Comparison of Code Constructions Suitable for High-Throughput Decoding

Sergei Semenov
HiSilicon
Lund Research Center, Sweden
e-mail: sergei.semenov@huawei.com

Jiye Liang
HiSilicon
Beijing, China
e-mail: liangjiye@huawei.co

Mingxu Zhang
HiSilicon
Beijing, China
e-mail: zhangmingxu2@hisilicon.com

***Abstract*—Two classes of codes allowing high-throughput decoding: Spatially Coupled Low Density Parity-Check (SC-LDPC) codes and staircase codes are compared. Accumulate Repeat-Jagged (ARJ)-based SC-LDPC codes provide better performance and lower complexity for Soft-Decision Decoding (SDD). However, the serious drawback of this construction is the severe performance degradation with code rate increase. The decoding complexity of SDD might still be too high to provide very high throughput. Reed-Solomon (RS)-based staircase codes under Hard-Decision Decoding (HDD) provide quite good performance with low decoding complexity. Moreover, the performance changes very smoothly with code rate increase.**

Keywords-SC-LDPC codes; Generalized Product Codes; staircase codes; high-throughput decoding.

I. INTRODUCTION

Future Beyond-5G use cases are expected to require wireless speeds in the Terabit/s range. This sets a number of tough challenges on the physical layer and especially on the Forward Error Correction (FEC). The code constructions used in current 3GPP specs can hardly be used to provide this level of throughput under channel conditions considered in use cases. Apparently, some specific requirements should be applied to the choice of a code construction allowing high throughput decoding. The decoding complexity should be low enough and the decoding algorithm should be suitable for a high level of parallelization. Especially useful for high throughput decoding are codes with high *locality* property allowing the decoder to use structures that are independent of code length in terms of complexity, storage requirements and latency.

Spatially coupled codes are known for both high locality and high performance. In this paper, we are comparing two code constructions, one of which represents the class of SC-LDPC codes and another the Generalized Product Codes (GPC). The comparison is done not only in terms of performance/complexity, but also the ability to create the code family that can be easily adapted to different code rates is considered.

The rest of this paper is organized as follows. Section II describes the code constructions considered in this paper. Section III describes the decoding of the considered codes. Section IV addresses the comparison of the considered code constructions with the focus on ability to provide high throughput. The conclusions close the article.

II. CONSIDERED CODE CONSTRUCTIONS

The main principle of spatial coupling is that the codewords $\mathbf{v}_t$ of the block code defined by the parity-check matrix $\mathbf{H}$, instead of being encoded independently, are interconnected (coupled) with their neighbors at times $t-1, t-2, \dots, t-w$ during the encoding procedure. This is done in such a way that the sequence satisfies the condition

$$\mathbf{v}_t\mathbf{H}_0^T(t) + \mathbf{v}_{t-1}\mathbf{H}_1^T(t) + \cdots + \mathbf{v}_{t-w}\mathbf{H}_w^T(t) = \mathbf{0}, \quad (1)$$

where matrices $\mathbf{H}_0(t), \mathbf{H}_1(t), \dots, \mathbf{H}_w(t)$ result from the decomposition of the original matrix $\mathbf{H}$ [6]:

$$\mathbf{H}_0(t) + \mathbf{H}_1(t) + \cdots + \mathbf{H}_w(t) = \mathbf{H}, \ \forall t. \quad (2)$$

It is easy to verify that both code constructions considered in this paper satisfy (1) and (2), therefore, they define spatially coupled codes.

A. SC-LDPC codes

The idea of SC-LDPC codes was proposed in [1]. It can be interpreted as generalization of block and convolutional coding where the convolutional coding is applied to the words of some block code rather than to information symbols. One of the possible ways of constructing SC-LDPC codes is constructing with the help of coupling of protographs. Recall that a protograph can be considered as a graph representing the general structure of a graph-based code, e.g., LDPC code. A protograph can be represented with the help of a Base-Graph (BG) matrix $\mathbf{B}$ where the element $b_{i,j}$ shows the number of edges connecting Check Node (CN) i and Variable Node (VN) j in the protograph. The base-graph matrix $\mathbf{B}$ can be translated to the parity-check matrix $\mathbf{H}$ by substituting each element in $\mathbf{B}$ by the corresponding permutation matrix of size $(M \times M)$, where M is the code lifting size. If the number of edges connecting CN i and VN j is more than 1 and ,e.g., is

equal to k, the corresponding matrix of size $(M \times M)$ should be obtained as a sum of k permutation matrices. For example, consider the protograph shown in Figure 1.

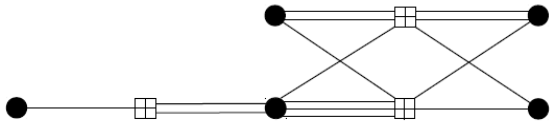

Figure 1. Example of an ARJ protograph.

The corresponding base-graph matrix and the example of parity-check matrix of LDPC code with lifting size $M = 3$ are represented in (3) and (4):

$$\mathbf{B} = \begin{bmatrix} 2 & 0 & 0 & 0 & 1 \\ 3 & 1 & 1 & 1 & 0 \\ 1 & 2 & 1 & 2 & 0 \end{bmatrix}, \quad (3)$$

$$\mathbf{H} = \begin{bmatrix} 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}. \quad (4)$$

Notice that, without lifting, the protograph shown in Figure 1 with BG matrix (3) cannot represent any code and matrix (4) is not a unique representation of BG (3). It is possible to construct the SC-LDPC code by splitting the original BG into a set of BG matrices $\mathbf{B}_i$ in such a way that

$$\mathbf{B} = \sum_{i=0}^{w} \mathbf{B}_i. \quad (5)$$

where w is the SC memory.

