Syndrome Based Check Node Processing of High Order NB-LDPC Decoders

Philipp Schläfer and Norbert Wehn
Microelectronic Systems Design Research Group
University of Kaiserslautern, Germany
{schlaefer,wehn}@eit.uni-kl.de

Matthias Alles and Timo Lehnigk-Emden
Creonic GmbH
Kaiserslautern, Germany
info@creonic.com

Emmanuel Boutillon
Lab-STICC, CNRS UMR 6285
Université de Bretagne Sud, France
emmanuel.boutillon@univ-ubs.fr

***Abstract*—Non-binary low-density parity-check codes have superior communications performance compared to their binary counterparts. However, to be an option for future standards, efficient hardware architectures must be developed. State-of-the-art decoding algorithms lead to architectures suffering from low throughput and high latency. The check node function accounts for the largest part of the decoders overall complexity. In this paper a new, hardware aware check node algorithm is proposed. It has state-of-the-art communications performance while reducing the decoding complexity. Moreover the presented algorithm allows for partially or even fully parallel processing of the check node operations which is not applicable with currently used algorithms. It is therefore an excellent candidate for future high throughput hardware implementations.**

I. Introduction

Upcoming standards like 5G will increase the demands on throughput and latency of communication systems. Especially applications like the Tactile Internet [1] require a significantly reduced latency. At the same time, error free transmission has to be guaranteed, which requires forward error correction schemes. Low-Density Parity-Check (LDPC) codes where first proposed by R.G. Gallager in 1963 [2] and rediscovered by D. Mackay and others in 1996 [3]. In the following almost two decades a lot of research has been carried out in this field. Today many commercial standards (WiMAX, WiFi, DVB-C2, DVB-S2X, DVB-T2) make use of LDPC codes. Very long binary LDPC codes have been proven to perform close to the Shannon limit. However when considering short blocks of only some hundred bit length for low latency applications, they suffer under degradation in communications performance of more than 0.5 dB. The extension of binary LDPC codes to Galois Fields (GF(q)) with $q > 2$ is a promising approach to solve this problem. Moreover the symbols of high-order modulation schemes can be directly mapped to the decoders input symbols. Thus an additional gain in communications performance is observed for systems combining high-order modulation and Non-Binary Low-Density Parity-Check (NB-LDPC) codes [4]. The performance gain of NB-LDPC codes comes at the cost of significantly increased decoding complexity. The decoding can be performed by message passing algorithms like Belief Propagation (BP). However the complexity increases with the size of the GF(q). A straightforward implementation of the BP algorithm has a complexity of $\mathcal{O}(q^2)$ [5]. In the last years several approaches have been proposed to reduce the decoding complexity without sacrificing the communications performance. Algorithms working in the Fourier domain [6] [7] have an excellent communications performance, but are still too complex for efficient hardware architectures. Symbol flipping algorithms [8] [9] in general have low complexity but suffer from heavily degraded communications performance. Approaches based on stochastic decoding [10] [11] have been presented as an alternative decoding method but introduce very high decoding latency. An extension of the well known binary Min-Sum algorithm to the non-binary domain, called Extended Min-Sum (EMS) algorithm [12] [13] [14] gives the best compromise between hardware complexity and communications performance. Therefore in this paper we will focus on the EMS algorithm which is the most promising starting point for efficient architectures.

To achieve the required throughput of today's applications, executing the decoding algorithms in software is not sufficient. Dedicated hardware architectures become mandatory. The largest complexity in the EMS algorithm is the computation of the Check Node (CN). State-of-the-art architectures apply a so called Forward-Backward (FWBW) scheme [15] to process the CN. A serial calculation is carried out to reduce the hardware cost and to allow for reuse of intermediate results during the computation. However this scheme introduces high latency and degrades the throughput. This effect increases significantly when the size of the GF(q) or the CN degree grows.

In this paper we propose a new hardware aware algorithm to perform the CN processing within the EMS decoding, we call it Syndrome-Based (SYN) CN processing. To reduce the initially high complexity we present several algorithmic modifications. While achieving slightly better communications performance than state-of-the-art hardware aware decoding algorithms, the SYN CN processing has a lower complexity. Moreover it allows for increased parallelism of the CN computation. Thus the SYN CN processing is the first hardware aware algorithm for NB-LDPC decoding, enabling low latency and high throughput decoder architectures.

The paper is structured as follows. In Section II we review the decoding of NB-LDPC codes making use of the EMS algorithm. Section III describes the state-of-the-art FWBW hardware aware decoding approach. Our new algorithm is presented in Section IV and further optimizations are discussed in Section V. Finally Section VI presents a comparison with other decoding methods in means of complexity and communications performance. Section VII concludes the paper.

II. EMS Decoding

This section reviews the EMS algorithm to give an overview of the complete decoding process.

