Extended-Forward Architecture for Simplified Check Node Processing in NB-LDPC Decoders

Cédric Marchand, Emmanuel Boutillon, Hassan Harb, Laura Conde-Canencia, Ali Al Ghouwayel

Extended-Forward Architecture for Simplified Check Node Processing in NB-LDPC Decoders

Cédric Marchand, Emmanuel Boutillon,
Hassan Harb and Laura Conde-Canencia
Lab-STICC, CNRS UMR 6285
Université de Bretagne Sud, Lorient, France
Email: emmanuel.boutillon@univ-ubs.fr

Ali Al Ghouwayel
CCE Department
Lebanese International University (LIU), Beirut, Lebanon.
Email: ali.ghouwayel@liu.edu.lb

***Abstract*—This paper focuses on low complexity architectures for check node processing in Non-Binary LDPC decoders. To be specific, we focus on Extended Min-Sum decoders and consider the state-of-the-art Forward-Backward and Syndrome-Based approaches. We recall the presorting technique that allows for significant complexity reduction at the Elementary Check Node level. The Extended-Forward architecture is then presented as an original new architecture for efficient syndrome calculation. These advances lead to a new architecture for check node processing with reduced area. As an example, we provide implementation results over GF(64) and code rate 5/6 showing complexity reduction by a factor of up to 2.6.**

***Index Terms*—NB-LDPC, Check Node, Syndrome-Based, Foward-Backward, VLSI.**

I. Introduction

Low-Density Parity-Check (LDPC) codes [1] have now been adopted for a wide range of standards (WiMAX, WiFi, DVB-C2, DVB-S2X, DVB-T2) because of their near-channel-capacity performance. However, this performance is mainly obtained for long codeword lengths. The last decade witnessed a great deal of research effort devoted to the extended version of LDPC codes defined over (GF(q), $q > 2$). These codes called Non-Binary (NB) LDPC codes have shown strong potential in error correction capability with moderate and short codeword lengths [2]. NB-LDPC codes retain the benefits of steep waterfall region (typical of convolutional turbo-codes) and low error floor (typical of binary LDPC). Compared to their binary counterparts, NB-LDPC codes generally present higher girths, which leads to better decoding performance. Moreover, the NB nature of such NB-LDPC codes makes them suitable for high-spectral-efficiency modulation schemes where the constellation symbols are directly mapped to GF(q) symbols. This mapping bypasses the marginalization process of binary LDPC codes that causes information loss. These characteristics place NB-LDPC codes as serious competitors of classical binary LDPC and Turbo-Codes in future wireless communication and digital video broadcasting standards. However, NB-LDPC codes suffer from high decoding complexity. In the NB decoder each message exchanged between processing nodes is an array of values, each value corresponding to a GF element. From an implementation point of view, this leads to a highly increased complexity compared to binary LDPC.

TABLE I
CN architectures based on the EMS algorithm.

CN architectures		Input processing	
Abbreviation	Name	Normal	PreSorted
T-EMS	Trellis EMS	[6]	-
FB	Forward-Backward	[4],[5],[9]	[10]*
SB	Syndrome-Based	[7]*,[8]*, [11]	[12]*
EF	Extended Forward-SB	This paper	This paper

The extension of the well-known binary Min-Sum algorithm to the NB domain, called Extended Min-Sum (EMS), represents an interesting compromise between hardware complexity and error correction performance [3], [4]. The computational bottleneck in the EMS algorithm is the Check Node (CN) processing. State-of-the-art architectures apply the Forward-Backward (FB) algorithm [4], [5] to efficiently implement the CN. The FB approach uses Elementary Check Nodes (ECN) in a 3-layer structure to reduce the hardware cost of the CN as well as the number of computations, as it is possible to reuse intermediate results. However, the FB CN structure suffers from high latency and low throughput rate. The Trellis-EMS (T-EMS) introduced in [6] reduces the latency of the FB computation but presents a hardware complexity that significantly increases with q when a parallel implementation is considered. The Syndrome-Based Check Node (SB CN) algorithm, recently presented in [7] [8], is an efficient method to perform in parallel the CN computations for high order Galois fields ($q \geq 16$). However, the complexity of the SB CN algorithm is dominated by the number of syndromes to be computed, which increases quadratically with d_c. This limits its interest for high coding rates, i.e. high d_c values.

