Hardware-Friendly Implementation of Soft Information Set Decoders

A. Gortan, R. P. Jasinski, W. Godoy Jr., V. A. Pedroni
UTFPR, Dept. of Electronics Engineering
Curitiba – PR, Brazil

***Abstract*— This article has two main purposes: (i) to introduce and evaluate a modified version of the Dorsch soft decoding algorithm for block codes, based on information sets, and (ii) to examine the complexity of implementing this kind of decoder in hardware (more specifically, in FPGA devices). The modifications introduced in the algorithm lead to an optimized circuit size when the algorithm is implemented in hardware. Indeed, the physical implementation and its detailed analysis represent a major departure from traditional decoder analysis, increasingly important as more frequently such specialized functions are embedded into the hardware, in top-performance systems. It is demonstrated that, in the worst case, $n - d_{min} + 1$ iterations suffice to find the most reliable information set, from which only a very small fraction of all 2^k possible candidate codewords needs to be tested to achieve near maximum likelihood decoding (MLD) performance. These conclusions are confirmed by simulations on a C(48, 24, 12) block code, then used as guidelines in the physical implementations of several decoders, in order to test their real-time operation. A detailed circuit diagram is presented, along with experimental results indicating the number of logic cells and registers needed to implement the decoder in a high-end FPGA, for various code sizes.**

Keywords - soft-decision; information set decoding; Dorsch algorithm; hardware implementation; FPGA; VHDL.

I. INTRODUCTION

The use of information sets for decoding linear block codes was first proposed by Prange [1], followed by several other researchers [2]-[12], leading to an extensive family of related decoding techniques. Their common goals are to reduce the number of candidate codewords, to obtain better candidates, and to reduce the code's computational complexity.

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

$\mathbf{G}$ 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, $\mathbf{G}$ can be represented as $\mathbf{G} = [\mathbf{I} \mid \mathbf{P}]$, where $\mathbf{I}$ is a $k \times k$ identity matrix and $\mathbf{P}$ is a $k \times (n - k)$ parity matrix. An information set (IS) is defined as any set of k linearly independent (LI) columns in $\mathbf{G}$ [5].

Due to its relatively simple software implementation, of particular interest is the Dorsch algorithm [4], as well as a variant based on ordered statistics decoding introduced by Fossorier [7-8]. A modified, hardware-friendlier version of the former is introduced, which greatly reduces the circuit size, making real-time applications feasible. A detailed analysis of its time complexity is presented, which shows that the number of trials to find the most reliable information set is bounded by $n - d_{min} + 1$, therefore a small number. It is also shown that, once the IS has been found, the decoding of any codeword requires only an order-1 search about the most reliable information word to achieve near MLD performance, thus limiting the total search space to just $k + 1$ candidates, which is another very small number. These conclusions are confirmed by simulations on the extended binary quadratic residue (48, 24, 12) block code presented in [13], and then used as guidelines in the physical implementations of the decoders in an FPGA device, in order to test their real-time operation.

II. HARDWARE-FRIENDLIER INFORMATION SET DECODER

This section introduces the modified version of the Dorsch algorithm, which allows a more compact hardware implementation.

The core of information set decoding with soft decision can be roughly described as follows.

A) Extract, from the received codeword, the hard decoded sequence $\mathbf{r}$ and the corresponding reliability sequence $\mathbf{s}$.

B) Based on $\mathbf{s}$, select the k most reliable symbols in $\mathbf{r}$ and disregard the remaining $n - k$ symbols.

C) Re-encode the k most reliable symbols using a new generator matrix $\mathbf{G}_n$, derived from the original $\mathbf{G}$ and equivalent to it, but with unit columns in the k most reliable positions.

One way of obtaining $\mathbf{G}_n$ is by inverting the matrix formed by the k elected columns of the original $\mathbf{G}$, then multiplying the result by $\mathbf{G}$. A major problem in this procedure is that not all sets of k columns from $\mathbf{G}$ are LI, so inversion might not be possible. In such a case, another set of k symbols is chosen, and the process is repeated until k LI columns (an IS) are found.

Another approach, with much simpler computations and for which a guaranteed small search space is demonstrated, is

described next. The generator matrix is manipulated using Gauss-Jordan transformations, which can reduce any row or column to a unit vector. The algorithm is summarized in Fig. 1 and briefly described below, with a (7, 4) code used as an example, whose generator matrix **G** is shown in Fig. 1(a).

