A Systolic LLR Generation Architecture For Non-Binary LDPC Decoders

Ali Al Ghouwayel, Emmanuel Boutillon
Université Européenne de Bretagne, UBS, Lab-STICC CNRS
56100 Lorient, FRANCE
Tel: +33-2-9787-4566, Fax: +33-2-9787-4500
E-mail: emmanuel.boutillon@univ-ubs.fr

***Abstract*—Non-Binary LDPC codes offer higher performances than their binary counterpart but suffer from higher decoding complexity. A solution to reduce the decoding complexity is the use of the Extended Min-Sum algorithm. The first step of this algorithm requires the generation of the first n_m largest Log-Likelihood Ratio (LLR), sorted in increasing order, of each received symbol. In the case where GF(q) symbols are transmitted using a BPSK modulation, we propose a simple systolic architecture that generates the sorted list of symbols.**

I. Introduction

Non-binary Low Density Parity Check (NB-LDPC) codes over GF(q), with $q > 2$, are known to outperform binary LDPC codes for short and medium length [1]. Moreover, for high spectral efficiency modulation, channel symbol can be mapped directly into code symbol, avoiding the information loss generated by the marginalisation process used in the binary LDPC case (gain of 1.6 dB, in the MIMO case, are reported in [2]).

These advantages of such a Non-Binary coding scheme come at the expense of increased hardware complexity. However, a recent decoding algorithm called Extended Min-Sum (EMS) Algorithm was proposed by Declercq [3] which presents a significant reduction of the computation complexity of the decoding process while keeping a good performance. This algorithm, instead of treating the q LLR values associated to the complete list of GF(q) symbols, considers the sorting of the LLR messages and only treats the n_m greatest values and proposes an offset compensating the truncated messages.

One of the key components of the NB-LDPC decoder implementing the EMS algorithm is the LLR generation circuit and its accompanying sorter. This paper considers the design and the hardware implementation of an efficient circuit dedicated to generate the LLR values sorted in increasing/decreasing order.

The rest of this paper proceeds as follows. In section II, we review some basic notions and definitions related to the LLR computation and we define a new simplified formula allowing the generation of the LLR values in an ordered list. In Section III, we discuss the proposed sorting algorithm. Section IV describes the hardware design and gives a complexity evaluation of the proposed LLR generation circuit implemented on Virtex 4 FPGA devices. Finally, a conclusion ends the paper.

This work is supported by INFSCO-ICT-216203 DAVINCI "Design And Versatile Implementation of Non-binary wireless Communications based on Innovative LDPC Codes [www.ict-davinci-code.eu] funded by the European Commission under the Seventh Framework Programme (FP7).

II. Definition of the Log-Likelihood Ratio

This section describes the algorithm implemented in the LLR circuit which provides an ordered list of the smallest n_m LLR values and their associated GF(q) symbols from the q binary LLR values received from the channel.

Let $X = (x_0\ x_1\ ...\ x_{m-1})$ be a GF($q = 2^m$) symbol composed by $m = \log_2(q)$ binary symbols and let $Y = (y_0\ y_1\ ...\ y_{m-1})$ be the symbol received from the channel so that $Y = (y_i = B(x_i) + w_i)_{i=0,..,m-1}$, where w_i is a realization of a white Gaussian noise of variance σ^2 and $B(x) = 2x-1$ is the BPSK modulation that associates symbol -1 to bit $x = 0$ and +1 to bit $x = 1$. For an Additive White Gaussian Noise channel with a noise of variance σ^2, $\ln(P(Y|X))$ is given by:

$$\ln(P(Y|X)) = \ln(\prod_{i=0}^{m-1} P(y_i|x_i))$$

$$= m\ln(\frac{1}{\sqrt{2\pi}\sigma}) - \sum_{i=0}^{m-1}\Big(\frac{(y_i - B(x_i))^2}{2\sigma^2}\Big) \quad (1)$$

Let $\tilde{X}$ be the symbol of GF(q) that maximizes $\ln(P(Y|X))$, i.e. $\tilde{X} = \{\arg\max_{X\in GF(q)}, P(Y|X)\}$. Using equation (1), $\tilde{X}$ is given by $\tilde{X} = (\tilde{x}_i = D(y_i))_{i=0..m-1}$, where D denotes the Hard Decision on y, i.e. $D(y) = 0$ if $y < 0$, $D(y) = 1$ otherwise.