Then, the BG of the SC-LDPC code can be written in the form of

$$\mathbf{B}_L = \begin{bmatrix} \mathbf{B}_0 & & & & & & & \\ \mathbf{B}_1 & \mathbf{B}_0 & & & & & & \\ \vdots & \ddots & \ddots & & & & & \\ \mathbf{B}_w & \cdots & \mathbf{B}_1 & \mathbf{B}_0 & & & & \\ & \ddots & & \ddots & \ddots & & & \\ & & \ddots & & \ddots & \mathbf{B}_0 & & \\ & & & \ddots & & \mathbf{B}_1 & & \\ & & & & \ddots & \vdots & & \\ & & & & & \mathbf{B}_w & & \end{bmatrix}, \quad (6)$$

where L is the number of transmitted consecutive blocks (initial protographs), i.e., defining the block code of length MLb_v with the number of CNs $M(L + w)b_c$, where b_v is the number of BG VNs and b_c is the number of BG CNs. The rate of the corresponding code is

$$R_L = \frac{Lb_v - (L+w)b_c}{(L+w)b_c}. \quad (7)$$

As can be seen from the structure of the BG matrix (6), the CN degrees at both ends of the code are lower. It should be noted that CNs with lower degrees (at the ends) improve the BP decoding. This irregularity caused by the fact that w top and w bottom blocks of CNs in (6) are of lower degrees is one of the main reasons for the excellent performance of SC-LDPC codes [4]: the small-degree CNs serve as starting points for the convergence of the iterative decoding process such that a "wave" of reliable information propagates towards the middle of the codewords.

A BP decoding analysis of SC-LDPC codes shows that the performance of the iterative decoder is improved significantly by spatial coupling. In fact, the results in [4] and [5] show that asymptotically, as $L \to \infty$, the BP threshold is boosted to that of the optimal Maximum a Posteriori (MAP) decoder. Since the MAP thresholds of regular LDPC ensembles with increasing node degrees are known to converge to capacity, it follows that spatial coupling provides a new way of provably achieving capacity with low-complexity iterative BP decoding [6].

The Quasi-Cyclic LDPC (QC-LDPC) codes are of special interest due to the fact that they have efficient encoder and decoder implementations. If each $(M \times M)$ permutation matrix substituting non-zero elements in BG is a circulant, the corresponding matrix **H** defines a QC-LDPC code. In [2]-[3], the QC-LDPC codes were thoroughly studied and the effective code construction with ensemble minimum distance growing linearly with block size was proposed. This code family is called ARJ construction. An example of a BG representing ARJ construction is shown in Figure 1 [3].

The ARJ-based SC-LDPC code with memory $w = 2$ was chosen for the comparison in this paper. As it was mentioned above the Belief Propagation (BP) decoding threshold approaches capacity for all SC-LDPC codes with $L \to \infty$. Except this the ARJ-based SC-LDPC codes show better than regular code based SC-LDPC codes minimum distance growth rate [8]. That means minimum distance increases linearly with block length n and for ARJ-based SC-LDPC this increase is more than for SC-LDPC codes based on regular codes.

After simple optimization, the following partitioning of the BG (3) was chosen

$$\mathbf{B}_0 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 \end{bmatrix},$$

$$\mathbf{B}_1 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 \end{bmatrix}, \quad (8)$$

$$\mathbf{B}_2 = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \end{bmatrix}.$$

In case lifting sub-matrices are chosen to be a circulant (leading to QC code), the optimization of the lifting

permutation sub-matrices is reduced to the choice of the circulant powers. The corresponding lifting powers were optimized for lifting size $M = 8$.

B. Staircase codes

Staircase codes and braided block codes comprise the two most known representatives of the GPC [17]. Moreover, both constructions provide quite similar performance. For this reason, it was decided to limit the scope of the GPC study to staircase codes only.

Staircase codes introduced in [15] can also be considered as an example of spatially coupling principle. The staircase code construction combines ideas from recursive convolutional coding and block coding. Staircase codes are completely characterized by the relationship between successive matrices of symbols. Specifically, consider the (infinite) sequence $B_0, B_1, B_2, \ldots$ of $(m \times m)$ matrices B_i. For simplicity, consider each matrix B_i as a binary matrix (it can be generalized to non-binary case as well).

Block B_0 is initialized to a reference state known to the encoder–decoder pair, e.g., an $(m \times m)$ matrix of zero symbols. Furthermore, select a conventional linear block code (e.g., Hamming, Bose–Chaudhuri–Hocquenghem (BCH), RS, etc.) in systematic form to serve as the component code; this code C, is selected to have block length $2m$ symbols, of which r are parity symbols. Encoding proceeds recursively on the B_i. For each i, $m(m-r)$ information symbols (from the streaming source) are arranged into the $(m-r)$ leftmost columns of B_i; we denote this submatrix by $B_{i,L}$. Then, the entries of the rightmost r columns (this submatrix is denoted by $B_{i,R}$) are specified as follows.

1) Form the $\left(m \times (2m-r)\right)$ matrix $A = \left[B_{i-1}^T\ B_{i,L}\right]$.

2) The entries of $B_{i,R}$ are then computed such that each of the rows of the matrix $\left[B_{i-1}^T\ B_{i,L}\ B_{i,R}\right]$ is a valid codeword of C. That is, the elements in the jth row of $B_{i,R}$ are exactly the r parity symbols that result from encoding the $2m-r$ "information" symbols in the jth row of A.

Generally, the relationship between successive blocks in a staircase code satisfies the following relation: for any $i \geq 1$, each of the rows of the matrix $[B_{i-1}^T\ B_i]$ is a valid codeword in C. An equivalent description, from which the term "staircase codes" originates is suggested in Figure 2, in which (the concatenation of the symbols in) every row (and every column) in the "staircase" is a valid codeword of C.
The rate of a staircase code is

$$R = 1 - \frac{r}{m}, \tag{9}$$

since encoding produces r parity symbols for each set of $m-r$ "new" information symbols.

At the end of the information sequence, termination can be used to protect the final information block. In this case, after L information blocks enter the encoder, Λ additional all-zero blocks enter the encoder. Note that the all-zero blocks are not sent over the channel, but the resulting parity bits are transmitted.

