

Fast Column Message-Passing Decoding of Low-Density Parity-Check Codes

Saleh Usman¹, Member, IEEE, and Mohammad M. Mansour¹, Senior Member, IEEE

Abstract—A new fast column message-passing (FCMP) schedule for decoding LDPC codes is presented and investigated in this brief. FCMP converges in half the number of iterations compared to existing serial decoding schedules, has a significantly lower computational complexity than residual-belief-propagation (RBP)-based schedules, and consumes less power compared to state-of-the-art schedules. An FCMP decoder architecture supporting IEEE 802.11ad (WiGig) LDPC codes is presented. The architecture is synthesized using the TSMC 40 nm CMOS technology node and operates at a clock frequency of 200 MHz. The decoder achieves a throughput of 8.4 Gbps while consuming 72 mW of power. This results in an energy efficiency of 8.6 pJ/bit, which is the best-reported energy-efficiency in the literature for a WiGig LDPC decoder.

Index Terms—Layered LDPC decoding, fast column message-passing, low-power LDPC decoding, IEEE 802.11ad.

I. INTRODUCTION

HIGH-THROUGHPUT LDPC decoding is required for contemporary wireless communication standards, like the 5G New Radio [1], [2], IEEE 802.11n/ac/ax (WiFi) [3], and IEEE 802.11ad (WiGig) [4], among others. Power consumption of VLSI implementation of an LDPC decoder increases with throughput, especially if delivered at high clock speeds. It is well known that hardware architectures operating at lower clock speeds consume less power, and therefore, are highly desirable for embedded SoC applications. Such SoCs typically operate in a clock speed range of 200–300 MHz, to meet the power budget and close timing across all process, voltage, and temperature (PVT) corners in modern CMOS process nodes [5]. However, the throughput of an SoC decreases at lower clock speeds. Therefore, improving the energy efficiency (power/throughput) of multi-Gbps LDPC decoders with such constraints on clock speeds becomes a challenging task.

Fast-converging LDPC decoding simultaneously enables low-power and high-throughput VLSI implementation by reducing the number of decoding iterations required for convergence. The reduced number of iterations decreases processing latency, leading to a higher decoder throughput at lower clock speed. Fast-convergence also minimizes power consumption because of the reduction in the number of computations performed. The originally proposed LDPC decoding

schedule is a flooding schedule [6]. Serial LDPC decoding schedules later emerged in the literature that converge in half the number of decoding iterations compared with the flooding schedule [7], [8]. Serial decoding is divided into two types, row message-passing (RMP) [7], [9], [10], and column message-passing (CMP) [8], [11], based on whether a row or column order of the sparse-parity-check-matrix (SPCM) is followed.

To further accelerate the convergence rate of serial decoding schedules, several approaches that determine the best processing order of variable nodes or check nodes for serial decoders have emerged in the literature. A maximum mutual-information-increase-based schedule is presented in [12], while a column-weight based fixed scheduling for irregular LDPC codes is investigated in [13]. Informed fixed scheduling (IFS) that finds an ordering that ensures the maximum number of updated messages is utilized within a single iteration is introduced in [14]. These schedules, however, attain marginal improvement in convergence speed over serial schedules.

Informed dynamic scheduling (IDS) employs residual belief propagation (RBP), and updates only those nodes that correspond to the maximum residuals of check-nodes messages [15]. Node-wise RBP (NWRBP) is also introduced in [15] to address the “greediness” of the RBP. Motivated by the IDS approach, several techniques with alternative message selecting and updating methods, have been presented in the literature. Examples include the silent-variable-node-free (SVNF) scheduling [16], the dynamic-silent-variable-node-free (D-SVNF) [17], the tabu search-based dynamic scheduling (TSDS) [18], and the two-reliability-metrics based RBP (TRM-TVRBP) scheduling [19]. These fast-decoding schedules accelerate the convergence rate of LDPC decoding, but the added cost of computing the residuals and the required search operations is far from trivial in VLSI implementations. Interlaced column-row message-passing (ICRMP) also speeds up the convergence of an LDPC decoder by a factor of two compared to RMP and CMP [20]. However, ICRMP requires average-variable-node-degree times more clock cycles to complete one decoding iteration compared with CMP, which limits the ICRMP in achieving the high-throughput and low-latency requirements of contemporary LDPC decoding.

