Parallel Generation of Most Reliable LLRs of a Non-Binary Symbol

Hassan Harb, Ali Chamas Al Ghouwayel, Emmanuel Boutillon

Parallel Generation of Most Reliable LLRs of a Non-Binary Symbol

Hassan Harb, Ali Chamas Al Ghouwayel, *Member, IEEE* and Emmanuel Boutillon, *Senior Member, IEEE*

***Abstract*—The first task of Non-Binary decoders (LDPC or Turbo) is to generate the Log-Likelihood Ratio (LLR) of the received Non-Binary symbols defined over Galois Fields GF(q >2). In the Extended Min-Sum decoding algorithm, the intrinsic information associated to a given received symbol is a sorted list of the n_m most reliable Non-Binary GF symbols along with their associated reliability values. In this paper we present a fully parallel LLR generation algorithm, an enabler for very high throughput decoding, that processes one received symbol per clock cycle. We provide complexity figures for LLR architectures designed over GF(q) of sizes 64, 256 and 1024, as well as different values of n_m. Compared to the related state of the art architecture, FPGA synthesis results show that the proposed parallel architecture improves the hardware efficiency by a factor ranging from 7 up to 15.**

***Index Terms*—NB-LDPC codes, EMS algorithm, LLR value.**

I. Introduction

Binary Low Density Parity Check (LDPC) [1] codes and Binary Turbo-Codes [2] have been adopted in many communication standards such as WiMAX, WiFi, DVB-C, DVB-S2X, DVB-T2, among others. Nevertheless, with short codewords, performance of binary codes start to degrade compared to theoretical achievable performance. A solution to mitigate this problem is to replace binary codes by Non-Binary (NB) codes defined over GF(q), $q = 2^m$ [3] [4]. NB-LDPC codes are efficient for short and moderate codeword lengths [3]. This high error correction capability is obtained thanks to higher girths which is an inherent feature of NB-LDPC codes. These characteristics position NB-LDPC codes as serious competitors of classical binary LDPC and Turbo-Codes in future wireless communication and digital video broadcasting standards. However, this competitive edge does not come for free; it entails high computational complexity making their hardware implementation a very challenging task. In [5] and [6], the authors propose the Extended-Min Sum (EMS) algorithm to decode NB-LDPC code with a reduced complexity. The principle of the EMS is to truncate the LLR messages associated to a symbol from q (the Galois Field size) down to n_m ($n_m \ll q$). The Bubble Check algorithm [7] reduces the CN complexity from $\mathcal{O}(q^2)$ down to $\mathcal{O}(4n_m)$.

The first step in the EMS algorithm is the generation of the n_m most reliable candidate GF(q) symbols. Feeding the decoder with n_m symbols in one clock cycle is a necessity for highly parallel processing-based decoder architectures. In this paper, we propose a parallel LLR generation architecture able to generate a predefined number, n_m, of symbols at each clock cycle in the specific case where the channel generates binary LLRs (Binary Phase Shift Keying (BPSK) modulation for example). Compared to the state of the art [8], the proposed architecture offers a hardware efficiency gain factor ranging from 7 to 15.

The rest of this paper is organized as follows. Section II reviews the basic notions and definitions related to the LLR computation. Section III discusses the proposed parallel architecture and Section IV presents the synthesis results. Finally, Section V concludes the paper.

Hassan Harb is with the Université de Bretagne Sud - Lab-STICC, UMR 6285 CNRS - Lorient, France, and Lebanese University - Faculty of Sciences, Hadat, Beirut, Lebanon.

Ali Chamas Al Ghouwayel is with the School of Engineering, Lebanese International University (LIU) and International University of Beirut (BIU), Beirut, Lebanon.

Emmanuel Boutillon is with the Université de Bretagne Sud - Lab-STICC, UMR 6285 CNRS - Lorient, France.