Let us consider an (N, K) NB-LDPC code over GF(q) where N is the codeword length and K the number of information symbols. The code is defined by a sparse parity check matrix H with N columns, $M = N - K$ rows and elements $h_{m,n}$. The transmitted codeword consists of the codeword symbols $c = (c_1, c_2, \ldots, c_N), c_i \in \text{GF}(q)$. The decoder receives the noisy representation of the codeword symbols, $y = (y_1, y_2, \ldots, y_N)$.The decoding process can be partitioned in four main parts, the initialisation, the Variable Node (VN) update, the CN update and the permutations in the Permutation Nodes (PNs). It is important to mention that in contrast to a binary LDPC decoder instead of a single Logarithmic Likelihood Ratio (LLR) a set of q LLRs is exchanged between the nodes on a single edge. This fact accounts for the significant increase in complexity compared to binary LDPC decoding.

The first step in the EMS algorithm is the calculation of the symbol LLRs for the received codeword. For each received symbol y_i a set of q LLR values is calculated. Under the assumption that all GF(q) symbols are equiprobable, the initialization LLRs for VN v are calculated as follows:

$$\begin{aligned} \forall x \in \text{GF}(q) \left\{ L_v[x] = ln\left(\frac{P(y_v|c_v = \tilde{x}_v)}{P(y_v|c_v = x)}\right)\right\} \\ \text{with } \tilde{x}_v = \max_{\forall x \in \text{GF}(q)} \{P(y_v|c_v = x)\} \, . \end{aligned} \tag{1}$$

Given this definition it follows that $L_v[\tilde{x}_v] = 0$ and $L_v[x] \geq 0$ with an increasing LLR representing a decreasing symbol reliability. For simplified reading, in the following we use an array to represent the message sets. Messages from VN to PN are denoted as U_{vp}, messages from PN to CN as U_{pc}. Respectively messages from CN to PN are called V_{cp} and messages from PN to VN as V_{pv}.

In the first iteration the VNs simply forward the channel values (Eq. (1)) to the according CNs. The VN outputs are calculated as $U_{vp}[x] = L_v[x]$. In all other iterations the VN operation is to combine the channel values L_v with the d_v incoming message sets V_{pv}, where d_v is the VN degree. The updated extrinsic messages U_{vp} for VN v are calculated as follows:

$$U_{vp}[x] = L_v[x] + \sum_{t=1, t\neq p}^{d_v} V_{tv}[x], \forall x \in \text{GF}(q), p = 1 \ldots d_v \, . \tag{2}$$

The VN function is to sum up all values for a certain Galois field element received from the connected CNs and the according channel information $L_v[q]$. This however has to be performed for all q elements of the GF(q). To achieve the same message structure as before (LLR $= 0$ for the most reliable symbol, increasing LLR values for less reliable symbols), a normalisation of the U_{vp} messages with respect to the most reliable symbol has to be applied at the output of the VN.

The next step in the decoding is the permutation according to the parity check matrix H. The permutation of the VN v outputs U_{vp} is defined as:

$$U_{pc}[x] = U_{vp}[h_{c,v}^{-1} \cdot x], \forall x \in \text{GF}(q), p = 1 \ldots d_v \, , \tag{3}$$

where $h_{c,v}$ is the Galois field element at row c and column v of H, U_{pc} represents the input for CN c.

In the CN update d_c edges from U_{pc} are processed. The outputs for CN c are calculated as follows:

$$\begin{aligned} V_{cp}[x] = \min_{\forall \text{ permutations of } x_t} \sum_{t=1; t\neq p}^{d_c} U_{tc}[x_t] \\ \text{with } \sum_{t=1; t\neq p}^{d_c} x_t = x, \\ \forall x \in \text{GF}(q), p = 1 \ldots d_c \, . \end{aligned} \tag{4}$$

For every Galois field element x all possible input permutations fulfilling the parity check constraint are evaluated. The parity check constraint is given by the sum of the according Galois field elements. From all valid combinations the one with the highest reliability (smallest LLR) is chosen. Again, this task has to be performed for all q elements of the Galois field. The CN computation is the most complex part of the decoding and has a complexity of $\mathcal{O}(q^2)$ when calculated straightforward with the FWBW scheme [15].

Before one iteration is completed the outputs of CN c must be reverse permuted.

$$V_{pv}[x] = V_{cp}[h_{c,v} \cdot x], \forall x \in \text{GF}(q), p = 1 \ldots d_c \, . \tag{5}$$

This closes the loop and another iteration starts. The processing of a block is stopped as soon as a valid codeword is detected. To make this decision the estimated symbols $\hat{x}$ need to be computed for each VN v:

$$\hat{x}_v = \min_{x \in \text{GF}(q)} \left(L_v[x] + \sum_{p=1}^{d_v} V_{pv}[x] \right) \, . \tag{6}$$

In state-of-the-art EMS decoding one important simplification is applied. As it has been shown in [12] the sets of q messages exchanged between VNs and CNs can be truncated to carry only the n_m most reliable values per edge without sacrificing the communications performance. This approach significantly reduces the implementation complexity and is used for the algorithms presented in the following sections.

