FPGA and VLSI Implementation of a Decoding Based on Information-Set

Jefferson Rodrigo Schuertz[1], Sibilla Batista da Luz França[2]
Dept. of Electrical Engineering
Federal University of Paraná
Curitiba, Brazil
jeffersonschuertz@ufpr.br[1], sibilla@eletrica.ufpr.br[2]

***Abstract*—The growing number of mobile devices raised the volume of data in communication systems. To preserve the integrity of data error correction codes (ECC) are used, allowing the identification and correction of errors in the messages transferred or stored. A variety of algorithms for information decoding were studied in the last years. This paper describes a circuit design of two building blocks of a decoder based on information-set, previously proposed in the literature. The decoding algorithm used has a performance like maximum likelihood decoding (MLD), its implementation uses fewer hardware resources. The method comprises extracting the symbols considered being more reliable to construct a reduced set of candidates codewords, decreasing the number of comparisons to recover the message. The first block designed demodulates the received words and sorts the symbols based on its reliability, while the second block creates a matrix (G_{new}), used to generate the candidate codewords. The circuit was first described in VHDL and implemented in a Virtex 5 FPGA, and then, its synthesis and layout were made in the 130 nm technology, using the IC design tools from Cadence.**

Keywords—Error correction codes, Decoder, Information-set, FPGA, VLSI.

I. INTRODUCTION

To some applications are imperative to ensure that the data is transmitted, received, and stored appropriately and integrity. However, channel effects can affect data integrity at some point, whether during storage or transmission [1]. In this scenario, it is common to use error-correcting codes (ECC). These codes were created to improve communication performance, correcting errors arising from events such as interference and noise. To recover the information, eventually corrupted, redundancy bits are inserted to the message during the coding process, allowing recovery in the decoding process, since the number of errors does not exceed the code correction limit [2-3].

The ECC can be divided into two main categories: block codes and convolutional codes [3]. Traditionally, the block codes, covered in this work, are represented as **C**(*n*,*k*), where *n* is the number of bits of the codeword and *k* is the number of bits of the information (with $n - k$ redundancy bits).

A representation of the ECC is shown in figure 1. In the message source, a message **u** is generated and then, it is encoded resulting in **c**. Subsequently, the codeword is transmitted through a noisy channel. The possibly corrupted version of **c** is named **c***. Due to the redundancy bits, the message can be restored by the decoder using a specific algorithm, and, finally, **u**' is delivered to the receiver.

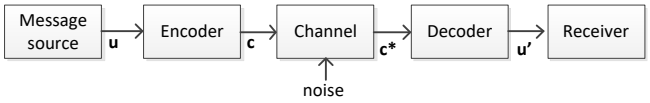

Fig. 1. Communication Channel.

The maximum likelihood method (MLD) is an optimum decoding algorithm. However, it is costly to the many necessary comparisons. Then, alternative algorithms, with performances comparable to MLD, have been extensively studied in the literature [3-11]. Some of these algorithms are based on information-set (IS) [4-11]. The most reliable symbols of the message are used to obtain an information-set and then generate a set of the most likely candidate codewords. The reduced set of candidates codewords allows fewer comparisons to obtain the correct codeword (recovered message) if compared to MLD (2^k comparisons). According to [4], an information-set is any set of *k* linearly independent columns in the generator matrix **G**.

This work presents the hardware implementation of some parts of the decoding algorithm proposed in [6], without stop criterion, targeting FPGA and ASIC. This paper is structured as follows. Section II presents a description of the alternative chosen algorithm. Section III describes the architecture of the building blocks from the decoder. Next, section IV presents the development and obtained results. Finally, section V presents the conclusion.

II. DECODING ALGORITHM

The algorithm presented in [6] uses the most reliable symbols of the received message to build partial matrices, to then generate a set of $k + 1$ candidate codewords. It is wise to demonstrate the operation of the algorithm with the following example: Consider the code **C**(7,4) with the following generator matrix:

$$\boldsymbol{G} = \begin{vmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 \end{vmatrix} \quad (1)$$

The message **u** = [1010] is encoded into a codeword **c**, that is given by multiply **u** by the **G**, resulting in **c** = [1010001].

