A 14 ns-Latency 9 Gb/s 0.44 mm^2 62 pJ/b Short-Blocklength LDPC Decoder ASIC in 22FDX

Darja Nonaca*, Jérémy Guichemerre*, Reinhard Wiesmayr*, Nihat Engin Tunali◇, and Christoph Studer*

*Department of Information Technology and Electrical Engineering, ETH Zurich, Switzerland; ◇Independent Researcher

***Abstract*—Ultra-reliable low latency communication (URLLC) is a key part of 5G wireless systems. Achieving low latency necessitates codes with short blocklengths for which polar codes with successive cancellation list (SCL) decoding typically outperform message-passing (MP)-based decoding of low-density parity-check (LDPC) codes. However, SCL decoders are known to exhibit high latency and poor area efficiency. In this paper, we propose a new short-blocklength multi-rate binary LDPC code that outperforms the 5G-LDPC code for the same blocklength and is suitable for URLLC applications using fully parallel MP. To demonstrate our code's efficacy, we present a 0.44 mm^2 GlobalFoundries 22FDX LDPC decoder ASIC which supports three rates and achieves the lowest-in-class decoding latency of 14 ns while reaching an information throughput of 9 Gb/s at 62 pJ/b energy efficiency for a rate-1/2 code with 128-bit blocklength.**

I. Introduction

Fifth-generation (5G) wireless systems include ultra-reliable low-latency communication (URLLC) [1] to support applications in industry automation, autonomous driving, and telemedicine. Achieving low latency requires codes with short blocklengths for which polar codes with successive cancellation list (SCL) decoding are suitable candidates [2]. However, SCL decoding is highly sequential, which results in decoder implementations that achieve long latency and rather poor area efficiency [3], [4]. In contrast, low-density parity-check (LDPC) codes [5] are adopted in virtually all modern communication systems because message-passing (MP)-based decoders achieve high throughput and excellent area- and energy-efficiency while delivering near capacity performance in the long blocklength regime [6]. However, MP-based LDPC decoders typically do *not* perform well with short-blocklength codes, and despite their potential use for URLLC applications, little is known about short-blocklength LDPC codes that achieve low error rates with MP-based decoding and corresponding high-throughput, low-latency hardware implementations.

A. Contributions

We propose a novel short-blocklength binary quasi-cyclic (QC) LDPC code [7] that supports three rates. Our code is optimized to achieve good error-correcting performance with MP-based decoding, which enables efficient hardware implementations that achieve low latency. To demonstrate our code's effectiveness, we propose an ASIC implementation of a flooding-schedule MP-based decoder that processes each decoding iteration in a fully parallel fashion. To improve error-rate performance, we employ machine-learning-optimized edge-adaptive normalized min-sum (ANMS) check-node updates [8]; to improve energy efficiency, we utilize early termination (ET); and to improve throughput without sacrificing latency, we use pipeline interleaving [9, Sec. 3.7.6] to process two independent codewords simultaneously. Our decoder has been fabricated in GlobalFoundries' 22FDX technology, occupies an area of only 0.44 mm^2, and measurements reveal information throughputs of 9, 16 and 22 Gb/s at record latencies of 13.78, 16.06, and 17.52 ns for code rates 1/2, 2/3, and 3/4, respectively.

This work has received funding from the Swiss State Secretariat for Education, Research, and Innovation (SERI) under the SwissChips initiative. The authors thank GlobalFoundries for providing silicon fabrication through the 22FDX University Program. The authors also thank Oscar Castañeda for his assistance with the ASIC design flow. Contact author: D. Nonaca (e-mail: dnonaca@iis.ee.ethz.ch).

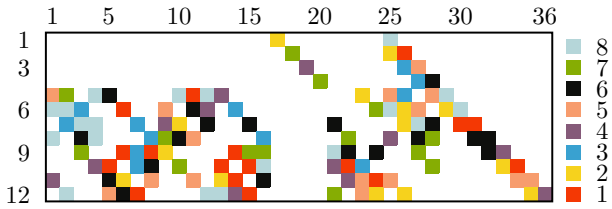


Fig. 1: Base graph of the proposed short-blocklength binary quasi-cyclic LDPC code with lifting factor $Z = 8$. Each square represents a cyclically shifted 8×8 identity matrix with the shift amount indicated by the corresponding color. The entire matrix is used for blocklengths of 288 coded bits; for blocklengths 224 and 160 coded bits, we remove the first 8 and 16 block columns, respectively.

