Decoder Error Propagation Mitigation for Spatially Coupled LDPC Codes

Min Zhu*, David G. M. Mitchell†, Michael Lentmaier‡, and Daniel J. Costello, Jr.§

*State Key Laboratory of ISN, Xidian University, Xi'an, P. R. China, zhunanzhumin@gmail.com
†Klipsch School of Electrical and Computer Engineering, New Mexico State University, Las Cruces, NM, USA, dgmm@nmsu.edu
‡Department of Electrical and Information Technology, Lund University, Lund, Sweden, michael.lentmaier@eit.lth.se
§Department of Electrical Engineering, University of Notre Dame, Notre Dame, IN, USA, dcostel1@nd.edu

Abstract—In this paper, we introduce two new methods of mitigating decoder error propagation for low-latency sliding window decoding (SWD) of spatially coupled low density parity check (SC-LDPC) codes. Building on the recently introduced idea of *check node (CN) doping* of regular SC-LDPC codes, here we employ *variable node (VN) doping* to fix (set to a known value) a subset of variable nodes in the coupling chain. Both of these doping methods have the effect of allowing SWD to recover from error propagation, at a cost of a slight rate loss. Experimental results show that, similar to CN doping, VN doping improves performance by up to two orders of magnitude compared to undoped SC-LDPC codes in the typical signal-to-noise ratio operating range. Further, compared to CN doping, VN doping has the advantage of not requiring any changes to the decoding process. In addition, a log-likelihood-ratio based *window extension algorithm* is proposed to reduce the effect of error propagation. Using this approach, we show that decoding latency can be reduced by up to a significant fraction without suffering any loss in performance.

I. INTRODUCTION

Spatially coupled low-density parity-check (SC-LDPC) codes, a type of LDPC convolutional code [1], have been shown to achieve *threshold saturation*, i.e., the suboptimal *belief propagation (BP)* iterative decoding threshold of SC-LDPC code ensembles over memoryless binary-input symmetric-output channels coincides with the *maximum a posteriori probability (MAP)* threshold of their underlying LDPC block code (LDPC-BC) ensembles [2]–[5]. Further, *regular* SC-LDPC code ensembles not only have capacity approaching iterative decoding thresholds, but they are asymptotically good, i.e., their minimum distance grows linearly with frame length [6]. Therefore, SC-LDPC codes combine the best features of both *regular* and *irregular* LDPC-BCs.

SC-LDPC codes can be formed by applying a protograph-based construction technique [6]. In this paper we consider SC-LDPC codes constructed by coupling together a sequence of L disjoint (J, K)-regular LDPC-BC protographs into a single coupled chain, where infinite L results in an *unterminated* coupled chain and finite L results in a *terminated* coupled chain. Without loss of generality, we consider an example of constructing (3,6)-regular SC-LDPC codes. We begin with an independent (uncoupled) sequence of (3,6)-regular LDPC-BC protographs with *base matrix* $\mathbf{B} = [3, 3]$. Fig. 1 shows the resulting unterminated (3,6)-regular SC-LDPC code chain obtained by applying the *edge-spreading* technique of [6] to
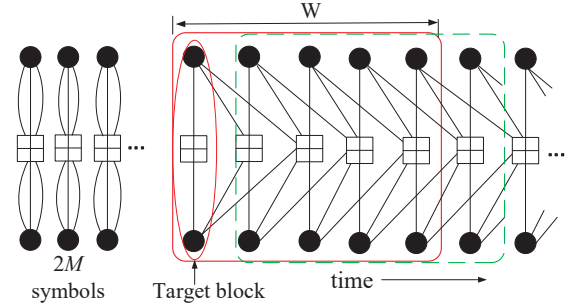

Fig. 1. A (3,6)-regular SC-LDPC code protograph obtained from an underlying LDPC-BC protograph with base matrix $B = [3, 3]$. The black circles represent variable nodes, and the "plus" squares represent check nodes. (a) A sequence of independent (uncoupled) protographs; (b) Spreading edges to the $m = 2$ nearest neighbors.

the uncoupled protographs. The edge spreading is defined by a set of *component base matrices* $\mathbf{B}_0 = \mathbf{B}_1 = \mathbf{B}_2 = [1 \ \ 1]$ that must satisfy $\mathbf{B} = \mathbf{B}_0 + \mathbf{B}_1 + \mathbf{B}_2$. In general, an arbitrary edge spreading must satisfy $\mathbf{B} = \sum_{i=0}^{m} \mathbf{B}_i$, where m is referred to as the *coupling width*. Applying the lifting factor M to the SC-LDPC protograph of Fig. 1 results in an unterminated ensemble of (3,6)-regular SC-LDPC codes in which each time unit represents a *block* of $2M$ coded bits (variable nodes).

