AN EFFICIENT VLSI IMPLEMENTATION FOR THE SOFT INFORMATION-SET DECODING ALGORITHM

Sibilla B. L. França, Ricardo P. Jasinski, Volnei A. Pedroni

UTFPR, Dept. of Electronics Engineering
Curitiba–PR, Brazil

ABSTRACT

Error-correcting codes are present in basically all modern data communications and data storage systems. When top-performance is required, the corresponding algorithms (encoder + decoder) are implemented in hardware. This paper describes a VLSI implementation for an information-set-based error correcting decoder. More specifically, we present the modified Gauss-Jordan elimination block, which constitutes the main block of the ongoing work on the complete soft information-set decoder. The motivation for this work is that such decoder can achieve near-MLD performance with fewer computations, but no VLSI implementation has been developed for it yet.

1. INTRODUCTION

Error-correcting codes are a common method to increase reliability in data transmission and storage operations. By properly encoding the information, errors induced by a noisy channel or medium can be reduced to any desired level [1]. This is made possible by the addition of extra bits, called *redundancy bits* or *parity bits*, so the number of bits at the encoder outputs (n) is always higher than at its inputs (k) [2].

Figure 1 illustrates the use of an error-correcting code in data transmission [2]. Initially, the original message (**u**) is encoded, resulting in a codeword (**c**) with more bits. After the data is transmitted through a noisy channel, the received message (**c***) may be corrupted, but it can be reconstituted if the errors introduced are within the code's correction capability. In Figure 1, an original message with $k = 6$ bits was encoded into a codeword with $n = 9$ bits, of which the 3rd bit was corrupted during transmission. On the decoder side, the original message (**u**) was reconstructed thanks to the redundant bits added by the encoder.

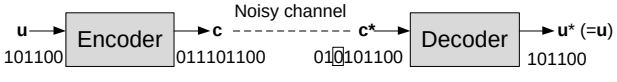

Figure 1. Typical usage of error-correcting codes.

Error-correcting codes can be classified into *block codes* and *convolutional codes*. Block codes operate on a fixed number of bits at a time (called a *message*) and do not require memory from previous decoding operations. Convolutional codes, on the other hand, operate on sequences of data, and each encoded block depends not only on the corresponding k-bit message but also on m previous message blocks [1].

Several approaches have been investigated to address the problem of decoder computational complexity. The use of information sets is one attempt to reduce the computations required to decode a codeword, by considering only a fraction of the total message bits. This approach was introduced by Prange [3], and followed by several researchers [4][5][6]. Due to its relatively simple software implementation, of particular interest is the Dorsch algorithm [7], as well as a variant based on ordered statistics decoding introduced by Fossorier [5].

In this paper, we present an efficient VLSI implementation for this kind of decoder. The proposed circuit is a modified, hardware-friendlier version of the Dorsch algorithm, and was originally introduced in [8].

2. INFORMATION-SET DECODING

Consider a $\boldsymbol{C}(n, k, d_{min})$ linear block code with codewords $\mathbf{c}_i$ ($i = 0$ to 2^k-1), minimum Hamming distance d_{min}, and generator matrix **G** (of size $k \times n$). The encoding procedure consists in multiplying a message vector $\mathbf{u}_i$ (with k bits) by **G** to produce a corresponding codeword $\mathbf{c}_i \in C$ (with n bits). The decoder receives **c***, a possibly corrupted version of $\mathbf{c}_i$, from which it extracts a hard decoded sequence $\mathbf{r}_i$, along with a reliability measure (based on the actual analog value) of each symbol. The latter is needed in order to rank the symbols in $\mathbf{r}_i$ according to their reliabilities, originating the sorting sequence $\mathbf{s}_i$ and thus allowing the use of soft decision.

G consists of k linearly independent columns (usually, the identity matrix $\mathbf{I}_k$) plus $n - k$ columns (linearly dependent on the previous ones) responsible for adding the redundancy. In blocked form, **G** can be represented as $\mathbf{G} = [\mathbf{I} \mid \mathbf{P}]$, where **I** is a $k \times k$ identity matrix and **P** is a $k \times (n - k)$ parity matrix.

An *information set* (IS) is defined as any set of k linearly independent (LI) columns in **G** [3]. Because of the redundancy added by the encoder, it is not necessary to consider all n bits in order to decode a received word. The core principle of soft information set decoding is to use only the k most reliable symbols in **c***, which are used to reconstruct the original source message $\mathbf{u}_0$.