II. Definition of the Log-Likelihood Ratio

In this paper, we consider the transmission of Non-Binary codeword over GF(q), $q = 2^m$, using a BPSK modulation through an Additive White Gaussian Noise (AWGN) Channel of noise variance σ^2. A GF symbol X of the codeword is represented by a binary vector of size m, i.e., $X = (x_0, x_1, ..., x_{m-1})$. Each coordinate of this vector is modulated as $B(x_i) = (-1)^{x_i}$, $i = 0, 1, \ldots, m-1$, and transmitted over the AWGN channel. At the receiver side, the received message is $R = (r_0, r_1, \ldots, r_{m-1})$, where $r_i = B(x_i) + w_i$ and w_i, $i = 0, 1, \ldots, m-1$ are independent realizations of a Gaussian variable $\mathcal{N}(0, \sigma)$. Following [9], and knowing the received vector R and that the GF symbols are equiprobable, we define the LLR $L(X)$ of a GF symbol X as

$$L(X) = \log\left(\frac{P(R|\bar{X})}{P(R|X)}\right), \tag{1}$$

where $\bar{X}$ represents the hard decision over R, i.e. $\bar{x_i} = 1$ if $r_i < 0$, 0 otherwise, $i = 0, 1, \ldots, m-1$. The expression (1) can be developed as

$$L(X) = \sum_{i=0}^{m-1} \log\left(P(r_i|\bar{x}_i)\right) - \log\left(P(r_i|x_i)\right). \tag{2}$$

Since in the AWGN channel, $P(r|x) = \frac{1}{\sqrt{2\pi}\sigma} e^{\frac{-(r-B(x))^2}{2\sigma^2}}$, (2) reduces to

$$L(X) = \frac{1}{2\sigma^2} \sum_{i=0}^{m-1} (r_i - B(x_i))^2 - (r_i - B(\bar{x}_i))^2. \tag{3}$$

Thus,

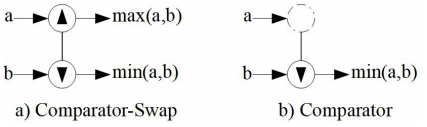

Fig. 1: Comparator-Swap (CS) and Comparator (C).

$$L(X) = \sum_{i=0}^{m-1} \frac{2}{\sigma^2} |r_i| \Delta(\bar{x}_i, x_i), \tag{4}$$

with $\Delta(a,b) = 0$ if $a = b$, 1 otherwise. In the following, the LLR value $\frac{2}{\sigma^2} r_i$ in (4) will be denoted by y_i. Note that the decision value $\bar{y}_i$ on y_i remains equal to $\bar{x}_i$.

Example: let $Y = (-6, +9, -2, +12, +11, -7)$ be the received quantized LLR values of a symbol over GF(64). Then $\bar{Y} = (1, 0, 1, 0, 0, 1)$. Let $X_0 = (0, 0, 0, 0, 0, 0)$, then $L(X_0) = 6 + 2 + 7 = 15$, since X_0 differs from $\bar{Y}$ in positions 1, 3 and 6. The $n_m = 5$ best possible decisions, based on Y, will be $\mathcal{U} = \{U_i\}_{i=0,1,2,3,4}$, with $U_i = (U_i^{\oplus}, U_i^{+})$ a couple composed of a GF value $U_i^{\oplus}$ and its associated LLR $U_i^{+} = L(U_i^{\oplus})$. In this example, $U_0 = ((1, 0, 1, 0, 0, 1), 0)$, $U_1 = ((1, 0, 0, 0, 0, 1), 2)$, $U_2 = ((0, 0, 1, 0, 0, 1), 6)$, $U_3 = ((1, 0, 1, 0, 0, 0), 7)$ and finally $U_4 = ((0, 0, 0, 0, 0, 1), 8)$.