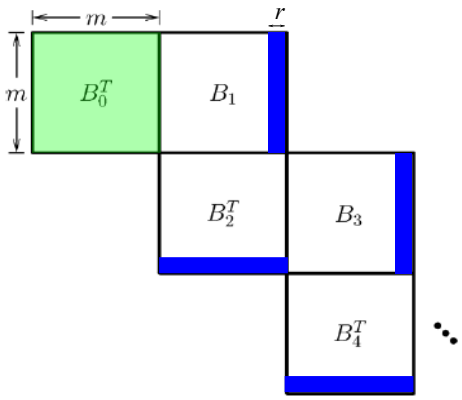

Figure 2. "Staircase" visualization of staircase codes [15].

Then, the actual rate of the staircase code, including the tail, is given by

$$R = \frac{m-r}{m+\Lambda\frac{r}{L}}. \tag{10}$$

The staircase codes are well suited for HDD. In this case, low complexity syndrome decoder can be used for decoding a component code. Efficient, high-throughput table-lookup based methods for decoding the component codes are highlighted in [15]. On the other hand, SDD is also possible to use for a component code decoder [16].

It is easy to verify that the structure of the staircase code parity-check matrix is very similar to the structure of SC-LDPC code parity-check matrix (6):

$$\mathbf{H}_{St} = \begin{bmatrix} \mathbf{H}_2 & \mathbf{H}_1 & \mathbf{H}_0 & \mathbf{0} & \ldots & \ldots & \mathbf{0} & \ldots \\ \mathbf{0} & \mathbf{0} & \mathbf{H}_2 & \mathbf{H}_1 & \mathbf{H}_0 & \mathbf{0} & \mathbf{0} & \ldots \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{H}_2 & \mathbf{H}_1 & \mathbf{H}_0 & \ldots \\ \vdots & \vdots & \vdots & \vdots & \mathbf{0} & \mathbf{0} & \mathbf{H}_2 & \ddots \end{bmatrix}. \tag{11}$$

The difference is that, in the staircase code parity-check matrix, the number of $(2m \times 2m)$ submatrices $\mathbf{H}_i$ is exactly 3, and each bunch of rows in (11) is shifted by the size of 2 sub-matrices. One more attractive property of the staircase codes is that the minimum distance of the staircase code is no less than d_{min}^2, where d_{min} is the minimum distance of a component code [15].

III. Decoding of the Considered Codes

In this study, the SC-LDPC codes were decoded with the help of SDD only. Decoding of the staircase codes was considered for both SDD and HDD.

A. Decoding of SC-LDPC codes

The convolutional structure of SC-LDPC allows to define a latency constrained decoder using a sliding window of size W. Consider the blocks of VNs of size Mb_v. Due to the convolutional structure of matrix (6) two VN blocks with indices i and j, such that $j \geq i + w + 1$, do not share any parity-check equation, i.e., VNs from these blocks cannot be

connected to the same CN. Window decoder exploits this property of the convolutional parity-check matrix of SC-LDPC code to define a decoder that deals with W received blocks such that $W \geq w + 1$. It has been shown in [7] that SC-LDPC codes decoded with a window decoder outperforms LDPC block codes under equal latency.

The most common window decoding uses the VN-centered strategy where a decoding window of size W is defined by the set of VNs for W consecutive blocks (of size Mb_v each). Some VNs on the left-hand side of the decoding window then share CNs with VNs that have already moved out of that window. CN → VN messages sent along the corresponding edges are not updated any longer, i.e., they are read-only. Similarly, some VNs on the right-hand side of the decoding window share CNs with VNs that have not yet been processed by the WD. The messages along the corresponding edges are also not updated in the current window. In terms of parity-check matrix, the sliding window decoder of size W operates on a section of WMb_c rows (CNs) and WMb_v columns (VNs) of the matrix $\mathbf{H}_L$, corresponding to W coupled blocks. Figure 3 depicts the decoding window of size 4 when assuming an SC-LDPC code with memory $w = 2$. The grey rectangles correspond to the parts of parity-check matrix not involved in the decoding process. The parts of parity-check matrix corresponding to currently (at moment t) updated VNs are marked with red. Other parts of parity-check matrix belonging to the same window are marked with blue. Performing updates on VNs in the window still requires access to messages sent along edges connected to previously decoded VNs, i.e., parts of matrix depicted by green. However, those accesses are read-only. Moving the sliding window to the next position means shifting it down by Mb_c CNs and right by Mb_v VNs.

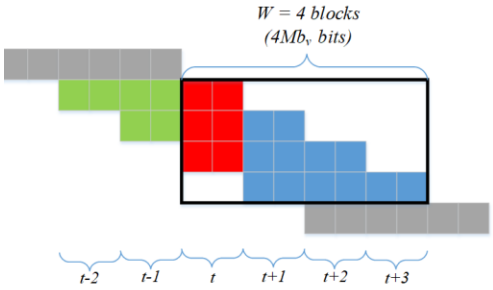

Figure 3. Parts of parity-check matrix involved in window decoding.

The window BP decoder consists of WMb_c CNs and WMb_v VNs. The decoding performance depends on the sub-block size b_v and the memory w, rather than on L. This *locality* property allows using decoder structures that are independent of L in terms of complexity, storage requirements and latency. Since the window size W is a decoder parameter, it can be varied without changing the code, providing a flexible trade-off between performance and latency [7]. In general, the storage requirements for the decoder reduces by a factor of $\frac{L}{W}$ compared to the BP decoder operating on the length of the whole codeword.

It is assumed that the window decoder uses all iterations locally inside one window position and only after fully decoding target VNs, the window is shifted to the next position. That makes it possible to unroll the iterations, which can significantly decrease the overall latency up to N_{ItMax} times (where N_{ItMax} is the maximum number of iterations), keeping the area requirements not too high. The unrolled window decoder requires CN network of $N_{ItMax}WMb_c$ CN processors, rather than $N_{ItMax}LMb_c$ CN processors in case of decoding the whole codeword.

The complexity of a decoder can be roughly estimated as follows.