3. HARDWARE-ORIENTED DECODING ALGORITHM

The modified information set decoder with soft decision, based on [5] and [7] but with much simpler computations, is presented in this section. The algorithm

is summarized in Fig. 2 and is briefly described below using a (7, 4) code whose **G** matrix is that in Fig. 2(a).

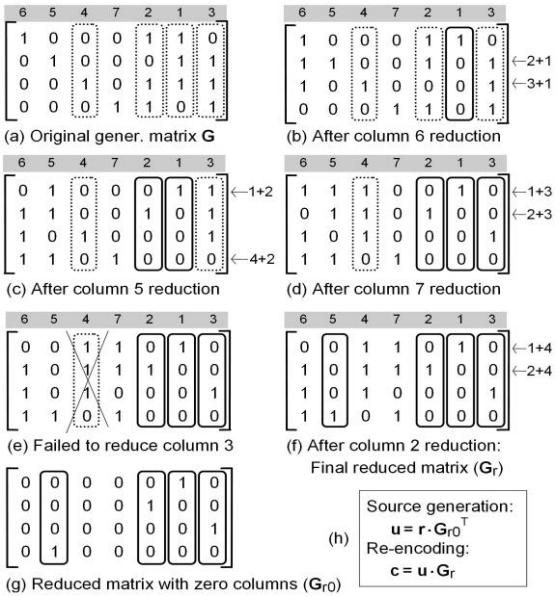

Fig. 2. Modified Gauss-Jordan elimination.

A) Extract from the received codeword the hard decoded sequence $\mathbf{r}_i$ and the corresponding reliability sequence $\mathbf{s}_i$. The values of $\mathbf{s}_i$ are marked at the top of Fig. 2(a), where '1' indicates the most reliable column.

B) Using Gauss-Jordan transformations, reduce the k most reliable columns (MRCs) of **G** to unit vectors. This step is repeated until k LI columns are found, and is illustrated in Figs. 2(b)-(f). Column 6 (the MRC) was reduced in Fig. 2(b), column 5 (the next MRC), in Fig. 2(c), then column 7, in Fig. 2(d). In Fig. 2(e), the algorithm failed to reduce column 3, indicating that the set is not LI. Column 3 was then replaced with column 2 (the next MRC), which was successfully reduced in Fig. 2(f), resulting in a fully reduced matrix $\mathbf{G}_r$.

C) Create the matrix $\mathbf{G}_{r0}$, which is simply $\mathbf{G}_r$ with all unselected columns zeroed. This is shown in Fig. 2(g).

D) Multiply **r** by $\mathbf{G}_{r0}$ to attain the source message (that is, $\mathbf{u}_0 = \mathbf{r} \times \mathbf{G}_{r0}$, as indicated in Fig. 2(h)).

E) Construct the k remaining candidate messages by simply flipping one bit of $\mathbf{u}_0$ at a time (a very simple procedure in hardware). The total number of candidate messages is $k + 1$, represented by $\mathbf{u}_i$ (i = 0 to k).

F) Finally, re-encode each candidate message using $\mathbf{c}_i = \mathbf{u}_i \times \mathbf{G}_r$. Measure the Euclidean distance between each codeword $\mathbf{c}_i$ and **r**, in order to decide the winner.

This algorithm was introduced in [8] and provides a performance comparable to that of maximum-likelihood decoding (MLD), but with considerably less computations. For the **C**(48, 24, 12) code used in that work, the coding gain difference is less than 0.1 dB relative to MLD.

4. HARDWARE ARCHITECTURE

The hardware architecture (Fig. 3) comprises five main blocks: (1) input sorting and demodulation, (2) modified Gaussian elimination (or reduction) of the **G** matrix, (3) candidate messages generation, (4) candidate codewords generation and (5) best candidate selection.

Block (1) receives an analog word **x**, and produces the reliabilities vector **s** and the received word, demodulated in a hard-decision fashion (**r**).

Block (2) performs a modified Gauss elimination on the generator matrix **G**. The elimination proceeds until k LI columns are found, in the order dictated by the reliabilities vector **s**. Block (2) outputs are the reduced matrix $\mathbf{G}_r$ and the transformation matrix $\mathbf{G}_{r0}$, which will extract only those bits in the positions of the selected information set when multiplied by the received word **r**.

Block (3) produces a set of candidate messages for evaluation. First, the received word **r** is multiplied by $\mathbf{G}_{r0}^T$, in order to produce the base candidate $\mathbf{u}_0$. Then, other k candidates are generated by flipping one bit of $\mathbf{u}_0$ at a time, originating the list of candidate messages $\mathbf{U}_C$.

Next, block (4) computes a list of k+1 candidate codewords ($\mathbf{C}_C$), re-encoding the candidate messages by multiplying matrices $\mathbf{U}_C$ and $\mathbf{G}_r$.

