Towards Fully Pipelined Decoding of Spatially Coupled Serially Concatenated Codes

Mojtaba Mahdavi, Liang Liu, Ove Edfors, Michael Lentmaier, Norbert Wehn, Stefan Weithoffer

Towards Fully Pipelined Decoding of Spatially Coupled Serially Concatenated Codes

Mojtaba Mahdavi*, Liang Liu*, Ove Edfors*, Michael Lentmaier*, Norbert Wehn†, Stefan Weithoffer‡
*Department of Electrical and Information Technology (EIT), Lund University, Lund, Sweden
†Department of Electrical and Computer Engineering, Technische Universität Kaiserslautern
‡IMT Atlantique, Lab-STICC, UMR CNRS 6285, F-29238 Brest, France
Email: *firstname.lastname@eit.lth.se, †wehn@eit.uni-kl.de, ‡stefan.weithoffer@imt-atlantique.fr

***Abstract*—Having close-to-capacity performance and low error floor, even for small block lengths, make spatially coupled serially concatenated codes (SC-SCCs) a very promising class of codes. However, classical window decoding of SC-SCCs either limits the minimum block length or requires a large number of iterations, which increases the complexity and constrains the degree to which an SC-SCC decoder architecture can be parallelized. In this paper we propose *jumping window decoding* (JWD), an algorithmic modification to the scheduling of decoding such that it enables pipelined implementation of SC-SCCs decoder. Also, it provides flexibility in terms of block length and number of iterations and makes them independent of each other. Simulation results show that our scheme outperforms classical window decoding of both SC-SCCs and uncoupled SCCs, in terms of performance. Furthermore, we present a fully pipelined hardware architecture to realize JWD of SC-SCCs along with area estimates in 12nm technology for the respective case study.**

I. Introduction

Support for extreme data rates in wireless systems beyond 5G (B5G) is essential for many new applications, such as data kiosks, high capacity wireless backhauls, and wireless virtual and augmented reality [1]. These demanding throughput requirements directly translate to the digital baseband signal processing where channel coding is a main contributor of computational complexity [2] and is subject to major restrictions in terms of silicon area. Achieving Tbps throughputs with FEC schemes that rely on soft decoding require highly parallel and deeply pipelined decoder hardware architectures. Although fully pipelined turbo decoders are expected to reach Tbps for medium block sizes and low number of iterations, they suffer from a large silicon area for block sizes above 100 bits [3], [4]. Spatially coupled schemes, allowing construction of a larger code of almost arbitrary length from a much smaller code, may offer a way around this block size limitation.

From a code design perspective, it has been shown that spatial coupling improves the decoding threshold of LDPC codes [5], [6]. Similarly, the concept of spatial coupling has been applied to turbo-like codes, where threshold saturation has also been proven [7]. It has been shown that the performance improvement through spatial coupling is more pronounced in serially concatenated codes (SCCs) compared to the parallel concatenated codes (PCCs) [8]. Due to close-to-capacity performance and low error floor, spatially coupled SCCs (SC-SCCs) are selected as the focus of this paper.

From a hardware design perspective, it is worth mentioning that decoding of spatially coupled codes can be done block-wise using window decoding [9], [10], which allows the decoder to be constructed from a set of traditional decoders working on the block size of the smaller code, called sub-decoders. These sub-decoders are then chained together and coupled accordingly. This streaming-like decoding has the potential to combine good error correction performance with high throughput and lends itself to a pipelined implementation.

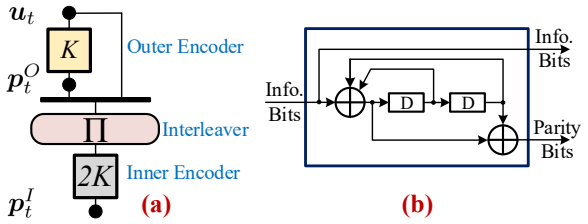

Fig. 1. (a) Compact graph representation of SCC. (b) Structure of RSC encoder, which is used for the inner and outer encoders.

However, limitations for fully pipelined hardware architectures with the highest throughput, that already exist for the uncoupled case, such as maximum block size and maximum number of iterations also exist for the coupled case. On the other hand, classical window decoding for spatially coupled turbo-like codes either imposes a minimum block size or a large number of decoding iterations per sub-block of each window. Both may render an implementation ineffective.