Motivated by the above shortcomings of existing LDPC decoding schedules, this brief presents a new fast-converging LDPC decoding schedule with hardware-friendly features that are amenable for efficient VLSI implementation. The proposed schedule is dubbed fast column message-passing (FCMP), converges in half the number of iterations compared with RMP and CMP, and does not require any costly search and residual-computation operations. An FCMP decoder architecture targeting IEEE 802.11ad (WiGig) LDPC codes is then proposed and synthesized. The synthesized architecture operates at a clock frequency of 200 MHz, achieves a throughput of

Manuscript received June 11, 2020; revised October 4, 2020; accepted December 23, 2020. Date of publication January 6, 2021; date of current version June 29, 2021. This brief was recommended by Associate Editor F. J. Kurdahi. (Corresponding author: Saleh Usman.)

The authors are with the Department of Electrical and Computer Engineering, American University of Beirut, Beirut 1107 2020, Lebanon (e-mail: salehusman@iee.org; mmansour@iee.org).

Color versions of one or more figures in this article are available at <https://doi.org/10.1109/TCSII.2021.3049733>.

Digital Object Identifier 10.1109/TCSII.2021.3049733

1549-7747 © 2021 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission.

See <https://www.ieee.org/publications/rights/index.html> for more information.

8.4 Gbps, and an energy-efficiency of 8.6 pJ/bit, which is the best-reported energy-efficiency of an IEEE 802.11ad LDPC decoder.

II. BACKGROUND AND RELATED WORK

Consider an LDPC code with its associated bipartite graph $\mathbf{G}(\mathbf{V} \cup \mathbf{C}, \mathbf{E})$, where vertex sets \mathbf{V} and \mathbf{C} represent variable and check nodes, respectively, and \mathbf{E} represents an edge set, connecting the vertices. Let L_{cv} be the message sent from check node c to variable node v , and Q_{vc} the message sent from variable node v to check node c .

A. Column Message-Passing (CMP) Schedule

The CMP schedule updates the nodes as follows:

- Initialize all variable-node messages as

$$\begin{aligned} P_v^{(0)} &= \text{LLR}[v] \triangleq \frac{2r_v}{\sigma^2} \quad \forall v \in \mathbf{G}, \\ L_{cv}^{(0)} &= 0 \quad \forall c \in \mathbf{G}, \end{aligned} \quad (1)$$

where r_v is v th received soft symbol and σ is the estimated standard deviation of channel noise.

- For all variable nodes, at iteration $k \geq 1$
 - Update all check nodes $c \in \mathcal{R}[v]$ connected to v as

$$Q_{vc}^{(k)} = P_v^{(k)} - L_{cv}^{(k-1)}, \quad (2)$$

$$L_{cv}^{(k)} = \Psi^{-1} \left(\sum_{v' \in \mathcal{R}[c] \setminus v} \Psi \left(|Q_{v'c}^{(k)}| \right) \right), \quad (3)$$

where $\Psi(x) \triangleq 0.5 \log(\tanh(x/2)) = \Psi^{-1}(x)$, and $\mathcal{R}[c]$ denotes the set of d_c variable-nodes connected to check-node c .

- Update variable node v as follows

$$P_v^{(k)} = \text{LLR}[v] + L_{cv}^{(k)}, \quad \forall c \in \mathcal{R}[v]. \quad (4)$$

Here $\mathcal{R}[v]$ denotes the set of d_v check-nodes connected to variable-node v .

- Hard decisions are made in the final iteration based on the signs of (4).

B. Check-Node Computations

Several low-complexity approximations of (3) have emerged in the literature [21], the most common of which is the offset min-sum approximation [22], [23]:

$$L_{cv}^{(k)} = \Delta_{cv} \cdot \max \left\{ \left(\min_{v' \in \mathcal{R}[c] \setminus v} |Q_{v'c}^{(k-1)}| \right) - \beta, 0 \right\}, \quad (5)$$

$$\Delta_{cv} \triangleq \prod_{v' \in \mathcal{R}[c] \setminus v} \text{sign}(Q_{v'c}^{(k-1)}), \quad (6)$$

where β is a correction factor determined empirically. If \min_1 , \min_2 are the magnitudes of the first and second minima among the inputs of a check node, and v_{m1} is the index of the first minimum, then (5) can equivalently be written as:

$$L_{cv}^{(k)} = \Delta_{cv} \cdot \begin{cases} \max(\min_2 - \beta, 0) & \text{if } v = v_{m1}, \\ \max(\min_1 - \beta, 0) & \text{otherwise.} \end{cases} \quad (7)$$

Here the sign of each input, $\text{sign}(Q_{v'c})$, is a single bit; therefore, the product Δ_{cv} in (6) is accomplished by just taking the Exclusive-OR (**XOR**) of all the bits.