Finally, block (5) evaluates all k+1 re-encoded words in order to select the best possible candidate. A soft-distance between each candidate $\mathbf{c}_{cj}$ and the received analog word **x** is calculated as the sum of all individual bit distances. Finally, the codeword with the smallest distance is selected as the output of the decoder.

The modified Gaussian elimination is central to the decoding algorithm, and as such block 2 is presented with more detail in Fig. 4. Matrix $\mathbf{G}_{r0}$ is a simple byproduct of $\mathbf{G}_r$, and was omitted from the diagram.

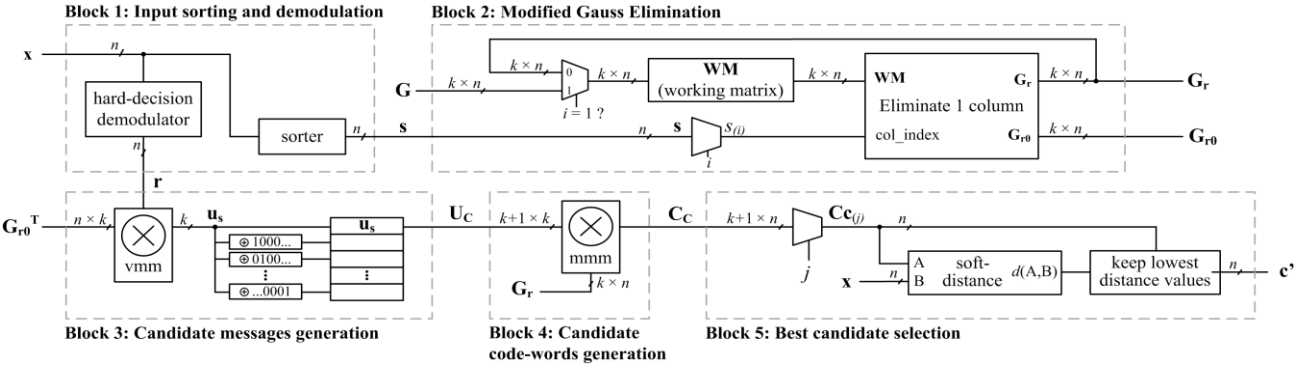

Fig. 3. Hardware block diagram.

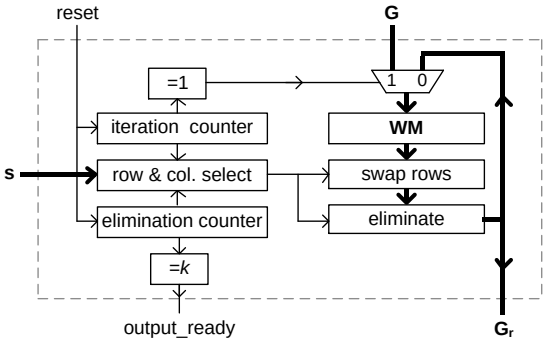

Fig.4. Block 2 implementation diagram.

The elimination process is iterative and controlled by two counters. The *iteration counter* increases at every cycle, and is used as an index into the reliabilities vector **s** (starting with the most reliable element). The *elimination counter* increases only when a successful elimination occurs (i.e., the current column is LI); when its value reaches *k*, the elimination is completed.

Scratch values are maintained in a *working matrix* (**WM**), which is registered at each iteration. **WM** is initially loaded with the generator matrix **G**, on the first processing cycle. The two blocks that follow are purely combinational, and perform the elementary operations applicable to binary matrices [8]: interchanging two rows in order to get a pivot element at a given position, and adding two rows to reduce each column to a unit vector. Following this combinational operations, the matrix is fed back, and registered for the next iteration.

5. VLSI IMPLEMENTATION

A VLSI implementation for the circuit shown in Fig. 3 and corresponding to the algorithm from Section 3 is described next. This part of the work has the following objectives:

(1) To implement a complete information-set decoder with soft-decision capability; to our knowledge, this is the first VLSI implementation for this kind of decoder.

(2) To create a custom library of cells and hardware blocks tailored for the implementation of decoders; this will allow the investigation of tradeoffs and fine tuning of several implementation parameters.

(3) To evaluate the resulting circuit performance, including a comparison with the same circuit synthesized to different FPGA families. The FPGA version was implemented in VHDL, and is described in [8].

Our VLSI implementation uses a 0.5µm CMOS SCM3M_SUBM technology with 5V supply. At this moment, we have concluded the creation of the standard cells library, as well as the implementation of block 2, responsible for the modified Gauss-Jordan elimination.