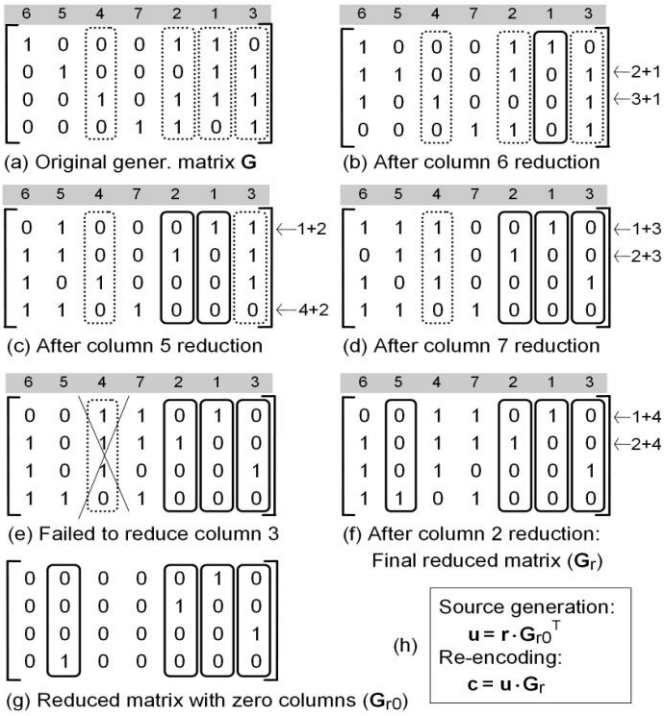

Figure 1. Modified information set decoding algorithm.

A) Extract from the received codeword the hard decoded sequence **r** and the corresponding reliability sequence **s**. Note that the rank values ($\mathbf{s}_i$) are marked at the top of Fig. 1(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. Even though there is no guarantee that the k MRCs are LI, the process does not need to be restarted; just replace the least reliable among the MRCs with the next MRC and proceed from there. This is illustrated in Figs. 1(b)-(f). Column 6 (the MRC) was reduced in Fig. 1(b), column 5 (the next MRC), in Fig. 1(c), then column 7, in Fig. 1(d). In Fig. 1(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. 1(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 (Fig. 1(g). Although this matrix is not strictly necessary for the decoding procedure, it eases the hardware implementation, since it can be used to extract from **r** only the bits in the positions of the selected information set.

D) Multiply **r** by $\mathbf{G}_{r0}^T$ to attain the source message (that is, $\mathbf{u}_0 = \mathbf{r}\times\mathbf{G}_{r0}^T$, as indicated in Fig. 1(h)).

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

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

It will be demonstrated in the next section that the number of iterations to find the most reliable IS in the proposed algorithm is never larger than $n - d_{min} + 1$, and that only a small fraction of all the 2^k possible candidate codewords are indeed sufficient to produce practical MLD decoding performance.

III. Algorithm Analysis and Demonstrations

Three fundamental questions will be discussed and answered in this section:

A) What is the maximum number of columns in **G** that must be inspected until an information set is guaranteed to be found?

B) What is the maximum number of columns in **G** that picked randomly are guaranteed to be LI?

C) What is the likelihood of needing to run the maximum number of trials derived in (A) until an IS is found?

A. Maximum number of columns in **G** *that must be inspected until an IS is guaranteed to be found*

This can be answered using theorem 1.4.15 of [11], which demonstrates that any set with $n - d_{min} + 1$ columns from **G** is guaranteed to contain an IS. Another proof can be obtained as follows. Say that **u** is an information word, so its corresponding codeword is

$$\mathbf{c} = \mathbf{u}\cdot\mathbf{G}. \tag{1}$$

This codeword can be rearranged in another codeword $\boldsymbol{c}_r$, with all z zeros of $\boldsymbol{c}$ in the initial z positions and the $n - z$ ones in the final $n - z$ positions. Rearranging then **G** in the same way, we obtain $\mathbf{G}_r$, which obviously obeys

$$\mathbf{c}_r = \mathbf{u}\cdot\mathbf{G}_r. \tag{2}$$

The codeword $\mathbf{c}_r$ can be written as the concatenation of an all-zero vector **a** and an all-one vector **b**, that is $\mathbf{c}_r = [\mathbf{a} \mid \mathbf{b}]$. Likewise, $\mathbf{G}_r$ can be constructed with the concatenation of a z-column matrix **A** with another $(n–z)$-column matrix **B**. Therefore, (3) can be written as

$$[\mathbf{a}_z \mid \mathbf{b}_{n-z}] = \mathbf{u}\cdot[\mathbf{A}_{k\times z} \mid \mathbf{B}_{k\times(n-z)}], \tag{3}$$

or, equivalently,

$$\mathbf{a}_z = \mathbf{u}\cdot\mathbf{A}_{k\times z} = \mathbf{0}, \tag{4}$$
$$\mathbf{b}_{n-z} = \mathbf{u}\cdot\mathbf{B}_{k\times(n-z)}. \tag{5}$$