In this work we present *jumping window decoding* (JWD), a new decoding schedule for window decoding of SC-SCC turbo codes that enables, for the first time, fully pipelined implementation. We discuss a conceptual hardware architecture and give implementation estimates for the 12nm technology node based on a characterization of the compute kernels. The remainder of this paper is structured as follows: Section II gives background on SC-SCCs and Section III discusses the classical window decoding and its applicability to fully pipelined decoder hardware architectures. Then, Section IV introduces the proposed JWD, along with a detailed performance evaluation for hardware friendly code design and decoding schedule parameters, as well as 12nm implementation estimates for a fully pipelined decoder hardware architecture with JWD scheduling.

II. Spatially Coupled Serially Concatenated Codes

A. Serially Concatenated Codes (SCCs)

SCCs can be described by a compact graph [11] as shown in Fig. 1(a), where the information and parity sequences are rep-

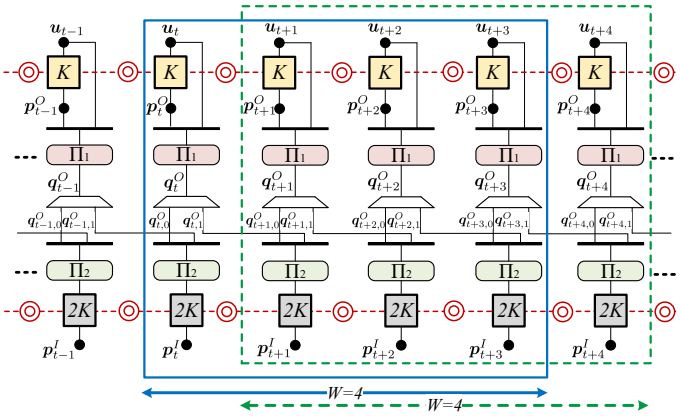

Fig. 2. Compact graph representation of an infinite chain of SC-SCC for $m=1$ and block length of K. Two decoding windows with $W=4$ are shown.

resented by black circles (*variable nodes*). The convolutional codes are shown by squares (*factor nodes*) and labeled by the corresponding trellis lengths. The SCC encoder consists of two convolutional encoders, called *outer* and *inner* encoders, that are connected in series with an interleaver. These encoders are realized by a recursive systematic convolutional (RSC) encoder with generator polynomial (1, 5/7) (see Fig. 1(b)).

The outer encoder receives the information sequence, $\boldsymbol{u}_t$, of length K bits and produces the K-bit outer parity sequence, $\boldsymbol{p}_t^O$. Then, the information and outer parity sequences are reordered by the interleaver, Π, to generate

$$\boldsymbol{q}_t^O = \Pi(\boldsymbol{u}_t, \boldsymbol{p}_t^O). \tag{1}$$

The inner encoder gets the $2K$-bit sequence $\boldsymbol{q}_t^O$ and produces the inner parity sequence, $\boldsymbol{p}_t^I$, of length $2K$ bits. Finally, the output of the SCC encoder is $\boldsymbol{v}_t = (\boldsymbol{u}_t, \boldsymbol{p}_t^O, \boldsymbol{p}_t^I)$, which is called a *code block* and will be transmitted over the channel.

To decode the inner and outer trellises the BCJR algorithm can be employed. To support efficient hardware implementation, the BCJR counterpart in the logarithmic domain, the Log-MAP algorithm, and its simplified version *max-Log MAP* (MLM) algorithm [3], [4] are typically used.

B. Spatially Coupled Serially Concatenated Codes (SC-SCCs)

The SC-SCC encoder is constructed by coupling $m+1$ component encoders, where m is the coupling memory. Fig. 2 illustrates the compact graph representation of an SC-SCC with coupling memory $m = 1$. Each component encoder consists of an outer encoder, an inner encoder, a demultiplexer, and two interleavers, which are connected together as follows.