With the hypothesis that the GF(q) symbols are equiprobable, the reliability $L(X)$ of a symbol X may be defined as the LLR of the probability of X related to the probability of symbol $\tilde{X}$:

$$L(X) = \ln\Big(\frac{P(Y|X)}{P(Y|\tilde{X})}\Big) \quad (2)$$

which, using (1), can be developed as:

$$L(X) = \sum_{i=0}^{m-1}\Big(-\frac{(y_i - B(x_i))^2}{2\sigma^2} + \frac{(y_i - B(\tilde{x}_i))^2}{2\sigma^2}\Big)$$

$$= -\frac{1}{2\sigma^2}\sum_{i=0}^{m-1}\Big(2y_i(B(\tilde{x}_i) - B(x_i))\Big) \quad (3)$$

By definition of $\tilde{X}$, $L(X)$ is a negative number. In order to deal with positive numbers, the quantity $\overline{L}(X) = -L(X)$ will be considered in what follows, as in [4]. Using (3), $\overline{L}(X)$ can be written as:

$$\overline{L}(X) = \frac{2}{\sigma^2} \sum_{i=0}^{m-1} |y_i| \Delta_i \quad (4)$$

where $\Delta_i = x_i$ XOR $\tilde{x}_i$, i.e. $\Delta_i = 0$ if x_i and $\tilde{x}_i$ have the same sign, 1 otherwise.

This approach which represents the inverse logic of the conventional LLR computation, associates the lower LLR value to the most reliable GF(q) symbol. Its main advantage is that it avoids the normalization step needed to keep the numerical stability of the extrinsic LLR during the decoding process.

The first step of the EMS algorithms [3] requires the generation of the first n_m minimum values of $\overline{L}(X)$ which is not a trivial problem. An elegant algorithm has been proposed by Fossorier et al. in [5]. Unfortunately, this algorithm is more software oriented, since it builds the LLR list dynamically, and is not adapted to hardware implementation mainly when a parallel approach is considered. Thus, a hardware oriented algorithm has become a necessity when the hardware implementation of the NB-LDPC decoder has become possible.

III. Proposed algorithm

Let us tackle the problem of generation of the n_m lowest $\overline{L}(X)$ values and their associated GF(q) symbols sorted in increasing order. In this paper, we propose an iterative construction of the list by considering, at stage c, only the first c coordinates of X (c varying from 1 to m). Let us define $\overline{L}(X^c) = \frac{2}{\sigma^2} \sum_{i=0}^{c-1} |y_i| \Delta_i$ the partial LLR of X using only its first c coordinates. Let λ_c be the set of couples $(\overline{L}(X^c), X^c)$ sorted in increasing order according to the $\overline{L}(X^c)$ values. For the sake of simplicity, the multiplicative constant factor $\frac{2}{\sigma^2}$ will be omitted in the following. For $c = 1$, λ_1 is then equal to: $\lambda_1 = \{(0, d_0), (|y_0|, \bar{d}_0)\}$ where $\bar{d}$ means not(d). At stage $c = 2$, the list λ_2 can be constructed from λ_1 in two steps. The first step is the expansion of λ_1 into two lists λ_2^0 and λ_2^1 by appending the second coordinate with value (0,d_1) and $(|y_1|, \bar{d}_1)$. The obtained lists are respectively

$$\lambda_2^0 = \{(0, d_0 d_1), (|y_0|, \bar{d}_0 d_1)\} \quad (5)$$

$$\lambda_2^1 = \{(|y_1|, d_0 \bar{d}_1), (|y_0| + |y_1|, \bar{d}_0 \bar{d}_1)\}. \quad (6)$$

The second step is the merging of λ_2^0 and λ_2^1 to create λ_2. For example, if $|y_1| > |y_0|$, then λ_2 will be equal to $\lambda_2 = \{(0, d_0 d_1), (|y_0|, \bar{d}_0 d_1), (|y_1|, d_0 \bar{d}_1), (|y_0| + |y_1|, \bar{d}_0 \bar{d}_1)\}$.