Each CN input should be updated once for each layer. Considering the structure of sub-matrices (8) we can assume that each VN in window is updated on average 3 times before being used in CN processing, which translates to $3WMb_v$ additions ($120W$ additions for $M = 8, b_v = 5$) per window per iteration. On average, each CN is connected to 5 VNs, i.e., each CN processing involves on average 20 box-plus operations for SP or 20 min operations for MS (since the CN output should be generated for each of 5 connected VNs and each CN output involves $(5 - 1) = 4$ box-plus or min operations). That gives $20WMb_c = 480W$ box-plus or min operations per window per iteration. We can assume that on average $5M$ LLRs are updated at each layer, which translates to $5MWb_c = 120W$ additions per window per iteration. Then, the overall decoding complexity can be estimated as $240LN_{ItMax}W$ additions and $480LN_{ItMax}W$ box-plus or min operations for SP and MS algorithm correspondingly, where L is the number of blocks (code length is $MLb_v = 40L$), and N_{ItMax} denotes maximum number of iterations. The required memory can be estimated as $WMb_c = 24W$ elements to store CN outputs in current window.

B. *Decoding of staircase codes*

Staircase codes in this study were decoded both with SDD and HDD methods. In SDD, the one-sweep optimal decoding [10] was used for a component code decoding. In HDD mostly some low complexity modifications of Bounded Distance Decoding (BDD) were applied for decoding of a component code.

Similarly to the SC-LDPC code decoding, the SDD of the whole staircase code is based on the concept of windowing decoding. Consider the iterative decoding for window of length $W = 3$ for simplicity. Assuming that the target block is B_i, blocks B_{i-1}, B_i and B_{i+1} are involved in the iterations. Denote by $\mathbf{L}_j^{(ch)}$ the channel Log-Likelihood Ratios (LLRs) corresponding to $(m \times m)$ block B_j, by $\mathbf{L}_j^{(app)}$ the a posteriori probability (APP) LLRs corresponding to APPs obtained after decoding the component codes, by $\mathbf{L}_j^{(a)}$ the a priori LLRs, and by $\mathbf{L}_j^{(e)}$ the extrinsic LLR corresponding to block B_j. For window length $W = 3$ the extrinsic LLRs corresponding only to block B_i are exchanged in iterations. At first half-iteration the input for the first m symbols of the received sequence $\mathbf{y}$ is formed by LLRs chosen from $\mathbf{L}_{i-1}^{(app)}$ and the input for symbols $y_{m+1}, \ldots, y_n$ is formed by the LLRs from the sum $\mathbf{L}_i^{(a)} + \mathbf{L}_i^{(ch)}$.

We assume that block B_{i-1} is already decoded, or in case B_1 is the target block, block B_0 is known a priori. At first iteration a priori LLRs are assumed to be zero, i.e., $\mathbf{L}_i^{(a)} = \mathbf{0}_{m\times m}$. After decoding of m codewords, the extrinsic LLRs are formed:

$$\mathbf{L}_i^{(e)} = \mathbf{L}_i^{(app)} - \mathbf{L}_i^{(a)}, \quad (12)$$

and the extrinsic LLRs $\mathbf{L}_i^{(e)}$ are provided for the second half-iteration as an a priori information. At second half-iteration first m input LLRs are chosen from the sum $\mathbf{L}_i^{(a)} + \mathbf{L}_i^{(ch)}$, where $\mathbf{L}_i^{(a)}$ is substituted by the extrinsic LLRs (12) obtained at first half iteration. The input LLRs corresponding to symbols $y_{m+1}, \dots, y_n$ are chosen from the LLRs $\mathbf{L}_{i+1}^{(ch)}$. After decoding, the extrinsic LLRs are provided as an a priori LLRs for the next iteration.

Now consider the computation complexity of staircase code SDD. The number of operations (additions and multiplications) required for one-sweep decoding is half of needed for the Bahl, Cocke, Jelinek and Raviv (BCJR) algorithm and can be estimated as $O(n2^{n-k})$ per one codeword, i.e., $O(2m(W-1)n2^{n-k})$ operations per iteration per target block. The required memory corresponds to storage of one section of the trellis, i.e., 2^{n-k}. Calculation of the extrinsic information requires $2m^2(W-2)$ additions per iteration per block. Then, the overall computation complexity can be estimated as $LN_{ItMax}\big(2m^2(W-2) + O(2m(W-1)n2^{n-k})\big)$ operations, where L denotes the number of blocks and N_{ItMax} maximum number of iterations. Obviously, the overall complexity of SDD grows exponentially with $(n-k)$. Thus, only codes with the low correcting capability can be considered for a component code.

HDD imposes serious performance loss in comparison with SDD (usually around 2 dB). On the other hand, using HDD of a component code can be very attractive from the computational complexity point of view. Another argument supporting HDD is that it provides more flexibility in using different codes as a component code. As it was mentioned previously the computational complexity of the syndrome-based decoding grows exponentially with the syndrome size and therefore the choice of the codes that can be used as a component code is very limited. Of course, another possible option is to use BP algorithm for the component code decoding. However, this option also limits the choice of a component code since most part of known good codes are not suitable for BP decoding due to low girth. That limits us to use LDPC codes as component code but usually short LDPC codes provide quite low performance and in case of using longer LDPC codes as component code the decoding complexity grows fast as well. One of the most attractive features of HDD is that it is possible to use low-resolution exchange messages for iterative decoding.

The simplest iterative decoding algorithm of staircase codes is based on the bounded-distance decoding (BDD) [15]. Usually it is called iterative BDD (iBDD) algorithm or intrinsic message passing.

In [11], Extrinsic Message Passing (EMP) algorithm was proposed. It improves the iBDD performance with almost negligible decoding complexity increase.

Recently, several hybrid decoding schemes combining SDD and HDD architectures have been proposed, which provide a suitable performance-complexity tradeoff between SDD and HDD. The unifying idea of these schemes is to employ HDD as the decoding core, while exploiting some level of soft information to improve the overall decoder performance. Examples of such an approach can be found in [12] - [14].