III. Proposed architecture

This section describes the new proposed parallel architecture that generates the first n_m most reliable LLR values with their associated GF symbols in parallel. The key idea is to work in 3 steps. The first step is to replace the received unstructured set of LLR $Y = (y_0, y_1, \dots y_{m-1})$ by an equivalent structured set $S^+ = (s_0^+, s_1^+, \dots, s_{m-1}^+)$ containing the absolute values of Y sorted in ascendant order. The second step is to find the set $\hat{\mathcal{I}}$ of the n_m smallest LLR values along with their associated GF values from the structured set S^+. Finally, the last step is to transform the set of solutions $\hat{\mathcal{I}}$ of the structured problem back to the set of solutions $\mathcal{U}$ of the initial problem. In order to ease the reading of the paper, GF symbol and vector in the structured domain will be noted with a hat.

A. From unstructured to structured channel observations

The first step is then to sort the absolute values of the received LLR $|y_i|$, $i = 0...m-1$, using the odd-even sorting algorithm[1] [10] of size m. Fig. 1.a) shows the considered symbol to refer to a Comparator-Swap (CS) while the one shown in Fig. 1.b) refers to a simple Comparator (C). Fig. 2 shows the fully parallel pipelined sorter that receives the absolute values $|y_i|$, sorts them, and generates the couples $s_i = (s_i^+, \pi(i))$, $i = 0, \dots, m-1$, so that $\pi(i) \in \{0, \dots, m-1\}$, $s_i^+ = |y_{\pi(i)}|$ and $0 \le s_0^+ \le s_1^+ \le \cdots \le s_{m-1}^+$. The permutation π is used later in step 3 to transform the solution of the structured problem back to the solution of the initial problem.

With the example of previous section, from the set of binary LLR $Y = (-6, +9, -2, +12, +11, -7)$, we obtain the set $S^+ = (2, 6, 7, 9, 11, 12)$ and the associated permutation $\pi = (2, 0, 5, 1, 4, 3)$.

[1]Other types of sorting algorithm can also be selected.

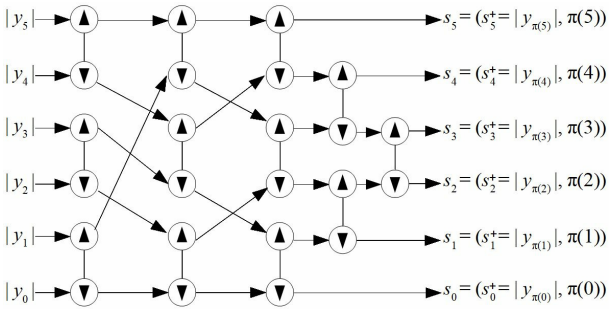

Fig. 2: Sorter architecture of the observed bits.

B. Solution of the structured problem

The first step is to compute the LLR values associated to a predefined set of symbols $\hat{\mathcal{J}}^{\oplus}$, where $\hat{\mathcal{J}}^{\oplus}$ is defined offline in such a way that the n_m most reliable intrinsic symbols for any S^+ are always computed. The design of $\hat{\mathcal{J}}^{\oplus}$ takes advantage of the fact that the LLR values of S^+ are positive and sorted. Then, the LLR values $\hat{\mathcal{J}}^{+}$ of $\hat{\mathcal{J}}^{\oplus}$, computed from S^+, are sorted in increasing order and the set of solutions $\hat{\mathcal{I}} = (\hat{\mathcal{I}}^{\oplus}, \hat{\mathcal{I}}^{+})$ of the structured problem is simply extracted as the smallest n_m values of $\hat{\mathcal{J}}^{+}$.