III. PROPOSED FCMP SCHEDULE

For hardware implementation of LDPC decoders, check-node processing is significantly simplified using the “min-sum” approximation, given in (5) and (7). In the min-sum approximation, each check node identifies the first and second minimum magnitudes among its inputs, and the check-node outputs are then generated by combining these minima with their corresponding signs, determined separately. The min-sum approximation simplifies the check-node processing for RMP, where each row of the SPCM directly corresponds to a check node. For CMP, however, complexity reduction is not significant, and several techniques to leverage the benefits of min-sum based CMP check-node processing have emerged, but the benefits come at the expense of potential degradation in decoding performance [24]–[27].

In CMP, each check node in the neighboring set of a given variable node determines two minima, and the message propagated by each check node to that variable node is determined using these minima. The same minima, however, can be used to determine messages for other variable nodes, which are in the neighboring sets of the check nodes being processed. The proposed FCMP schedule relies on messages that can be generated for other variable nodes using already determined minima, instead of applying existing complexity reduction techniques [26] in the CMP. This potential utilization of check-node outputs creates an advantage in terms of convergence speed, in contrast to the loss in decoding performance incurred by applying the complexity reduction techniques.

Motivated by the above observation, this brief proposes the FCMP schedule that propagates messages to all neighboring variable nodes of a check node, instead of just to the variable node for which the check node is processed. Referring to Fig. 1, the process starts by considering the first variable-node v_1 and updating it using the messages from check-nodes c_1 and c_3 ; processing up to this point is similar to CMP. Next comes the part which makes FCMP different than the CMP, and essentially amplifies its convergence speed. Check-nodes c_1 and c_3 also propagate the messages to their neighboring variable nodes, other than v_1 , and these variable nodes also get updated. The processing order of the proposed FCMP schedule is depicted in Fig. 1 for the first and second variable nodes. In Fig. 1, the nodes and edges taking part in the update process are highlighted as red. Variables v_{ij} represents the edge connecting the i th check node to its j th neighboring variable node. For example, v_{13} is the connection of the first check node to its third neighboring variable node.

The process of propagating newly generated messages at the check nodes to all connected variable nodes completes a sub-cycle (or sub-iteration) of information exchange that started from the first variable node and concluded at some set of variable nodes, depending on the structure of the SPCM. In summary, one sub-iteration of the proposed FCMP schedule, for a particular variable node, processes all check nodes connected to that variable node and propagates the check-node messages to their neighboring-sets of variable nodes in a parallel fashion. The number of sub-iterations equals the number of variable nodes constitutes one iteration of FCMP. One iteration of the FCMP requires $d_c - 1$ times more **XOR** operations and $d_c - 1$ times more addition operations, compared to the CMP, with d_c being the check-node degree. The pseudo-code of the proposed FCMP schedule is summarized in **Algorithm 1**. In the pseudo-code, d_{v_j} represents the degree of j th variable node, and c_i denotes the i th neighboring check node to the

Algorithm 1 Proposed FCMP Schedule

```

1: for all  $v, c \in \mathbf{G}$  do ▷ Initialization loop
2:    $P_v^{(0)} \leftarrow LLR[v]$ 
3:    $L_{c_v}^{(0)} \leftarrow 0$ 
4: end for
5: for  $k = 1$  to  $K$  do ▷ Iteration loop
6:   for  $j = 1$  to  $N_v$  do ▷ Variable-node (VN) loop
7:     parfor  $i = 1$  to  $d_{v_j}$  do Parallel loop
8:        $L_{c_i v'}^{(k)} \leftarrow \text{Eq. (3)}$  for all  $v' \in \mathcal{R}[c_i]$ 
9:       for all  $v' \in \mathcal{R}[c_i]$  do ▷ VN updates
10:         $P_{v'}^{(k)} \leftarrow \text{Eq. (4)}$ 
11:      end for
12:    end parfor
13:  end for
14: end for
15: for all  $v \in \mathbf{G}$  do ▷ Decision making
16:    $\Lambda[v] \leftarrow P_v \geq 0$ 
17: end for
    
```

variable-node v_j . The parallel loop of FCMP, demarcated by a bounding box in the pseudo-code, reduces the critical path delay by processing the check nodes connected to a particular variable node in parallel.

IV. COMPLEXITY ANALYSIS

In CMP, the processing of one variable node involves d_v **XOR** operations and d_v addition operations at the variable node, along with the operations to determine the two minima and apply offset correction. The process is repeated for all the N_v variable nodes. If V_c and C_c are the number of variable-node to check-node (V2C) operations and check-node to variable-node (C2V) computations, respectively, then for a single iteration, V_c and C_c are given as:

$$V_c = N_v \times d_v = E, \quad (8)$$

$$C_c = N_v \times d_v \times (X + 2 \times M) = E. \quad (9)$$

In (9), X represents the **XOR** operation count, while M corresponds to the operation count of computing a single minimum with the offset correction. Since $X + 2 \times M$ is considered a single C2V computation, therefore, $C_c = E$.