At time instant t the outer encoder receives a block of information bits, $\boldsymbol{u}_t$, and generates the outer parity sequence, $\boldsymbol{p}_t^O$. Then, the pair of $(\boldsymbol{u}_t, \boldsymbol{p}_t^O)$ are permuted by *Interleaver 1* to generate a $2K$-bit sequence, $\boldsymbol{q}_t^O = \Pi_1(\boldsymbol{u}_t, \boldsymbol{p}_t^O)$. For coupling memory m, this sequence is divided into $m+1$ subsequences of equal length, i.e., $\boldsymbol{q}_{t,0}^O, \boldsymbol{q}_{t,1}^O, \ldots, \boldsymbol{q}_{t,m}^O$. The first subsequence, i.e. $\boldsymbol{q}_{t,0}^O$, is used as a part of the input of the inner encoder at time t and the other ones, $\boldsymbol{q}_{t,1}^O, \ldots, \boldsymbol{q}_{t,m}^O$, are used to generate the input of the next inner encoders at time $t+1, \ldots, t+m$, respectively. Therefore, at time t the sequence

$$\tilde{\boldsymbol{q}}_{t'}^O = (\boldsymbol{q}_{t,0}^O, \boldsymbol{q}_{t-1,1}^O, \ldots, \boldsymbol{q}_{t-m,m}^O) \tag{2}$$

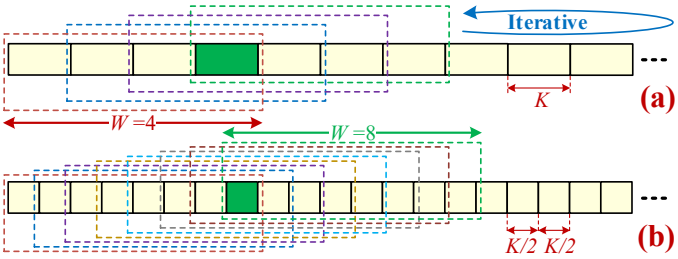

Fig. 3. Classical window decoding (WD) for SC-SCC with a latency of $4K$ bits. (a) $\{W = 4, K, \mathcal{L} = 4K\}$ and (b) $\{W' = 8, K' = K/2, \mathcal{L} = 4K\}$.

is produced by the current and previous m component encoders, which will be permuted using *Interleaver 2* (Π_2). The permuted sequence is sent to the inner encoder at time t to produce the inner parity sequence, $\boldsymbol{p}_t^I$ (see Fig. 2). Finally, the output of the SC-SCC encoder at time t is $\boldsymbol{v}_t = (\boldsymbol{u}_t, \boldsymbol{p}_t^O, \boldsymbol{p}_t^I)$.

Since the RSC encoder in Fig. 1(b) has the code rate $R = 1/2$, the overall rate of the SCC and SC-SCC encoders is $R = 1/4$. To achieve $R = 1/3$, which is considered in the following, a part of $\boldsymbol{p}_t^I$ is punctured. Moreover, we consider a non-terminated encoder, in which the final states of inner and outer encoders at time t are used as the initial states of the inner and outer encoders at time $t+1$. This concept is shown by double circles (*state variable nodes*) in Fig. 2.

III. Fully Pipelined Window Decoding

A. Window Decoding

To decode a stream of spatially coupled code-blocks, the *window decoding* (WD) approach has been proposed [10], [12]. To give an example, we consider a decoding window of length $W=4$ code blocks, which starts at time t and ends at $t+W-1$, as shown by a solid rectangle in Fig. 2. The leftmost block in the window is referred to as the *target block*. For all blocks with indices $t' = t, \cdots, t+W-1$, first the inner, then the outer decoder perform decoding as follows.

The inner decoder at time t' is fed by three sequences: the LLRs of the inner parity bits, the LLRs of the sequence defined in (2), and the a-priori LLRs. The inner decoder then produces the extrinsic LLRs and sends them back to the connected outer decoders. Similarly, the outer decoder receives three sequences: the LLRs of information bits, the LLRs of outer parity bits, and a-priori LLRs. It produces extrinsic LLRs and sends them to the connected inner decoders. Here, the a-priori LLRs of inner(outer) decoder are obtained using extrinsic LLRs of its coupled outer(inner) decoders (for simplicity, the extrinsic LLR exchange is not shown in Fig. 2). This process is repeated I_{WD} times for the current window, to decode the target block, $\boldsymbol{u}_t$, where I_{WD} is the number of iterations per window position. Then, the window is moved by one block to decode the target block $\boldsymbol{u}_{t+1}$ (see dashed rectangle in Fig. 2).