B. Relevant Prior Work

The literature describes numerous LDPC decoder designs, which mostly target long blocklengths. Reference [10] provides place-and-route results for a fully parallel LDPC decoder supporting a blocklength of 2048 bits, which unrolls all iterations to achieve extremely high throughput. Reference [11] targets the same blocklength and achieves state-of-the-art area and energy efficiency at the expense of reduced throughput with a partially parallel decoder architecture. However, only little is known about decoders targeting short LDPC codes, with the exception of [12], [13]. Similar to [12], [13], our decoder is rate-configurable, and similar to [12], we use early termination (ET) to save power and pipeline interleaving to improve throughput. In contrast to [12], [13], we propose a novel LDPC code of even shorter blocklength and a decoder ASIC that achieves the lowest latency reported in the literature. In addition, our decoder achieves excellent area efficiency and error rate performance and is, thus, suitable for URLLC.

Polar codes and corresponding SCL decoders are known to perform well with short blocklengths. In contrast to the state-of-the-art polar decoder implementations from [3], [4], we implement a hardware-friendly LDPC decoder, achieving orders

of magnitude lower latency. Furthermore, our decoder produces soft outputs on the coded bits, which is useful for hybrid automatic repeat request or iterative detection and decoding.

Specifically for URLLC applications, there exists the recently proposed block orthogonal sparse superposition (BOSS) decoder [14]; this decoder achieves low throughput and is area inefficient. In comparison to all of the discussed state-of-the-art designs, our decoder achieves the shortest decoding latency while reaching the highest throughput among all designs targeting short blocklengths; see Sec. V for a comparison.

II. LDPC Decoding

We consider forward error correction with binary block codes, where $\mathbf{b} \in \{0,1\}^k$ is the information bit vector of length k to be transmitted and $\mathbf{c} = \mathbf{G}\mathbf{b} \in \{0,1\}^n$ the resulting codeword of length n. Here, $\mathbf{G}$ is the generator matrix, typically derived from the parity-check (PC) matrix $\mathbf{H}$ for which every codeword satisfies $\mathbf{H}\mathbf{c} = \mathbf{0}$. Instead of transmitting $\mathbf{c}$, one often transmits $\mathbf{c}' \in \{0,1\}^{n'}$ with $n' \leq n$ by *puncturing* (removing) a fixed set of bits from $\mathbf{c}$ that will later be retrieved by the decoder. With puncturing, the code rate is given by $R = k/n'$.

The coded and punctured bits $\mathbf{c}'$ are mapped to the symbol sequence $\mathbf{x}$ and transmitted over a channel. The receiver then obtains $\mathbf{y} = h(\mathbf{x})$, where h models the channel. At the receiver, the entries y_i of $\mathbf{y}$ are used to compute the a-posteriori likelihood of each bit c_i, from the codeword $\mathbf{c}$, described by the log-likelihood ratio (LLR) $\ell_i = \log\left(\frac{\mathbb{P}(c_i=-1|y_i)}{\mathbb{P}(c_i=+1|y_i)}\right)$; the LLR values associated with the punctured bits are set to zero.

In what follows, we focus on LDPC codes [5], [6], which can be represented as a bipartite graph consisting of variable nodes (VNs) associated with the LLR values, and check nodes (CNs) associated with the parity checks (corresponding to the rows of $\mathbf{H}$). LDPC codes can be decoded using MP [6] by iteratively propagating beliefs along the edges of the bipartite graph in an iterative fashion. We implement the flooding MP schedule, which alternates between updating the messages associated with all CNs and all VNs, together with the edge-adaptive normalized min-sum approximation (NMS) [15]

$$\ell\Big(c_j = \bigoplus_{k\in N(i)\setminus j} c_k\Big) \approx \alpha_{ij} \times \prod_{k\in N(i)\setminus j} \operatorname{sign} \ell_k \times \min_{k\in N(i)\setminus j} |\ell_k| \,. \quad (1)$$

Here, $N(i)$ is the set of the VNs connected to the ith CN and α_{ij} is a predefined scaling factor associated with the edge connecting the ith CN and the VN associated to c_j. The scaling factors are obtained off-line through machine learning training as introduced in [8], with additional fine-tuning utilizing SNR deweighting and a BLER loss function as proposed in [16].[1] We terminate the decoding process either if a maximum number of iterations $I_{\max}$ is reached or if a valid codeword is found.