To reduce the decoding latency and memory, *sliding window decoding (SWD)* was proposed for SC-LDPC codes in [7], where a standard BP flooding schedule is applied to all the nodes in the window. For example, in Fig. 1, the rectangular box represents a decoding window of *size W* (blocks). To decode, a BP flooding schedule is applied to all the nodes in the window for some fixed number of iterations, or until some *stopping criterion* is met, the *target block* of $2M$ symbols in the first window position is decoded according to the signs of their *log-likelihood ratios* (LLRs), and the window shifts one time unit (block) to the right (see Fig. 1). Decoding continues in the same fashion until the entire chain is decoded, where the decoding latency in bits is given by $2MW$.

In order to reduce decoding latency and memory, the window size W should be chosen as small as possible. In [8] the authors experimentally showed that near optimal performance can be maintained at higher *signal-to-noise ratios* (SNRs) as long as $W \geq 6\eta$, where $\eta = m + 1$ is *decoding constraint length*. When low latency operation is desired, however, at

typical operating SNRs, smaller values of W can sometimes result in infrequent but severe *decoder error propagation*. Error propagation is triggered when, after a block decoding error occurs, the decoding of the subsequent block is also affected, which in turn can cause a continuous string of block errors, resulting in an unacceptable loss in performance. This is particularly damaging for very long code chains or for streaming applications. Klaiber et al. in [9] proposed to adapt the number of decoder iterations and/or shift the window position in order to combat decoder error propagation for SC-LDPC codes. For a related class of spatially coupled codes, viz. *braided convolutional codes (BCCs)*, with SWD [10], a *window extension algorithm*, a *synchronization* mechanism, and a *retransmission* strategy were all used to mitigate error propagation [11]. More recently, Zhu et al. proposed a *check node (CN) doped* SC-LDPC code design [12] to limit error propagation. A disadvantage of these approaches is that they all require some modification of the decoding process.

The CN doped code design was motivated by the fact that the boundaries of a coupled chain have lower degree CNs, which has the effect of propagating more reliable information throughout the chain during iterative decoding. Inserting occasional lower degree CNs in a code chain, i.e., CN doping, has the same effect, thus allowing the decoder to recover from error propagation, although the shape of the decoding window must be altered at the doping points.

In a similar vein, known (or fixed) variable nodes in a coupled chain can also aid the iterative decoding process. This motivates us to propose a new class of *variable node (VN) doped* SC-LDPC codes in this paper, which operate by inserting occasional fixed (known) VNs in a code chain, thus allowing the decoder to recover from error propagation, with the added advantage of leaving the shape of the decoding window unchanged. The CN doping and VN doping code designs introduce occasional irregularities in the coupled chain, which results in some rate loss. However, since code doping is primarily useful for long or unterminated chains, the rate loss associated with doping is very slight.[1] We present numerical results showing that, similar to CN doping, VN doping improves performance by up to two orders of magnitude compared to undoped SC-LDPC codes in the typical SNR operating range.

We also adapt the *log-likelihood ratio (LLR)* based window extension algorithm, first proposed in [10] for BCCs, to combat error propagation in SWD of SC-LDPC codes. This approach, which requires the size of the decoding window to occasionally be extended, but involves no rate loss, is shown to be capable of reducing the decoding latency by a significant fraction without suffering any loss in performance.

II. DECODER ERROR PROPAGATION

During SWD of SC-LDPC codes, when a block of target symbols at time t is decoded, the window shifts to include

---

[1]Since termination itself serves to truncate error propagation, the effect on short chains is minimal.