In the literature, most evaluations of the finite length performance have considered only coupling memory $m = 1$ [10], [11], although the asymptotic thresholds are slightly improved for larger m [7]. We have demonstrated in [12] how to use higher coupling memories without increasing the complexity and latency. To this end, for a certain coupling memory, m, the window size should be chosen as $W = 2m+1$, which as

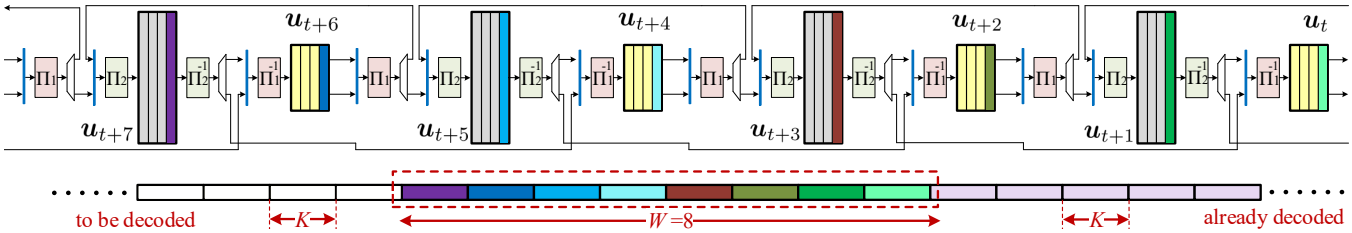

Fig. 4. High-level schematic of a decoder pipeline for fully pipelined decoding of SC-SCCs with alternating HI pipelines functioning as inner- and outer decoders connected through interleavers, Π, and deinterleavers, Π^{-1}. In this example, $n_{\text{HI}} = 8$, $I_{\text{eff}} = 4$ are considered and the code blocks are color coded.

we have shown in [12], leads to the best possible performance in the corresponding latency of

$$\mathcal{L} = K \cdot W = K \cdot (2m + 1), \qquad \text{(bits)}. \tag{3}$$

This enables us to trade between coupling memory and block length in a fixed latency without sacrificing the code strength. Thus, a given latency can be achieved by either large or small blocks. Fig. 3(a) and (b) illustrate this concept for two cases with $\{W = 4, K\}$ and $\{W' = 8, K' = K/2\}$, respectively, which both have the same latency of $\mathcal{L} = 4K$ and their optimal coupling memories are determined by (3).Since there is $W - 1$ blocks overlap between successive windows, the larger window size, e.g., Fig. 3(b), leads to a larger complexity if the same I_{WD} is used. We define an *effective number of iterations*,

$$I_{\text{eff}} = W \cdot I_{\text{WD}}, \tag{4}$$

which specifies how often the BCJR is run to decode a certain block. As a result, to have the same complexity in different SC-SCC scenarios, the same I_{eff} should be used, which can be achieved by either small W and large I_{WD} or large W and small I_{WD} as imposed by (4).

B. Fully Pipelined Iteration Unrolled Hardware Architecture

Decoding at more than 100 Gbps has been demonstrated for PCCs in a *fully pipelined iteration unrolled XMAP decoder architecture* (UXMAP) [3], [4]. In this architecture the decoding of the PCC is unrolled onto a pipeline with several *half-iteration* (HI) stages corresponding to the iterative processing. The HI stages themselves are pipelined implementations of the MLM algorithm. Assuming a completely filled pipeline, this leads to one decoded block per clock cycle at the output. In [3], the HI stages were implemented not as monolithic pipelines, but as P parallel pipelines, each processing sub-blocks of K/P for $K = 128$, and [4] investigated optimal sizes for K/P for $K \leq 512$. In contrast to [3], [4], a UXMAP decoder architecture for SC-SCCs needs two different types of HI stages. One for the inner decoder, which processes $2K$ bits, and one for the outer decoder, processing K bits. Consequently, for a given K, the area complexity of an SCC-UXMAP decoder can be expected to be at least $1.5\times$ that of a PCC-UXMAP, which further limits the maximum block size and the number of iterations. Due the unevenness in the trellis length of the inner and outer decoder iteration stages, the inner decoder iteration stages must, in addition, be implemented with different degrees of parallelism (i.e., via [3]) for full pipeline utilization. Moreover, in order to realize the spatial coupling, the feedback connections need to be introduced between the HI stages, as illustrated in Fig. 4. For simplicity, there is no distinction made in Fig. 4 between connections for extrinsic, parity, and channel values and, a coupling depth of $m = 1$ is assumed. The lower bound on the required number of HI stages, n_{HI}, is given by (4) to $I_{\text{eff}} = W \cdot I_{\text{WD}}$, provided $m \leq I_{\text{eff}} = W \cdot I_{\text{WD}}$. Thus, at least I_{eff} sub-blocks of an SC-SCC encoded data stream are present in the decoder pipeline with n_{HI} HI stages. In order to fully utilize the pipeline, while preserving the spatial coupling, the number of independently coupled streams fed to the decoder must equal to the pipeline depth of the HI stages.