III. Custom Short-Blocklength QC-LDPC Code

To achieve low error rates with MP-based decoding, we propose a custom short-blocklength binary quasi-cyclic (QC) [7] LDPC code derived from the family of rate-compatible protograph-based AR4A codes [17]. The parity-check matrix of the rate $3/4$ code is represented by the protograph

$$\mathbf{H}_{\text{proto}} = \begin{bmatrix} 0 & 0 & 0 & 0 & 2 & 0 & 2 & 0 & 0 \\ 3 & 1 & 3 & 1 & 0 & 1 & 3 & 2 & 0 \\ 1 & 3 & 1 & 3 & 0 & 3 & 1 & 0 & 2 \end{bmatrix}, \quad (2)$$

where each entry represents the constraints on connectivity of the variable nodes and check nodes. For example, the variable nodes in the first column would only be connected to three check nodes in the middle row and a single check node in the last row. Code construction consists of two stages: In the first stage, each entry in the protograph matrix $\mathbf{H}_{\text{proto}}$ is expanded (lifted) by a lifting factor $Z = 4$ with the progressive edge growth (PEG) algorithm [18]. At this stage, the matrix is not QC, but has increased girth for improved MP decoding performance. In the second stage, the PEG expanded PC matrix is lifted by $Z = 8$ with the approximate cycle extrinsic (ACE) message degree algorithm [19], by setting $d_{\text{ACE}} = 3$ and $\eta_{\text{ACE}} = 13$; this ensures that length $2\times d_{\text{ACE}}$ cycles have an ACE message degree of 13. The larger the ACE message degree, the better a cycle is connected to the rest of the graph. The resulting PC matrix consists of cyclically shifted identity matrices of size $Z = 8$ (cf. Fig. 1). Combining the PEG and ACE algorithms is known to achieve near Shannon performance [17].

Our proposed code is depicted in Fig. 1 and supports three different code lengths for three different rates: $R = 1/2$, $R = 2/3$, and $R = 3/4$. To adjust the code rate (and length), the columns of the PC matrix $\mathbf{H}$ are progressively removed. For $R = 3/4$, the entire PC matrix $\mathbf{H}$ is used; for $R = 2/3$ and $R = 1/2$, the first 64 and 128 columns are removed, respectively. In this code, we puncture the information bits $[1\,{:}\,32]$, $[65\,{:}\,96]$, and $[129\,{:}\,160]$ for rates $R = 1/2$, $R = 2/3$, and $R = 3/4$, respectively. This procedure leads to the code lengths $(k, n') = (64, 128)$, $(k, n') = (128, 192)$, and $(k, n') = (192, 256)$, for rates $R = 1/2$, $R = 2/3$, and $R = 3/4$, respectively.

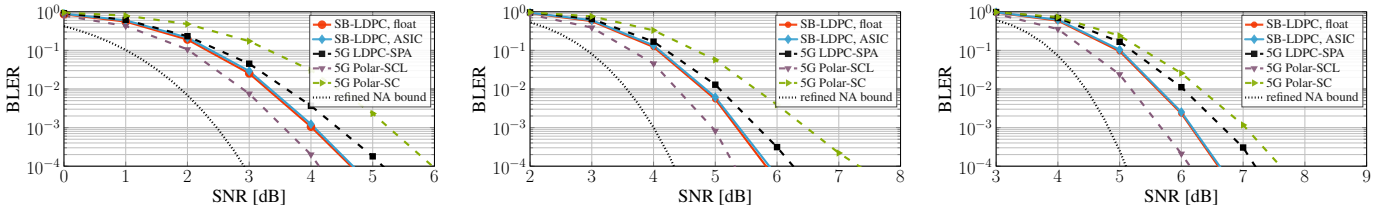
To demonstrate the efficacy of our short-blocklength LDPC (SB-LDPC) code, we simulate a memoryless, real-valued, AWGN channel with binary phase-shift keying. Fig. 2 evaluates the block error rate (BLER) of the proposed SB-LDPC code with ANMS decoding for rates $1/2$, $2/3$, and $3/4$. We compare the BLER of our decoding method applied to our SB-LDPC code with the theoretical bound called refined normal approximation (NA) [20], the 5G LDPC code decoded with the sum-product MP algorithm (SPA), and the 5G polar codes with SC and SCL decoding. For every LDPC decoder, we use the flooding scheduling with $I_{\max} = 10$ iterations; for polar SCL, we use a list size 8. Fig. 2 reveals that for all three rates, our code and decoding scheme exhibit a loss of approximately 1.5 dB relative to the refined NA bound and 0.5 dB from 5G polar codes with SCL decoding. Depending on the rate, we achieve a 0.3 to 0.5 dB gain over 5G LDPC codes with SPA decoding and a 0.75 to 1.25 dB gain over 5G polar codes with SC decoding.Fig. 2 also demonstrates that our fixed-point hardware design exhibits virtually no BLER loss.

