previous step. How extrinsic information is transmitted and the way it is exploited by decoders to make their decision are not yet closed. The iterative decoding process that we use follows the Pyndiah's scheme [9][10] (figure 13).

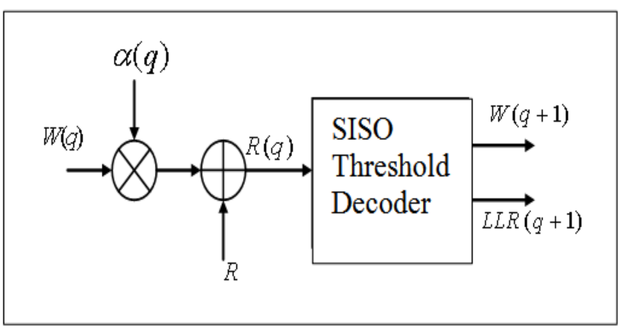

Figure 13. Component decoder

The soft input and respectively the soft output of the qth step (iteration) of the iterative decoding are given by:

$$R(q) = R + \alpha(q)W(q)$$

$$LLR(q+1) = 4\frac{E_s}{N_0}R(q) + W(q+1)$$

Where R represents the quantified received word and $W(q)$ the extrinsic information calculated by the previous decoder.

B. Implementation of parralel iterative threshold decoding

The final design of the parallel iterative decoder is described in VHDL and embedded on FPGA using the Altera's Quartus II tool. For the complexity of the global entity, the area occupied by the fully parallel iterative show in table 2 for the two codes DSC (21, 11) and DSC (73, 45) with 4 and 5 bits quantification. The figure 14 gives an example of functional simulation on which it is reported that the information presented at the input of the circuit decoder will be decoded on two rising edges of the clock for a two iterations, and for a number of iterations n_ite, latency is of n_ite clock cycles.

We were able to simulate and verify the decoding of 2000 words used for each SNR following a simulation protocol. Figure 15 (resp. Fig.16) shows the performances of our iterative decoder using the quantified data of 5 bits and α(q) fixed for the DSC(21, 11) (resp. DSC(73, 45)) code.

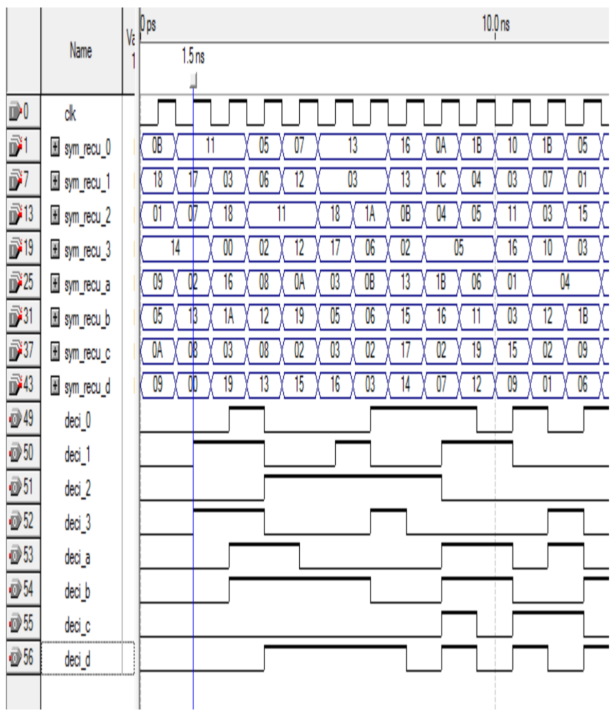

Figure 14. Functional simulation of our decoder for two iterations

Table 2. Surface comparison between 4 and 5-bit quantization for two decoder with two iterations

	DSC (73, 45)		DSC (21, 11)	
	Complexity (LEs)	*Number of E/S*	*Complexity (LEs)*	*Number of E/S*
q=4bits	87798	366	7350	106
q=5bits	106139	439	9219	127

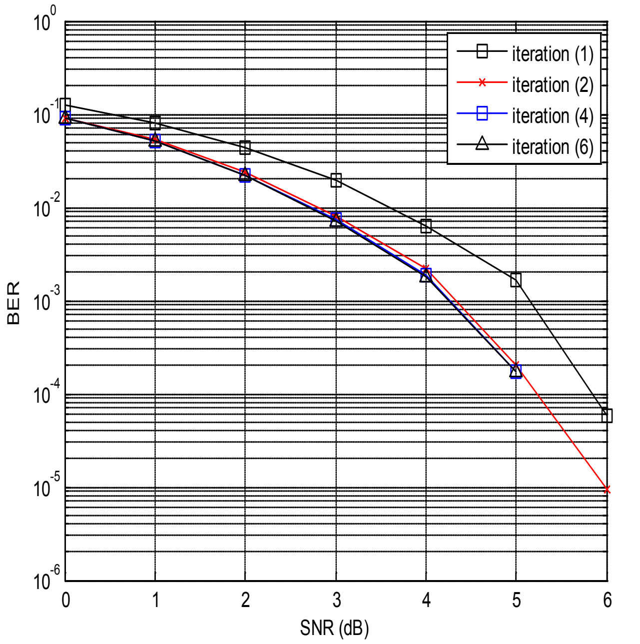

Figure 15. Performance of our implemented fully parallel simple iterative decoder (DSC (21, 11) data quantified in q=5 bits alpha=0.5)