1) Offline construction of $\hat{\mathcal{J}}^{\oplus}$: Let $\hat{A}^{\oplus}$ and $\hat{B}^{\oplus}$ be two vectors of size m and S^+ a vector of m positive ordered bit-LLR values. $\hat{A}^{\oplus}$ is said to dominate $\hat{B}^{\oplus}$ if and only if, among all possible realizations of S^+, $L(\hat{A}^{\oplus}) \le L(\hat{B}^{\oplus})$. For example, with $m = 3$, $\hat{A}^{\oplus} = (1, 1, 0)$ dominates $\hat{B}^{\oplus} = (1, 0, 1)$ because $L(\hat{A}^{\oplus}) = s_0^+ + s_1^+$ while $L(\hat{B}^{\oplus}) = s_0^+ + s_2^+$ and, by construction, $s_1^+ \le s_2^+$, thus $\hat{A}^{\oplus}$ will be always selected before $\hat{B}^{\oplus}$. The notion of dominance can be defined formally by the existence of an injective function π between the set of indices $\phi_{\hat{A}^{\oplus}}$ of the positions of 1s in $\hat{A}^{\oplus}$ and the set of indices $\phi_{\hat{B}^{\oplus}}$ of positions of 1s in $\hat{B}^{\oplus}$ so that, for all $e \in \phi_{\hat{A}^{\oplus}}$, $\pi(e) \ge e$. The explicit construction of π is not described due to lack of space but it can be simply constructed offline using a greedy algorithm.

The relation of dominance gives only a partial order over the set of binary vectors of size m. For example, $\hat{C}^{\oplus} = (0, 0, 1)$ and $\hat{D}^{\oplus} = (1, 1, 0)$ can give either $L(\hat{C}^{\oplus}) > L(\hat{D}^{\oplus})$ or $L(\hat{C}^{\oplus}) < L(\hat{D}^{\oplus})$ (for example, with $S^+ = (2, 3, 12)$ and $S^+ = (2, 3, 4)$, respectively). For given values of m and n_m, it is possible to generate formally the set $\hat{\mathcal{J}}_m^{\oplus}(n_m)$ of all vectors of size m that are dominated by at most n_m vectors (note that a vector is dominated by itself). Thus, for any realization of S^+, the set $\hat{\mathcal{J}}_m^{\oplus}(n_m) = \{\hat{J}_0^{\oplus}, \hat{J}_1^{\oplus}, \dots, \hat{J}_{n_J-1}^{\oplus}\}$ of cardinality n_J, is guaranteed to contain the n_m m-binary vectors associated to the n_m smallest LLR values. Note that the proposed notion of dominance is already presented in a different context in [11] to define the position of frozen bits in the construction process of polar codes.

Table I shows all the elements of the set $\hat{\mathcal{J}}_6^{\oplus}(12)$ that constitute all the possible candidates needed to extract $n_m = 12$ symbols over GF(64). In this case, the set $\hat{\mathcal{J}}_6^{\oplus}(12)$ is of cardinality $n_J = 17$.

Figure 3 shows the evolution of n_J as a function of n_m for

TABLE I: The $n_J = 17$ elements of $\hat{\mathcal{J}}_6^{\oplus}(12)$, associated $\Phi_{\hat{J}^{\oplus}}$ sets, literal expressions of $\hat{J}^+$ and Numerical Application (NA) for $S^+ = (2, 6, 7, 9, 11, 12)$.