If a linear binary code is used as a component code very simple syndrome decoding can be used for decoding of a component code. The complexity of the component code decoder in this case is defined by the complexity of a syndrome calculation, which requires $(n-k)n$ XOR operations. The drawback of this method is quite high memory requirements, which are $O(2^{n-k})$ elements. In this case, the decoding algorithm is stick to one particular code.

If BCH codes are used as a component code, it is possible to apply the algebraic decoding. In this case, the computational complexity of the algorithm can be estmated by the number of operations in Galois field $GF\big(2^{\log_2 n}\big)$. The overall number of operations in Galois field (including calculation of error values that is redundant for BCH codes) can be estimated as $6nt + 9t^2$, where t is the number of correctable errors (error correcting capability of the code). In this case, the memory requirements are practically negligible since the memory is used only for storing the coefficients of polynomials of power no more than $(n-k)$.

In the same manner as in SDD, windowed decoding can be used in HDD as well. In this case, decoding of one target block requires decoding of $2m(W-1)$ codewords per iteration. If the EMP is used for HDD the only additional opeartions needed to calculate the messages is some small amount of logic operations. In case hybrid approach, e.g., like in [13] is used, the required additional complexity to calculate the messages is not negligible. For example, calculation of LLRs according to [13] requires $2m^2(W-2)$ additions per iteration per block. Then, the overall computation complexity can be estimated as $LN_{ItMax}(2m(W-1)(n-k)n)$ XOR operations and $LN_{ItMax}\big(2m^2(W-2)\big)$ additions, if syndrome decoder is used and as $LN_{ItMax}\big(2m(W-1)(6nt+9t^2)\big)$ operations in Galois field if algebraic decoding is applied for decoding of a component code. If EMP (or any other binary message passing) is used $2m^2(W-2)$ bits is enough to store the messages. In case of hybrid approach, the required memory amount should be increased according to message resolution, e.g., for ternary message passing in [13] the memory size increases up to $4m^2(W-2)$ bits. In case the syndrome decoding is used, additional $O(2^{n-k})$ memory elements are required. In case of calculation parallelization, the amount of memory should be increased according to the number of parallel processes.

IV. COMPARISON OF THE CONSIDERED CODES

In this section, we will try to compare the considered codes taking into account how easy the code or code family can be

adapted to different code rates. We will try to compare both performance and decoding complexity of the codes with similar parameters.

A. SDD of both SC-LDPC and staircase codes

The most obvious way of adapting the code to the different code rates is to construct a mother code of low code rate and then puncture parity-check bits to obtain codes of higher rates.

For comparison the following mother codes were constructed.

SC-LDPC ARJ based mother code was constructed in line with the description in Section II with the following parameters:

- Memory size $w = 2$;
- Lifting size $M = 8$;
- Number of blocks $L = 55$;
- Code length $N_{SC-LDPC} = 2200$;
- Mother code rate $R_{SC-LDPC_{init}} = 0.38$.

Windowed SPA layered decoding with window size $W = 9$ blocks and maximum number of iterations $N_{ItMax} = 5$ was used. Floating point messages are used for message passing.

In the staircase mother code, extended $(32, 21)$ BCH code capable of correcting 2 errors was used as a component code $(m = 16,\ r = 11)$. The staircase mother code parameters are:

- Number of blocks $L = 8$;
- Number of terminating blocks $\Lambda = 0$;
- Code length $N_{St} = 2048$;
- Mother code rate $R_{St_{init}} = 0.31$.

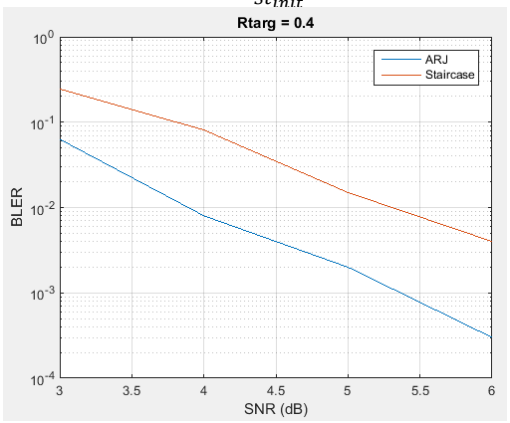

Figure 4. Performance comparison of ARJ based SC-LDPC ($N_{SC-LDPC} = 2200$) and staircase code ($N_{St} = 2048$) at Rtarg = 0.4. SDD.

The staircase code was decoded with the help of the SDD described in Section III.B with window size $W = 3$ blocks and maximum number of iterations $N_{ItMax} = 5$. The syndrome based one-sweep decoding was applied for decoding of a component code. Floating point messages are used for message passing.

Simulation results for different code rates R_{targ} are represented in Figure 4, 5 and 6.

As can be seen from Figure 4, 5 and 6, with target code rate $R_{targ} = 0.4$, the SC-LDPC code outperforms the staircase code by more than 1 dB. However, with increasing target code rate, this performance gap decreases and at $R_{targ} = 0.7$ it is negligible. This can be explained by the fact that the initial mother code rate of the ARJ-based is higher, but puncturing deteriorates its performance faster than the performance of the staircase code.

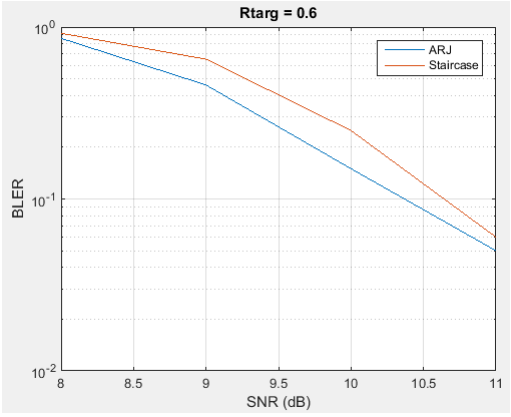

Figure 5. Performance comparison of ARJ-based SC-LDPC ($N_{SC-LDPC} = 2200$) and staircase code ($N_{St} = 2048$) at Rtarg = 0.6. SDD.