Starting with the VHDL description, the first step was translating the RTL code into logic gates. Although a logic synthesis tool (such as Cadence Encounter – RTL Compiler) would be recommended for this task, one of our goals was to develop a better understanding of the steps involved in the entire process. Thus, for each circuit block, the following tasks were performed:

(1) The VHDL description was translated into hardware diagrams, one diagram for each VHDL process.

(2) Each diagram was implemented in terms of simple gates and registers. These diagrams are used to create circuit schematics based on transistors, some of which are shown in Figs. 5(a) and 5(c). As can be seen, all circuits with the exception of the flip-flops use CMOS logic. This is a critical design step, since it involves the definition of each component configuration. For example, one of the most crucial decisions concerned the flip-flop to be employed in the circuits. Due to its single-clock-phase, small size, and glitch free operation, the GF-TSPC (glitch free true single-phase clock) flip-flop [9] was chosen (Fig. 5(c)).

(3) The corresponding spice code was written and simulated for each circuit from step (2), in order to confirm their correct behavior and timing response.

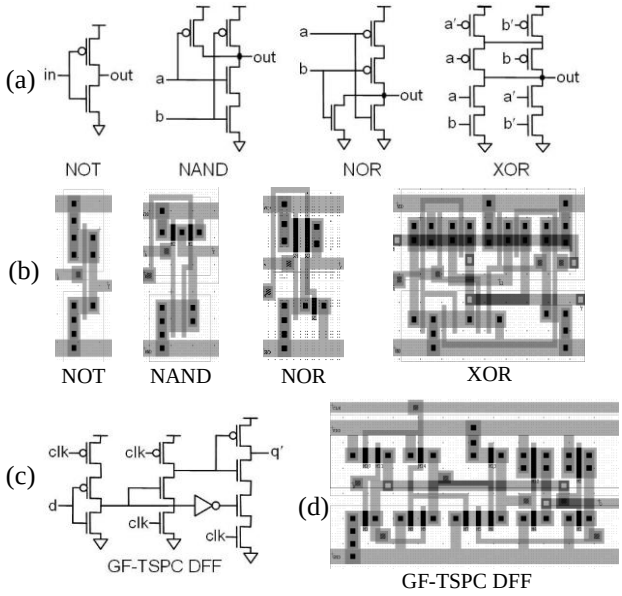

Figure 5. Standard cells: (a) Logic gates schematic; (b) Logic gates layout; (c) GF-TSPC schematic; (d) GF-TSPC layout.

(4) A gates and registers layout library was created, containing the project's standard cells (logic gates, flip-flops, multiplexers, adders, counters, comparators, etc.). Some examples can be seen in Figs. 5(b) and 5(d).

(5) The low-level circuits were placed and routed according to the diagrams obtained in steps (1) and (2). The resulting circuit (Fig. 6) was verified one more time, by performing a layout versus schematic comparison.

6. RESULTS

The complete layout of the Gaussian elimination circuit is shown in Fig. 6, with the main parts of the algorithm outlined. As can be seen, a large area is used by the combinational logic that performs the modified Gaussian elimination (regions (*b*) and (c)). Additionally, each output matrix ($\mathbf{G_r}$ and $\mathbf{G_{r0}}$) occupies a significant area, due to the flip-flops required to store each element value.

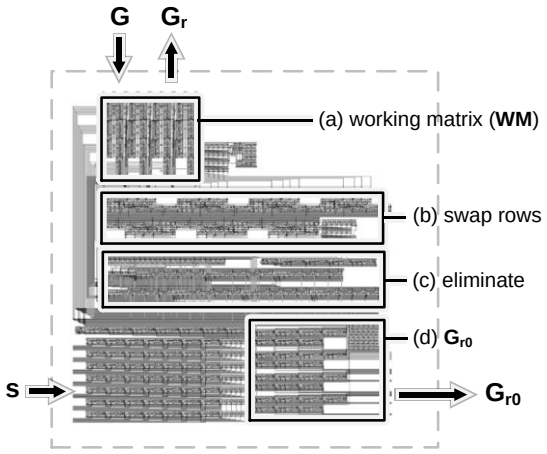

Figure 6. Layout of modified Gauss-Jordan block.