For the FCMP schedule, messages computed at check nodes, connected to a particular variable node, are further propagated to their neighboring variable nodes, as illustrated in Fig. 1. If V_f and C_f are the number of V2C and C2V computations of the FCMP, respectively, then for a single iteration, V_f and C_f are given by the following equations:

$$V_f = N_v \times d_v \times d_c = d_c \times E, \quad (10)$$

$$C_f = N_v \times d_v \times (d_c \times X + 2 \times M). \quad (11)$$

By comparing (8) with (10), and (9) with (11), it is evident that the FCMP requires d_c times additions and **XOR** operations at the variable-nodes and check-nodes sides, respectively, compared to the CMP schedule. Addition and **XOR** are simple operations, compared to the minima determination operations; the number of these operations is the same for the FCMP and CMP schedules. The computational complexities of various LDPC decoding schedules are summarized in Table I. In Table I, $r(v)$ denotes the operations required to calculate the variable-node residuals. The comparison of the complexities reveals that the proposed FCMP schedule,

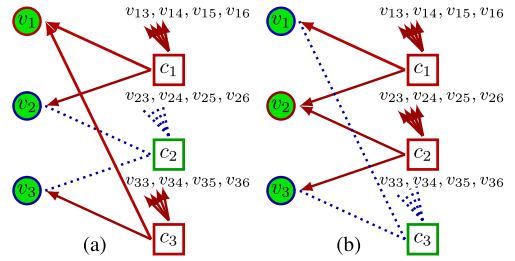


Fig. 1. Bipartite graph of FCMP schedule for variable-nodes v_1 and v_2 : (a) neighboring check-nodes c_1, c_3 of v_1 update all neighboring variable nodes; (b) neighboring check-nodes c_1, c_2 of v_2 update all neighboring variable nodes.

TABLE I
COMPUTATIONAL COMPLEXITIES OF LDPC DECODING SCHEDULES

| Schedules | V2C Computations | C2V Computations | Search Operations | $r(v)$ |
|-------------|------------------|------------------------------|-------------------|-------------|
| RMP | E | E | 0 | 0 |
| CMP | E | E | 0 | 0 |
| NWRBP* | $E(d_v - 1)$ | $E[(d_c - 1) + 1]$ | $M_c(E - 1)$ | 0 |
| SVNF* | $E(d_v - 1)$ | $E[(d_c - 1) + 1]$ | $E(d_v - 1)d_c$ | 0 |
| TSDS* | $E(d_v - 1)$ | $E(d_c - 1)$ | $-d_v - 1$ * | 0 |
| TRM-TVRBP** | $E/2$ | $5(E/4)$ | $N_v(N_v - 1)$ | $5(E/4)$ |
| RM-RBP** | E | $E \cdot d_c$ | $+E(d_c - 1)$ | $(d_c - 1)$ |
| ICRMP | $E \cdot d_c$ | $E + \text{XOR Ops}^\dagger$ | 0 | 0 |
| Prop. FCMP | $E \cdot d_c$ | $E + \text{XOR Ops}^\dagger$ | 0 | 0 |

* The complexity expressions have been taken from Table I of [18].

** The complexity expressions have been taken from Table 2 of [19].

* T_L is the length of the tabu list of the TSDS schedule.

† Bitwise XOR are simple operations, compared to other tabulated operations.

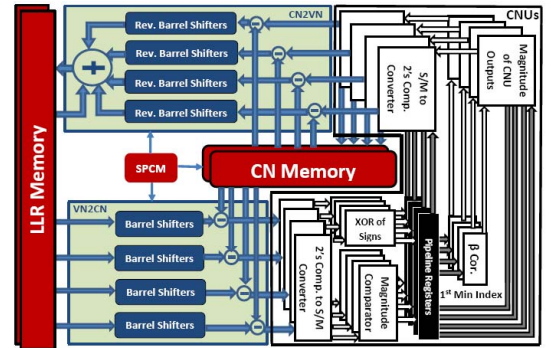


Fig. 2. A block diagram of an FCMP decoder architecture, which processes two frames simultaneously, for IEEE 802.11ad LDPC codes.

along with the CMP, RMP, and ICRMP schedules, does not introduce any costly search or $r(v)$ operations. The rest of the fast-converging schedules, NWRBP, SVNF, TSDS, TRM-TVRBP, and RM-RBP, introduce costly search and $r(v)$ operations. The computational complexities of the FCMP and ICRMP schedules are the same; however, the proposed FCMP schedule processes its neighboring sets of check-nodes in parallel, which enables the FCMP to achieve significantly higher throughput than ICRMP.