Recently, we showed that sorting the input vector of the CN according to a reliability criterion [12] [10] allows significant reduction of the hardware complexity of the CN architecture without affecting the performance. This technique, named *presorting*, has been successfully combined with the SB CN leading to the PreSorted SB CN (or PS-SB CN) architecture [12] and with the FB CN to give the PS-FB CN [10]. Table I summarizes the different CN architectures based on the EMS algorithm and introduces the architecture proposed in this paper, i.e. the Extended-Forward CN, with and without

application of the presorting technique. The references with an asterisk correspond to previous contributions of the authors.

The rest of the paper is organized as follows: Section II briefly introduces NB-LDPC codes as well as the Min-Sum and the Extended Min-Sum algorithms. It also reminds the principles of the Forward-Backward and the Syndrome-Based CN architectures. Section III presents the new Extended-Forward approach and how the presorting technique can lead to a more efficient architecture. Performance and synthesis results are presented in Section IV, for comparison with the state-of-the-art architectures. Finally, section V concludes the work.

II. NB-LDPC codes and Extended Min-Sum decoding

This Section first introduces NB-LDPC codes and their notations. NB-LDPC decoding is then considered with the description of the Min-Sum algorithm and its extended version.

A. NB-LDPC codes

An NB-LDPC code is a linear block code defined over a Galois Field of size q, GF(q), and with a sparse parity-check matrix $\mathbf{H}$ of size $M \times N$. The M rows of the matrix $\mathbf{H}$ refer to M parity-check equations. The i^{th} parity-check equation is expressed as

$$\sum_{k=1}^{d_c(i)} h_{i,j_i(k)} x(j_i(k)) = 0,$$

where $d_c(i)$ in the number of non-zero GF(q) values which are denoted as $\{h_{i,j_i(k)}\}_{k=1\ldots d_c(i)}$ and $\{x(j_i(k))\}_{k=1\ldots d_c(i)}$ is a subset of size $d_c(i)$ of the N GF(q) symbols of the code. A NB-LDPC code is regular if the number d_c of non-zero values per row and the number d_v of non-zero values per column are constant. The authors in [13] showed that robust NB-LDPC codes can be constructed with $d_v = 2$. In that case, assuming a full rank parity check matrix $\mathbf{H}$, the rate of the code is $r = 1 - 2/d_c$. An (M, N) GF(q) NB-LDPC code has a binary counterpart of size $(M \times m, N \times m)$ with $m = \log_2(q)$.

B. Min-Sum algorithm

For simplicity, the algorithms are described only at the CN level. The reader can refer to [14] for a complete description of the variable node and edge node processors.

Let us define a CN equation of degree d_c in GF(q) as $e_1 \oplus e_2 \oplus e_3 \oplus \ldots \oplus e_{d_c} = 0$ where the operator $\oplus$ represents addition over GF(q). Fig. 1 shows a CN for $d_c = 4$ and the associated messages. Inputs e_i, $i \in 1, \ldots, d_c$ take their values on the alphabet of size q. The *a priori* information about variable e is the discrete probability distribution $P(e = x)$, $x \in$ GF(q). Each element of the probability distribution E associated to e can be expressed in the log domain as the Log Likelihood Ratio (LLR) $e^+(x)$ defined as

$$e^+(x) = -\log\left(\frac{P(e = x)}{P(e = \bar{x})}\right), \quad (1)$$

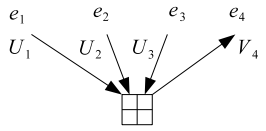

Fig. 1. Message notation on a CN

where $\bar{x}$ is the hard decision on e obtained by taking the most probable GF symbol, i.e. $\bar{x} = \arg\max_{x \in \text{GF}(q)} P(e = x)$. By definition of the LLR, we have: $e^+(\bar{x}) = 0$ and $\forall x \in$ GF(q), $e^+(x) \geq 0$. The distribution (or message) E associated to e is thus $E = \{e^+(x)\}_{x \in \text{GF}(q)}$.

The Min-Sum algorithm computes the LLR value of the i^{th} output for the GF symbol x as $v_i^+(x)$

$$v_i^+(x) = \min\left\{ \sum_{i'=1, i' \neq i}^{d_c} e_{i'}^+(x_{i'}) \mid \bigoplus_{i'=1, i' \neq i}^{d_c} x_{i'} = x \right\}, \quad (2)$$

where $x_{i'} \in$ GF(q) for $i' = 1, \ldots, d_c$, $i' \neq i$.