---

the most recent block of received symbols at time $t + W$, and decoding commences on the block of target symbols at time $t + 1$. During the decoding of the time $t + 1$ block, for a coupling width of m, the final LLRs of the m past decoded blocks, from time $t - m + 1$ to time t, remain involved in the decoding process, although these LLRs are no longer updated, as illustrated in Fig. 2, which depicts the base parity-check matrix of a (3, 6)-regular SC-LDPC code with $W = 3$ and $m = 2$.
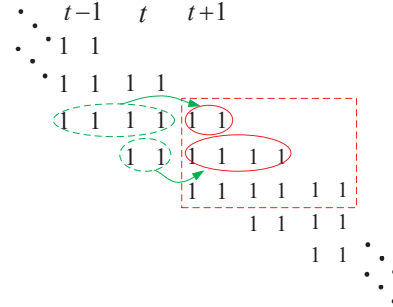

Fig. 2. The final variable node LLRs at times $t-m+1$ to t are used to update the check nodes in the window during the decoding of the target symbols at time $t + 1$.

Under normal operation, decoding proceeds with correctly decoded blocks until such time as a block of target symbols contains one or more LLRs with incorrect signs when the window shifts, thus resulting in a *block decoding error*. Typically, if only a few symbols have incorrect LLRs and most of the correct LLRs have large magnitudes, the LLRs of the incorrectly decoded block will have only a small effect on the decoding of the next block, and the decoder will recover and continue to correctly decode subsequent blocks, assuming most of the symbols in the window have large and correct LLRs. This type of operation results in randomly distributed error blocks.

However, if an error block contains many incorrect LLRs, particularly if they have large magnitudes and a significant number of the LLRs associated with the correct symbols are small, those "bad" LLRs may negatively affect the decoding of the next block of target symbols, causing a block error that would not have occurred under normal operating conditions. This in turn can trigger additional block errors, resulting in an *error propagation* effect, i.e., a continuous sequence of incorrectly decoded blocks.

In an application where information is transmitted in *frames* of a small fixed length L (in time units), with graph termination (reduced check node degrees for m time units) following the last block of transmitted variable nodes, any error propagation will be limited and decoding will start fresh with the next frame. However, if L is large, a significant number of blocks could be affected by error propagation, thus severely degrading performance. In a streaming application, with no termination, the situation could be catastrophic, resulting in a *block error rate (BLER)* that asymptotically tends to 1.
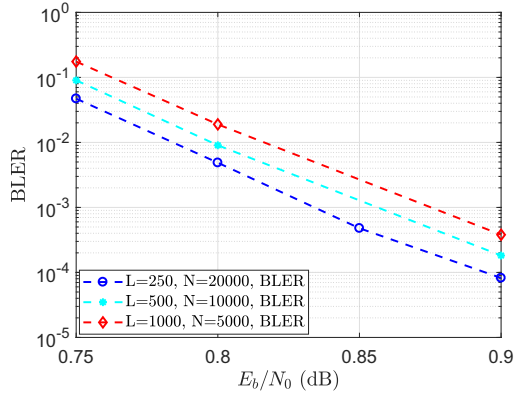
Fig. 3. SWD BLER performance of a (3,6)-regular SC-LDPC code for three different combinations of frame length and number of frames simulated, all with the same total number of simulated blocks.

We now give an example illustrating the effect of this error propagation. The simulated BLER performance of SWD of a (3,6)-regular SC-LDPC code based on the coupled protograph in Fig. 1 is shown in Fig. 3, where $W = 18$ and $M = 2000$. The figure represents the simulation of a total of $LN = 5 \times 10^6$ blocks (or $2MLN = 2 \times 10^{10}$ bits), where L is the frame length and N is the number of frames simulated, for three different combinations of L and N. From the figure, we observe that, with increasing L, the BLER performance becomes worse, even though there are relatively few error-propagation frames overall, thus confirming the above observation.[2]

III. ERROR PROPAGATION MITIGATION

In this section, we briefly review CN doping and then describe the new VN doping and window extension error propagation mitigation methods in detail.

A. A Brief Review of CN Doped SC-LDPC Codes

In order to combat error propagation in SWD of SC-LDPC codes, a new CN doped SC-LDPC code design was proposed in [12]. The key idea of CN doped SC-LDPC codes lies in occasionally inserting additional check nodes into the protograph of a regular SC-LDPC code, which is referred to as check node doping. The resulting structured irregularity limits error propagation by emulating graph termination, as noted earlier. We now use (3,6)-regular SC-LDPC codes as an example to briefly review of the CN doping process and the corresponding decoding scheme.