In the general case, during the expansion process, the k^{th} couple $(\overline{L}(X^{c-1})(k), X^{c-1}(k))$ of λ_{c-1} is expanded as:

$$\lambda_c^0(k) = (\overline{L}(X^{c-1})(k) + 0, X^{c-1}(k) \& d_{c-1}) \quad (7)$$

$$\lambda_c^1(k) = (\overline{L}(X^{c-1})(k) + |y_{c-1}|, X^{c-1}(k) \& \bar{d}_{c-1}), \quad (8)$$

where & stands for the binary append operator.

Since both λ_c^0 and λ_c^1 are sorted, the creation of λ_c is simple. These two ordered lists are iteratively compared. At each step, each list sends its smallest remaining LLR value. Those two LLR values are compared and the couple with the smallest LLR is retrieved from the corresponding list and added to the output list λ_c. This algorithm can be performed efficiently in a serial pipe-line architecture. One can note that, since only n_m lowest values are required by the EMS algorithm, the size $s(c)$ of list λ_c is equal to $\min(2^c, n_m)$.

Let us consider the following example of LLR generation for $m = 4$, $n_m = 10$ and $Y = (-7, 8, 12, -3)$. The hard decoding of Y leads to $D = (0, 1, 1, 0)$. Since $|y_0| < |y_1|$, the list λ_2 is equal to λ_2={(0, 01), (7, 11), (8, 00), (15, 10)}.

The expansion of λ_2 gives λ_3^0={(0, 011), (7, 111), (8, 001), (15, 101)} and λ_3^1={(12, 010), (19, 110), (20, 000), (27, 100)}. Merging those two lists gives: λ_3={(0, 011), (7, 111), (8, 001), (12,010), (15, 101), (19, 110), (20, 000), (27, 100)}

The extension of λ_3 gives: λ_4^0={(0, 0110), (7, 1110), (8, 0010), (12, 0100), (15, 1010), (19, 1100), (20, 0000), (27, 1000)} and λ_4^1={(3, 0111), (10, 1111), (11, 0011), (15, 0101), (18, 1011), (22, 1101), (23, 0001), (30, 1001)}. Taking the $n_m = 10$ highest value of λ_4^0 and λ_4^1 leads to the final list: λ_4={(0, 0110), (3, 0111), (7, 1110), (8, 0010), (10, 1111), (11, 0011), (12, 0100), (15, 1010), (15, 0101), (18, 1011)}.

The direct approach (computation of all q LLRs and sorting process to extract the first n_m values) leads to a complexity in $q \log_2(q)$. With the proposed method, the overall complexity is reduced to $n_m \log_2(q)$. Moreover, the proposed algorithm can be implemented in a simple systolic hardware architecture.

IV. Hardware Architecture of the LLR circuit

In this section we describe the systolic hardware architecture serially generating the sorted LLR list. This architecture is composed of m stages working in pipeline mode and each stage consists of one Processing Element (PE). Figure 1-a illustrates the architecture for $m = 4$. As shown in this Figure, the c^{th} PE receives two inputs: the channel binary observation y_{c-1} with its corresponding sign and the list λ_{c-1} from the $(c-1)^{th}$ PE. As an output, the c^{th} PE generates the list λ_c to be fed to the $(c+1)^{th}$ PE of the next stage.