C. EMS algorithm

The main characteristic of the EMS algorithm is to truncate message E from q values to the n_m most reliable ones, with $n_m \ll q$. The resulting message U is composed of n_m couples sorted in increasing order of LLRs (the higher the LLR value, the lower the reliability). Fig. 1 illustrates the principle of CN processing. Input U of a CN is a list $\{U[j]\}_{j=0\ldots n_m-1}$ of couples $U[j] = (U^+[j], U^\oplus[j])$, where $U^+[j]$ denotes the j^{th} smallest LLR value of E and $U^\oplus[j]$ denotes its associated GF element, i.e., $e^+(U^\oplus[j]) = U^+[j]$. Note also that $U^+[0] = 0$, $U^\oplus[0] = \bar{x}$, and that $j \leq j' \Rightarrow U^+[j] \leq U^+[j']$. The same representation is used for each output V of a CN.

The EMS algorithm is a simplification of the Min-Sum algorithm and can be described in two steps:

1) **Evaluate:** eq. (2) is modified by replacing $x_{i'} \in$ GF(q) by $x_{i'}$ in the set of available GF data, i.e., $x_{i'} \in U_i^\oplus$, with $U_i^\oplus = \{U_i^\oplus[j]\}_{j=0,1,\ldots,n_m-1}$:

$$v_i^+(x) = \min\left\{ \sum_{i'=1, i' \neq i}^{d_c} U_{i'}^+[j_{i'}] \mid \bigoplus_{i'=1, i' \neq i}^{d_c} U_{i'}^\oplus[j_{i'}] = x \right\}, \quad (3)$$

where $j_{i'} \in \{0, 1, \ldots n_m - 1\}$ for $i' = 1, 2, \ldots d_c$, $i' \neq i$.

2) **Sort:** the $v_i^+(x)$ are sorted in increasing order and the first n_m smallest values are kept to generate the output vector V_i.

D. Forward-Backward CN

As described in [4], the FB consists of three layers of $d_c - 2$ ECNs each. An ECN processes a single output C as a function of two inputs A and B. Intermediate results of the ECNs are reused in the later stages to avoid re-computations, thus reducing the amount of processing to generate each output message.

The ECN processing [15] can be described in three steps.

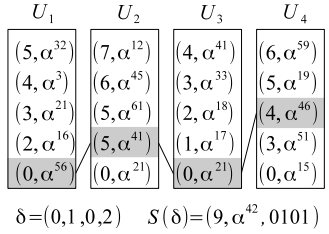

Fig. 2. Example of a deviation path

1) **Addition:** For each couple of index $(a,b) \in \{0,1,\ldots,n_m-1\}^2$, the output tuple $C_{a,b} = (C^+(x), x)$ is computed as

$$C_{a,b} = (A^+[a] + B^+[b], A^{\oplus}[a] \oplus B^{\oplus}[b]) \quad (4)$$

2) **Sorting**: The couples $(c^+(x), x)$ are sorted in increasing order of $c^+(x)$.
3) **Redundancy Elimination (RE)**: In case of two couples with the same GF value, the one with the higher LLR is suppressed during this RE step.

For the sake of clarity, the three ECN steps are represented using the symbol $\boxplus$ and the ECN equation is

$$C = A \boxplus B. \quad (5)$$

E. Syndrome-Based CN

The SB CN algorithm [7] relies on the definition of a deviation path and its associated syndrome. In the sequel, $n_{m,in}$ (resp. $n_{m,out}$) refers to the size of the input (resp. output) vector of a CN.

A **deviation path**, denoted by δ, is defined as a d_c-tuple of integer values between 0 and $n_{m,in}-1$, i.e. $\delta = (\delta(1), \delta(2), \ldots, \delta(d_c))$, with $\delta(i) \in \{0,1,\ldots,n_{m,in}-1\}$, $i = 1,2\ldots,d_c$.

The SB considers a set Δ of deviation paths on the d_c input messages to compute associated syndromes.