Fig. 4 shows the construction and SWD schedule of CN doped (3,6)-regular SC-LDPC codes. The red VNs at time $t = \tau_1$ spread their three edges to the CNs at times $t = \tau_1 + 1$, $t = \tau_1 + 2$, and $t = \tau_1 + 3$; the red VNs at time $t = \tau_2$ spread their three edges to the CNs at times $t = \tau_2 + 2$, $t = \tau_2 + 3$, and $t = \tau_2 + 4$, and so on. To decode a CN doped (3,6)-regular SC-LDPC code, the window shifting schedule of SWD applied to

---

[2]Fig. 3 represents only a narrow range of SNRs, below the threshold of the underlying LDPC-BC, where error propagation presents a significant problem. For larger values of E_b/N_0 and/or W, SWD typically recovers from an error burst.

the doped coupled chain is altered compared to standard SWD. When a doping point (red VN pair) becomes the target block, the window shifts by one VN time unit to include one new block of VNs, as before, but it shifts by two CN time units to include two new blocks of CNs (and thus still including the same total number of CNs), as illustrated in Fig. 4.

B. VN Doped SC-LDPC Codes

As an alternative to introducing occasional reduced-degree check nodes in the coupled chain (called *CN doping*) [12], here the encoder fixes (set to "0") occasional variable nodes in the coupled chain, called *VN doping*, as shown Fig. 5, where each time unit represents a *block* of $2M$ coded symbols. The VNs at time $t = \tau_1$ (the green empty circles) are doped by setting the $2M$ coded bits corresponding to these VNs to be "0". As a result, the CNs at times $t = \tau_1, \tau_1 + 1, \tau_1 + 2$ (colored red and shaded) can be viewed as degree-4, rather than degree 6, CNs, thus emulating CN doping without actually altering the graph structure. Similarly, if the VNs at time $t = \tau_2$ are doped, the CNs at times $t = \tau_2, \tau_2 + 1, \tau_2 + 2$ (colored red and shaded) can be viewed as degree-4 CNs.

If n_c and n_v denote the total number of CNs and the total number of unknown VNs in VN doped SC-LDPC codes, respectively, and if there are d doped positions, the design rate of VN doped SC-LDPC codes with frame length L and d doped VNs is

$$R_{L,\text{doped}} = 1 - \frac{n_c}{n_v} = 1 - \left(\frac{L+m}{L-d}\right)(1-R), \quad (1)$$

where $R = 1 - J/K$ is the design rate of the uncoupled protograph [6]. Compared to the design rate $R_L = 1 - \left(\frac{L+m}{L}\right)(1-R)$ of undoped SC-LDPC codes [6], we see from (1) that the design rate of VN doped SC-LDPC codes is smaller, i.e., VN doping results in some *rate loss*, similar to CN doping.

The decoding process is the same as for undoped codes, except that the doped code symbols and their positions are treated as known, i.e., during the decoding process we set the LLRs of the doped symbols to be a large constant negative value. These known bits have the effect of transmitting perfectly reliable information to their neighbour nodes, thus helping the decoder recover from error propagation. An important implementation advantage of VN doping over CN doping is that the shape of the decoding window remains unaltered.

C. Window Extension Algorithm

In this section, rather than altering the code design, we mitigate error propagation by occasionally extending the size of the decoding window. By experimentally recording decoder behavior during error propagation, we find that the average LLR magnitudes of the blocks are typically near zero, a phenomenon also observed for BCCs in [11].