i	$\hat{J}_i^{\oplus}$	$\phi_{\hat{J}_i^{\oplus}}$	$\hat{J}_i^+ = L(\hat{J}_i^{\oplus})$	NA
0	$(0,0,0,0,0,0)$	$\emptyset$	0	$\hat{J}_0^+ = 0$
1	$(1,0,0,0,0,0)$	$\{0\}$	s_0^+	$\hat{J}_1^+ = 2$
2	$(0,1,0,0,0,0)$	$\{1\}$	s_1^+	$\hat{J}_2^+ = 6$
3	$(0,0,1,0,0,0)$	$\{2\}$	s_2^+	$\hat{J}_3^+ = 7$
4	$(0,0,0,1,0,0)$	$\{3\}$	s_3^+	$\hat{J}_4^+ = 9$
5	$(0,0,0,0,1,0)$	$\{4\}$	s_4^+	$\hat{J}_5^+ = 11$
6	$(0,0,0,0,0,1)$	$\{5\}$	s_5^+	$\hat{J}_6^+ = 12$
7	$(1,1,0,0,0,0)$	$\{0,1\}$	$s_0^+ + s_1^+$	$\hat{J}_7^+ = 8$
8	$(1,0,1,0,0,0)$	$\{0,2\}$	$s_0^+ + s_2^+$	$\hat{J}_8^+ = 9$
9	$(0,1,1,0,0,0)$	$\{1,2\}$	$s_1^+ + s_2^+$	$\hat{J}_9^+ = 13$
10	$(1,1,1,0,0,0)$	$\{0,1,2\}$	$s_0^+ + s_1^+ + s_2^+$	$\hat{J}_{10}^+ = 15$
11	$(1,0,0,1,0,0)$	$\{0,3\}$	$s_0^+ + s_3^+$	$\hat{J}_{11}^+ = 11$
12	$(0,1,0,1,0,0)$	$\{1,3\}$	$s_1^+ + s_3^+$	$\hat{J}_{12}^+ = 15$
13	$(1,1,0,1,0,0)$	$\{0,1,3\}$	$s_0^+ + s_1^+ + s_3^+$	$\hat{J}_{13}^+ = 17$
14	$(0,0,1,1,0,0)$	$\{2,3\}$	$s_2^+ + s_3^+$	$\hat{J}_{14}^+ = 16$
15	$(1,0,0,0,1,0)$	$\{0,4\}$	$s_0^+ + s_4^+$	$\hat{J}_{15}^+ = 13$
16	$(1,0,0,0,0,1)$	$\{0,5\}$	$s_0^+ + s_5^+$	$\hat{J}_{16}^+ = 14$

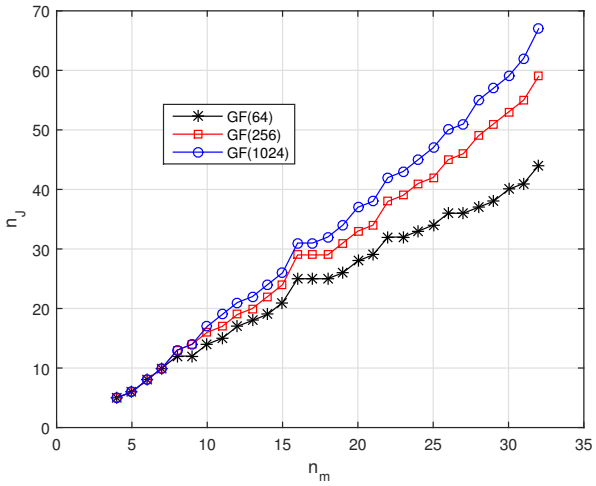

Fig. 3: n_J versus n_m in case of $m = 6$, 8 and 10 (GF(64), GF(256) and GF(1024) respectively).

several values of m. It is important to note that n_J is greater than n_m, but less than $2n_m$ for $m \leq 10$, i.e., it increases almost linearly with n_m for practical values of n_m.

2) Computation of the list of LLR values $\hat{\mathcal{J}}^+$: Based on S^+ and the set $\hat{\mathcal{J}}^{\oplus}$, the n_J values of $\hat{\mathcal{J}}^+$ are computed using a set of adders wired according to the addition of s_i^+s indicated in the third column of Table I. The next step is to sort the predefined potential candidates $\hat{\mathcal{J}}_i^+ = \{\hat{J}_i^+\}_{i=0,1,\ldots,n_J-1}$, to generate the list of n_m sorted LLRs $\hat{\mathcal{I}} = \{(\hat{I}_k^{\oplus}, \hat{I}_k^+)\}_{k=0,\ldots,n_m-1}$.

3) Sorting of the potential candidates: The sorting process is first presented for the case where $m = 6$ and $n_m = 12$, then the method is generalized for any m and n_m values.