V. PROPOSED FCMP DECODER ARCHITECTURE

An LDPC decoder architecture based on the proposed FCMP schedule is presented in this section. The decoder supports IEEE 802.11ad LDPC codes [4]. The block diagram of the proposed FCMP decoder architecture is shown in Fig. 2. The architecture replicates the processing units, in comparison with legacy architectures, [7], [23], [28], [29], by a factor of

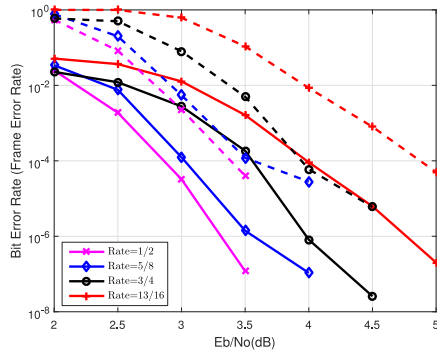


Fig. 3. Fixed-point decoding performance of proposed FCMP architecture for IEEE 802.11ad (WiGig), with $Q_L = 7$, $Q_C = 5$, $I_{\max} = 2$. Solid and dotted lines represent BERs and FERs, respectively.

4, which is the maximum column-degree of the base matrix of IEEE 802.11ad LDPC codes, as defined in [4]. The processing units are replicated by 4 to process the neighboring set of check nodes of a particular variable node in a single clock cycle, as required in FCMP. A Check Node Unit (CNU) implements (7) and 4 CNUs are shown in black-and-white in Fig. 2. Each CNU is further composed of 42 subunits that operate in parallel to fully exploit the Quasi-Cyclic (QC) structure of WiGig LDPC codes. Thus 168 CNUs in total are used in the decoder. For WiGig LDPC codes, the maximum degree of a check node is 16, and therefore, each Barrel Shifter and Rev. Barrel Shifter, shown in Fig. 2, is instantiated 16 times in the decoder implementation. The maximum check-node degree also defines 16 inputs of each CNU. Adders and subtractors, shown in VN2CN and CN2VN modules of Fig. 2, are replicated by 42×16 as per the QC expansion factor and maximum check-node degree of WiGig LDPC codes.

The architecture is fully pipelined with a pipeline depth of 2 and processes two frames simultaneously without any idle cycles. The decoder is pipelined in a way that balances the two pipeline stages of the proposed architecture. Registers-based memory modules are used to support the decoder's high-bandwidth requirement, and the number of memory modules is doubled to store the messages of two frames. Registers-based LLR Memory also implements one stage of pipelining. The design of memory modules supports the reading and writing of messages to and from multiple blocks of processing units. In contrast to RBP-based schedules, the proposed FCMP schedule could be implemented with hardware elements of legacy LDPC decoders, as the adders, subtractors, and barrel shifters, shown in Fig. 2, are standard processing elements.

A. Decoding Performance

The fixed-point decoding performance of the proposed FCMP-architecture, targeted for IEEE 802.11ad (WiGig), is plotted in Fig. 3. The design parameters; the number of quantization bits for variable-nodes messages $Q_L = 7$, number of quantization bits for check-nodes messages $Q_C = 5$, $\beta = 1$, and $I_{\max} = 2$ are determined during the design exploration phase. By considering Fig. 3, it is evident that the proposed architecture converges to a BER value of 10^{-6} at $E_b/N_0 = 3.3$, 3.6, 4.0, and 4.8 dB, for code rates 1/2, 5/8, 3/4, and 13/16, respectively. The architecture presented in [30], for example, achieves a BER value of 10^{-6} at $E_b/N_0 = 4.6$, 4.6, 5.0, and 5.4 dB, for code rates 1/2, 5/8, 3/4, and 13/16, respectively, while operating at 202 MHz and running for $I_{\max} = 10$ decoding iterations. Thus, the proposed architecture based on the

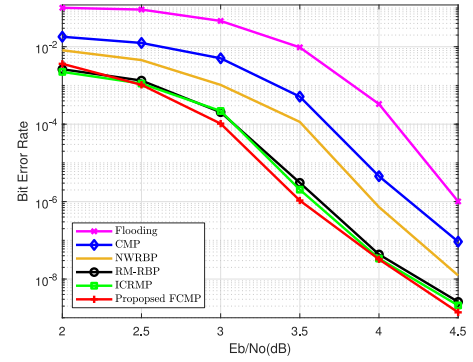


Fig. 4. Floating-point BER comparison of proposed FCMP schedule with other schedules for 802.11ad (WiGig) LDPC codes, rate=5/8, $I_{\max} = 100$.