III. FWBW CN Processing

In this section we will first review the state-of-the-art decoding method for the CN calculation, the FWBW scheme.

The FWBW method applies a divide and conquer approach to the CN processing. Each CN processes d_c edges at a time following Eq. (4). The FWBW scheme splits the processing in three layers of $d_c - 2$ so called Elementary Check Nodes (ECNs). Each ECN processes only two edges at a time. Fig. 1 shows the resulting structure for a six input CN. Intermediate results of the ECNs are reused in the later stages and avoid recomputations. This significantly reduces the problem size and allows for efficient architectures for the ECN [16]. The processing of the ECN is based on the assumption that the input sets are sorted according to their LLR. Therefore the search space for the best elements can be reduced systematically. In [17] the authors present a low complexity scheme for the ECN processing which can be implemented efficiently. However, due to the serial processing it requires n_m clock cycles to perform the ECN task. For the complete CN an additional latency penalty of $2(d_c - 2) + 4$ clock cycles is

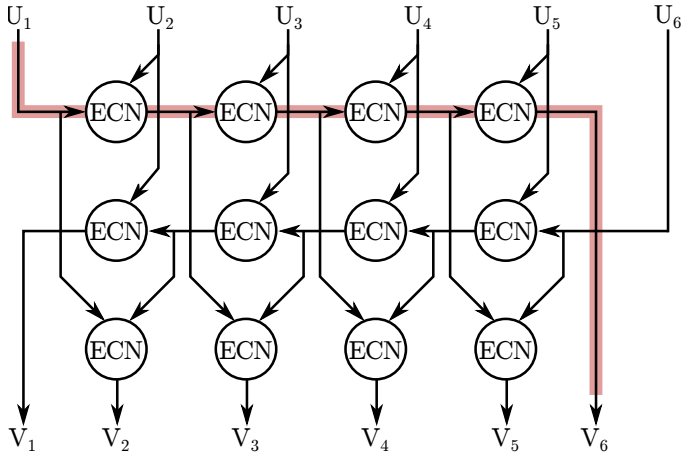

Fig. 1: CN architecture implementing the FWBW scheme. The red line marks one of the serial processing paths.

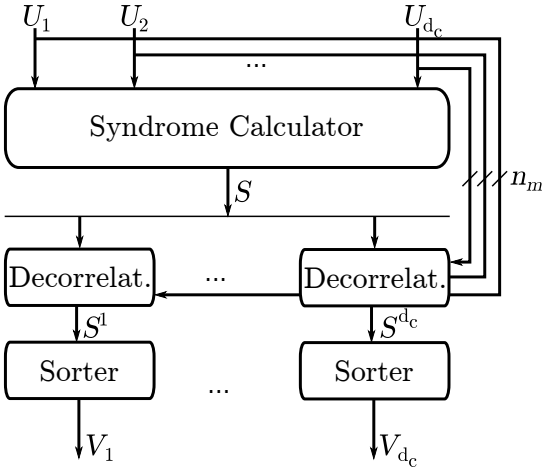

Fig. 2: Syndrome-based CN processing

introduced by the structure pointed out in Fig. 1, see [16] for a detailed timing analysis. The overall computation time in clock cycles for one CN is then calculated as follows:

$$T_{FWBW} = n_m + 2(d_c - 2) + 4\,. \tag{7}$$

This is the drawback of the FWBW approach. With increasing n_m and d_c the processing time for a CN rises and leads to high latency and low throughput of the complete NB-LDPC decoder. Thus the FWBW method for the CN processing is only an option for moderate sizes of the GF (n_m is closely coupled to q) and low Code Rates (CRs) as d_c increases significantly for high rate codes. Moreover a parallelization of the FWBW processing is hardly possible which makes low latency decoding infeasible.

IV. Syndrome-Based CN Processing

As discussed before, the state-of-the-art way of CN processing with the FWBW scheme has several drawbacks. Architectures making use of the FWBW scheme suffer from low throughput and high latency. Todays approaches to solve these issues are limited to small GF(q)s with $q \leq 16$ which have only small gain in Frame Error Rate (FER) compared to their binary counterparts [14]. Only with Galois fields of high-order significantly higher communications gains can be achieved. Therefore we propose a new algorithm, the so called SYN CN processing which can also be applied to Galois field sizes of practical interest ($q \geq 64$). The SYN CN cannot be only implemented in a serial, but also in a partially or even fully parallel fashion to achieve high throughput and low latency.