The first three outputs are always $\hat{I}_0 = \hat{J}_0$, $\hat{I}_1 = \hat{J}_1$ and $\hat{I}_2 = \hat{J}_2$. For the remaining 9 outputs, we propose the sorter architecture shown in Fig. 4. The sorter receives the 14 remaining elements $\hat{J}_i$, $i = 3, \ldots, 16$, and extracts the first 9 sorted symbols having the smallest LLR values. The relation of dominance gives some a priori information on the relative order of the elements of $\hat{J}$. In fact, for a given couple (a,b), if $\hat{J}_a^{\oplus}$ dominates $\hat{J}_b^{\oplus}$, then by definition, $\hat{J}_a^+ \leq \hat{J}_b^+$. This a priori information can be exploited to reduce the number of comparators and multiplexers. Therefore, the 14 elements are split up into 5 sets based on the LLR value as: $\{\hat{J}_3^+ \leq \hat{J}_4^+ \leq \hat{J}_5^+ \leq \hat{J}_6^+\}$, $\{\hat{J}_7^+ \leq \hat{J}_8^+ \leq \hat{J}_9^+ \leq \hat{J}_{10}^+\}$, $\{\hat{J}_{11}^+ \leq \hat{J}_{12}^+ \leq \hat{J}_{13}^+\}$, $\{\hat{J}_{14}^+\}$ and $\{\hat{J}_{15}^+ \leq \hat{J}_{16}^+\}$.

TABLE II: Upper bound of N_{cs} and N_l to extract $\hat{\mathcal{I}}$ from $\hat{\mathcal{J}}$ for different values of n_m and GF(q).

	n_m	4	8	12	16	20	24	28	32
GF(64)	n_J	5	12	17	25	28	33	37	44
	N_l	1	7	7	11	11	13	13	16
	N_{cs}	1	13	28	103	125	147	162	205
GF(256)	n_J	5	13	19	29	33	41	49	59
	N_l	1	8	12	12	14	14	16	16
	N_{cs}	1	47	82	118	147	181	207	247
GF(1024)	n_J	5	13	21	31	37	45	55	67
	N_l	1	8	16	18	18	20	20	22
	N_{cs}	1	47	167	203	283	348	399	444

The sorter architecture is based on the odd-even algorithm [10]. It is composed of $N_l = 7$ layers of CSs. The overall complexity is $N_{cs} = 28$ CSs (CSs and Cs are not differentiated) with a critical path of $T = N_l \times T_{CS} = 7 \times T_{CS}$, where T_{CS} is the critical path of one CS.

Going back to the previous example, the obtained set $\hat{\mathcal{I}}$ is equal to the $n_m = 12$ couples (see Table I): $\hat{\mathcal{I}} = \{(\hat{J}_0^{\oplus}, 0)$, $(\hat{J}_1^{\oplus}, 2)$, $(\hat{J}_2^{\oplus}, 6)$, $(\hat{J}_3^{\oplus}, 7)$, $(\hat{J}_7^{\oplus}, 8)$, $(\hat{J}_4^{\oplus}, 9)$, $(\hat{J}_8^{\oplus}, 9)$, $(\hat{J}_5^{\oplus}, 11)$, $(\hat{J}_{11}^{\oplus}, 11)$, $(\hat{J}_6^{\oplus}, 12)$, $(\hat{J}_9^{\oplus}, 13)$, $(\hat{J}_{15}^{\oplus}, 13)\}$.

To generate the sorting architecture that extracts $\hat{\mathcal{I}}$ from $\hat{\mathcal{J}}$ in the general case, we propose a generic method. It consists in starting with an odd-even sorter of size $2^{\lceil \log_2(n_J) \rceil}$, i.e., the lowest power of 2 greater than or equal to n_J, then pruning each CS receiving pre-ordered inputs or having unused outputs. Table II summarizes the sorter complexity for several values of n_m and different Galois fields. Note that the mapping between the elements of $\hat{\mathcal{J}}$ and the inputs of the odd-even sorter impacts significantly the overall complexity. Finding the optimal mapping is still an open question. When defined, this optimal mapping may lead, for some GF and n_m configurations, to lowering the complexity figures of N_{cs} and/or N_J proposed in Table II.