To take advantage of this observation, we can allow the window size W to change dynamically in SWD of SC-LDPC codes, from an initial size W_{init} to a maximum size W_{max}, thereby causing additional decoding resources to be employed
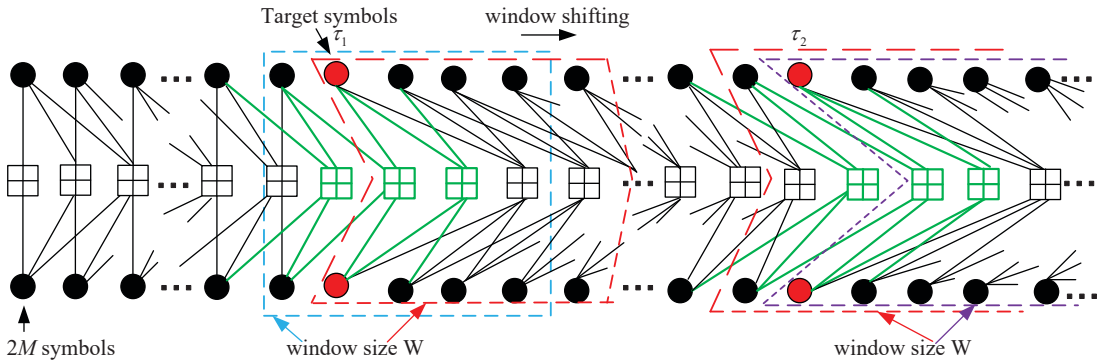
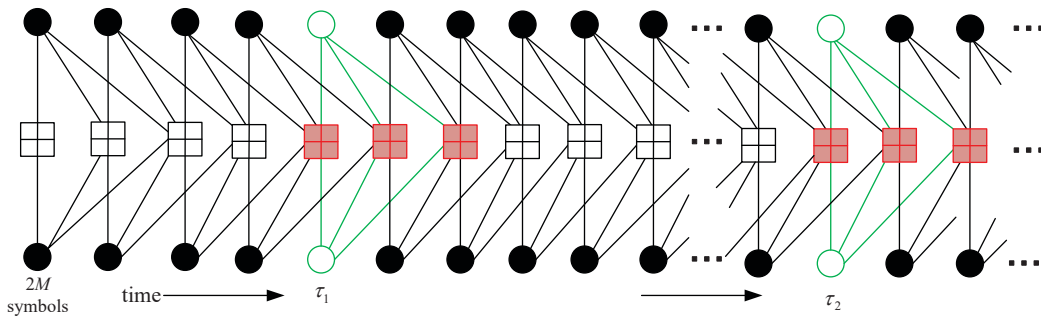Fig. 4. The construction and SWD schedule for CN doped (3,6)-regular SC-LDPC codes.

Fig. 5. VN doping for a (3,6)-regular SC-LDPC code with occasional fixed variable nodes spaced throughout the coupled chain.

when a potential error propagation condition is detected. To formalize this process, we denote the decision LLRs of the $2M$ coded bits in the ith block of the current window after some fixed number I of iterations by $\boldsymbol{\ell}^i = \left(\ell_0^i, \ell_1^i, \ldots, \ell_{2M-1}^i\right)$, $i \in \{t, t+1, \ldots, t+W-1\}$. Then the average LLR magnitude of the $2M$ code bits in block i after I iterations is given by

$$\bar{\ell}^i = \frac{1}{2M} \sum_{j=0}^{2M-1} \ell_j^i. \qquad (2)$$

We also define the *observation span* τ as the number of consecutive blocks in the decoding window over which the average LLRs magnitude is to be examined.

Now assume the current decoding window, with $W = W_{\text{init}}$, covers the blocks ($2M$ bits each) from time t to $t + W - 1$. After I iterations, if any of the average LLR magnitudes of the first τ blocks in the current window, $1 \leq \tau \leq W$, is lower than a predefined *threshold* θ, i.e., if

$$\bar{\ell}^i < \theta, \qquad \text{for any } i \in \{t, t+1, \ldots, t+\tau-1\}, \quad (3)$$

then the target block is not decoded, the window size W is increased by 2 time units[3], and the decoding process restarts. If none of the first τ blocks is satisfies (3), the target block is

decoded and the window shifts by 1 time unit. If the current window size reaches W_{max}, then the target block is decoded, the window shifts by 1 time unit, and the window size is reset to W_{init}.[4] Fig. 6 shows how the window extension scheme works during the SWD of (3,6)-regular SC-LDPC codes. The method is described in detail in Algorithm 1.