Considering that codeword **c** is BPSK modulated and transmitted through a noisy channel, the received codeword, possibly corrupted, is called **c***. The values of the received symbols are then quantified in 3 bits. The most reliable are the symbols with values 0 and 7, while the least reliable are the symbols with values 3 and 4. The symbols are classified according to his reliability (level 0 to 3), being 3 the most reliable and 0 the least reliable.

In this example, the codeword received is $\mathbf{c}^*$ = [6, 1, 5, 3, 0, 0, 7]. The first step of the algorithm is hard-decoding, thus $\mathbf{r}$ = [1010001]. A vector called $\mathbf{r}_{partial}$ is obtained selecting the k most reliable symbols of $\mathbf{c}^*$, therefore $\mathbf{r}_{partial}$ = [0100].

The next step is the construction of $\mathbf{G}_{partial}$, consisting of k columns of $\mathbf{G}$. The select columns are 0, 4, 5, and 6, corresponding to those of the k most reliable symbols of $\mathbf{c}^*$ in decreasing reliability order. Thus, $\mathbf{G}_{partial}$ corresponds to the matrix:

$$\boldsymbol{G}_{partial} = \begin{vmatrix} 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{vmatrix} \quad (2)$$

After the construction of $\mathbf{G}_{partial}$, it is necessary to check if this matrix has an inverse matrix. If it does not have, another $\mathbf{G}_{partial}$ matrix must be generated through another combination of the k most reliable columns of $\mathbf{G}$. When the inverse matrix is obtained, it is possible to generate the $\mathbf{G}_{new}$ matrix, through the product between $\mathbf{G}$ and the $\mathbf{G}^{-1}$partial:

$$\boldsymbol{G}_{new} = \begin{vmatrix} 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 \end{vmatrix} \quad (3)$$

The first candidate codeword is obtained by the product of $\mathbf{G}_{new}$ and $\mathbf{r}_{partial}$, thus $\mathbf{c}_0$ = [1010001]. The others k candidate codewords are obtained by the product of $\mathbf{G}_{new}$ and the $\mathbf{r}_{partial}$ (after bit-flipping of each symbol). The last step is to compare each candidate codeword with the received codeword ($\mathbf{c}^*$). The one with the shortest soft-distance is defined as the winning codeword, that is, the recovered message. In this example, the most similar codeword was $\mathbf{c}_0$.

III. Hardware Architecture

To implement the hardware (FPGA and ASIC) of some parts of the algorithm proposed in [6], a hardware architecture for the decoder was proposed, based on another information-set decoder presented in [7, 9].

The hardware architecture of the decoder consists of four blocks, as presented in figure 2. The Block I receives the input (incoming message $\mathbf{c}^*$) and performs the hard-decoding (vector $\mathbf{r}$) and reliability sorter of symbols (vector $\mathbf{s}$). The Block II receives the outputs of Block I and selects a possible partial matrix $\mathbf{G}_{partial}$. The determinant of this matrix is checked, and the process is repeated until a valid $\mathbf{G}_{partial}$ is obtained. Then, the inversion of $\mathbf{G}_{partial}$ is performed by the modified Gauss Jordan elimination. The inverse matrix $\mathbf{G}_{inv}$ is then multiplied by $\mathbf{G}$ resulting in the matrix $\mathbf{G}_{new}$. The $\mathbf{r}_{partial}$ vector is also constructed in this block, based on the information-set positions. The Block III receives the $\mathbf{r}_{partial}$ and generates the $k + 1$ candidate codewords. Block IV selects the best candidate codeword considering the soft-distance between each candidate codeword and the received message ($\mathbf{c}^*$).

This work presents results for blocks I and II. Blocks III and IV are still in development, so far, only the FPGA implementation has been performed.