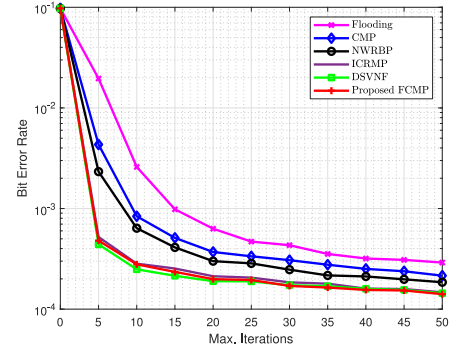


Fig. 5. Comparison of BER vs. number of iterations for WiGig LDPC codes, rate 1/2, at $E_b/N_0 = 2.25$ dB.

FCMP schedule achieves a better decoding performance and higher energy efficiency than [30].

VI. SIMULATION AND VLSI SYNTHESIS RESULTS

BER performance for the RMP, ICRMP, RBP-based, and proposed FCMP decoding schedules is assessed by considering IEEE 802.11ad (WiGig) LDPC codes. The BER plots against E_b/N_0 and the maximum number of iterations are shown in Fig. 4 and Fig. 5, respectively. The plots reveal that the proposed FCMP schedule converges faster than RMP and CMP schedules and achieves the same decoding and convergence performance as the RBP-based schedule, although the computational complexity of the proposed FCMP schedule is significantly lower than the RBP-based schedules.

The proposed hardware architecture of the FCMP schedule for IEEE 802.11 ad LDPC codes, shown in Fig. 2, is synthesized using the TSMC 40 nm CMOS process for a target clock speed of 200 MHz. The throughput θ is calculated as:

$$\theta = \frac{F_L \times N_F}{I_{\max} \times C} \times f_{\text{clk}}. \quad (12)$$

Here, F_L is the frame-length, N_F is the number of frames, I_{\max} is the maximum number of iterations, C is the number of cycles to complete one decoding iteration, and f_{clk} is the clock frequency. For the proposed architecture, $F_L = 672$, $N_F = 2$, $I_{\max} = 2$, $C = 16$, and $f_{\text{clk}} = 200$ MHz, which result in a throughput of 8.4 Gbps. The proposed architecture exploits the quasi-cyclic structure of WiGig LDPC codes and completes the FCMP processing for one column of the base matrix in one cycle. There are 16 columns of the base matrix, and therefore, $C = 16$ has been used for the throughput calculation.

TABLE II
COMPARISON WITH STATE-OF-THE-ART

| | This work | [31] | [32] | [30] | [33] |
|-----------------------------|-----------|------|-------|------|------|
| Implementation | Synthesis | ASIC | ASIC | ASIC | ASIC |
| Tech. (nm) | 40 | 65 | 40 | 28 | 28 |
| Clk. Freq. (MHz) | 200 | 360 | 220 | 202 | 260 |
| Frame Length | 672 | 672 | 672 | 672 | 672 |
| Messages Bitwidth | 5, 7 | 5 | 5 | 5 | 5 |
| Iterations | 2 | 10 | 7 | 10 | 3.75 |
| Code Rate for Power Measur. | 1/2 | 1/2 | 13/16 | 1/2 | 1/2 |
| Through. (Gbps) | 8.4 | 6 | 6.2 | 6.78 | 12 |
| Area (mm ²) | 1.7* | 1.6 | 0.8 | 1.99 | 0.63 |
| Power (mW) | 72* | 374 | 203 | 279 | 180 |
| En. Eff. (pJ/bit) | 8.6 | 23** | 32.7 | 41† | 15† |

* The area and power of the synthesized netlist are 1.44 mm² and 60 mW, respectively, and increased by 20% for comparison with ASIC implementations of state-of-the-art decoders.

** Energy-efficiency is calculated after scaling throughput and power by 1.6 and 0.6, respectively, to account for the 65 nm technology.

† Results are not scaled as Dennard's rules are not valid below 65 nm.