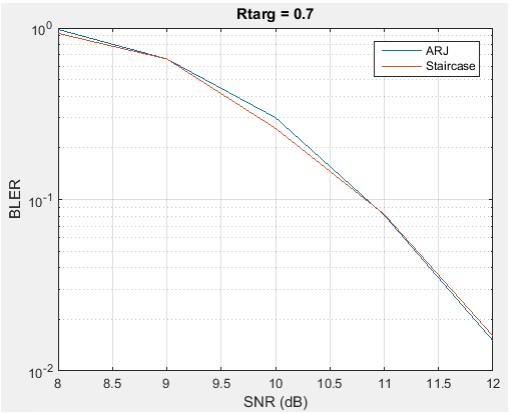

Figure 6. Performance comparison of ARJ-based SC-LDPC ($N_{SC-LDPC} = 2200$) and staircase code ($N_{St} = 2048$) at Rtarg = 0.7. SDD.

Comparing the decoding complexity of used decoders according to estimates in Section III.A and Section III.B we can observe that the complexity of the staircase decoder is about 3 - 4 times higher than the complexity of the SC-LDPC decoder (depending on how to estimate the complexity of the box-plus operation).

Taking into account the lower computational complexity and better performance of the ARJ-based SC-LDPC code, it would be the obvious choice for SDD. On the other hand, the drawback of this code family is the serious performance degradation with target rate increase. Most probably with some optimization efforts it would be possible to find some specific puncturing patterns that can provide not so fast performance degradation but it will take time to find optimal distribution of punctured bits between the sub-matrices of the parity-check matrix.

Copyright (c) IARIA, 2022. ISBN: 978-1-61208-956-0

B. HDD for staircase codes

As it was mentioned in Section A, the complexity of SDD for staircase codes is too high and the performance is inferior to the performance of the ARJ-based SC-LDPC code at low code rates. However, the performance of staircase codes does not drop as fast as the performance of the SC-LDPC codes with the code rate increase.

Since our goal is to consider the codes that can provide high throughput, it might be interesting to consider the possibility of using HDD. As it was discussed in Section III.B, the staircase codes are especially attractive for the HDD since some good codes allowing low-complexity algebraic decoding can be used as a component code. Moreover, the performance drop due to applying HDD rather than SDD can be partly compensated by usage of more powerful codes, which are not possible to use in SDD due to prohibitively high decoding complexity.

For example, it is possible to consider RS codes as component codes. Except the good performance one of the attractive properties of the RS codes is that they belong to the class of Maximum Distance Separable (MDS) codes for which any k symbols of a codeword, where k is the number of information symbols, forms the information sequence. Due to this property all puncturing patterns for RS codes are equally good and there is no need in designing special puncturing patterns when adapting the mother code for the target code rate.

In this section, we consider the comparison of longer codes. We again apply SDD for decoding of the SC-LDPC code, but of lower complexity. We consider Min-Sum (MS) algorithm rather than Sum-Product Algorithm (SPA) and decrease the exchanged message resolution. High resolution of the exchanged messages can hinder achieving high throughput since it will require a lot of additional wiring in comparison with the low resolution message passing leading to serious limitation in possible parallelizing of computations. Thus, the binary message passing is especially attractive for high throughput decoders. In our simulations, we decrease the message resolution to 3 bits.

The parameters of the SC-LDPC ARJ-based mother code used in the simulations are as follows:

- Memory size $w = 2$;
- Lifting size $M = 8$;
- Number of blocks $L = 510$;
- Code length $N_{SC-LDPC} = 20400$;
- Mother code rate $R_{SC-LDPC_{init}} = 0.398$.

Windowed MS layered decoding with window size $W = 18$ blocks and maximum number of iterations $N_{ItMax} = 5$ was used. 3 bits messages are used for message passing.

In the staircase mother code, extended $(32, 23)$ RS code over the $GF(2^5)$ capable of correcting 4 errors was used as a component code, i.e., the code length in bits is $32 \cdot 5 = 160$ $(m = 80\ bits,\ r = 45\ bits)$. The staircase mother code parameters are:

- Number of blocks $L = 3$;
- Number of terminating blocks $\Lambda = 1$;
- Code length $N_{St} = 22800$;
- Mother code rate $R_{St_{init}} = 0.368$.

The staircase code was decoded with the help of the binary message passing with window size $W = 3$ blocks and maximum number of iterations $N_{ItMax} = 5$, the algebraic decoding of the component RS code was used. Notice, that despite the exchanged messages have 5 bit resolution, this is 5 bit message per $GF(2^5)$ symbol, i.e., per 5 bits. Thus, we can consider it as a binary message passing.

Performance comparison of the ARJ-based SC-LDPC code (MS decoding, 3 bit message) and $(32, 23)$-RS-based staircase code ($m = 80\ bits$, 1 bit message) is shown in Figures 7, 8 and 9.

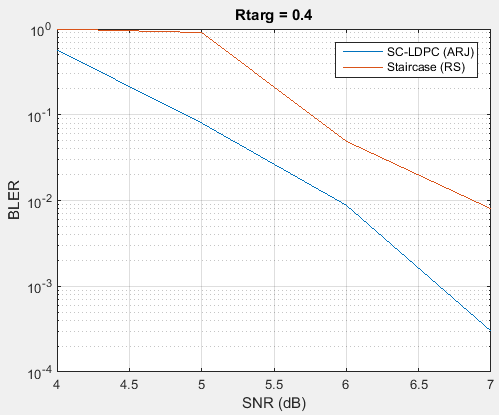

Figure 7. Performance comparison of ARJ based SC-LDPC ($N_{SC-LDPC} = 20400$) and RS-based staircase code ($N_{St} = 22800$) at Rtarg = 0.4. MS decoder for SC-LDPC, HDD for staircase code.