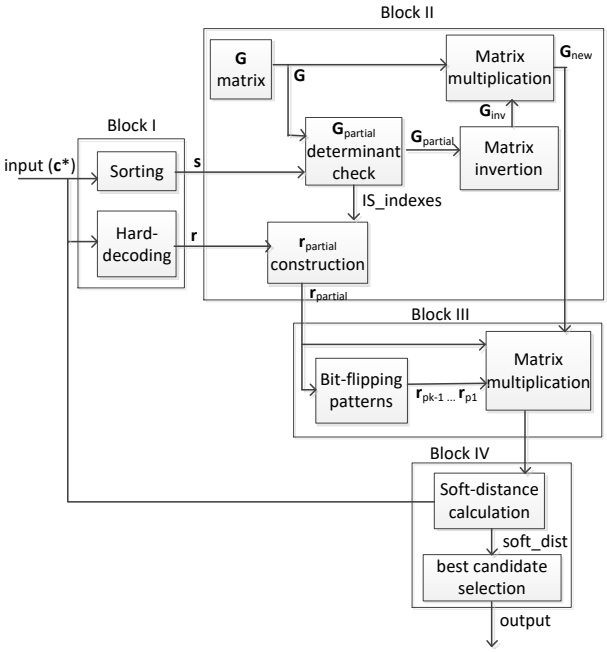

Fig. 2. Hardware Architecture.

IV. Development and Results

A. FPGA design

Block I perform the first steps of the algorithm. In this block, the hard-decision of the received codeword $\mathbf{c}^*$ is performed. Another activity implemented is the sorting of the most reliable symbols. To sort the most reliable symbols in descending order, the classic sorting circuit based on the Insertion Sort algorithm was used [13]. The circuit is presented in figure 3, where d is the symbol to be sorted and $\mathbf{s}$ the sorted output. The $\mathbf{s}$ vector has two pieces of information the reliability level of each symbol and its respective position in the original input vector. The number of cycles required to sort the symbols is equal to the number of elements of the codeword $\mathbf{c}^*$.

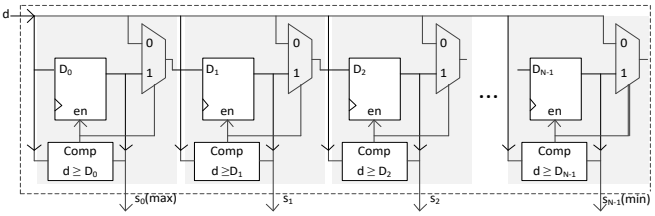

Fig. 3. Sort Insertion Method in Hardware.

The project was implemented using the Xilinx ISE design tool for a Virtex 5 FPGA. The resource usage of Block I for the $\mathbf{C}(7,4)$ code is presented in Table I (no DSP block or BRAM were used).

TABLE I. Block I FPGA Results

Code	LUTs	Registers	Fmax (MHz)	Latency
$\mathbf{C}(7,4)$	40	55	336.86	7 cycles

The simulation results of Block I for the same code are shown in figure 4, whereas the input is $\mathbf{c}^*$ = [7, 5, 4, 3, 2, 1, 0]. The vector $\mathbf{r}$ represents the data obtained with the hard decision, while $\mathbf{s}$ represents the indexes of the ordered confidence vector. The output signal ready indicates when the

vector **s** is completed ordered and, the block signal output is ready to be used by Block II.

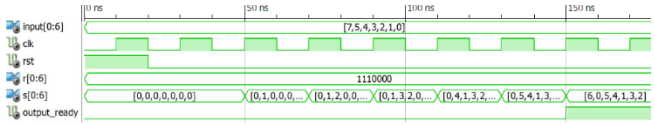

Fig. 4. Block I simulation result.