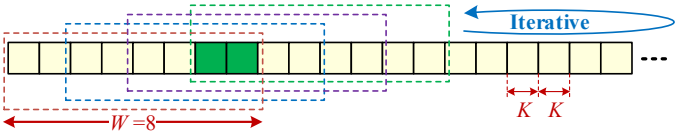

Fig. 5. Proposed JWD for SC-SCC in Fig. 3(b) with $\{W = 8, K' = K/2\}$.

C. Challenges for Fully Pipelined Classical Window Decoding

In summary, the classical window decoding of SC-SCCs in a fully pipelined decoder architecture is even more constrained in terms of block size, K, and effective number of iterations, I_{eff}, than in the PCC case. Moreover, in case of large decoding windows, W, constraining I_{eff} leads to a very low number of iterations within each decoding window, I_{WD}, following from (4), which results in a poor error correcting performance. To avoid this, larger I_{WD} and consequently larger I_{eff} should be used, which in turn increases the complexity. On the other hand, limiting the decoder to smaller values of W, which would allow for larger I_{WD}, would require larger blocks to benefit from a larger overlap between decoding windows and to keep latency unchanged according to (3). However, the larger block lengths increase the silicon area significantly, as discussed in Section IV-B. Also, small window sizes prevent from employing higher coupling memories since W is upper bounded to $2m + 1$ as discussed in Section III.

Moreover, since I_{WD} depends on W and consequently the block length (see (4)), the architecture of WD has to be flexible to adjust I_{WD} according to K, which increases hardware cost. To overcome these challenges, we propose below *jumping window decoding*, a schedule that enables a fixed I_{eff} without sacrificing I_{WD} or requiring large K.

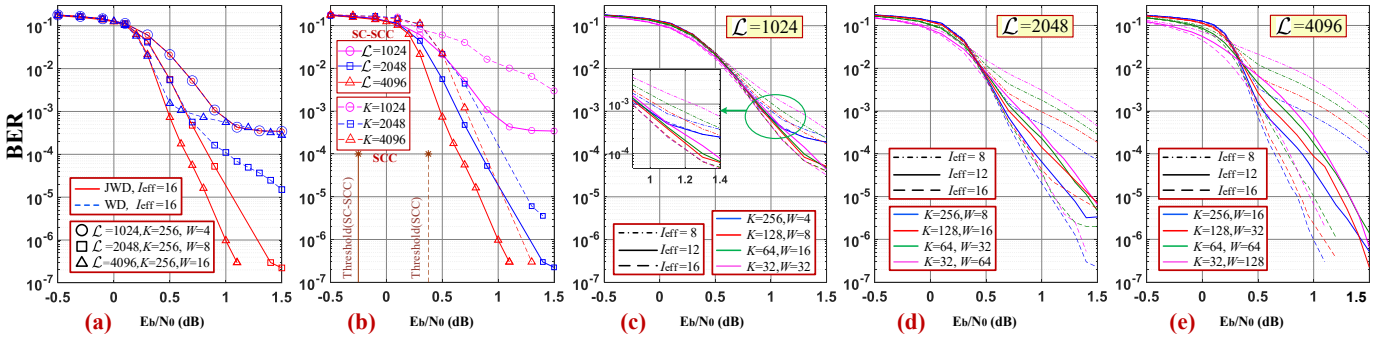

Fig. 6. (a) Performance comparison between proposed JWD and traditional WD. (b) Performance comparison of SC-SCC (with JWD) and uncoupled SCCs. Performance evaluation of JWD for different values of I_{eff}, W, and K in latency of (c) $\mathcal{L} = 1024$, (d) $\mathcal{L} = 2048$, and (e) $\mathcal{L} = 4096$ bits.