From (4), we conclude that $rank(\mathbf{A}) < k$, because there is a set of rows from **A**, given by **u**, whose sum is zero. Since the row and columns ranks are alike, we conclude that among the z columns of **A** there cannot be k LI columns. If we then take a codeword with z zeros and select the corresponding z columns of **G**, a set of z columns without

an information set will be attained. Since the largest value of z is

$$z_{max} = n - d_{min}\,, \tag{6}$$

equation (6) represents the size of the largest set without an IS. Consequently, the maximum (worst case) number of iterations (*NI*) is given by:

$$NI_{max} = n - d_{min} + 1\,. \tag{7}$$

B. Maximum number of columns from G that selected randomly are still guaranteed to be LI

This is the *heft* of **G** (the largest value of t such that any set of t columns from **G** are LI). The heft of the parity matrix **H** of a code is known to be $d_{\min} - 1$. Since **G** is in turn the parity matrix of its dual code, then $heft(\mathbf{G}) = d_{\min}^{\perp} - 1$, where $d_{\min}^{\perp}$ is the minimum Hamming distance of the dual code (in the particular case of self dual codes, this value is the same for both, that is, $d_{\min}$). For example, for the (48, 24, 12) code, $heft(\mathbf{G}) = 11$, so any 11 columns from **G** in this code are guaranteed to be LI.

The values of $heft(\mathbf{G})$ for some well known codes are listed in Table I, which also exhibits the largest number of columns that is guaranteed to contain an IS (that is, $n - d_{\min} + 1$, as determined in section III-A).

TABLE I. MINIMUM NUMBER OF LI COLUMNS AND MAXIMUM NUMBER OF COLUMNS NEEDED TO OBTAIN AN IS

Code	$d_{\min}$	$heft(\mathbf{G})$	$n - d_{\min} + 1$
(7, 4, 3)	3	3	5
(15, 7, 5)	5	3	11
(23, 12, 7)	7	7	17
(24, 12, 8)	8	7	17
(48, 24, 12)	12	11	37

The knowledge of $heft(\mathbf{G})$ is important because it tells that the first $d_{\min}^{\perp} - 1$ most reliable columns of **G** will never have to be replaced during the process of obtaining an IS. Thus any column replacement, should it be needed, will happen from column $d_{\min}^{\perp}$ to column $n - d_{\min}$. Taking again the (48, 24, 12) code as an example, which is a self-dual code, its 11 MRCs can be automatically assigned to the IS. Since in this case an IS requires 24 LI columns, the 12th MRC is then tested, then the 13th MRC, and so on, until 24 LI columns are finally obtained. However, this procedure will never require inspection beyond the 37th MRC. Furthermore, as will be illustrated in section III-C, the probably of attaining an IS (that is, k LI columns) in the first k columns is high (~34% in the present example), with the probability rapidly surpassing 99% with the inspection of just a few additional columns beyond the k initial ones.

C. Likelihood of inspecting the maximum number of columns derived in (A) before an IS is found

We have demonstrated that the number of columns from **G** that might be needed to inspect until an IS is found lies between k and $n - d_{\min} + 1$. For the (48, 24, 12) code, it ranges from 24 to 37, although simulations performed on this code show that the maximum number is very unlikely to be needed.

In order to evaluate the number of columns inspected by the decoder, a sequence of randomly generated codewords subjected to an additive white Gaussian noise channel was applied to its input. Simulation results from 10^6 iterations on the (48, 24, 12) code are plotted in Fig. 2. As can be observed, in 34% of the cases an IS was found using just the first k (=24) most reliable columns. With just one more MRC, 64.5% of the cases were covered. Note also that, with 30 columns, the success rate was 99.54%. Hence, as expected, the number of trials is generally small, with the maximum number rarely needed (zero occurrences in the present simulation with 10^6 iterations). Finally, it is important to note that, even if the maximum number of columns were needed, it would still be a reasonably low, manageable value.