A. Structure of the PE

The PE constitutes the core of the proposed LLR circuit. It consists of two expansion modules, two First-In-First-Out (FIFO) memories and one comparator selecting the minimum of the two FIFO's outputs as shown in Figure 1-b where the third stage PE is illustrated. The input/ouput of this stage represent the intermediate LLR computation of the example described in previous section. The 3^{rd} PE serially receives (one new couple every clock cycle) the ordered list λ_2 from stage 2. The first step is the expansion of λ_2 into lists λ_3^0 and λ_3^1 as described in the previous section. Once the first elements of λ_3^0 and λ_3^1 are stored in the FIFOs, the merging process can start. At each clock cycle, the two outputs of the FIFOs are compared and the couple having the smaller LLR value is selected to be fed in the list λ_3 and a pull signal is set to 1 to get a new couple at the output of the considered FIFO.

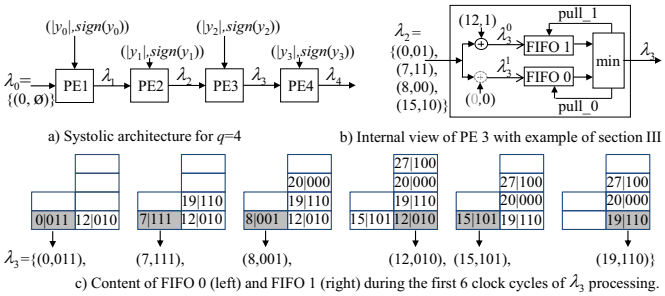

Fig. 1. LLR circuit block diagram designed over GF(16).

Figure 1-c illustrates the temporary contents of the FIFOs during the first six clock cycles of the 3^{rd} stage processing. The highlighted memory case of the FIFOs represents the content to be selected at each clock cycle. As previously discussed, the FIFO 0 contains the first elements of the LLR list. Thus, it is possible to empty the FIFO 0 and in this case the remaining contents of the FIFO 1 are systematically selected to complete the list λ_3.

One can note, that, during the first $s(c-1)$ clock cycles, FIFO 0 and FIFO 1 receive one new couple every clock cycle and one couple is output, either from FIFO 0 or FIFO 1. Let $f^0(c)$ and $f^1(c)$ be the minimum sizes required by the FIFO 0 and the FIFO 1 of the c^{th} PE respectively. Let us first consider the size $f^1(c)$ when $s(c)$ is even. In the worst case, the first $s(c)/2$ couples of λ_c are retrieved from FIFO 0. At that point, FIFO 1 contains the $s(c)/2$ couples sufficient to complete the $s(c)/2$ remaining elements of λ_c. If $s(c)$ is odd, $(s(c)-1)/2$ elements of λ_c at most can come from FIFO 1. Thus, in the general case, $f^1(c) \leq \lfloor s(c)/2 \rfloor$, where $\lfloor x \rfloor$ represents the greatest integer smaller or equal to x.

The determination of $f^0(c)$ needs a more careful examination. Let us consider the case where $s(c-1) = s(c)$. In the worst case, the first $\lfloor s(c)/3 \rfloor$ couples are retrieved from FIFO 0, then the next $\lfloor s(c)/3 \rfloor$ couples are retrieved from FIFO 1. At that point, both FIFOs have already received $2\lfloor s(c)/3 \rfloor$ couples. The output of FIFO 0 and FIFO 1 are respectively $\lambda_c^0(\lfloor s(c)/3 \rfloor + 1)$ and $\lambda_c^1(\lfloor s(c)/3 \rfloor + 1)$. In all cases, the content of FIFO 0 is enough to retrieve the next $\lfloor s(c)/3 \rfloor$ values needed to complete the list λ_c. If $s(c)$ is not a multiple of 3, $s(c)$ is still equal to $\lfloor s(c)/3 \rfloor$. In this case the $(2\lfloor s(c)/3 \rfloor + 1)^{th}$ element of λ_c will be retrieved from FIFO 0, which freed room for the $(3\lfloor s(c)/3 \rfloor + 1)^{th}$ element coming from λ_{c-1} and so on. If $s(c-1) = s(c)/2$ it can be shown that, in that case, $f^0(c) = s(c)/4$. In summary, $f^0(c)$ varies between $f^0(c) = s(c)/4$ and $f^0(c) = s(c)/3$.