The basic structure of the SYN CN processing is depicted in Fig. 2. In the first step of the algorithm the syndromes are calculated. In contrast to the classical syndrome definition, in the following we refer with the term syndrome to the sum of one GF(q), LLR tuple from each input. As for each input U, n_m tuples can be chosen, there is not just one syndrome but a set of syndromes which we call S. Individual syndromes are distinguished by the elements which are chosen for the sum. Let K_{ic} be the set of n_m most reliable field elements from the ith edge of CN c. The syndrome reliability $R(x_1 \dots x_{d_c})$ and according Galois field element $G(x_1 \dots x_{d_c})$ for CN c are calculated as follows:

$$R(x_1 \dots x_{d_c}) = \sum_{t=1}^{d_c} U_{tc}[x_t], x_t \in K_{tc} \tag{8}$$

$$G(x_1 \dots x_{d_c}) = \sum_{t=1}^{d_c} x_t, x_t \in K_{tc} \tag{9}$$

One syndrome SYN is then defined as follows:

$$\text{SYN}(x_1 \dots x_{d_c}) = \left\{R(x_1 \dots x_{d_c}), G(x_1 \dots x_{d_c})\right\} \tag{10}$$

The syndrome set S contains all valid syndromes:

$$S = \left\{\text{SYN}(x_1 \dots x_{d_c}) : \forall x_1 \dots x_{d_c} \in K_{tc}\right\} \tag{11}$$

Calculating the syndromes in S as the sum of elements over all input edges (Eq. (8) and Eq. (9)), disregards one of the basic concepts of BP algorithms: In- and output of the same edge must not be correlated. Thus an additional step in the SYN CN processing is the decorrelation of in- and output:

$$R^i(x_1 \dots x_{d_c}) = R(x_1 \dots x_{d_c}) - U_{ic}[x_i], x_i \in K_{ic} \tag{12}$$

$$G^i(x_1 \dots x_{d_c}) = G(x_1 \dots x_{d_c}) - x_i, x_i \in K_{ic} \tag{13}$$

The result is a dedicated syndrome set S^i for every output i, which has no correlation with input i.

$$\text{SYN}^i(x_1 \dots x_{d_c}) = \left\{R^i(x_1 \dots x_{d_c}), G^i(x_1 \dots x_{d_c})\right\} \tag{14}$$

$$S^i = \left\{\text{SYN}^i(x_1 \dots x_{d_c}) : \forall x_1 \dots x_{d_c} \in K_{tc}\right\} \tag{15}$$

Once the S^i sets are computed, they are sorted by their syndrome reliability, represented by the LLRs. This gives direct access to the n_m most reliable syndromes which constitute the CN output sets V_{cp}.

The algorithm we proposed is an alternative to the conventional FWBW processing. It is the first efficient approach for high-order Galois field decoding, allowing for massive parallel implementations and thus high throughput and low latency. However, without special treatment the calculation of the syndrome set S and the sorting of S^i introduce a high complexity. It has to be reduced to make the algorithm attractive for an efficient hardware implementation.

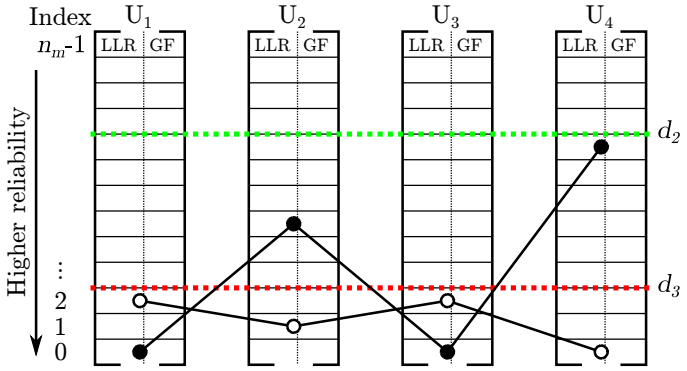

Fig. 3: Exemplary syndromes with two (filled circles) and three deviations (open circles) and maximum ranks d_2 and d_3.

V. Complexity Reduction of the Syndrome-Based CN Processing

In this section we are presenting an approach to reduce the complexity of the afore introduced SYN CN algorithm. The target is to allow the algorithm to be implemented efficiently in hardware. Therefore we discuss algorithmic modifications for simplifications of the syndrome set generation and the sorting while maintaining the communications performance.

A. Reducing the syndrome set cardinality:

The first step is to optimize is the calculation of the syndrome set S. For the output computation only the most reliable values of S are used which makes the computation of all other syndromes superfluous. Thus a smart reduction of the cardinality of S, i.e. $|S|$, can significantly reduce the overall complexity of the algorithm without sacrificing the communications performance.

The first step for a reduction of $|S|$ is the separation of syndromes with high reliability from ones with low reliability. In the following, a concept similar to the configuration sets introduced in [18] [19] is applied to the computation of S. We define $d_c + 1$ deviation sets D_i with $i \in 0 \dots d_c$ and separate the syndrome set in sub-sets:

$$S = \bigcup_{i=0}^{d_c} D_i \,. \tag{16}$$

Each set contains only syndromes deviating in exactly i elements from the most reliable element as shown in Fig. 3. The subset D_0 contains only one syndrome, which is the sum of the most reliable elements from all inputs. These sub-sets structure the data in a way that allows for easier access to syndromes with high reliability. Fig. 4 shows the average LLR values of the syndromes in the sorted deviation sets D_i. One can observe, that the distribution of reliable LLRs depends on the Signal-to-Noise Ratio (SNR). However, syndromes with more than two deviations i.e. D_3 and D_4 have such a low reliability that they rarely contribute to the generation of the outputs. Thus we can limit the calculation of sub-sets D_i to the ones with a low amount of deviations.