The syndrome $S(\delta)$ associated to deviation path δ is defined by the 3-tuple $(S^+(\delta), S^{\oplus}(\delta), S^D(\delta))$, with

$$S^+(\delta) = \sum_{i=1}^{d_c} U_i^+[\delta(i)],\ S^{\oplus}(\delta) = \bigoplus_{i=1}^{d_c} U_i^{\oplus}[\delta(i)] \text{ and} \quad (6)$$

$$S^D(\delta)[i] = \begin{cases} 0, & \text{if } \delta(i) = 0 \\ 1, & \text{otherwise} \end{cases}, \quad (7)$$

where $S^+(\delta)$ is an LLR value, $S^{\oplus} \in$ GF(q), and $S^D(\delta)$ is a binary vector of size d_c called Discard Binary Vector (DBV).

An example of deviation path δ and its associated syndrome $S(\delta)$ is shown in Fig. 2.

After syndromes computation, they are sorted and then decorrelated for each output message V_i, $i = 1,...,d_c$. The SB CN architecture is given in Fig. 3. In particular, decorrelation is performed by d_c Decorrelation Units (DU). DUs are represented in parallel to show the inherent parallelism of the SB CN. Fig. 3 also shows the schematic operations of the DUi. S^D is the d_c-wide bit vector that indicates for which output edges the syndrome should be discarded during the decorrelation process. A simple reading of bit i in the binary vector S^D validates or not the syndrome for the output edge i.

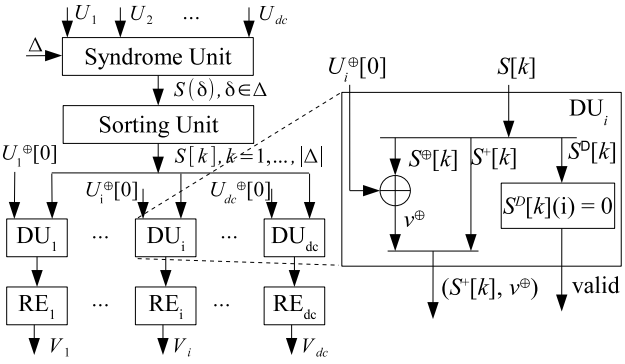

Fig. 3. Syndrome-based CN processing (left part), details of the DU unit (right part)

F. Presorting technique

The presorting technique consists in permuting the input vectors as a function of $U_i^+[1], i = 1,2,...,d_c$. Fig. 4 illustrated an application of the presorting algorithm on the EMS-based CN, where $n_m = 4$ and $d_c = 4$. In this particular example, the hatched tuples represent tuples that are not involved in the computation of the first 8 syndromes. Compared to the standard CN, the presorting process requires extra hardware: a d_c-input vector sorter and two permutation networks (or switches). However, it allows some simplifications in the CN itself, globally leading to a complexity reduction of the whole CN processing. The presorting technique was applied to both FB [10] and SB [12], leading to significant complexity reductions in the decoder CN architectures.

III. New CN architecture: syndrome computation with Extended-Forward processing

In this section, we present an architecture that dynamically generates a set of syndromes, keeping the complexity linear with d_c. The output of a forward layer in a FB decoder processes all input except last one U_{d_c}. The idea is to add an ECN to merge the forward layer with U_{d_c}. The output of this Extended Forward (EF) generates in fact a set of sorted syndromes S_b. The syndrome computation is thus equivalent to the equation

$$S_b = \boxplus_{i=1}^{d_c} U_i. \quad (8)$$

The associated EF CN architecture with $d_c = 6$ is presented in Fig. 5. Thus $d_c - 1$ ECNs with parameters (n_m, n_m, n_m) will provide n_m syndromes sorted in increasing order of their associated LLR values.

S_b can also be computed in $\log_2(d_c)$ layers of ECNs using a tree structure as shown in Fig. 6 or in any semi-parallel