IV. VLSI Architecture

A. Architecture Overview

To achieve high throughput at low latency, we implement the flooding schedule in a fully parallel fashion per MP iteration.

[1] As in [8], we optimize one set of trainable CN-to-VN edge-weights for the scaling parameters α_{ij} that are shared among all MP iterations.



(a) $R = 1/2, n = 160, n' = 128$ (b) $R = 2/3, n = 224, n' = 192$ (c) $R = 3/4, n = 288, n' = 256$

Fig. 2: BLER comparison at a fixed blocklength and rate of the proposed short blocklength (SB) LDPC code using the update rule in (1) with the 5G LDPC code using the SPA update rule with the same number of decoding iterations $I_{\max} = 10$, and with 5G polar codes with SC and SCL decoding using list size 8. Our SB-LDPC code is 1.5 dB away from the refined NA bound and outperforms the 5G LDPC code by $0.3 - 0.5$ dB at $0.1\,\%$ BLER for all three rates.

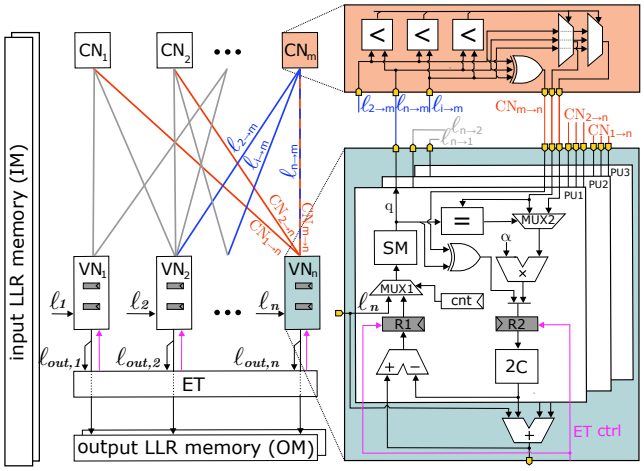


Fig. 3: Architecture of the proposed SB-LDPC decoder. The input/output LLR memories contains two sets of LLR values associated to two independent codewords. In this example, $\mathrm{VN_n}$ contains three PUs as it connects to three CNs ($\mathrm{CN_1}$, $\mathrm{CN_2}$, and $\mathrm{CN_m}$); each PU contains two pipeline registers, which enable us to process two codewords at the same time. The ET block triggers pipeline register freezing in case a valid codeword is found.

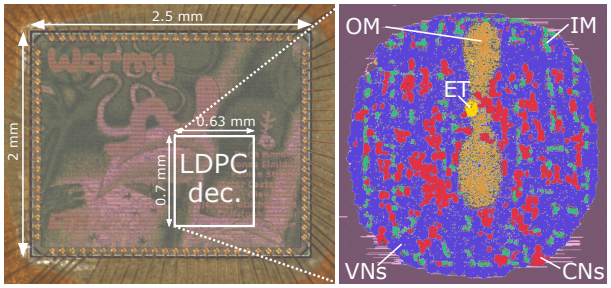


Fig. 4: ASIC micrograph (left) containing our SB-LDPC decoder and detailed area breakdown (right). OM, IM, and ET are output LLR memory, input LLR memory, and early termination block, respectively. Each VN block contains PUs, one for each VN-CN connection from the given VN block.

The left side of Fig. 3 provides an architecture overview. Our decoder consists of $N = 288$ VN blocks (one for each code bit of the longest blocklength) and $M = 96$ CN blocks. VN processing and CN processing together carry out NMS [15] MP as in (1). To improve throughput, we deploy pipeline interleaving [9, Sec. 3.7.6] with pipeline registers in the VN-blocks; this allows simultaneous processing of two independent codewords while shortening the critical path. To improve energy efficiency, we utilize ET, which dynamically freezes the appropriate pipeline stages to suppress switching activity.