Figure 2 shows the timing diagram of the LLR computation of the example described in section III. As shown in this Figure, the LLR circuit operates in a synchronous way with a clock signal. The signal $startin$ indicates the start of the LLR computation and the signal $load\_y_i$ indicates the availability of the y_i data at the input. The circuit has a Latency of $m = 4$ cycles, the start of the generation of the output is indicated by signal $startout$. After $n_m + 4 - 1 = 13$ cycles the (LLR,GF) couples are generated. As previously mentioned, the LLR circuit operates in a pipeline manner where it can starts the processing of the second symbol while the last stage of the circuit is producing the (LLR,GF) couples of the first symbol. Thus as shown in Figure 2, after a delay of one cycle the first (LLR,GF) couple associated to the second symbol is produced. This delay cycle is required to reinitialize the FIFOs.

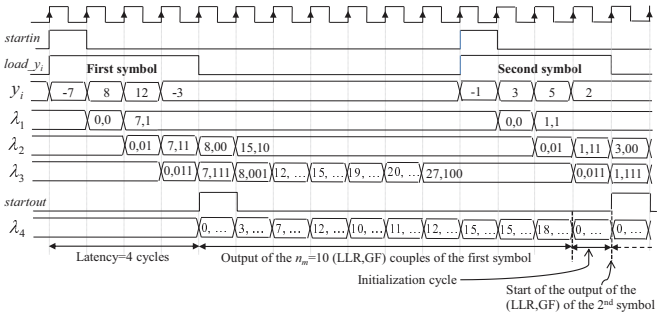

Fig. 2. Timing diagram of the LLR computation over GF(16), $n_m = 10$.

B. Implementation results

A generic architecture of the proposed systolic architecture has been developed and successfully validated. The architecture accepts any Galois Field size q and any value $n_m \leq q$. The internal binary word size (n_b, n_s) of the (LLR,GF) couple is also parameterized. This architecture has been used in a FPGA based GF(64) NB-LDPC code implementation, with $q = 6$, $n_m = 12$, $n_b = 6$ and $n_s = 6$. On a Virtex 4 (XC4VLX15), it requires 275 slices and operates at a frequency of 149 MHz.

V. Conclusions

We have presented a novel and efficient hardware design of the LLR computation circuit. The proposed circuit is the first of its kind to be designed. It performs the LLR computation in a systolic way and produces an ordered list of (LLR, GF) couples. This architecture has been implemented in a design of a GF(64) NB-LDPC code. It can be also used for other codes, like the soft Reed Solomon decoder described in [6].

References

[1] M. Davey and D. J. C. MacKay "Low density parity check codes over GF(q)", *IEEE Commun. Lett., vol. 2, pp. 165, 1998.*

[2] S. Pfletschinger, D. Declercq, "Getting Closer to MIMO Capacity with Non-Binary Codes and Spatial Multiplexing", *IEEE Globecom, Miami, Florida, USA, Dec. 2010.*

[3] D. Declercq and M. Fossorier, "Decoding algorithms for Nonbinary LDPC codes over GF(q)", *IEEE Transactions on Communications, Vol. 55, No. 4, pp. 633-643, Apr. 2007.*

[4] V. Sevin, "Min-max decoding for non binary LDPC codes". *Proc. Int. Symp. on Information Theory, ISIT2008, Toronto, Canada, pp. 960-964, Jul. 2008.*

[5] A. Valembois and M. Fossorier, "An improved method to compute lists of binary vectors that optimize a given weight function with application to soft-decision decoding", *IEEE communications Letters, Vol. 5, No. 11, pp. 456-458, Nov. 2001.*

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