IV. NUMERICAL RESULTS

In order to illustrate the effectiveness of the proposed mitigation methods, the bit error distribution per block of a typical error-propagation frame in SWD of the VN doped and undoped (3,6)-regular SC-LDPC codes of Fig. 1 sent over an AWGN channel with BPSK signaling is plotted in Fig. 7, where $M = 1000$, $L = 500$, and $W = 18$. The figure clearly shows that VN doping truncates the error propagation at the doping point in the center of the frame, whereas the errors continue to the end of frame in the undoped case. Similar behavior has been observed for CN doping (see [12]) and window extension.

The *bit error rate (BER)* and BLER performance of VN doping with rate $R = 0.49699$, CN doping with rate $R = 0.497$ [12], and undoped with rate $R = 0.499$ (3,6)-regular SC-LDPC codes of Fig. 1 is shown in Fig. 8, where the termination length is $L = 500$ and the window size is

---

[3]Increasing the window size by 2 time units was determined experimentally to be the best compromise between performance and complexity.

[4]Note that, since the window size can vary, the decoding latency can be characterized by the average window size.
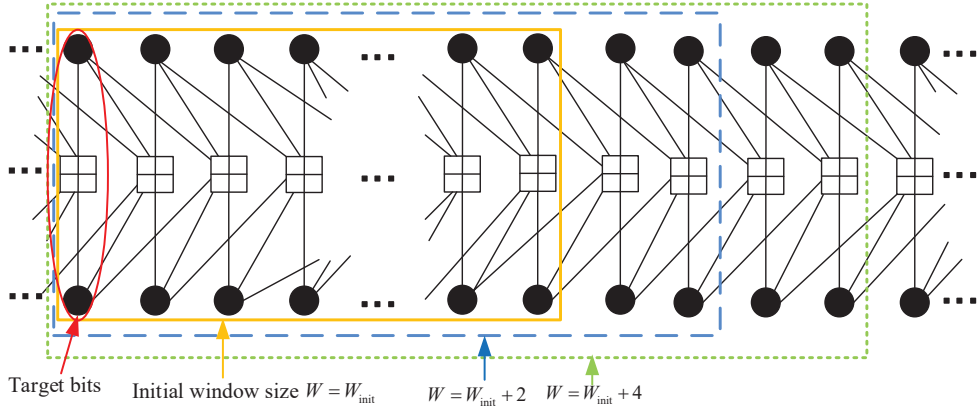
Fig. 6. Sliding window decoder with the window extension algorithm for (3,6)-regular SC-LDPC codes.

Algorithm 1 Window Extension

1: Assume that the block at time t is the target block in a window decoder of size W initialized with the channel LLRs of w received blocks. Set $W = W_{\text{init}}$ initially, and let τ, θ, W_{init}, and W_{max} be parameters.

2: Every time I_{max} iterations are finished, calculate the absolute average LLR value of each block in the current window according to (2).

3: **if** $\bar{\ell}^i < \theta$ for any $i \in \{t, t+1, \ldots, t+\tau-1\}$ **then**

4:     **if** $W < W_{\text{max}}$ **then**

5:        The decoder accepts two new blocks from the channel. The target block is still the block at time t, the new blocks are at times $t+W$ and $t+W+1$, and, the window size is set to $W = W + 2$.

6:        For the old $W-2$ blocks in the window, all the LLRs are maintained.

7:        Restart decoding process, and go to step 2.

8:     **else**

9:        Go to step 12.

10:    **end if**

11: **else**

12:    Decode the target block, set $W = W_{\text{init}}$, and shift the window by one time unit.

13:    Continue decoding and go to step 2.

14: **end if**

$W = 18$. We observe that both VN doping and CN doping gain approximately two orders of magnitude in BER and one order of magnitude in BLER compared to the undoped code at those SNR operating points of interest (below the threshold of the underlying LDPC-BC). Also, the fact that the performance of both doping methods is essentially equivalent corroborates our earlier observation that VN doping emulates the CN doping process while not requiring any alteration to the shape of the decoding window.