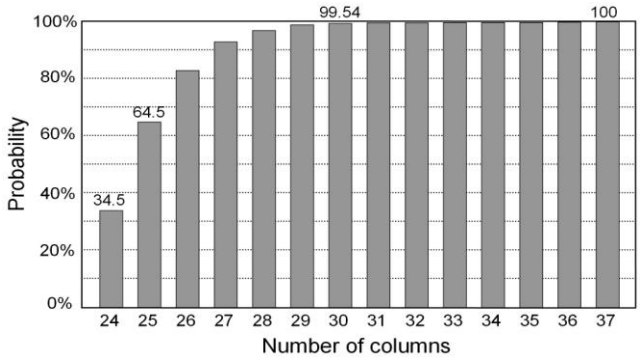

Figure 2. Probability of finding an IS as a function of the number of columns inspected for the (48,24,12) code, based on simulation results from 10^6 iterations.

IV. HARDWARE IMPLEMENTATION

The algorithm described in Section III was implemented as a hardware description in VHDL language, and synthesized to an FPGA in order to evaluate the cost and performance of a physical implementation. The devised hardware architecture (Fig. 3) comprises 5 main blocks: (1) input sorting and demodulation, (2) modified Gauss elimination on the **G** matrix, (3) candidate messages generation, (4) candidate codewords generation and (5) best candidate selection.

The implemented VHDL code is completely generic, allowing for easy experimentation on the configurable parameters. Among the parameters that can be configured at compile time are the generator matrix **G** and the quantization levels of the input analog word.

Block (1) receives an analog word **x** transmitted through the channel and produces two outputs: the reliabilities vector **s**, and the received word demodulated in a hard-decision fashion (**r**). The demodulation process is trivial, and consists in inspecting the sign bit (most significant bit) of each binary-encoded input symbol. Vector **s** is generated by a linear insertion sorter, based on the architecture described in [14]. Since the analog values are ordered as they are shifted into the sorter, block (1) outputs become available after n clock cycles.

The reliabilities vector **s** is promptly used as an input for block (2), which performs a modified Gauss elimination on the generator matrix **G**. Instead of sequentially eliminating the matrix columns from left to right, the processing order is dictated by the reliabilities vector **s**. The elimination steps proceed until k linearly independent columns are found. Since it is not always true that the k most reliable columns are linearly independent, the block outputs will be available somewhere between k and $n - d_{min} + 1$ clock cycles, as demonstrated in item III. Block (2) outputs are the resulting matrix $\mathbf{G_r}$ (derived from **G** after k successful elimination steps) and the transformation matrix $\mathbf{G_{r0}}$, which can be multiplied by the received word **r** in order to extract only those bits in the positions of the selected information set.

Block (3) produces a set of k+1 candidate messages, based on the received word **r** and the information set selected in block (2). First, the received word is multiplied by $\mathbf{G_{r0}}^T$, in order to produce a candidate message $\mathbf{u_0}$. Then, another k candidate messages will be generated, by flipping each bit of $\mathbf{u_0}$ one at a time. This process is entirely combinational, and so the list of candidate messages $\mathbf{U_C}$ is generated in one clock cycle.

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 $\mathbf{G_r}$. This process is also combinational, and takes place in one clock cycle.

Finally, block (5) evaluates all of the 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 described in [10]: for each received symbol represented as a 3-bit quantized level r, the bit distance is 7-r to a code bit value '1', and r to a code bit value '0'. The total distance between these two words is the sum of all individual bit distances. After evaluating this measure for each candidate, the codeword with the smallest distance to the received analog word is selected as the output of the decoder. Each candidate word is evaluated in one clock cycle, and as such the total processing time for block (5) is k+1 clock cycles.

The amount of time required to decode one incoming word can be calculated by summing up the clock cycles required by each block. Since block (2) processing can take a variable amount of clock cycles, we have:

$$\#\ of\ cycles_{\ MIN} = n + 2k + 3 \tag{8}$$

$$\#\ of\ cycles_{\ MAX} = 2n - d_{min} + k + 4 \tag{9}$$

It should be noted that although each received word takes a significant amount of clock cycles to be completely decoded, the implemented architecture operates in a pipelined way, and as such the average throughput is dictated by the number of clock cycles required by the most demanding stage (block 2), which is given in (7).