Spice simulations were performed to verify the correct circuit behavior. Fig. 7 shows the results for a Gaussian elimination using the same input values as those in Fig. 2. The four lower waveforms show two columns of **WM**; column 5 is represented with a dashed line, while column 6 appears as a solid line. Their original values are 1011 (col. 5) and 1110 (col. 6), which is in accordance with Fig. 2(a). After column 6 is eliminated, the new values are 1101 (col. 5) and 1000 (col. 6), corresponding to Fig 2(b). Next, after eliminating column 5, the new values are 0100 (col. 5) and 1000 (col. 6). The fact that each column became a unit vector indicates that the elimination worked as expected. Also, it can be seen that each column remains stable for the rest of the simulation, which is also consistent with the expected behavior.

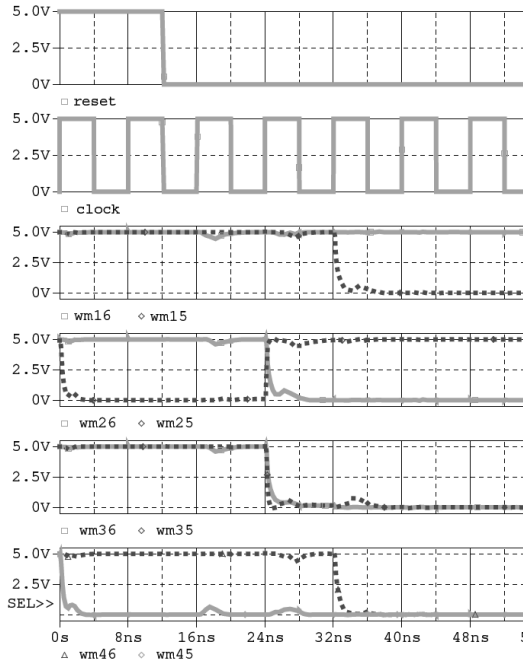

Figure 7. Simulation results for the modified Gaussian elimination (first two columns).

Despite having completed only block 2 so far, its construction demanded the definition, testing, and layout of several standard cells, which will be used also in the other blocks, so the work is at a much more advanced stage than it might appear at first. The silicon area used by this block is approximately 0.98 mm^2. Considering that in the FPGA implementation this block corresponds to 48% of the decoder area, we can estimate that the complete implementation will require about 2.04 mm^2.

In order to determine the maximum operating frequency of the circuit, a series of simulations were performed varying the clock period until a practical limit was found, and below which the decoder functioning was not reliable. It was determined that the circuit can operate correctly up to a frequency of 147 MHz. Comparing these results with those obtained in [8] for an FPGA of the Stratix III family (159.1 MHz), we can see that the implemented circuit, despite employing a low-end technology (0.5 μm), presents a performance comparable to that of a high-end FPGA (within 8%).

7. CONCLUSION

We have presented a VLSI implementation for the Gauss-Jordan elimination circuit, which is the main component of the target information-set-based decoder. First, a library of standard cells (for the entire project) was developed, followed by the layout and test of the entire block. The total block area (in 0.5 μm CMOS) is 0.98 mm^2, with a maximum operating frequency of 147 MHz, comparable to that reported for a high-end FPGA.

8. REFERENCES

[1] S. Lin, J. J. Costello, *Error Control Coding: Fundamentals and Applications*, Prentice-Hall, USA, 1983.

[2] V.A. Pedroni, *Digital Electronics and Design with VHDL*, Morgan Kaufmann, USA, 2008.

[3] E. Prange, "The use of information sets in decoding cyclic codes," IRE Trans. on Information Theory, Vol. IT-8, pp. 5-9, Sep. 1962.

[4] J. Coffey, R. Goodman, "The complexity of information set decoding," IEEE Trans. on Information Theory, Vol. IT-36, No. 5, pp. 1031-1037, Set. 1990.

[5] M. Fossorier, S. Lin, "Soft-decision decoding of linear block codes based on order statistics," IEEE Trans. on Information Theory, Vol. IT-41,No. 5, pp. 13791396, Sep. 1995.

[6] M. Fossorier, "Reliability-based soft-decision decoding with iterative information set reduction," IEEE Trans. on Information Theory,Vol. IT-48, No. 12, pp. 3101-3106, Dec. 2002.

[7] B. G. Dorsch, "A decoding algorithm for binary block codes and j-ary output channels," IEEE Trans. on Information Theory, Vol. IT-20, pp. 391-394, May 1974.

[8] A. Gortan ; W. Godoy Jr. ; R. P. Jasinski , V. A. Pedroni, "Hardware-Friendly Implementation of Soft Information Set Decoders," IEEE Int. Telecomm. Symp. (ITS), Manaus, 2010.

[9] Q. Huang, "Speed optimization of edge-triggered nine-transistor D-flip-flops for gigahertz single-phase clocks," IEEE Int. Symp. on Circuits and Systems, pp. 2118-2121, May 1993.