To illustrate the advantage of allowing a dynamic window size, Fig. 9 shows the performance of the (3,6)-regular SC-LDPC codes of Fig. 1 with and without window extension for
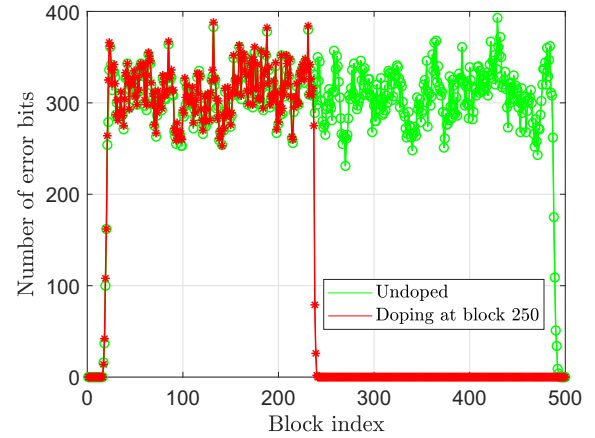

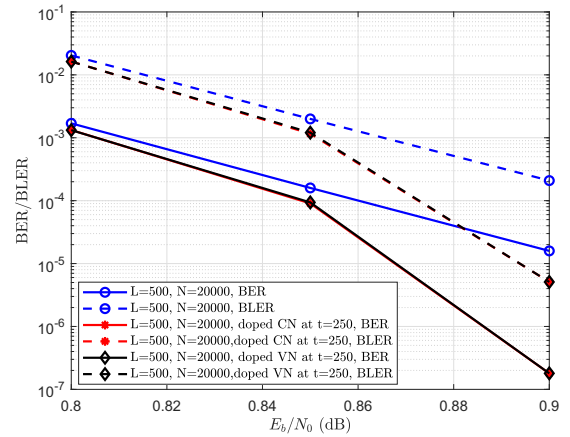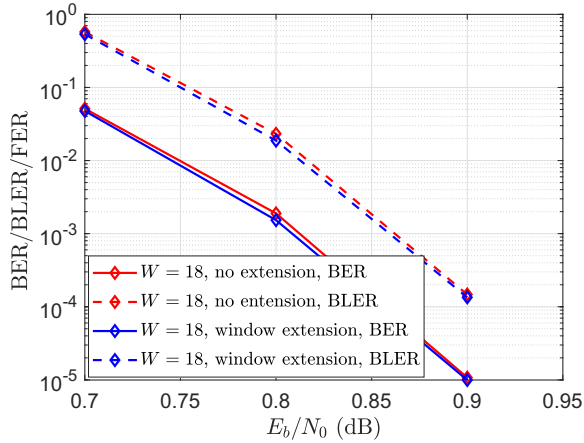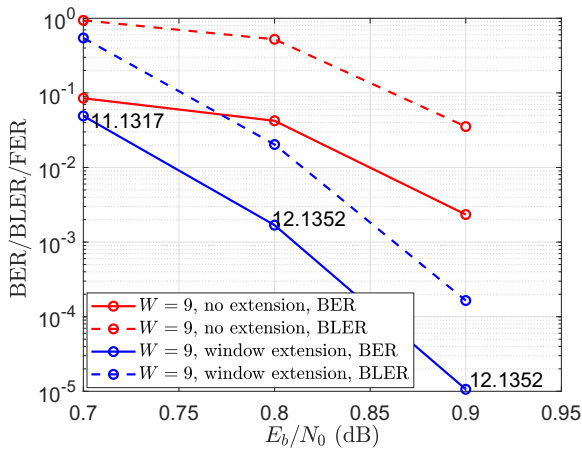Fig. 7. Bit error distribution per block.

Fig. 8. Performance comparison of CN doped, VN doped, and undoped SC-LDPC codes.

(a) $W = 18$

(b) $W = 9$

Fig. 9. BER and BLER performance of window extension with window sizes $W = 18$ and $W = 9$.

$L = 250$ and window sizes $W = 18$ and $W = 9$. From the figure, we see that, for $W = 18$, window extension gives only a slight improvement compared to not using window extension. However, for $W = 9$, window extension gains more than two orders of magnitude in both BER and BLER at SNR operating points of interest. From another point of view, by comparing Figs. 9(a) and 9(b), we see that the performance with $W = 9$ and window extension is roughly equivalent to that of $W = 18$ with no window extension, while the average window size (decoding latency), indicated on the BER curve with window extension as shown in Fig. 9(b), is reduced by about 1/3.