A comparison of the VLSI synthesis results of the proposed architecture with state-of-the-art IEEE 802.11ad LDPC decoder implementations is summarized in Table II. The comparison reveals that the proposed architecture consumes the least power while achieving a throughput that is comparable with other architectures. Owing to this minimal power consumption while maintaining a comparable throughput, the proposed FCMP architecture attains the best-reported energy efficiency of 8.6 pJ/bit for an IEEE 802.11ad LDPC decoder. Specifically, the proposed architecture consumes significantly less power because of its ability to converge in fewer iterations to achieve a specific FER/BER at a given E_b/N_0 . The switching activity of a decoder, and consequently, the power consumption is directly related to the number of decoding iterations required for convergence.

VII. CONCLUSION

A low-latency and energy-efficient multi-Gbps fast column message-passing LDPC decoding schedule that achieves fast convergence with reduced computational complexity, or consumes lower power to achieve a specific throughput, compared with other schedules in the literature, has been presented and investigated. An FCMP-based decoder architecture targeting IEEE 802.11ad (WiGig) LDPC codes has also been proposed. VLSI synthesis results using the TSMC 40 nm CMOS have revealed that the synthesized decoder achieves an energy efficiency of 8.6 pJ/bit while operating at 8.4 Gbps, which is the best-reported energy-efficiency of a WiGig LDPC decoder.

REFERENCES

- [1] T. Richardson and S. Kudekar, "Design of low-density parity check codes for 5G new radio," *IEEE Commun. Mag.*, vol. 56, no. 3, pp. 28–34, Mar. 2018.
- [2] *Multiplexing and Channel Coding, (V 15.3.0) Release 15*, 3GPP Standard TS 38.212, Oct. 2018.
- [3] *IEEE Standard for Local and Metropolitan Area Networks—Part 11: Wireless LAN (MAC) and Physical Layer (PHY)*, IEEE Standard 802.11, 2012.
- [4] *IEEE Draft Standard for Information Technology-Wireless LANs—Part 21: mmWave PHY Specification*, IEEE Standard 802.11ad, May 2010.
- [5] K. Okada *et al.*, "Full four-channel 6.3-Gb/s 60-GHz CMOS transceiver with low-power analog and digital baseband circuitry," *IEEE J. Solid-State Circuits*, vol. 48, no. 1, pp. 46–65, Jan. 2013.
- [6] R. G. Gallager, *Low-Density Parity-Check Codes*. Cambridge, MA, USA: MIT Press, 1963.
- [7] M. M. Mansour and N. R. Shanbhag, "High-throughput LDPC decoders," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 11, no. 6, pp. 976–996, Dec. 2003.
- [8] J. Zhang and M. Fossorier, "Shuffled belief propagation decoding," in *Proc. 36th Asilomar Conf. Signals Syst., Comput. (Asilomar)*, vol. 1, Pacific Grove, CA, USA, Nov. 2002, pp. 8–15.

