A Unified Forward Error Correction Accelerator for Multi-Mode Turbo, LDPC, and Polar Decoding

Yufan Yue, Tutu Ajayi, Xueyang Liu, Peiwen Xing, Zihan Wang, David Blaauw, Ron Dreslinski, Hun-Seok Kim
University of Michigan
{funkyyue,ajayi,marliu,xingpw,zihanw,blaauw,rdreslin,hunseok}@umich.edu

ABSTRACT

Forward error correction (FEC) is a critical component in communication systems as the errors induced by noisy channels can be corrected using the redundancy in the coded message. This paper introduces a novel multi-mode FEC decoder accelerator that can decode Turbo, LDPC, and Polar codes using a unified architecture. The proposed design explores the similarities in these codes to enable energy efficient decoding with minimal overhead in the total area of the unified architecture. Moreover, the proposed design is highly reconfigurable to support various existing and future FEC standards including 3GPP LTE/5G, and IEEE 802.11n WiFi. Implemented in GF 12nm FinFET technology, the design occupies 8.47mm^2 of chip area attaining 25% logic and 49% memory area savings compared to a collection of single-mode designs. Running at 250MHz and 0.8V, the decoder achieves per-iteration throughput and energy efficiency of 690Mb/s and 44pJ/b for Turbo; 740Mb/s and 27.4pJ/b for LDPC; and 950Mb/s and 45.8pJ/b for Polar.

CCS CONCEPTS

• **Hardware → Application specific integrated circuits**.

KEYWORDS

Turbo code, LDPC, Polar code, FEC decoder

ACM Reference Format:
Yufan Yue, Tutu Ajayi, Xueyang Liu, Peiwen Xing, Zihan Wang, David Blaauw, Ron Dreslinski, Hun-Seok Kim. 2022. A Unified Forward Error Correction Accelerator for Multi-Mode Turbo, LDPC, and Polar Decoding. In *ACM/IEEE International Symposium on Low Power Electronics and Design (ISLPED '22), August 1–3, 2022, Boston, MA, USA.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3531437.3539726

1 INTRODUCTION

Wireless communication copes with errors induced by noisy channels. In order to recover the error-free data, forward error correction (FEC) is adopted to insert redundancy into the original data so that the receiver can detect and correct errors using the redundant information. An FEC encoder takes an information block of length K and encodes it into a code block of length N. The code rate is defined by K/N.

Turbo, LDPC and Polar codes are three popular FEC classes actively used in recent years. Turbo codes are capacity-approaching codes that have been used in 3G UMTS [1] and 4G LTE[2]. LDPC codes have found wide applications in WiFi[5] and 3GPP 5G NR[6] standards. Polar code has been recently invented and adopted in the 3GPP 5G NR standard. A recent Cloud-RAN concept that incorporates standards such as 5G NR and UMTS and LTE requires multi-standard decoding capability while several multi-mode FEC decoders have been proposed in recent years. Niktash et al. proposed an application-specific processor based solution [18] for Viterbi, Turbo, LDPC and Reed-Solomon decoding. It achieves 3883pJ/b energy efficiency for Turbo and 435 pJ/b for LDPC excluding the memory power. The architecture, however, is not easily extendable to Polar decoding. Gentile et al. designed a reconfigurable ASIC decoder [19] for Turbo and LDPC, claiming energy efficiency of 931pJ/b for Turbo and 63.7pJ/b for LDPC. This design implements a windowed Turbo decoding to reduce the memory size and power overhead. Although this design supports Turbo and LDPC, the authors do not report reconfigurability to control various parameters in FEC. Naessens et al. introduced an ASIP Turbo and LDPC decoder [20] with 4070pJ/b for Turbo and 2410pJ/b for LDPC. None of the above designs supports Polar decoding, and some only implement very specific configurations defined in existing standards instead of allowing arbitrary parameters. In addition, prior multi-mode designs have a relatively large energy efficiency gap compared to dedicated single-mode ASIC designs.

To tackle the problems stated above, we make a critical observation that soft decision decoding algorithms for Turbo, LDPC, and Polar codes share substantial similarities including iterative decoding, usage of log-likelihood ratios (LLRs), deterministic schedules, and dataflow graphs that involve parallel computation and memory units, and metric updates using similar unit operations. In addition, code lengths of existing standards are not dramatically different, which strongly implies the potential benefits from memory sharing.

Based on these critical observations, we propose a novel area-efficient unified reconfigurable decoder that supports Turbo, LDPC, and Polar codes with user-configurable arbitrary parameters at the cost of moderately increased energy consumption compared to dedicated single-mode ASIC decoders.