B. Architecture Details

The right side of Fig. 3 provides the architecture details. Each VN block contains a multi-operand adder and processing units (PUs). Each PU is responsible for one of the VN-CN connections of the VN block (determined by the PC matrix in Fig. 1). Starting at pipeline register R1, we first convert the VN-to-CN messages (or the LLR values ℓ_i in the first iteration) from two's complement to sign magnitude values q (SM block). Then, these messages are sent to the appropriate CN blocks, which extract the smallest and second smallest absolute values of the messages coming from the connected VN blocks (see blue connections in Fig. 3). Each CN block also computes the associated parity check by XOR-ing all incoming sign bits. The CN processing is facilitated by the sign magnitude format. Back in the PU, the value q is compared to the minimum from the incoming CN. If they match, then MUX2 passes the second-smallest absolute value; if not, MUX2 passes the minimum. The selected value is then scaled using the predefined weight as in (1); since all of the weights are determined offline, only multiplications with constants are necessary. The resulting sign-magnitude value is then stored in pipeline register R2. After R2, the message is converted to two's complement (2C block) in preparation for summation. Each VN block then sums all the CN-to-VN messages previously processed by the internal PUs, including the associated input LLR ℓ_i, obtaining the intrinsic LLR $\ell_{\mathrm{out},i}$ associated with code bit c_i. The intrinsic LLRs are fed back to the PUs, where they are converted into extrinsic messages and stored in R1. Decoding is repeated for $I_{\max}$ iterations. The ET block uses the intrinsic LLR's sign bits to compute the PC. If ET is triggered for one of the two interleaved codewords (pink signal in Fig. 3), then the pipeline registers associated with the unterminated codeword keep their outputs for one additional clock cycle to avoid switching activity associated with the early-terminated codeword. If ET is triggered for both codewords, then all pipeline registers are frozen. To minimize the BLER implementation loss (cf. Fig. 2), we use fixed-point numbers with 4 integer bits, 2 fractional bits, and 1 sign bit. The multi-operand adder inside the VNs uses one additional integer bit.

V. Implementation Results and Comparison

Fig. 4 depicts our multi-project chip fabricated in Global-Foundries' 22 FDX™ FD-SOI technology. The proposed SB-LDPC decoder occupies an area of $0.44\,\mathrm{mm}^2$, including the input memory (IM) to store the two input LLR vectors and output memory (OM) for the two output LLR vectors, both of size 7×288 bits. At nominal 0.8 V core supply voltage and 25° C, the decoder achieves a maximum clock frequency of

TABLE I: ASIC measurement results and comparison with other decoder designs.