Another parameter for reduction of $|S|$ is the maximum allowed rank d_i of elements contributing to deviation D_i. The

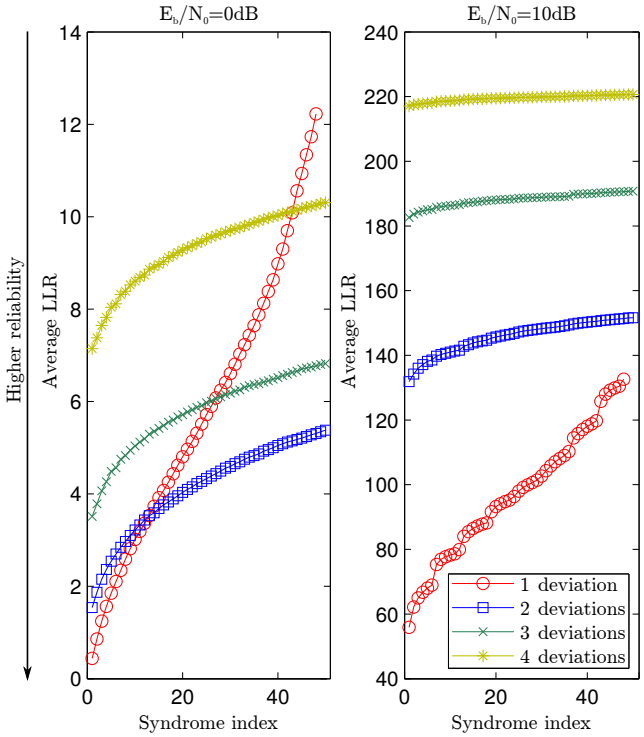

Fig. 4: Averaged LLR values of D_i for a GF(64), $d_c = 4$ code. For D_2, D_3 and D_4 we show only the 48 most reliable syndromes.

rank describes the position of the element in the sorted input set relative to the most reliable element. The most reliable element has the index zero. Less reliable elements have higher indices which describe their rank in the sorted list of LLRs. For the calculation of D_i only elements with indices less or equal to d_i are considered. The maximum allowed rank for a certain deviation can be set dynamically based on the LLR value of the elements or it is fixed, as a predefined parameter. For each deviation a different maximum rank can be set, see Fig. 3, e.g. the higher the number of allowed deviations, the lower the maximum rank of the deviations, $d_1 \geq d_2 \geq \dots \geq d_{d_c}$. The best trade off for d_i can be explored by simulation. Using this scheme implicitly keeps the best syndromes in each D_i and removes the less reliable ones. The cardinality of the subsets D_i can be calculated as follows:

$$|D_i| = \begin{cases} \binom{d_c}{i} \cdot (d_i)^i & \text{if } d_i \geq 1; i > 0 \\ 1 & \text{if } i = 0 \\ 0 & \text{else} \end{cases} \tag{17}$$

Combining both proposed techniques strictly reduces the cardinality of S and thus the computational complexity. The most reliable syndromes are calculated and only unreliable ones are removed. The parametrization for the number of deviations and their maximum rank is a critical step in the algorithm. Using for example only D_0, D_1 and D_2 with fixed ranks $d_0 = 0, d_1 = n_m - 1, d_2 = 2$, $d_c = 4$ and $n_m = 13$, shrinks $|S|$ from 28561 to 73. For a code in GF(64) this gives a very good trade-off between complexity and communications performance, see Section VI.

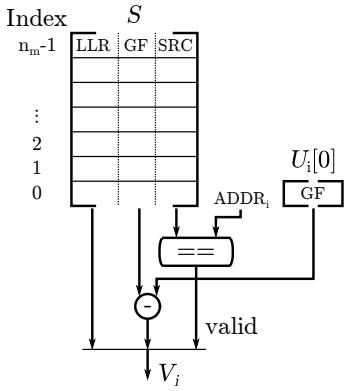

Fig. 5: Decorrelator for edge i.

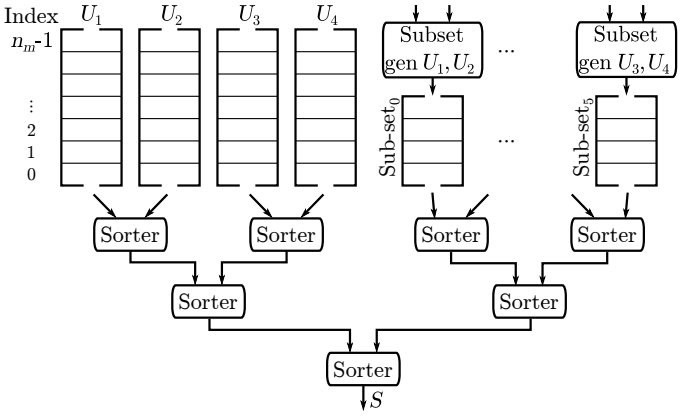

Fig. 6: Syndrome sorting for D_1 and D_2 with $d_c = 4$.