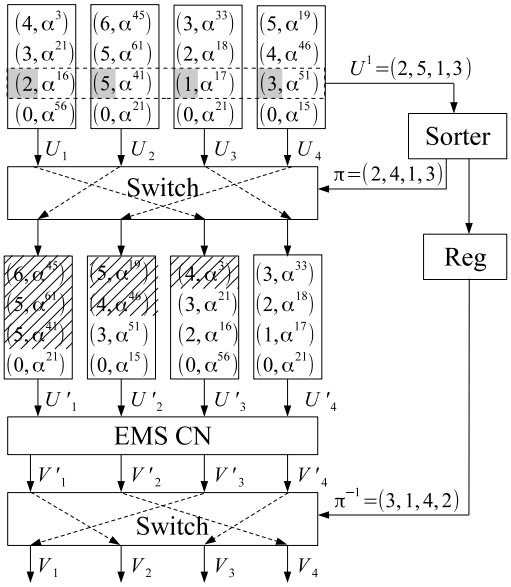

Fig. 4. Presorting principle applyed on an EMS-based CN architecture

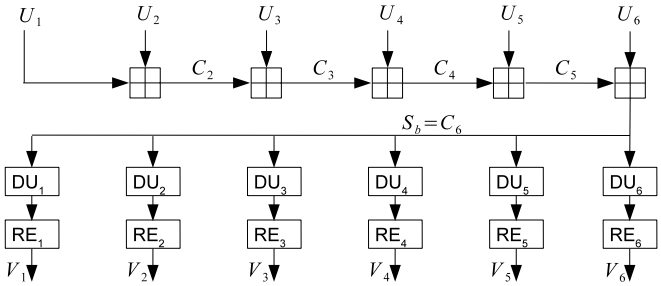

Fig. 5. EF CN Architecture

compromise. As shown in Fig. 5 and Fig. 6 the complexity is of $d_c - 1$ ECNs and is thus linear with d_c.

A. Construction of the Discard Binary Vector

Thanks to the use of ECNs, the computed syndrome set is sorted and can be directly applied to the decorrelation process.

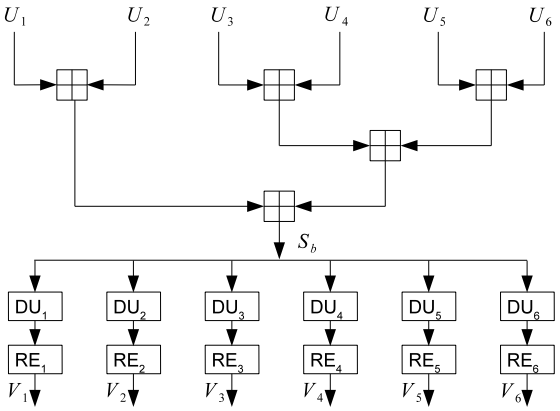

Fig. 6. EF CN with tree architecture

However, each ECN must perform an additional operation to generate a new term $c^D_{a,b}$, that is needed to construct the DBV S^D required in the decorrelation process ((7) and right part of Fig. 3). The ECN addition is modified as

$$C_{a,b} = (c^+_{a,b}, c^{\oplus}_{a,b}, c^D_{a,b}) = (A^+[a] + B^+[b], \\ A^{\oplus}[a] \oplus B^{\oplus}[b], \\ A^D[a] \| B^D[b]) \tag{9}$$

where $\|$ represents the concatenation operation of two binary vectors.

In the EF processing, the CN inputs are initialized with a DBV value of one bit, as follows: $U^D_i[j] = 1, \forall j \neq 0$ and $U^D_i[0] = 0$. Thanks to the DBV computation, the output of the EF processing is similar to (6) and (7) with the only difference that the set of deviation paths Δ_{EF} is input dependent, while it is predefined offline for the SB [7], [8].

B. The effect of RE in ECNs

In the EF processing, the number of computed syndromes is typically $3 \times n_{m,out}$ to compensate for the discarded redundant syndromes [7], [8]. However, even with this approach, the first simulation results of the EF algorithm showed significant performance degradation compared to the FB algorithm (red curve in Fig. 8). The reason of this performance degradation resides in the RE step performed by each ECN: since each ECN performs RE, no more than one syndrome can be associated to a given GF value. Let us consider two syndromes with same GF value x and different deviation paths at the input of a DU where the second syndrome is validated by the DU and not the first one. If the second (valid) syndrome has been discarded previously by a RE then no syndrome with GF value x can be validated and the default maximum LLR is associated with x leading to performance degradation. The solution for this problem is to make sure that there is no RE before DU and to perform RE after DU as shown in Fig. 5 and Fig. 6.