	This work $R=1/2$	This work $R=2/3$	This work $R=3/4$	Ghanaatian [10]	Zhang [11]	Milicevic [12]	Verma [13]	Teng [4]	Giard [3]	Kam [14]
Code/Algorithm	LDPC-ANMS			LDPC-LUT	LDPC-OMS	LDPC-MS	LDPC-OMS	Polar RNN-BP	Polar-SCL	BOSS
(k, n')	(64,128)	(128,192)	(192,256)	(1723,2048)	(1723,2048)	(336,672)	(352,528)	(128,256)	(512,1024)	(15,128)
Rate R	0.5	0.67	0.75	0.84	0.84	0.5	0.67	0.5	0.5	0.12
Max. iterations $I_{\max}$	10	10	10	5	14	10	10	5	-	-
Min. SNR [dB] @BLER$=10^{-3}$	4	5.6	6.2	6.90[a]	6.41[a]	4.38[a]	2.33[i]	BER reported	2.4[a]	-2.55[a]
Technology (L) [nm]	22	22	22	28	65	28	110	40	28	28
Fabricated?	yes	yes	yes	no	yes	yes	yes	yes	yes	yes
Supply voltage (V_s) [V]	0.8	0.8	0.8	1	1.2	0.9	1.2	0.9	0.9	0.95
Area [mm^2]	0.44	0.44	0.44	16.2	5.05	1.99	1.96	0.18	0.44	0.37
$f_{\max}$ no ET / ET [MHz]	1452 / 1356	1246 / 809	1142 / 776	862	700	202	72.7	225	308	590
Power @$f_{\max}$ no ET / ET [mW]	575 / 296	636 / 218	699 / 236	13350	- / 2800	408 / 283	150	12.8	23.3	33.3
Inf. throughput @$f_{\max}$ and $I_{\max}$ [Gb/s]	9.29[b]	15.94[b]	21.92[b]	494.68	40.13[d]	3.39	1.11	0.41	0.06	0.68
Latency @$I_{\max}$ [ns]	13.78[b]	16.06[b]	17.52[b]	69.6	137	793	120	310	7820	21.9
Area efficiency [Gb/s/mm^2]	21.12[b]	36.24[b]	49.83[b]	30.53	7.94[d]	1.70	0.56	2.28	0.14	1.84
Energy efficiency @$I_{\max}$ [pJ/b]	61.88[b]	39.88[b]	31.88[b]	26.99	69.78[c,d]	120.36[b]	134.74	31.11	355.73	48.66
Scaled area efficiency[e] [Gb/s/mm^2]	21.12[b]	36.24[b]	49.83[b]	62.95	204.95[d]	3.51	71	13.74	0.30	3.81
Scaled energy efficiency[f] [pJ/b]	61.88[b]	39.88[b]	31.88[b]	13.57	10.50[c,d]	74.72[b]	11.98	13.52	220.84	27.12
Scaled inf. throughput[g] @$f_{\max}$ and $I_{\max}$ [Gb/s]	9.29[b]	15.94[b]	21.92[b]	629.60	118.56[d]	4.31	5.56	0.74	0.08	0.87
Scaled latency[h] [ns]	**13.78**[b]	**16.06**[b]	**17.52**[b]	54.69	46.37[d]	623.08	24	170.5	6144.29	17.21

[a] Conversion from E_b/N_0 to SNR. [b] Considering decoding without ET. [c] Considering decoding with ET. [d] Average throughput reported; the average throughout value taken at 5.5 dB with only 1.5 iterations. [e] Technology scaling by S^3, [f] $S^{-1}U^{-2}$, [g] S, and [h] S^{-1}, where $S = L/22$ is the relative dimension to 22 nm and $U = V_s/0.8$ the relative core voltage to 0.8 V. [i] Reported BLER is below the refined NA bound [20] for the corresponding blocklength and rate.

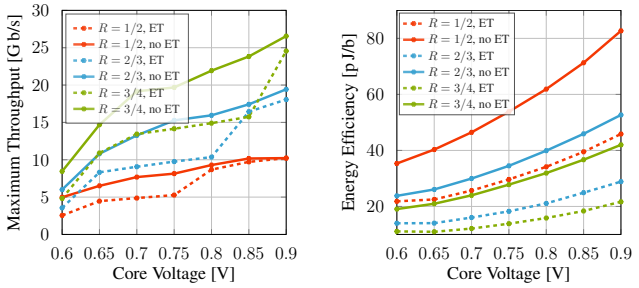


Fig. 5: Measured maximum throughput (left) and energy efficiency (right) achieved by our LDPC decoder for the three supported rates. The measurements for each rate were taken at the minimum SNR that achieves 0.1% BLER, i.e., 4 dB, 5.4 dB, and 6.2 dB for rates $R = 1/2$, $R = 2/3$, $R = 3/4$, respectively.

1.452 GHz, 1.246 GHz, and 1.142 GHz for the rates $R = 1/2$, $R = 2/3$, $R = 3/4$, respectively. Our ASIC achieves higher data rates with longer blocklengths, which activates more VN blocks; the power and critical path leads to a reduced clock frequency $f_{\max}$. With $I_{\max} = 10$, we achieve an information throughput $\theta_k = k f_{\max}/I_{\max}$ of 9.29, 15.94, and 21.92 Gb/s at rates $R = 1/2$, $R = 2/3$, $R = 3/4$, respectively, as shown on the left side of Fig. 5. ET is in the critical path and, when activated, reduces the throughput. Nonetheless, ET substantially improves energy efficiency; see Fig. 5.