B. Simplifying sorting:

One big drawback of the original SYN CN algorithm presented in Section IV is that every syndrome set S^i must be sorted separately to output the n_m most reliable syndromes. This is due to the decorrelation step applied before. To avoid the sorting of the decorrelated syndrome sets S^i, a simple but effective approach can be used. Instead of decorrelating every value, only syndromes using the most reliable element (LLR $= 0$) from the currently handled edge are considered. All other syndromes are not used for the current edge output. By this approach the order of the syndromes is not changed and it is sufficient to sort S instead of the d_c S^i sets. In addition, the LLR values are not modified in the decorrelation step which saves a real valued subtraction for every output message. Finally only the most reliable input element and not the complete input sets must be stored for the decorrelation.

Fig. 5 shows the schematic operations of one decorrelator. Each syndrome is denoted with the additional information about which of the input edges contributed to the syndrome with a deviation. SRC in Fig. 5 stores the edges where deviations occurred and ADDR$_i$ represents the current edge. A simple comparison evaluates if a deviation from the current edge was involved in the syndrome calculation and thus if the syndrome is valid for the current edge or not. Only if no deviation occurred on the current edge, the decorrelated message is marked as valid and used for the output V_i.

Even though the sorting has been reduced to the syndrome set S, there is more potential for simplification. Sorting S can be divided into sorting the deviation sets D_i and merging them. Especially for D_1 the sorting can be further simplified. This is achieved due to the previous knowledge we have of the input data. We implicitly know that the sets U_{pc} are sorted according to their LLRs. The sorting of D_1 thus is limited to merging d_c sorted sets. Performed serially, this is a trivial task that introduces only minimal hardware overhead.

For the higher-order deviations D_i for $i \geq 2$, the sorting can also be simplified because of the sorted input sets. Sorted sub-sets can be generated with little effort which only have to be merged to achieve the final set. An example of the sub-set generation for D_2 with $d_2 = 2$ is given in Fig. 7 which can be extended easily to other deviations and ranks. Once the sub-sets D_i are sorted, the outputs are generated by merging them iteratively as shown in Fig. 6. In the presented case with $d_c = 4$ exactly six sub-sets exist which are then merged for further processing.

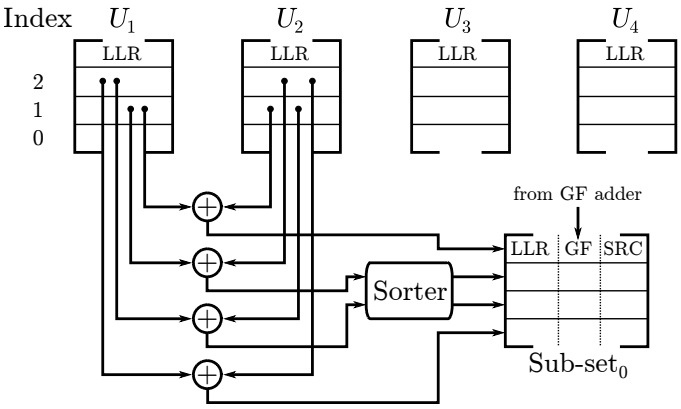

Fig. 7: Sub-set generation for D_2 with $d_2 = 2$, and SRC denoting the deviation positions.

The proposed algorithmic modifications result in a slightly different data flow, see Fig. 2 and Fig. 8. Summarized, three algorithmic transformations were introduced in this section to reduce the complexity:

- Significant reduction of $|S|$.
- Simplified sorting of S instead of S^i.
- No LLR subtractions and no storage for U_i in the decorrelation step.

VI. Communications Performance and Complexity Comparison

In the following section we discuss the quality of the proposed algorithm in means of communications performance and decoding complexity in means of required basic operations.

All results for the communications performance are obtained with a bit true C++ model using Binary Phase Shift Keying (BPSK) modulation and an Additive White Gaussian Noise (AWGN) channel. The simulated NB-LDPC code has a code word length N of 16 GF(64) symbols (96 bits), a CR of 0.5, $d_v = 2$ and $d_c = 4$. The code has been generated