C. Extended Forward CN with presorting

The presorting technique leads to significant hardware savings by reducing the number of candidate GF symbols in the ECNs [10]. This technique can also be applied to the EF CN architecture for complexity reduction thanks to the limited number of bubbles to be considered in each ECN. For this, we perform a statistical study based on Monte-Carlo simulation that traces the paths of the GF symbols that contribute to the output of the CN, on their way across the different ECNs. With this information we know, for each ECN, how often a specific bubble contributes to an output and we can then prune the bubbles that never or rarely contribute to an output. A short simulation on less than 10000 frames is more than enough to produce reliable statistics on the bubble order of importance. However, the best method on how to prune low-score bubbles is still an open question. We obtained good results by cumulating the sorted bubble scores and discard bubbles above a threshold given by a percentage of the total cumulated scores. In a second step, bubbles are

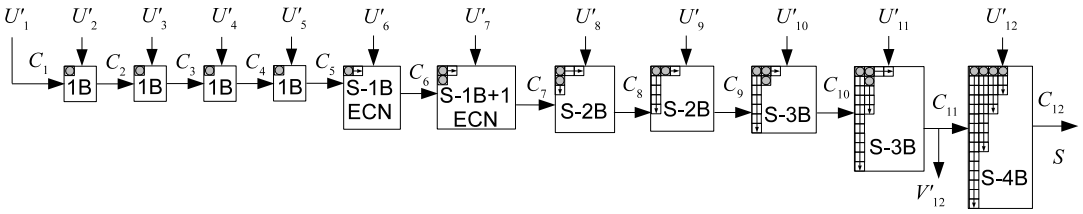

Fig. 7. Architecture of the proposed EF CN with $d_c = 12$, $n_b \leq 4$ ($n_{m,in} = 4$), $n_c^{12} = z = 20$.

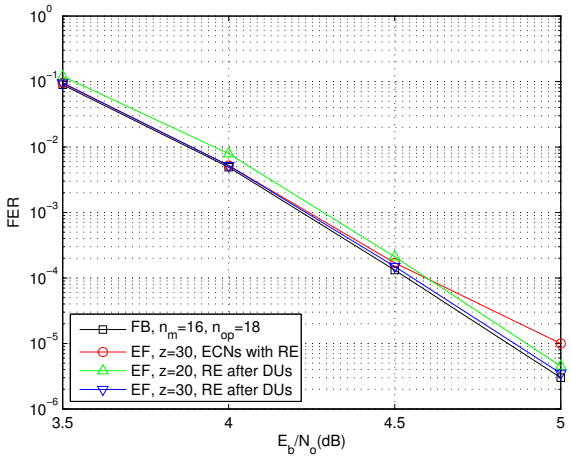

Fig. 8. Simulation results of NB-LDPC decoding algorithms for (576, 480) code over GF(64) and $d_c = 12$ under AWGN channel.

TABLE II
POST-SYNTHESIS RESULTS FOR DIFFERENT ECN ARCHITECTURES AND ENTITIES ON A XILINX VIRTEX 6 FPGA.

		Number of slices			F_{max}	Latency
		total	register	LUT	(MHz)	(cycles)
ECNs	1B	7	25	7	>500	1
	S-1B	17	53	23	>500	1
	S-1B+1	29	92	60	494	1
	S-2B	49	89	112	338	1
	S-3B	103	218	218	296	1
	S-4B	117	157	275	319	2
	S-4B RE	138	222	376	269	2
Entities	Sorter	160	325	368	325	6
	Switch	283	273	657	>500	1
	DU	21	55	14	>500	1
	RE	43	104	141	462	1
	Mult	8	18	19	>500	1

discarded when complexity can be reduced without significant performance degradation.

After this pruning process, the structure of some ECNs is greatly simplified.

- **1B:** only a single bubble is considered where C_i is given by $C_i = \{(0, U_i'^{\oplus}[0] \oplus C_{i-1}^{\oplus}[0], C_{i-1}^{D}[0]||0)\}$ for $i > 1$ and $C_1 = \{(0, U_1'^{\oplus}[0] \oplus U_2'^{\oplus}[0], 00)\}$.
- **S-1B:** it generates directly the n_b^i sorted output C_i as $C_i = \{(0, U_i'^{\oplus}[0] \oplus C_{i-1}^{\oplus}[0], C_{i-1}^{D}||0), (U_i'^{+}[b], U_i'^{\oplus}[b] \oplus C_{i-1}^{\oplus}[0], C_{i-1}^{D}||1)_{b=1...n_b^i-1}\}$. For this operation, a single GF-adder is required.
- **S-1B+1:** only one comparator and one GF-adder are required.
- **S-xB:** also known as S-bubble ECN. As described in [16], this architecture compares x bubbles per clock cycle. It is composed of x FIFO blocks, $x - 1$ comparators, x arithmetic adders and x GF-adders, implemented with XOR gates.