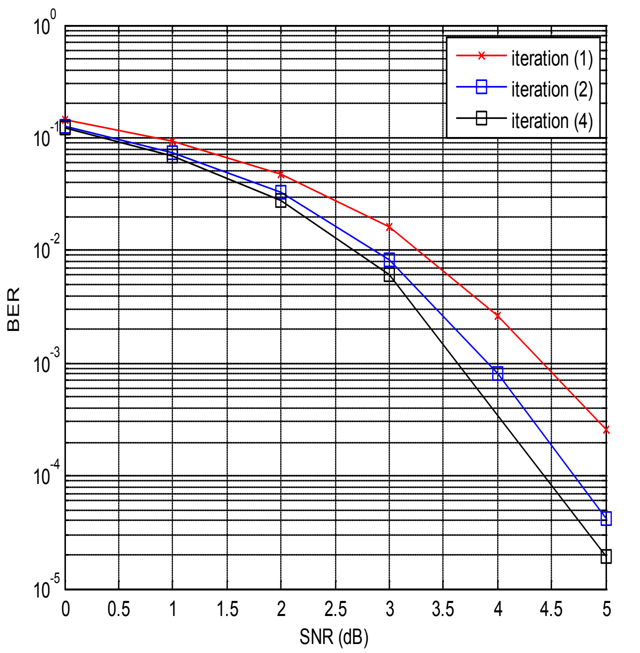

Figure 16. Performance of our implemented fully parallel simple iterative decoder (DSC (73, 45) data quantified in q=5 bits alpha=0.2)

6. Conclusions

The VHDL design and FPGA implementation of an iterative decoder for OSMLD codes have been proposed in this paper. This design allows reaching and achieving very high data rate while minimizing complexity. By adopting a combinatory architecture operating in pipelined and/or parallelized mode, the SISO threshold decoder reacts on the first active front of the clock with a complexity of 3672 LEs for DSC(21, 11) code and q=4bits. Furthermore the iterative decoder associated to the fully parallel SISO threshold decoders allows achieving high data rates, (2 clock cycles) for a two iterations with a complexity of 7350LEs for DSC(21, 11) code and q=4bits. As a perspective we plan to study the DSC turbo decoder.

REFERENCES

[1] C.Leroux, C.Jego, P.Adde et M.Jezequel,"Turbo décodage de codes produits très haut débit sur FPGA". Colloque GRETSI, 11-14 septembre 2007, TROYES, France.

[2] E. Piriou "Apport de la conception d'architecture flexible dédiée aux turbo codes en blocs", PhD thesis, ENST Bretagne, Jan 2007 (in Frensh).

[3] I. L Massey, “Threshold Decoding,” Cambridge, Mass.M.I.T. Press, 1963.

[4] M. Lahmer, "Décodage à seuil itératif des codes à logique majoritaire". PhD thesis, ENSIAS, Rabat, 2008, Morocco (in French).

[5] M. Belkasmi, M. Lahmer, F. Ayoub “Iterative Threshold Decoding of Product Codes Constructed From Majority Logic Decodable Codes ”, 2nd International Conf. on Information and Communication Technologies : From Theory to Applications, IEEE ICTTA’06, Apr. 2006, Damascus, Syria, pp. 2376 – 2381.

[6] J. Hagenauer, E. Offer, and L. Papke, "iterative decoding of binary block and convolutional codes", IEEE Trans. Inform. Theory, Mar. 1996, Vol. 42, pp. 429-446.

[7] M. Elamine,"Conception d'architecture rapides pour codes convolutifs en télécommunications: Application aux turbo-codes". PhD thesis, Metz university , France, 2003.

[8] C. Clarck and B. Cain,"Error-Correction Coding for digital communications," Plenum Press, 1983.

[9] R. Pyndiah, “Near optimum decoding of product codes: Block Turbo Codes", IEEE Trans. on Comm., vol 46, n°8, August 1998, pp. 1003-1010.

[10] P. Adde, R. Pyndiah, J.R. Inisan et Y. Sichez. " Conception d’un turbo décodeur de code produit ". Colloque GRETSI’97, Grenoble, France, pp 1003-1010.

[11] M.Elharoussi, A.Fouad, M. Belkasmi , “VHDL Design and FPGA Implementation of Weighted Majority Logic Decoders”, International Conference on Multimedia Computing and Systems (ICMCS’11) proceeding ,pp 397- 401, April 07-09,2011, Ouarzazate, Morocco.

[12] M. Elharoussi, I. Chana, M. Belkasmi "VHDL design and FPGA implementation of a fully parallel BCH SISO decoder", 5th International Symposium on I/V Communications and Mobile Networks (ISIVC), Sept. 30, -Oct. 2, 2010, Rabat, Morocco.

[13] M.Elharoussi, M.Belkasmi, "Implantation FPGA d’un turbo décodeur construit à partir des codes à logique majoritaire pour des applications Haut débit", International Workshop on Codes, Cryptography and Comminuction Systems

(WCCC'11), June 16-17,2011, ENSIAS, Rabat, Morocco(in French).

[14] R. Lucas, M. Bossert and M. Breitbach, "On Iterative Soft-Decision Decoding of Linear Binary Block Codes and Product Codes", IEEE Journal on selected areas in communications, February 1998, Vol. 16, N° 2, pp. 276-296.

[15] R. Lucas, M. P. C. Fossorier, Yu Kou, and Shu Lin, "Iterative Decoding of One-Step Majority Logic Decodable Codes Based on Belief Propagation", IEEE Trans. Commun, Vol. 48, No. 6, pp.931-937, June 2000.

[16] M. Belkasmi, M. Lahmer, "Iterative Threshold Decoding of One-Step Majority Logic Decodable block Codes", ISSPIT Conf, December 15-18, 2007, pp. 668-673 Cairo, Egypt.