V. CONCLUSIONS

In this paper, we proposed two new methods of mitigating decoder error propagation in SWD of SC-LDPC codes: VN doping from the code design aspect and window extension from the decoder design aspect. The first, VN doping, takes advantage of occasional fixed (known) variable nodes in the

protograph to allow SWD to recover from error propagation, without requiring any alteration to the shape of the decoding window (in contrast to the previously proposed CN doping). As a result, the BER and BLER performance of VN doped SC-LDPC codes was shown to improve by up to two orders of magnitude at SNR operating points of interest. The second method, window extension, adapts the window size dynamically to bring more decoding resources to bear in error propagation conditions. Simulation results show that window extension with $W = 9$ can reduce the decoding delay (latency) on average by about 1/3 compared to using $W = 18$ without window extension, while maintaining the same BER and BLER performance.

REFERENCES

[1] A. J. Felström, and K. Sh. Zigangirov, "Time-varying periodic convolutional codes with low-density parity-check matrix," *IEEE Trans. Inf. Theory*, vol. 45, no. 6, pp. 2181-2191, Sep. 1999.

[2] M. Lentmaier, A. Sridharan, D. J. Costello, Jr., and K. Sh. Zigangirov, "Iterative decoding threshold analysis for LDPC convolutional codes," *IEEE Trans. Inf. Theory*, vol. 56, no. 10, pp. 5274-5289, Oct. 2010.

[3] S. Kudekar, T. J. Richardson, and R. L. Urbanke, "Threshold saturation via spatial coupling: Why convolutional LDPC ensembles perform so well over the BEC," *IEEE Trans. Inf. Theory*, vol. 57, no. 2, pp. 803-834, Feb. 2011.

[4] S. Kudekar, T. J. Richardson, and R. L. Urbanke, "Spatially coupled ensembles universally achieve capacity under belief propagation," *IEEE Trans. Inf. Theory*, vol. 59, no. 12, pp. 7761-7813, Dec. 2013.

[5] S. Kumar, A. J. Young, N. Maoris, and H. D. Pfister, "A proof of threshold saturation for spatially-coupled LDPC codes on BMS channels," in *Proc. 50th Annu. Allerton Conf. Commun., Control, Comput.*, Monticello, IL, USA, Oct. 2012, pp. 176-184.

[6] D. G. M. Mitchell, Michael Lentmaier, and Daniel J. Costello, "Spatially coupled LDPC codes constructed from protographs," *IEEE Trans. Inf. Theory*, vol. 61, no. 9, pp. 4866-4889, July 2015.

[7] A. R. Iyengar, M. Papaleo, P. H. Siegel, J. K. Wolf, A. Vanelli-Corolli, and G. E. Corazza, "Windowed decoding of protograph-based LDPC convolutional codes over erasure channels," *IEEE Trans. Inf. Theory*, vol. 58, no. 4, pp. 2303-2320, Apr. 2012.

[8] K. Huang, D. G. M. Mitchell, L. Wei, X. Ma, and D. J. Costello, "Performance comparison of LDPC block and spatially coupled codes over GF (q)," *IEEE Trans. on Communications*, vol. 63, no. 3, pp. 592-604, Mar. 2015.

[9] K. Klaiber, S. Cammerer, L. Schmalen, and S. t. Brink, "Avoiding burst-like error patterns in windowed decoding of spatially coupled LDPC codes," in *Proc. IEEE 10th Int. Symp. on Turbo Codes & Iterative Inf. Processing (ISTC)*, Hong Kong, China, 2018, pp. 1-5.

[10] M. Zhu, D. G. M. Mitchell, M. Lentmaier, D. J. Costello, Jr., and B. Bai, "Braided convolutional codes with sliding window decoding," *IEEE Trans. on Communications*, vol. 65, no. 9, pp. 3645-3658, Sept. 2017.

[11] M. Zhu, D. G. M. Mitchell, M. Lentmaier, D. J. Costello, Jr., and B. Bai, "Combating error propagation in window decoding of braided convolutional codes," in *Proc. IEEE Int. Symp. Information Theory*, Vail, CO, USA, June 17-22, 2018, pp. 1380-1384.

[12] M. Zhu, D. G. M. Mitchell, M. Lentmaier, and D. J. Costello, Jr., "A novel design of spatially coupled LDPC codes for sliding window decoding," to appear in *Proc. IEEE Int. Symp. Information Theory*, Los Angeles, CA, USA, June 21-26, 2020.