Note that, as shown in Fig. 7, V'_{12} can be directly obtained from C_{11}.

IV. PERFORMANCE AND COMPLEXITY COMPARISON

A. Performance

We consider bit-true Monte-Carlo simulations over the AWGN channel with a BPSK modulation scheme. Extrinsic and intrinsic LLR messages are quantified on 6 bits and *a posteriori* LLRs on 8 bits. The maximum number of iterations is set to 10. Fig. 8 shows the Frame Error Rate (FER) performance of the FB and the EF algorithms for a GF(64)-LDPC code of size (576, 480). This code is regular with $d_v = 2$ and $d_c = 12$. As a reference, the FB decoder uses the S-bubble algorithm as in [16] with 4 bubbles, $n_m = 16$ and $n_{op} = 18$, where n_{op} represents the number of outputs of an ECN before RE. The simulation EF CN with RE performed in ECNs shows a performance degradation and it shows that RE should not be done in ECN but after decorrelation as demonstrated in sub-section III-B and shown in Fig. 5 and Fig. 6. Simulation with $z = 30$ shows negligible performance degradation. For the EF CN with $z = 20$, the ECNs are as represented in Fig. 7 and simulation shows that performance versus complexity compromise can be found.

B. Implementation results

For hardware complexity evaluation, we implemented the different CN architectures and CN sub-unit on a Xilinx VIRTEX 6 FPGA device. The 6 types of ECNs used in PS EF CN represented in Fig. 7 and the 4S-bubble with RE used in FB CN were synthesized to obtain the results presented in Table II. The sorter and the switch (Fig. 4), the DU and the RE (Fig. 5) and the GF multiplier implementation results are also presented. For all implementations, the LLR and GF values are quantified on 6 bits. The latency in cycles (or number of pipeline stage) between input and output of each entity is also presented.

Table III summarizes the implementation results for the considered architectures for GF(64) and $d_c = 12$. In some

TABLE III
POST SYNTHESIS RESULTS FOR GF(64) CN ARCHITECTURES

CN GF(64)	Number of slices			F_{max} (MHz)	L_{CN} (cycles)
	Total	REG	LUT		
FB RE	3904	6019	10314	227	22
PS FB RE	2729	4119	6733	229	20
EF RE	2633	4628	6768	217	24
PS EF RE	1987	3151	4814	222	19
PS EF	1495	2383	3616	227	19

decoder implementations [9], [15], when code with $d_v = 2$ are considered, the VNs are cascaded after the CN and a RE is performed in the VN. Since that in the PS EF architecture the RE block is performed at each CN output (as shown in Fig. 5 and Fig. 6), and is performed again in the VN, the RE can be removed from the CN output without performance loss. The EF shows significant complexity reduction compared with standard FB CN implementation. The PS EF CN with RE and without RE show 1.96 and 2.6 times less complexity than the standard FB CN, respectively.

L_{CN} gives the latency in cycles of a CN. For the FB architecture, $L_{CN} = L_{mult} + (d_c - 2) \times L_{S-4B} + L_{div} = 22$. For PS EF architecture (Fig. 7), $L_{CN} = L_{Sorter} + L_{Switch} + L_{ECNs} + L_{DU} + L_{Switch} = 6+1+12+1+1 = 19$, considering that the multiplication is performed in the same cycle as the switch and the division is performed during the DU cycle.

C. Thoughput

High throughput can be obtained by processing p CNs in parallel in a layered decoder [17], [9]. Because we focus on code with $d_v = 2$, the VN update can be cascaded after the CN update as in [9], [15]. The throughput of the layered decoder is then given by $T(Mbps) = (K_b \times F_{max} \times p)/(M \times P_{CN} \times it_{avr})$ where p is the parallelism, it_{avr} is the average number of iteration and P_{CN} is the periodicity of the CN computations in a layered decoder. P_{CN} in a layered decoder as in [9] is given by $P_{CN} = L_{CN} + L_{VN} + n_{m,in}$ with $L_{VN} = 3 + n_{m,out} + n_{m,int}$ where $n_{m,int}$ is the number of intrinsic values considered in the second phase of the VN [9]. The (576,480) code allows a parallelism of up to 4 and simulation shown is Fig. 8 gives $it_{avr} = 1.2$ at SNR = 4.5. With $n_{m,int}$ set to 4, the throughput can reach 534 Mbps with the FB architecture and 544 Mbps with the PS EF architecture.