IV. Proposed Jumping Window Decoding (JWD)

The key idea of *jumping window decoding* (JWD) is to move the decoding window such that the same number of iterations per window position is used for any SC-SCC scenario, regardless of its block length, window size, and latency. This scheme demonstrates that employing small block lengths in a certain latency does not imply that the window has to be moved by small steps, which as stated in (4) leads to a large I_{eff}.

We consider the classical window decoding for the SC-SCCs with the smallest coupling memory, i.e., $m = 1$, as the reference design where the decoding window is moved by one block. According to (3) for a given latency this is corresponding to the smallest window size, i.e., $W_{\text{ref}} = 4$, and the biggest block and consequently the largest step size for shifting the window (see Fig. 3(a)). Our scheme makes it possible to perform the same number of iterations per window position, I_{JWD}, for an SC-SCC with arbitrary W and K, where

$$I_{\text{JWD}} = \frac{I_{\text{eff}}}{W_{\text{ref}}} \ . \tag{5}$$

Accordingly, after I_{JWD} iterations, the window is moved by

$$\Delta = \frac{W}{W_{\text{ref}}} \times K \qquad \text{(bits)}. \tag{6}$$

Thus, in case of equal latencies in the JWD the window is moved by the same step size, Δ, which is not necessary equal to one block and can be flexible while in the classical WD it is moved by one block, i.e, $\Delta = K$ bits. The processing flow of JWD is illustrated in Fig. 5, where the same SC-SCC scenario as the one in Fig. 3(b) is used. We will show in Section IV-B, a reduced I_{eff} without loss in error correcting performance, which reduces the silicon area of a pipelined architecture.

To clarify the concept of JWD, we present three groups of SC-SCCs in Table I, corresponding to $\mathcal{L} = 1024, 2048, 4096$, where each group includes several combinations of K and W. As an example, $I_{\text{eff}} = 16$, 12, and 8 are assumed to fix the complexity. It can be seen that the number of iterations per window in the traditional scheme, obtained from (4), is very low for small blocks. Also, I_{WD} should be rounded to the nearest integer, which leads to unequal complexity. Moreover, in many cases like the ones with $K = 32$ and 64 the traditional WD becomes impractical since $I_{\text{WD}} < 1$, which means the complexity budget has to be increased. By employing JWD, the number of iterations per window will be independent of K and W and thus the same number of iterations per window, I_{JWD}, is used for all scenarios in Table I. In this table, I_{WD}, I_{JWD}, and Δ are calculated using (4), (5), and (6), respectively by considering $W_{\text{ref}} = 4$ as the reference. However, other window sizes can alternatively be considered as the reference.

TABLE I
SC-SCC Scenarios with Fixed Latency and Complexity.

	I_{eff}	16, 12, 8	16, 12, 8	16, 12, 8	16, 12, 8
$\mathcal{L} = 1024$	K (bits)	256	128	64	32
	W	4	8	16	32
	m	1	3	7	15
	I_{WD}	4, 3, 2	2, 1.5, 1	1, 0.75, 0.5	0.5, 0, 0
	I_{JWD}	4, 3, 2	4, 3, 2	4, 3, 2	4, 3, 2
	Δ (bits)	256	256	256	256
$\mathcal{L} = 2048$	K (bits)	256	128	64	32
	W	8	16	32	64
	m	3	7	15	31
	I_{WD}	2, 1.5, 1	1, 0.75, 0.5	0.5, 0, 0	0, 0, 0
	I_{JWD}	4, 3, 2	4, 3, 2	4, 3, 2	4, 3, 2
	Δ (bits)	512	512	512	512
$\mathcal{L} = 4096$	K (bits)	256	128	64	32
	W	16	32	64	128
	m	7	15	31	63
	I_{WD}	1, 0.75, 0.5	0.5, 0, 0	0, 0, 0	0, 0, 0
	I_{JWD}	4, 3, 2	4, 3, 2	4, 3, 2	4, 3, 2
	Δ (bits)	1024	1024	1024	1024