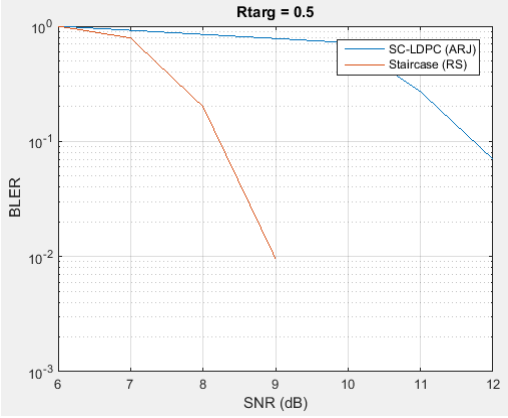

Figure 8. Performance comparison of ARJ based SC-LDPC ($N_{SC-LDPC} = 20400$) and RS-based staircase code ($N_{St} = 22800$) at Rtarg = 0.5. MS decoder for SC-LDPC, HDD for staircase code.

As can be seen from Figures 7, 8 and 9, the performance of the ARJ based SC-LDPC code deteriorates much faster with the target rate increase than the performance of the RS-based staircase code. At the same time, the performance of the RS-based staircase code changes quite smoothly with code rate increase.

Comparison of the decoding complexity depends on how to estimate the computational complexity of min operation and the operation in Galois field. Assuming that the computational complexity of the min operation is equal to

addition complexity, and the operation in Galois field has the same complexity (addition in Galois field is equivalent to XOR operation and multiplication is equivalent to addition on modulo $(n-1)$, where n is the RS code length, or both operations can be implemented with Look-Up Table (LUT)), we can estimate that the decoding complexity of the ARJ based SC-LDPC code is 7 – 8 times greater than that of RS-based staircase code.

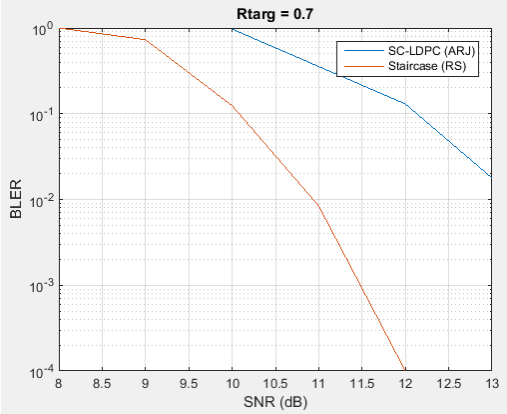

Figure 9. Performance comparison of ARJ based SC-LDPC ($N_{SC-LDPC} = 20400$) and RS-based staircase code ($N_{St} = 22800$) at Rtarg = 0.7. MS decoder for SC-LDPC, HDD for staircase code.

Moreover, with current parameters the amount of memory needed for storing the exchanged messages in window for ARJ based SC-LDPC code is about 200 times more than for the RS-based staircase code if the messages in BP decoder are stored in $(b_c \cdot M \cdot W \times b_v \cdot M \cdot W)$ matrix. More realistic is the case when each message simply consists of value and address. The number of messages in windowed MS decoding of SC-LDPC code can be estimated as $5b_c \cdot M \cdot W = 2160$. Assuming that there are $b_v \cdot M \cdot W = 720$ VNs in window, each CN→VN message comprises 3 bits bearing the value and 10 bits address. That gives $2160 \cdot 13 = 28080$ bits, which 4.3 times more than 6400 bits needed for messages in staircase decoder with window size 3.

One obvious drawback of the RS-based staircase codes is less flexibility with the choice of the code length since it should be a multiple of m^2. This value depends on the choice of the RS code. However, here some optimization is also possible. For example, for Galois field $GF(2^p)$ if p is not a prime the field element can be represented differently (e.g. as a $(1 \times p)$ vector or as a $(q_1 \times q_2)$ matrix, where $q_1 \cdot q_2 = p$). Then, one representation leads to $m = p2^{p-1}$ and another to $m = q_2 2^{p-1}$. In first case, $p2^{p-1}$ RS codes should be decoded at each half-iteration for each block. The second representation leads to necessity to decode $\frac{q_2 2^{p-1}}{q_1}$ codes. Of course, the performance in case of second representation should be less than in first case but it could worth of considering. An example of staircase codes based on different representation of $GF(2^4)$ symbol is shown in Figure 10.

Two representation of $GF(2^4)$ symbol ((2×2) and (1×4)) leads to the staircase codes with different block size and therefore different code length. Both codes in Figure 10 comprise 4 information blocks and 1 terminating block ($\Lambda = 1$). Obviously both codes have the same original rate $R = 0.3243$. Representation (2×2) gives the block size $m = 16$ (8 RS codes must be decoded in block), which leads to the original code length $N_{St} = 1184$, and representation (1×4) gives the block size $m = 32$ (32 RS codes must be decoded in block) leading to the original code length $N_{St} = 4736$. As can be seen from the performance curves in Figure 10, the behavior of both codes is quite similar with changing target code rate.

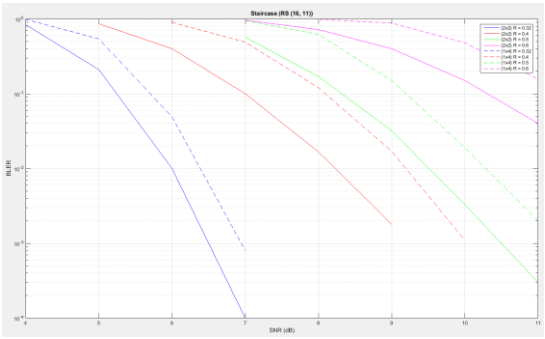

Figure 10. Performance of staircase code based on (16, 11) RS code with different representation of GF(2^4) symbol. (2×2) and (1×4).

In case performance provided by the HDD of RS codes is not enough, two other options can be considered. One of them is to apply soft decoding of RS codes based on Guruswami-Sudan (GS) list decoding [18]. Another is to consider LDPC codes as a component code. On the other hand, both options lead to severe complexity increase. For example, even applying Nielsen interpolation to GS, overall complexity of the list decoding is $O\left(n^2 \left(\frac{n}{k}\right)^{1/2} r^5\right)$ [19], which again narrows the choice of a component code.