The Block II starts with the construction of the $\mathbf{G}_{\text{partial}}$, obtained by joining the k most reliable columns of the **G**, they correspond to the first k elements of the vector **s**. Then, $\mathbf{G}_{\text{partial}}$ is verified to know if it is possible to get the inverse, k cycles are consumed to perform the $\mathbf{G}_{\text{partial}}$ reduction. After the elimination process, the determinant is calculated, if the matrix is reversible (determinant equal '1'), Gauss Jordan elimination method is performed, otherwise, more k clock cycles must construct other $\mathbf{G}_{\text{partial}}$ and check if it is reversible. The process to check if $\mathbf{G}_{\text{partial}}$ is reversible has been subdivided into three steps:

1) Adjust the pivot. Check if the pivot $X_{i,j}$ is valid (equal to '1'). If it is not, a row exchange is needed.
2) Check if there are elements different from '0' in the rows below and above the pivot.
3) Perform the XOR operation between the pivot row and the rows identified in the second step.

To perform the next task, that is to obtain the inverted matrix, an improved algorithm of the Gauss Jordan elimination method was used [12]. Others k clock cycles are necessary to complete the inversion operation. The method generates an augmented matrix of order $k \times 2 \cdot k$, where the first k columns are the same columns of $\mathbf{G}_{\text{partial}}$, while the rest are the columns of an identity matrix of order k. All steps are presented below:

1) Adjust the pivot. In this algorithm, the pivot will be always the $X_{1,1}$ element.
2) Check if there are elements different from '0' in the rows below and above the pivot and perform the XOR operation between the pivot row and the rows identified in step 2.
3) Perform the operation called shift left-up. Each element of the matrix must be moved one position to the left and one up.

Following the previously described tasks, the inverse of the matrix $\mathbf{G}_{\text{partial}}$ is obtained and, finally, the $\mathbf{G}_{\text{new}}$ is generated. The resource usage of Block II for **C**(7,4) code is presented in table II. No DSP block or BRAM were used.

TABLE II. BLOCK II FPGA RESULTS

Code	LUTs	Registers	Fmax (MHz)	Latency
C(7,4)	497	64	168.0	≥ 8 cycles

The simulation results of Block II for the same code are presented in figure 5. The inputs consist of the hard-decision vector **r** and the indexes of the ordered confidence vector **s** = [6, 0, 5, 4, 1, 3, 2]. The output $\mathbf{G}_{\text{new}}$ is obtained (for this vector **s**) after eight clock cycles, indicated by the signal output ready.

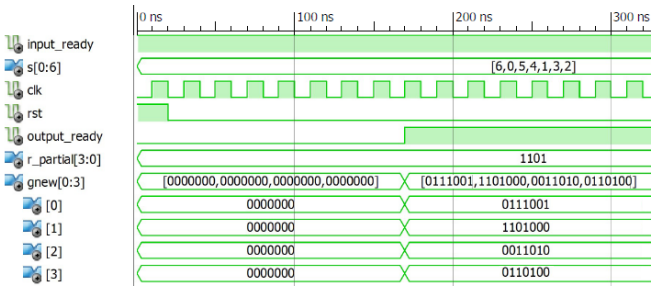

Fig. 5. Block II simulation result.

B. *Logical and Physical Synthesis for BiCMOS 130 nm*

The logical synthesis of Block I and II were performed for the **C**(7, 4) code using the Genus tool from Cadence in the GF BiCMOS 8HP 130 nm technology. In Table III are presented the results about the area, power consumption, and the number of cells for each block. The netlist generated by Genus was simulated in ModelSim and compared with the FPGA simulation results.

TABLE III. LOGICAL SYNTHESIS RESULTS FOR BICMOS 130 NM

Resources	Block I	Block II
Number of Cells	187	698
Area consumed (μm^2)	3007	7143
Block II	1.46	5.70

The physical synthesis of blocks I and II was performed for the **C**(7, 4) code using the Innovus tool from Cadence. Table IV shows the data on the use of resources to build the layout. The layout was verified throw geometry, connectivity, and DRC checks. The post-layout netlist generated was successfully simulated using the ModelSim tool.