A. Performance Evaluation

We have evaluated the performance of JWD in the SC-SCC scenarios listed in Table I. In each scenario, we use a coupling memory of $m = W/2 - 1$, which has been shown to give the best performance for a given $\{K, W\}$ [12]. Fig. 6(a) depicts the BER performance of a JWD and a traditional WD for the SC-SCCs in the first column of Table I. In case of $\mathcal{L} = 1024$ the BER curves of WD and JWD are exactly the same since $W = W_{\text{ref}}$ and therefore $\Delta = K$ for both cases. Also, Fig. 6(a) shows that JWD improves the performance by 0.5-0.9 dB and lowers the error floor, e.g., from 10^{-3} to lower than 10^{-6} in $\mathcal{L} = 4096$. However, it is worth noting that the performance improvement is not the only benefit of using JWD. As discussed in Section III-C, employing the traditional WD for small K is not possible if $I_{\text{eff}} < W$, i.e., low to moderate complexity budget. In such cases, even the lowest number of iterations, $I_{\text{WD}} = 1$, results in a large $I_{\text{eff}} = W$, which increases the complexity significantly, prohibiting efficient hardware implementation and achieving high throughput. The proposed scheme, on the other hand, makes it possible to

TABLE II
PLACE & ROUTE RESULTS OF THE MLM COMPUTE KERNELS.

	BMU	ACS	SOU^{I}	SOU^{O}
Technology	12nm FINFET LP			
V_{dd} (V) / T (°C) / Freq. (MHz)	0.72 / 125 / 1000			
Area (μm^2)	572	784	2025	2550

perform decoding for very short block lengths with a lower I_{eff}, i.e., lower complexity, while the performance is equal to or better than the traditional WD approach. Fig. 6(b) illustrates the performance of the proposed scheme and the uncoupled codes, SCCs, for different latencies. Also, the corresponding decoding thresholds, i.e., theoretical limit on the performance [11], are shown by vertical lines. It can be seen that in all cases the proposed scheme outperforms the SCCs by 0.2-0.8 dB. Note, that to have a fair comparison the same complexity is assumed by employing I_{eff}=16 for coupled and uncoupled codes and also the same latency is considered by adjusting K in the SCC equal to $\mathcal{L}$ in the SC-SCC.We have investigated the impact of I_{eff} on the performance of JWD for the scenarios listed in Table I. Fig. 6(c), (d), and (e) illustrate the BER curves for $\mathcal{L} = 1024, 2048,$ and 4096, respectively. It can be seen that the performance gap between $I_{\text{eff}} = 8$ and $I_{\text{eff}} = 12$ is larger than the gap between $I_{\text{eff}} = 12$ and $I_{\text{eff}} = 16$. Also, as expected, the performance is improved by increasing the latency. Note that the reason of the performance gap between different block lengths at a given latency is the poor performance of short interleavers. This gap could be smaller if the interleavers are designed jointly. The effective number of iterations trades between performance and silicon area, i.e., number of pipeline stages in Fig. 4. Thus, depending on the application, the most efficient value of I_{eff} is specified by considering a performance-area tradeoff.

B. Area Complexity Estimations

To illustrate this tradeoff and demonstrate the impact of JWD, we give hardware estimations for fully pipelined SC-SCCs. It is well established, that the area of fully pipelined implementations for the MLM is dominated by the compute kernels of the MLM [3], [4], [13]: The *branch metric unit* (BMU), *recursion unit* (RU) and the *soft output unit* (SOU) largely determine the area of the inner and outer HI stages:

$$\begin{aligned} A_{\text{HI}}^{\text{inner}} &= 2 \cdot K \cdot (2 \cdot A_{\text{BMU}} + 2 \cdot A_{\text{RU}} + A_{\text{SOU}^{\text{I}}}) + A_{\text{FIFO}}, \\ A_{\text{HI}}^{\text{outer}} &= K \cdot (2 \cdot A_{\text{BMU}} + 2 \cdot A_{\text{RU}} + A_{\text{SOU}^{\text{O}}}) + A_{\text{FIFO}} \end{aligned} \quad (7)$$

where the outer decoder stages use SOUs that generate extrinsic information on parities as well (SOU^{O}) and the inner decoder stages use SOUs, which only generate extrinsics on the information bits (SOU^{I}). Table II gives placed and route results for radix-4 compute kernels of a UXMAP decoder in 12nm technology for a target frequency of 1000 MHz. The overall area estimate for selected coding scenarios is given in Table III calculated from (7) and the total number of HI stages $n_{\text{HI}} = 2 \cdot I_{\text{eff}}$. Note, that for this qualitative estimate, A_{FIFO} is not taken into account. However, together with the error correcting performance results for the proposed JWD these qualitative estimates indicate, that the area consumption for a fully pipelined SC-SCC decoder can be drastically reduced through a reduction in I_{eff} without loss in BER performance.