V. CONCLUSION

The presented solutions efficiently incorporate Elementary Check Nodes into a Syndrome-based architecture. Moreover the CN inputs are presorted leading to drastic simplification of most of the ECNs. The resulting architecture benefits from flexibility and linear complexity of the Extended Forward solution and complexity reduction due to presorting. The final architecture shows area reduction by up to a factor of 2.6 compared with standard FB decoder using S-bubble ECNs.

REFERENCES

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

[2] M. C. Davey and D. J. C. MacKay, "Low density parity check codes over GF(q)," in *Information Theory Workshop, 1998*, Jun 1998, pp. 70–71.

[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, April 2007.

[4] 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.

[5] Y. L. Ueng, K. H. Liao, H. C. Chou, and C. J. Yang, "A high-throughput trellis-based layered decoding architecture for non-binary LDPC codes using max-log-QSPA," *IEEE Transactions on Signal Processing*, vol. 61, no. 11, pp. 2940–2951, June 2013.

[6] E. Li, D. Declercq, and K. Gunnam, "Trellis-based extended Min-Sum algorithm for non-binary LDPC codes and its hardware structure," *IEEE Transactions on Communications*, vol. 61, no. 7, pp. 2600–2611, July 2013.

[7] P. Schläfer, N. Wehn, M. Alles, T. Lehnigk-Emden, and E. Boutillon, "Syndrome based check node processing of high order NB-LDPC decoders," in *22nd International Conference on Telecommunications (ICT)*, April 2015, pp. 156–162.

[8] P. Schläfer, V. Rybalkin, N. Wehn, M. Alles, T. Lehnigk-Emden, and E. Boutillon, "A new architecture for high throughput, low latency nb-ldpc check node processing," in *Personal, Indoor, and Mobile Radio Communications (PIMRC), 2015 IEEE 26th Annual International Symposium on*, Aug 2015, pp. 1392–1397.

[9] C. L. Lin, S. W. Tu, C. L. Chen, H. C. Chang, and C. Y. Lee, "An efficient decoder architecture for nonbinary ldpc codes with extended min-sum algorithm," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 63, no. 9, pp. 863–867, Sept 2016.

[10] H. Harb, C. Marchand, A. Ghouwayel, A. Conde-Canencia, L., and E. Boutillon, "Pre-sorted forward-backward NB-LDPC check node architecture," in *2016 IEEE International Workshop on Signal Processing Systems (SiPS)*, Oct 2016, pp. 142–147.

[11] V. Rybalkin, P. Schläfer, and N. Wehn, "A new architecture for high speed, low latency NB-LDPC check node processing for GF(256)," in *2016 IEEE 83rd Vehicular Technology Conference (VTC Spring)*, May 2016, pp. 1–5.

[12] C. Marchand and E. Boutillon, "NB-LDPC check node with pre-sorted input," in *2016 9th International Symposium on Turbo Codes and Iterative Information Processing (ISTC)*, Sept 2016, pp. 196–200.

[13] C. Poulliat, M. Fossorier, and D. Declercq, "Design of regular (2,dc)-LDPC codes over GF(q) using their binary images," *IEEE Transactions on Communications*, vol. 56, no. 10, pp. 1626–1635, October 2008.

[14] 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.

[15] E. Boutillon, L. Conde-Canencia, and A. A. Ghouwayel, "Design of a GF(64)-LDPC decoder based on the EMS algorithm," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 60, no. 10, pp. 2644–2656, Oct 2013.

[16] O. Abassi, L. Conde-Canencia, A. A. Ghouwayel, and E. Boutillon, "A novel architecture for elementary-check-node processing in nonbinary ldpc decoders," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 64, no. 2, pp. 136–140, Feb 2017.

[17] Y. L. Ueng, C. Y. Leong, C. J. Yang, C. C. Cheng, K. H. Liao, and S. W. Chen, "An efficient layered decoding architecture for nonbinary QC-LDPC codes," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 59, no. 2, pp. 385–398, Feb 2012.