C. Inverse permutation of the GF values of $\hat{\mathcal{I}}$

The computation of the intrinsic message $\mathcal{U}$ from $\hat{\mathcal{I}}$ is straightforward. The couples $U_k = (U_k^{\oplus}, U_k^+)$, $k = 0, 1, \ldots, n_m - 1$ are obtained as $U_k^{\oplus} = (\hat{I}_k^{\oplus}(\pi^{-1}(i)) \oplus \bar{Y}(i))_{i=0,1,\ldots,m-1}$ and $U_k^+ = \hat{I}_k^+$. Following the previous example, $\pi = (2, 0, 5, 1, 4, 3)$ gives $\pi^{-1} = (1, 3, 0, 5, 4, 2)$. For $k = 0$, $U_0 = (\bar{Y}, 0)$. For $k = 1$, $\hat{I}_1^{\oplus} = (1, 0, 0, 0, 0, 0)$, then $(\hat{I}_1^{\oplus}(\pi^{-1}(i)))_{i=0,1,\ldots,m-1}$ is equal to $(0, 0, 1, 0, 0, 0,)$,

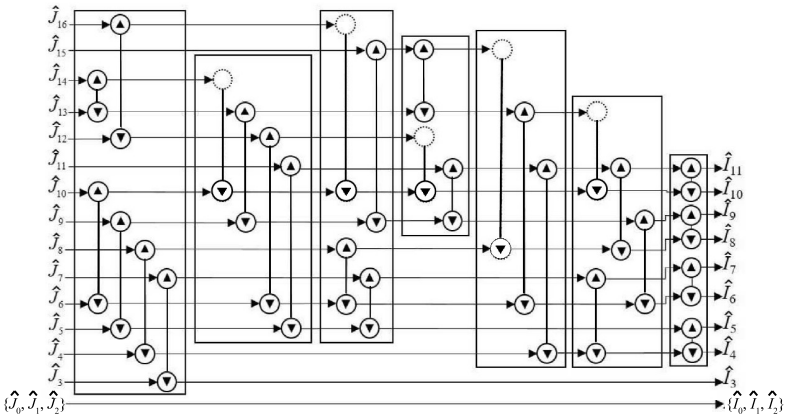

Fig. 4: Sorter Architecture generating the most reliable n_m intrinsic LLRs, GF(64), $n_m = 12$, $n_J = 17$.

TABLE III: Synthesis results of the GF(64)-LLR Generator on kintex 7 FPGA device, xc7k325t -2 fbg676.

n_m	Systolic [8]			Proposed			
	n_m/n_c	F	C	n_m/n_c	F	C	HER
4	0.8	250	322	4	530	222	15.37
12	0.923	250	322	12	500	1084	7.73

thus $U_1^{\oplus} = (0,0,1,0,0,0) \oplus (1,0,1,0,0,1) = (1,0,0,0,0,1)$. Since $U_1^{+} = \hat{I}_1^{+} = 2$, then $U_1 = ((1,0,0,0,0,1),2)$. The same process is repeated to obtain U_k, $k = 2, \ldots, n_m - 1$.

IV. Complexity analysis

This section presents the implementation results on a Xilinx Kintex7 (xc7k325t -2 fbg676) FPGA device, of the LLR generator over GF(64), $n_m = 4$ and 12. It is easy to modify the proposed architecture to fit all the cases of $n_m < 12$ since it is a matter of removing some CSs.