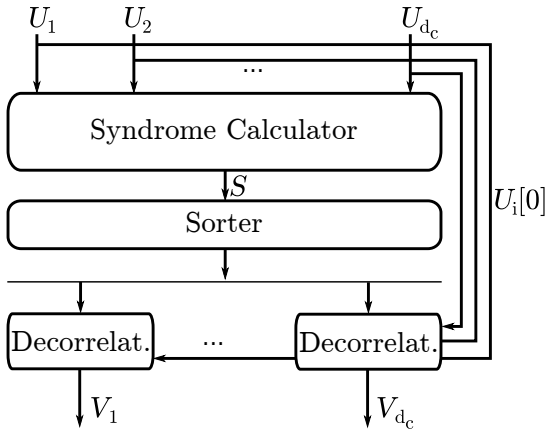

Fig. 8: Modified SYN CN processing after algorithmic transformation

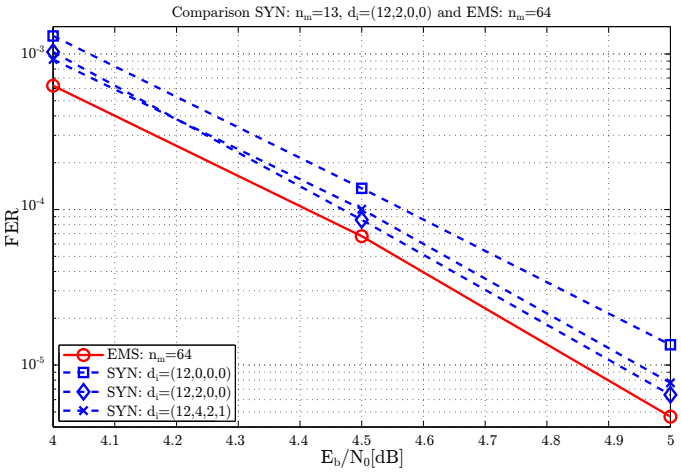

Fig. 9: Communications performance of the SYN CN compared to EMS decoding.

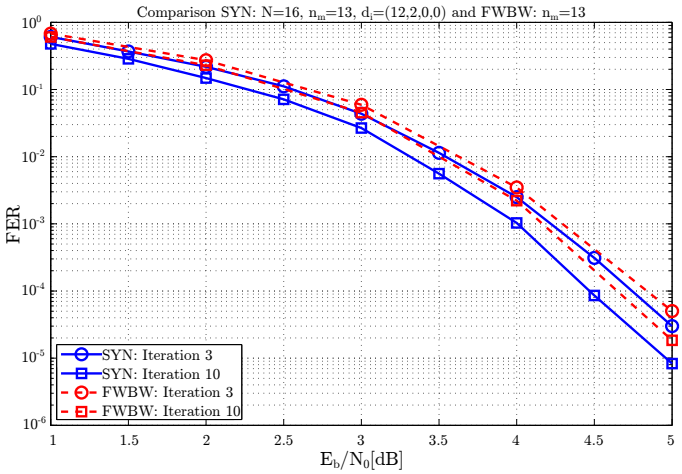

Fig. 10: Communications performance of the SYN CN compared to the state-of-the-art FWBW scheme.

TABLE I: Comparison of algorithmic complexity for GF(64) and GF(256). Parameters for GF(64) $n_m = 13$, $d_1 = n_m - 1$, $d_2 = 2$. Parameters for GF(256) $n_m = 61$, $d_1 = n_m - 1$, $d_2 = 3$.

Component	LLR add.	LLR comp.	GF add. (XOR)
FWBW CN GF(64)	102 (100%)	234 (100%)	102 (100%)
SYN CN GF(64)	24 (**24%**)	186 (**79%**)	271 (265%)
FWBW CN GF(256)	390 (100%)	1080 (100%)	390 (100%)
SYN CN GF(256)	60 (**15%**)	936 (**87%**)	787 (201%)

TABLE II: Comparison of possible hardware mappings.

Algorithm	FWBW CN	SYN CN
Hardware Mapping	serial	serial / partially parallel / fully parallel

based on the approach presented in [20]. Both, the FWBW and the SYN CN make use of truncated vectors of size $n_m = 13$, and perform a maximum of 10 iterations with a two-phase scheduling. A bit true fixed-point model with 8 bits for the LLR representation is implemented. Fig. 9 shows the performance of different implementations of the EMS algorithm compared with an optimal EMS decoding (no message truncation). For the syndrome based CN we have considered up to four deviations with $d_i = (d_1, d_2, d_3, d_4)$. A second comparison shows the difference with respect to the state-of-the-art hardware aware FWBW decoding. After ten iterations the syndrome based CN computation has a superior performance compared to the FWBW implementation, see Fig. 10.

To compare the complexity of the proposed algorithm with state-of-the-art decoding methods Table I lists the number of required basic operations. On the one hand, the SYN CN algorithm requires additional GF(q) additions. In a hardware architecture they can be implemented with simple XOR logic and thus are very cheap in means of required area. On the other hand, the SYN approach uses only a small fraction of adders (24%) and comparators (79%) for LLR values compared to the FWBW algorithm. These operations require real valued adders which are way more expensive than GF(q) adders. GF additions can be implemented as a bitwise XOR operation in hardware while a real valued addition requires complex adder architectures e.g. carry-save adders. Thus the comparison shows a significant benefit for the SYN CN algorithm. Finally our investigation shows, that this holds also for higher-order Galois fields.