TABLE III
AREA AND THROUGHPUT OF PIPELINES FOR DIFFERENT I_{eff}.

Block Size (K)	256	128	64	32
I_{eff}	Area (mm^2)‡			
16	26.66	13.33	6.66	3.33
12	20.00	10.00	5.00	2.50
8	13.33	6.66	3.33	1.66
Throughput† (Gbps)	204.8	102.4	51.2	25.6

‡ Area estimated based on Table II and (7), †: K/Freq.

V. CONCLUSION

In this paper, we highlighted the challenges of employing classical WD for the SC-SCCs. We proposed a new decoding approach, called *jumping window decoding* (JWD), which opens the door for the practical pipelined implementation of the SC-SCCs. We disscussed the necessary modifications to the UXMAP architecture and gave area estimates derived for different SC-SCCs based on PnR results in 12nm technology. Moreover, BER results show that our scheme achieves better performance compared to the classical WD of SC-SCCs and uncoupled SCCs. As such, JWD enables fully pipelined implementation for high throughput decoding of SC-SCCs.

ACKNOWLEDGMENT

This work was partially funded by the French National Research Agency TurboLEAP project (ANR-20-CE25-0007).

REFERENCES

[1] EPIC Project, "Enabling Practical Wireless Tb/s Communications with Next Generation Channel Coding (EPIC)," 2020, https://epic-h2020.eu/.

[2] C. Kestel, M. Herrmann, and N. Wehn, "When channel coding hits the implementation wall," in *2018 IEEE 10th International Symposium on Turbo Codes Iterative Information Processing (ISTC)*, 2018, pp. 1–6.

[3] S. Weithoffer, O. Griebel, R. Klaimi, C. A. Nour, and N. Wehn, "Advanced hardware architectures for turbo code decoding beyond 100 Gb/s," in *IEEE Wireless Comm. and Netw. Conf.(WCNC)*, 2020, pp. 1–6.

[4] S. Weithoffer, R. Klaimi, C. A. Nour, N. Wehn, and C. Douillard, "Fully pipelined iteration unrolled decoders the road to TB/S turbo decoding," in *ICASSP 2020 - 2020 IEEE Int. Conf. on Acoustics, Speech and Signal Pr. (ICASSP)*, 2020, pp. 5115–5119.

[5] 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. on Inf. Theory*, vol. 57, no. 2, pp. 803–834, Feb 2011.

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

[7] S. Moloudi, M. Lentmaier, and A. Graell i Amat, "Spatially coupled turbo-like codes," *IEEE Trans. on Inf. Theory*, vol. 63, no. 10, pp. 6199–6215, Oct 2017.

[8] S. Moloudi, M. Lentmaier, and A.Graell i Amat, "Spatially coupled turbo codes," in *2014 8th Int. Symp. on Turbo Codes and Iterative Inf. Pr. (ISTC)*, Aug 2014, pp. 82–86.

[9] N. U. Hassan, M. Schlüter, and G. P. Fettweis, "Fully parallel window decoder architecture for spatially-coupled LDPC codes," in *2016 IEEE Int. Conf. on Commun. (ICC)*, 2016, pp. 1–6.

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

[11] S. Moloudi, M. Lentmaier, and A.Graell i Amat, "Spatially coupled turbo-like codes: A new trade-off between waterfall and error floor," *IEEE Trans. on Comm.*, vol. 67, no. 5, pp. 3114–3123, May 2019.

[12] M. Mahdavi, M. Farooq, L. Liu, O. Edfors, V. Öwall, and M. Lentmaier, "The effect of coupling memory and block length on spatially coupled serially concatenated codes," in *2021 IEEE 93rd Vehicular Technology Conference (VTC)*, 2021.

[13] S. Weithoffer, M. Herrmann, C. Kestel, and N. Wehn, "Advanced wireless digital baseband signal processing beyond 100 Gbit/s," in *2017 IEEE Int. Workshop on Signal Pr. Systems (SiPS)*, 2017, pp. 1–6.