Table III compares the hardware cost of the proposed and the systolic one [8] in terms of Look Up Tables (LUTs). The complexity of the systolic architecture is constant since the number of stages is fixed and equal to m. The only thing that changes with n_m is the size of the FIFOs implemented in each stage which does not impact significantly the overall complexity of the systolic architecture. In order to compare the efficiency of both architectures, we evaluated a metric called Hardware Efficiency (HE). HE is defined as the number of sorted symbols per second per LUT, i.e., HE = $(n_m/n_c) \times F_{clk}/C$, where n_c denotes the number of clock cycles required to generate the n_m outputs, F_{clk} the clock frequency (in MHz) and C the number of LUTs required by the hardware with an FPGA implementation. For the proposed parallel architecture, $n_c = 1$, while in [8], n_c is equal to $n_m + 1$. To better illustrate the efficiency comparison, we have evaluated the HE Ratio (HER) defined as the ratio of the HE of the proposed architecture to the HE of [8]. The most right column of Table III shows that the proposed architecture outperforms the systolic architecture by an order of magnitude.

V. Conclusion

The paper has presented the design and implementation of a parallel low-hardware cost LLR generator. Theoretical complexity and performance analysis of the proposed architecture compared to the systolic architecture have been addressed. For any size of n_m, the proposed architecture requires the lowest area and offers the highest frequency, where a hardware efficiency gain ranging from 7 up to 15 is obtained. The architecture has been developed and presented in the context of non binary codes. It could be used also for high speed Chase algorithm [12].

References

[1] R. Gallager, "Low-density parity-check codes," Ph.D. dissertation, Cambridge, 1963.

[2] C. Berrou, A. Glavieux, and P. Thitimajshima, "Near Shannon limit error-correcting coding and decoding: Turbo-codes. 1," in *Proceedings of ICC '93 - IEEE International Conference on Communications*, vol. 2, May 1993, pp. 1064–1070 vol.2.

[3] M. C. Davey and D. J. C. MacKay, "Low density parity check codes over GF(q)," in *1998 Information Theory Workshop (Cat. No.98EX131)*, June 1998, pp. 70–71.

[4] C. Berrou, M. Jezequel, C. Douillard, and S. Kerouedan, "The advantages of non-binary turbo codes," in *Proceedings 2001 IEEE Information Theory Workshop (Cat. No.01EX494)*, Sep. 2001, pp. 61–63.

[5] D. Declercq and M. Fossorier, "Decoding Algorithms for Nonbinary LDPC Codes Over G(q)," *IEEE Transactions on Communications*, vol. 55, no. 4, pp. 633–643, April 2007.

[6] A. Voicila, D. Declercq, F. Verdier, M. Fossorier, and P. Urard, "Low-complexity decoding for non-binary LDPC codes in high order fields," *IEEE Transactions on Communications*, vol. 58, no. 5, pp. 1365–1375, May 2010.

[7] E. Boutillon and L. Conde-Canencia, "Bubble check: a simplified algorithm for elementary check node processing in extended min-sum non-binary ldpc decoders," *Electronics Letters*, vol. 46, no. 9, pp. 633–634, April 2010.

[8] A. A. Ghouwayel and E. Boutillon, "A Systolic LLR Generation Architecture for Non-Binary LDPC Decoders," *IEEE Communications Letters*, vol. 15, no. 8, pp. 851–853, August 2011.

[9] V. Savin, "Min-max decoding for non binary LDPC codes," in *2008 IEEE International Symposium on Information Theory*, July 2008, pp. 960–964.

[10] A. Farmahini-Farahani, H. J. D. III, M. J. Schulte, and K. Compton, "Modular Design of High-Throughput, Low-Latency Sorting Units," *IEEE Transactions on Computers*, vol. 62, no. 7, pp. 1389–1402, July 2013.

[11] G. He, J. Belfiore, I. Land, G. Yang, X. Liu, Y. Chen, R. Li, J. Wang, Y. Ge, R. Zhang, and W. Tong, "Beta-expansion: A theoretical framework for fast and recursive construction of polar codes," in *GLOBECOM 2017 - 2017 IEEE Global Communications Conference*, Dec 2017, pp. 1–6.

[12] A. Kabat, F. Guilloud, and R. Pyndiah, "On the Sensibility of the "Arranged List of the Most a Priori Likely Tests" Algorithm," in *MILCOM 2007 - IEEE Military Communications Conference*, Oct 2007, pp. 1–7.