Regarding future hardware architectures, the proposed algorithm generates a whole new design space. Both, serial and partially parallel architectures can be explored, see Table II. A serial processing may achieve higher efficiency in means of throughput per area, than state-of-the-art architectures but suffers from the same drawbacks, i.e. high latency and low throughput. Another possibility is the parallelization of the syndrome generation and sorting, leading to a high throughput and low latency architecture. First hardware experiments show promising results for the area efficiency of the proposed architectures. In the future, an extensive study on architectures will be carried out.

VII. Conclusion

We have presented a new hardware aware algorithm for the CN processing of NB-LDPC decoders. Our investigations have shown slightly better communications performance compared to state-of-the-art hardware aware decoding algorithms. In addition a comparison of the algorithmic complexity reveals that the proposed SYN CN processing has a lower complexity. The SYN CN algorithm is the first hardware aware CN processing, allowing for low latency and high throughput decoder architectures for high-order Galois fields. In the next step we will design hardware architectures based on the proposed algorithm to further explore its potential.

References

[1] G. Fettweis, "The Tactile Internet: Applications and Challenges," *IEEE Vehicular Technology Magazine*, vol. 9, no. 1, pp. 64–70, 2014. [Online]. Available: http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=6755599

[2] R. G. Gallager, "Low-Density Parity-Check Codes," *IRE Transactions on Information Theory*, vol. 8, no. 1, pp. 21–28, Jan. 1962.

[3] D. MacKay and R. Neal, "Near Shannon limit performance of Low-Density Parity-Check Codes," *Electronic Letters*, vol. 32, pp. 1645–1646, 1996.

[4] S. Pfletschinger and D. Declercq, "Getting Closer to MIMO Capacity with Non-Binary Codes and Spatial Multiplexing," in *Global Telecommunications Conference (GLOBECOM 2010), 2010 IEEE*, 2010, pp. 1–5. [Online]. Available: http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=5684077

[5] M. Davey and D. MacKay, "Low-Density Parity Check Codes over GF(q)," *IEEE Communications Letters*, vol. 2, no. 6, pp. 165–167, Jun. 1998.

[6] L. Barnault and D. Declercq, "Fast decoding algorithm for LDPC over GF(2q)," in *Proc. IEEE Information Theory Workshop*, Mar.–Apr. 2003, pp. 70–73.

[7] T. Lehnigk-Emden and N. Wehn, "Complexity Evaluation of Non-binary Galois Field LDPC Code Decoders," in *Proc. 6th International Symposium on Turbo Codes & Iterative Information Processing (ISTC'2010)*, Brest, France, Sep. 2010.

[8] Y. Lu, N. Qiu, Z. Chen, and S. Goto, "An efficient majority-logic based message-passing algorithm for non-binary LDPC decoding," in *Circuits and Systems (APCCAS), 2012 IEEE Asia Pacific Conference on*, 2012, pp. 479–482. [Online]. Available: http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=6419076

[9] F. Garcia-Herrero, D. Declercq, and J. Valls, "Non-Binary LDPC Decoder Based on Symbol Flipping with Multiple Votes," *IEEE Communications Letters*, vol. 18, no. 5, pp. 749–752, 2014. [Online]. Available: http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=6787156

[10] A. Ciobanu, S. Hemati, and W. Gross, "Adaptive Multiset Stochastic Decoding of Non-Binary LDPC Codes," *IEEE Transactions on Signal Processing*, vol. 61, no. 16, pp. 4100–4113, 2013. [Online]. Available: http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=6519310

[11] C.-W. Yang, X.-R. Lee, C.-L. Chen, H.-C. Chang, and C.-Y. Lee, "Area-efficient TFM-based stochastic decoder design for non-binary LDPC codes," in *Circuits and Systems (ISCAS), 2014 IEEE International Symposium on*, 2014, pp. 409–412. [Online]. Available: http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=6865152

[12] D. Declercq and M. Fossorier, "Decoding Algorithms for Nonbinary LDPC Codes Over GF," *IEEE Transactions on Communications*, vol. 55, no. 4, pp. 633–643, 2007. [Online]. Available: http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=4155118

[13] 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, 2010. [Online]. Available: http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=5464236

[14] 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, 2013. [Online]. Available: http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=6516166

[15] H. Wymeersch, H. Steendam, and M. Moeneclaey, "Log-domain decoding of LDPC codes over GF(q)," in *Proc. IEEE International Conference on Communications*, vol. 2, Jun. 2004, pp. 772–776.

[16] E. Boutillon, L. Conde-Canencia, and A. Al 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, 2013. [Online]. Available: http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=6595153

[17] 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, 2010. [Online]. Available: http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=5457396

[18] D. Declercq and M. Fossorier, "Extended minsum algorithm for decoding LDPC codes over GF(q)," in *Proc. International Symposium on Information Theory ISIT 2005*, Sep. 2005, pp. 464–468.

[19] E. Li, K. Gunnam, and D. Declercq, "Trellis based Extended Min-Sum for decoding nonbinary LDPC codes," in *Wireless Communication Systems (ISWCS), 2011 8th International Symposium on*, 2011, pp. 46–50. [Online]. Available: http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=6125307

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