V. CONCLUSIONS

The performance and complexity comparison of the ARJ based SC-LDPC codes and staircase codes shows that SDD SC-LDPC codes provide better performance and lower complexity than the staircase codes. However, the performance of SC-LDPC codes deteriorates very fast with code rate increase.

The usage of HDD together with the powerful RS codes brings a benefit of significant complexity decrease with an affordable performance loss. Moreover, the performance of the RS-based staircase codes changes quite smoothly with the code rate increase. One more benefit of HDD of the RS-based staircase codes is the possibility of usage of binary message passing, which decreases significantly the amount of data exchange between the nodes. The latter property is especially important for reaching a high throughput.

Therefore, the HDD of the RS-based staircase code can be considered as a good option for high-throughput decoding. In case higher performance will be needed, one of the interesting directions could be considered: the usage of LDPC codes as component codes in staircase code and BP decoding of a component code.

Copyright (c) IARIA, 2022. ISBN: 978-1-61208-956-0

REFERENCES

[1] A. Jimenez Felstrom and K. Zigangirov, "Time-varying periodic convolutional codes with low-density parity-check matrix," IEEE Trans. Inf. Theory, vol. 45, pp. 2181 – 2191, September 1999.

[2] D. Divsalar, C. Jones, S. Dolinar, and J. Thorpe, "Protograph Based LDPC Codes with Minimum Distance Linearly Growing with Block Size," IEEE Globecom 2005, pp. 1152 – 1156, 2005.

[3] D. Divsalar, S. Dolinar, C. Jones, and K. Andrews, "Capacity-approaching protograph codes," IEEE Journal on Select Areas in Communications, vol. 27, no. 6, pp. 876 – 888, 2009.

[4] M. Lentmaier, A. Sridharan, D. Costello, and K. Zigangirov, "Iterative decoding threshold analysis for LDPC convolutional codes," IEEE Trans. Inf. Theory, vol. 56, pp. 5274 –5289, Oct. 2010.

[5] S. Kudekar, T. Richardson, and R. Urbanke, "Threshold saturation via spatial coupling: Why convolutional LDPC ensembles perform so well over the BEC," IEEE Trans. Inf. Theory, vol. 57, pp. 803–834, Feb. 2011.

[6] M. Lentmaier, I. Andriyanova, N. Hassan, and G. Fettweis, "Spatial Coupling - A way to Improve the Performance and Robustness of Iterative Decoding," Proc. European Conf. Networks and Communications (EuCNC), pp. 1 – 4, 2015.

[7] N. Hassan, M. Lentmaier, and G. Fettweis, "Comparison of LDPC block and LDPC convolutional codes based on their decoding latency," in Proc. Int. Symp. on Turbo Codes & Iterative Inf. Proc., pp. 225 – 229, Aug. 2012.

[8] D. Mitchell, M. Lentmaier, and D. Costello, "Spatially coupled LDPC codes constructed from protographs," IEEE Trans. Inf. Theory , vol. 61, no. 9, pp. 4866 – 4889, 2015.

[9] K. Klaiber, S. Cammerer, L. Schmalen, and S. ten Brink, "Avoiding Burst-like Error Patterns in Windowed Decoding of Spatially Coupled LDPC," Proc. IEEE 10th International Symposium on Turbo Codes & Iterative Information Processing (ISTC), pp. 1 – 5, 2018.

[10] T. Johansson and K. Sh. Zigangirov, "A simple one-sweep algorithm for optimal APP symbol decoding of linear block codes," IEEE Trans. Inf. Theory, vol. 44, no. 7, pp. 3124–3129, Nov. 1998.

[11] Y. Jian, H. Pfister, K. Narayanan, R. Rao, and R. Mazahreh, "Iterative Hard Decision Decoding of Braided BCH Codes for High Speed Optical Communication," Proc. GLOBECOM, pp. 2376 – 2381, 2013.

[12] A. Sheikh, A. Graell i Amat, G. Liva, C. Häger, and H. D. Pfister, "On low-complexity decoding of product codes for high-throughput fiber-optic systems," in Proc. IEEE Int. Symp. on Turbo Codes & Iterative Inf. Proc. (ISTC), Hong Kong, pp. 1 – 5, Dec. 2018.

[13] A. Sheikh, A. Graell i Amat, G. Liva, and A. Alvarado, "Refined reliability combining for binary message passing decoding of product codes," arXiv, 2020. [Online]. Available:https://arxiv.org/abs/2006.00070, pp. 1 – 5, 2020.

[14] A. Sheikh, A. Graell i Amat, G. Liva, and A. Alvarado, "Novel High-Throughput Decoding Algorithms for Product and Staircase Codes based on Error-and- Erasure Decoding," arXiv, 2020, [Online]. Available: https://arxiv.org/abs/2008.02181, pp. 1 – 12, 2020.

[15] B. P. Smith, A. Farhood, A. Hunt, F. R. Kschischang, and J. Lodge, "Staircase codes: FEC for 100 Gb/s OTN," IEEE/OSA J. Lightw. Technol., vol. 30, no. 1, pp. 110–117, Jan. 2012.

[16] X. Dou, M. Zhu, J. Zhang, and B. Bai, "Soft-Decision Based Sliding-Window Decoding of Staircase Codes," IEEE 10th International Symposium on Turbo Codes & Iterative Information Processing, pp. 1 – 5, 2018.

[17] C. Häger, H. D. Pfister, A. Graell i Amat, and F. Brännström, "Density Evolution for Deterministic Generalized Product Codes on the Binary Erasure Channel at High Rates," IEEE Trans. Inf. Theory, vol. 63, no. 7, pp. 4357 – 4378, 2017.

[18] V. Guruswami and M. Sudan, "Improved Decoding of Reed-Solomon Codes and Algebraic Geometry Codes," IEEE Trans. Inform. Theory, vol. 45, no. 6, pp. 1757 – 1767, 1999.

[19] G. Kabatiansky, E. Krouk, and S. Semenov, "Error Correcting Coding and Security for Data Networks," Wiley, 2005.

Copyright (c) IARIA, 2022. ISBN: 978-1-61208-956-0