V. Results

A. Code Simulation Results

Simulations performed on C(24, 12, 8) and C(48, 24, 12) codes showed that the proposed algorithm indeed attains near MLD decoding performance. It was also observed the reduction of code gain as the number of candidates are reduced, starting from $k + 1$ (complete set) down to $k - d_{\min}^{\perp} + 1$ (the latter corresponds to the limit derived in part B of Sec. III). The omitted candidates are those obtained from inversion of the most reliable bit, so less likely to be incorrect.

The 13 curves obtained for the C(48, 24, 12) code are exhibited in Fig. 4. They were measured for $k + 1$ (= 25) candidates, then k candidates, next $k - 1$, and so on, down to $d_{\min}^{\perp} - 1$ (= 11) candidates. As expected, the verified gain reduction with respect to MLD for the complete set was very small (under 0.1 dB). Also, as expected, this number grew progressively as the number of candidates was reduced, reaching 0.85 dB when there were only 13 candidates left.

B. Hardware Results

The VHDL description was synthesized to a high-end Altera Stratix III FPGA (EP3SL70F780C2) for several code sizes, as shown in Table II. The correctness of the hardware implementation was confirmed by an exhaustive testbench simulation of the C(7,4) code, which yielded correct outputs for all 2^{21} possible input values (7 input symbols in 3-bit encoding).

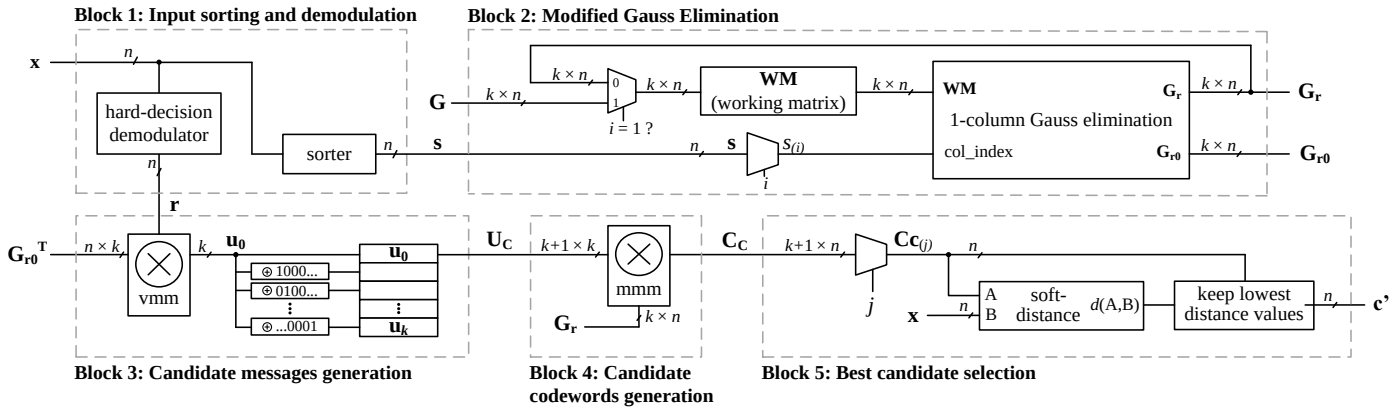

Figure 3. Decoder hardware diagram.

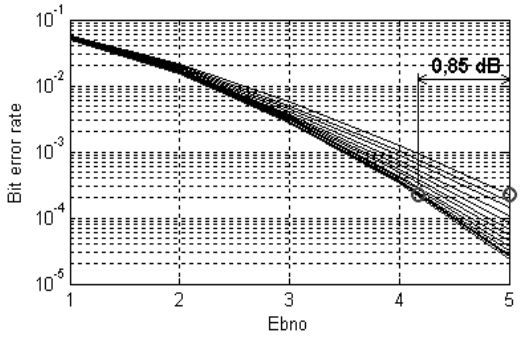

Figure 4. Comparison with MLD decoding and coding gain reduction for a smaller number of candidates.

TABLE II. SYNTHESIS RESULTS

Code	Registers	ALUTs	f_{MAX} (MHz)	Latency (cycles)	ttd_{MAX} (cycles)	Throughput (Mbps)
C(7,4,3)	291	443	159.1	19	5	127.3
C(15,7,5)	838	1,214	111.1	36	11	70.7
C(24,12,8)	1,954	3,273	84.2	56	17	59.4
C(48,24,12)	6,808	10,295	55.5	112	37	36.0
C(66,33,12)	12,387	17,933	50.1	157	55	30.0
C(78,39,14)	16,972	26,639	41.4	185	65	24.8

Regarding silicon area usage, it can be seen that logic resources utilization (look-up tables and registers) increases almost linearly (Fig. 5) with the product $n \times k$, which represents the generator matrix dimensions. It should be noted that even the large C(78,39,14) code fits in the smallest Stratix III device, indicating that the hardware implementation is highly area-efficient.