Tbl. I compares our SB-LDPC decoder with the state of the art. With respect to LDPC decoders for long blocklengths [10], [11], our ASIC achieves the shortest latency. With respect to short blocklength codes [3], [4], [12]–[14], our ASIC achieves the highest information throughput and is the most area efficient except for [13]. With respect to the LDPC decoders [10]–[13] for long and short blocklengths, our custom short-blocklength LDPC code and ASIC achieve lower BLER (considering equal code rates), except for [13], which reports a BLER below the refined NA bound. In summary, our SB-LDPC decoder achieves the shortest latency while being competitive in (scaled) throughput and area efficiency as well as BLER performance.

VI. Conclusions

We have proposed a new short blocklength LDPC code along with a decoder ASIC that relies on a parallel message-passing architecture. The proposed rate-configurable decoder outperforms existing LDPC decoders for short blocklengths in terms of BLER and achieves a throughput of 9, 16, and 22 Gb/s for rates $R = 1/2$, $R = 2/3$, and $R = 3/4$, respectively. Furthermore, our decoder achieves decoding latencies in the range of 14 ns to 18 ns—the shortest reported in the literature.

References

[1] X. Lin, "An overview of 5G advanced evolution in 3GPP release 18," *IEEE Comm. Stand. Mag.*, 2022.

[2] I. Tal *et al.*, "List decoding of polar codes," *IEEE Int. Symp. Inf. Theory*, 2011.

[3] P. Giard *et al.*, "PolarBear: A 28-nm FD-SOI ASIC for decoding of polar codes," *IEEE J. on Emer. and Sel. Topics in Circ. and Syst.*, 2017.

[4] C.-F. Teng *et al.*, "A 7.8–13.6 pJ/b ultra-low latency and reconfigurable neural network-assisted polar decoder with multi-code length support," *IEEE Trans. Circ. and Syst. I: Regular Papers*, 2021.

[5] R. Gallager, "Low-density parity-check codes," *IRE Trans. Inf. Theory*, 1962.

[6] D. MacKay, "Good error-correcting codes based on very sparse matrices," *IEEE Trans. Inf. Theory*, 1999.

[7] M. Fossorier, "Quasicyclic low-density parity-check codes from circulant permutation matrices," *IEEE Trans. Inf. Theory*, 2004.

[8] E. Nachmani *et al.*, "Deep learning methods for improved decoding of linear codes," *IEEE J. Sel. Topics in Sig. Proc.*, 2018.

[9] H. Kaeslin, "Top-down digital VLSI design: from architectures to gate-level circuits and FPGAs," Elsevier, 2015.

[10] R. Ghanaatian *et al.*, "A 588-Gb/s LDPC decoder based on finite-alphabet message passing," *IEEE Trans. VLSI Syst.*, 2018.

[11] Z. Zhang *et al.*, "An efficient 10GBASE-T ethernet LDPC decoder design with low error floors," *IEEE J. Solid State Circ.*, 2010.

[12] M. Milicevic *et al.*, "A multi-Gb/s frame-interleaved LDPC decoder with path-unrolled message passing in 28-nm CMOS," *IEEE Trans. on Very Large Scale Integration (VLSI) Syst.*, 2018.

[13] A. Verma *et al.*, "High-throughput and hardware-efficient ASIC-chip fabrication of reconfigurable LDPC/polar decoder for mMTC and URLLC 5G-NR applications," *IEEE Trans. Circ. and Syst. I: Reg. Papers*, 2024.

[14] D. Kam *et al.*, "2.8 a 21.9ns 15.7 Gbps/mm² (128,15) BOSS FEC decoder for 5G/6G URLLC applications," in *IEEE Int. Solid State Circ. Conf.*, 2024.

[15] J. Chen *et al.*, "Density evolution for two improved BP-based decoding algorithms of LDPC codes," *IEEE Comm. Letters*, 2002.

[16] R. Wiesmayr *et al.*, "Bit error and block error rate training for ML-assisted communication," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process. (ICASSP)*, Jun. 2023.

[17] D. Divsalar *et al.*, "Capacity-approaching protograph codes," *IEEE J. Sel. Areas in Comm.*, 2009.

[18] X.-Y. Hu *et al.*, "Progressive edge-growth Tanner graphs," in *IEEE Global Telecomm. Conf.*, 2001.

[19] T. Tian *et al.*, "Construction of irregular LDPC codes with low error floors," in *IEEE Int. Conf. Comm.*, 2003.

[20] G. Durisi *et al.*, *Transmitting short packets over wireless channels—an information-theoretic perspective*, Nov. 2020. [Online]. Available: https://gdurisi.github.io/fbl-notes/