- [9] M. M. Mansour and N. R. Shanbhag, "Memory-efficient turbo decoder architectures for LDPC codes," in *Proc. IEEE Int. Workshop Signal Process. Syst. (SiPS)*, San Diego, CA, USA, 2002, pp. 159–164.
- [10] M. M. Mansour, "A turbo-decoding message-passing algorithm for sparse parity-check matrix codes," *IEEE Trans. Signal Process.*, vol. 54, no. 11, pp. 4376–4392, Nov. 2006.
- [11] H. Kfir and I. Kanter, "Parallel versus sequential updating for belief propagation decoding," *Physica A, Stat. Mech. Appl.*, vol. 330, pp. 259–270, Dec. 2003.
- [12] H.-C. Lee and Y.-L. Ueng, "LDPC decoding scheduling for faster convergence and lower error floor," *IEEE Trans. Commun.*, vol. 62, no. 9, pp. 3104–3113, Sep. 2014.
- [13] C. A. Aslam, Y. L. Guan, and K. Cai, "Improving the belief-propagation convergence of irregular LDPC codes using column-weight based scheduling," *IEEE Commun. Lett.*, vol. 19, no. 8, pp. 1283–1286, Aug. 2015.
- [14] C. A. Aslam, Y. L. Guan, K. Cai, and G. Han, "Informed fixed scheduling for faster convergence of shuffled belief-propagation decoding," *IEEE Commun. Lett.*, vol. 21, no. 1, pp. 32–35, Jan. 2017.
- [15] A. I. V. Casado, M. Griot, and R. D. Wesel, "Informed dynamic scheduling for belief-propagation decoding of LDPC codes," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Glasgow, U.K., Jun. 2007, pp. 932–937.
- [16] H.-C. Lee, Y.-L. Ueng, S.-M. Yeh, and W.-Y. Weng, "Two informed dynamic scheduling strategies for iterative LDPC decoders," *IEEE Trans. Commun.*, vol. 61, no. 3, pp. 886–896, Mar. 2013.
- [17] C. A. Aslam, Y. L. Guan, K. Cai, and G. Han, "Low-complexity belief-propagation decoding via dynamic silent-variable-node-free scheduling," *IEEE Commun. Lett.*, vol. 21, no. 1, pp. 28–31, Jan. 2017.
- [18] X. Liu, C. Fan, and X. Chen, "Dynamic scheduling decoding of LDPC codes based on tabu search," *IEEE Trans. Commun.*, vol. 65, no. 11, pp. 4612–4621, Nov. 2017.
- [19] X. Liu, L. Zi, D. Yang, and Z. Wang, "Improved decoding algorithms of LDPC codes based on reliability metrics of variable nodes," *IEEE Access*, vol. 7, pp. 35769–35778, 2019.
- [20] S. Usman, M. M. Mansour, and A. Chehab, "Interlaced column-row message-passing schedule for decoding LDPC codes," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Washington, DC, USA, Dec. 2016, pp. 1–6.
- [21] J. Chen, A. Dholakia, E. Eleftheriou, M. P. C. Fossorier, and X.-Y. Hu, "Reduced-complexity decoding of LDPC codes," *IEEE Trans. Commun.*, vol. 53, no. 8, pp. 1288–1299, Aug. 2005.
- [22] M. Weiner, B. Nikolić, and Z. Zhang, "LDPC decoder architecture for high-data rate personal-area networks," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, Rio de Janeiro, Brazil, Jul. 2011, pp. 1784–1787.
- [23] S. Usman, M. M. Mansour, and A. Chehab, "A multi-Gbps fully pipelined layered decoder for IEEE 802.11n/ac/ax LDPC codes," in *Proc. IEEE Comput. Soc. Annu. Symp. VLSI (ISVLSI)*, Bochum, Germany, Jul. 2017, pp. 194–199.
- [24] Z. Cui, Z. Wang, X. Zhang, and Q. Jia, "Efficient decoder design for high-throughput LDPC decoding," in *Proc. IEEE Asia-Pac. Conf. Circuits Syst.*, Macao, China, Nov. 2008, pp. 1640–1643.
- [25] L. Jun, S. Jin, Z. Wang, and L. Li, "An improved min-sum based column-layered decoding algorithm for LDPC codes," in *Proc. IEEE Int. Workshop Signal Process. Syst. (SiPS)*, Tampere, Finland, 2009, pp. 238–242.
- [26] Z. Cui, Z. Wang, and X. Zhang, "Reduced-complexity column-layered decoding and implementation for LDPC codes," *IET Commun.*, vol. 5, no. 15, pp. 2177–2186, Oct. 2011.
- [27] X. Zhang *et al.*, "Column-layered message-passing LDPC decoder," U.S. Patent US20180062666 A1, Mar. 2018.
- [28] S. Kumawat, R. Shrestha, N. Daga, and R. Paily, "High-throughput LDPC-decoder architecture using efficient comparison techniques & dynamic multi-frame processing schedule," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 62, no. 5, pp. 1421–1430, May 2015.
- [29] S. Usman and M. M. Mansour, "An optimized VLSI implementation of an IEEE 802.11n/ac/ax LDPC decoder," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, Sevilla, Spain, Oct. 2020, pp. 1–5.
- [30] M. Milicevic and P. G. Gulak, "A multi-Gb/s frame-interleaved LDPC decoder with path-unrolled message passing in 28-nm CMOS," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 26, no. 10, pp. 1908–1921, Oct. 2018.
- [31] Y. S. Park, D. Blaauw, D. Sylvester, and Z. Zhang, "Low-power high-throughput LDPC decoder using non-refresh embedded DRAM," *IEEE J. Solid-State Circuits*, vol. 49, no. 3, pp. 783–794, Mar. 2014.
- [32] H. Motozuka, N. Yosoku, T. Sakamoto, T. Tsukizawa, N. Shirakata, and K. Takinami, "A 6.16Gb/s 4.7pJ/bit/iteration LDPC decoder for IEEE 802.11ad standard in 40nm LP-CMOS," in *Proc. IEEE Global Conf. Signal Inf. Process. (GlobalSIP)*, Orlando, FL, USA, Dec. 2015, pp. 1289–1292.
- [33] M. Weiner, M. Blagojevic, S. Skotnikov, A. Burg, P. Flatresse, and B. Nikolic, "27.7 A scalable 1.5-to-6Gb/s 6.2-to-38.1mW LDPC decoder for 60GHz wireless networks in 28nm UTBB FDSOI," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, San Francisco, CA, USA, Feb. 2014, pp. 464–465.