As for the timing, even though the latency to decode the first word can be large, once the pipeline is full the worst-case time-to-decode (ttd_{MAX}) is significantly shorter. For the C(78,39) code, the first output is available after 185 cycles, but subsequent codewords will be output in at most 65 clock cycles. The difference between the minimum and maximum number of cycles was small for all evaluated codes, with a largest value of 16.3% for the C(78,32,14) code. However, since the performance of the decoder is dominated by stage (2), this is equivalent to a 40% variation in throughput.

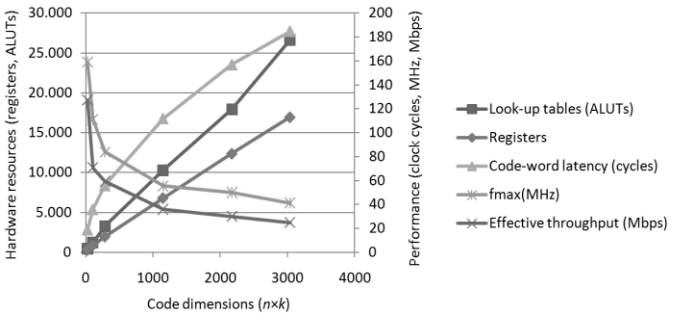

Figure 5. Performance and hardware resources usage for different code sizes.

Overall, the decoder performance can be better summarized by its effective throughput, defined as the number of information bits output by the decoder per second. Even for large codes, the hardware presents a good performance, with an effective throughput of 30 Mbps for the C(66,33,12) code.

VI. CONCLUSIONS

We have introduced and examined an efficient soft decoding algorithm for block codes based on information sets. It was demonstrated that with just $k + 1$ candidate codewords, easily generated in hardware with order-1 bit inversion, near MLD performance is achieved (simulation results confirmed a result less than 0.1 dB from MLD in a C(48, 24, 12) code). It was also shown that an IS is guaranteed to be found in at most $n - d_{min} + 1$ iterations (worst case), hence with very low time complexity, proper for real-time implementations.

The hardware version of the proposed algorithm was proven to exhibit the exact same behavior of the original software implementation. It was shown to be highly area-efficient in FPGAs; even the large C(78,39,14) code fits in the smallest device in the Stratix III family. For this code, the decoder was able to process an input word in at most 65 clock cycles, yielding an effective throughput of 24.8 Mbps.

REFERENCES

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

[2] G. D. Forney, Jr., "Generalized minimum distance decoding," IEEE Transactions on Information Theory, Vol. IT-12, No. 2, pp. 125-131, April 1966.

[3] D. Chase, "A class of algorithms for decoding block codes with channel measurement information," IEEE Transactions on Information Theory, Vol. IT-18, No. 1, pp. 170-182, Jan. 1972.

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

[5] G. Clark, J. Cain, Error-Correction Coding For Digital Communication, Plenum Press, 1981.

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

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

[8] M. Fossorier, S. Lin, J. Snyders, "Reliability-based syndrome decoding of linear block codes," IEEE Transactions on Information Theory, Vol. IT-44, No. 1, pp. 388-398, Jan. 1998.

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

[10] P. Sweeney, Error Control Coding From Theory to Practice, Wiley, 2002.

[11] W. Huffman, V. Pless, Fundamentals of Error-Correcting Codes, Cambridge University Press, 2003.

[12] W. Godoy, Jr., E. Wille, "A simple acceptance criterion for binary block codes soft-decision algorithms," Adv. Int. Conf. on Telecommunications and Int. Conf. on Internet and Web Applications and Services (AICT/ICIW), 2006.

[13] M. Jimbo, K. Shiromoto, "A construction of mutually disjoint Steiner systems from isomorphic Golay codes", Journal of Combinatorial Theory, Series A, Vol. 116, pp 1245–1251, Oct. 2009.

[14] L. Ribas, D. Castells, J. Carrabina, "A linear sorter core based on a programmable register file", XIX Conference on Design of Circuits and Integrated Systems - DCIS, pp. 635-640, France, 2004.