TABLE IV. PHYSICAL SYNTHESIS RESULTS FOR BICMOS 130 NM

Entity	Dimensions (µm)	Area (μm^2)	Fmax (MHz)	Density (%)
Block I	148 × 136	19,856	70	78.16
Block II	152 × 182	28,756	50	92.0

V. CONCLUSION

This paper presented an FPGA and VLSI implementation of a part of a soft-decoding algorithm based on information-set for block codes. To implement this algorithm, it was presented a hardware architecture consisting of four blocks. These blocks perform the hard-decision of the received message, extract the reliability of the symbols, create a matrix based on the most reliable symbols, generate a set of most likely candidate codewords and finally determine the best candidate. Results showed that the decoder was efficiently implemented in FPGA and ASIC, consuming a reduced number of logical resources. As future work, it is intended to finish blocks III and IV and join all blocks to obtain the complete decoder in FPGA and ASIC. Additionally, it is intended to synthesize the decoder for different codes sizes.

REFERENCES

[1] B. Sklar, "Digital Communications. Fundamentals and Applications", 2nd ed. Prentice Hall, 2017.

[2] S. Lin and D.J. Costello, "Error Control Coding: Fundamentals and applications", 2nd ed. Prentice Hall, June 2004.

[3] M. Fossorier, S. Lin, "Soft-decision decoding of linear block codes based on order statistics" Transactions on Information Theory, Vol. IT – 4I, No. 5, pp. 13791396, Sep. 1995.

[4] E. Prange, “The use of information sets in decoding cyclic codes” IEEE Transactions on information Theory, Vol. IT-8, pp. 5-9, Sep. 1962.

[5] Dorsch, B. "A decoding algorithm for binary block codes and J-ary output channels (Corresp.)," IEEE Transactions on Information Theory, vol. 20, no. 3, pp. 391-394, May 1974, doi: 10.1109/TIT.1974.1055217. Trans, Vol. IT-20, No. 3, May 1974, pp. 391–4.

[6] Brante, G. G. de O, Godoy W, Muniz, D.; “Information Set Based Soft-Decoding Algorithm for Block Codes”. IEEE Latin America Transactions, Latin-American Conference on Communications, Bogota, 2010.

[7] Gortan, A.; Godoy, W. Jr.; Jasinski, R. P.; Pedroni, V. A. “Achieving Near-MLD Performance with Soft Information-Set Decoders Implemented in FPGAs”. IEEE Asia Pacifc Conference on Circuits and SystemsCircuits and Systems (APCCAS), Kuala Lumpur, 2010.

[8] Jasinski, R. P.; Godoy JR., W.; Gortan, A.; França, S. B. L. ; Pedroni, V. A. . “Efficient Hardware Implementation of Advanced Soft Information-Set Decoders in FPGAs”. WSEAS Transactions on Communications, v. 12, p. 334-351, 2013.

[9] Scholl S. and N. Wehn, “Hardware implementation of a Reed-Solomon soft decoder based on information set decoding”. Design Automation and Test in Europe, December 2013.

[10] Karthik G.R. “Modified error insertion technique for information set based decoders” Conference: Advances in Computing, Communications and Informatics (ICACCI), 2013 International Conference, July 2013.

[11] Versfeld J., Y. Genga, and O. Oyerinde. (2019). “Improved Iterative Convergence Rate for Soft-Decision Bit-Level Reed-Solomon Decoders using Information Set Decoding”. IEEE Wireless Africa Conference (WAC), Pretoria, August 2019.

[12] Pedroni, V. A, Gortan A., Godoy W. Jr. “An Improved GF(2) Matrix Inverter with Linear Time Complexity,” International Conference on Reconfigurable Computing, 2010.

[13] Ribas, L; Castellis, D.; Carrabina, J. “A linear sorter core based on programmable register file”. Conference on Design of Circuits and Integrated Systems (DCIS’04), 2004.