2 FEC DECODING ALGORITHMS

To design a unified decoder, we studied various decoding algorithms for each code type to select a proper set of algorithms that allows hardware resource sharing without compromising the decoding performance. We briefly summarize the selected algorithms in this

ISLPED '22, August 1–3, 2022, Boston, MA, USA

ACM ISBN 978-1-4503-9354-6/22/08…$15.00
https://doi.org/10.1145/3531437.3539726

section.

2.1 Turbo Decoding Algorithm

A rate 1/3 Turbo encoder consists of two parallel convolutional encoders. It takes in the information bit sequence x^s of K bits and generates $N = 3K$ coded bits: $\{x^s, x^{p1}, x^{p2}\}$ where x^{p1} and x^{p2} are produced by interleaving outputs from two convolutional encoders.

Multiple decoding algorithms have been developed as summarized in [3]. We select the MLMAP algorithm due to its friendliness to hardware and good decoding performance compared to Soft-Output Viterbi Algorithm (SOVA) [26]. In MLMAP, backward propagation is first executed to calculate all backward state metrics $B_k(s)$ for all bit positions k and all states s, then the decoder runs forward propagation to calculate forward state metrics $A_k(s)$ along with extrinsic information L_k. During both backward and forward propagation, the branch metric $\Gamma_k(s, s')$ is calculated for the state transition between s and s'. The algorithm involves several iterations of the following calculations:

$$A_k(s) = \max_{s'=s_0,s_1} (A_{k-1}(s') + \Gamma_k(s', s)) \tag{1}$$

$$B_k(s) = \max_{s'=s_0,s_1} (B_{k+1}(s') + \Gamma_{k+1}(s, s')) \tag{2}$$

$$\Gamma_k(s, s') = \frac{1}{2}u_k L_k + \frac{1}{2}(x_k^s y_k^s + x_k^p y_k^p) \tag{3}$$

$$L_k = \max_{u_k=+1} (A_{k-1}(s') + \Gamma_k(s', s) + B_k(s)) - \max_{u_k=-1} (A_{k-1}(s') + \Gamma_k(s', s) + B_k(s)) \tag{4}$$

where u_k is the bit encoded by the edge (s, s'), and y_k^s (y_k^p) is the log likelihood ratio (LLR) of the received bit corresponding to x_k^s (x_k^p). The sign of converged L_k produces the decoded bit. We adopt the modulo-max [4] technique replacing max operations as it reduces the dynamic range of metric calculations for a limited bit width (8-bit in our design).

2.2 LDPC Decoding Algorithm

Quasi-Cyclic (QC) LDPC is a popular code construction type. Many LDPC codes are QC because it allows efficient (hardware) implementation with comparable decoding performance compared to non-QC [7]. A codeword x is generated by multiplying information bit sequence vector with a $N \times K$ binary generator matrix. A parity-check matrix $\mathbf{H}$ with dimension $K \times (N - K)$ satisfies $\mathbf{H}x^T = \mathbf{0}$ for a valid x. In addition, $\mathbf{H}$ in QC-LDPC consists of sub-matrices constructed by all-zero matrices or circular-shifted identity matrices of lifting size Z. $\mathbf{H}$ is characterized by a tanner graph using variable nodes (VN) representing code bits and check nodes (CN) representing parity bits. We adopt a popular Min-Sum (MS) [8] algorithm, which exhibits similar performance with relatively low complexity compared to alternatives. The Min step calculates LLRs η_{mn} from CN m to VN n; the Sum step calculates LLRs λ_{nm} from VN n to CN m:

$$\eta_{mn} = \prod_{k,\boldsymbol{H}(m,k)=1,k\neq m} sign(\lambda_{km}) \min_k(\lambda_{km}) \tag{5}$$

$$\lambda_{nm} = \sum_{k,\boldsymbol{H}(k,n)=1,k\neq n} \eta_{kn} + \eta_{0n} \tag{6}$$

where η_{0n} is the LLRs of the received n^{th} bit. The decoding is an iterative process passing LLRs from VN to CN, vice versa, updating η and λ.

2.3 Polar Decoding Algorithm

Various encoding and decoding algorithms for Polar are summarized in [9]. We select the Belief-Propagation (BP) algorithm which adopts an iterative approach to achieve good performance. The main advantage of using BP is the feasibility of an unified architecture that shares hardware with Turbo MLMAP and LDPC MS algorithms. An example decoding trellis and its computation unit is shown in Fig. 1 which has $n = \log N$ columns. In each half iteration, a computation unit reads two left-propagating LLRs L and two right-propagating LLRs R, and updates two LLRs in its propagation direction. During t^{th} iteration, following computations are performed in each computation unit:

$$L_{i+1,j}^{(t+1)} = f(L_{i,2j}^{(t)}, L_{i,2j+1}^{(t)} + R_{i+1,j+N/2}^{(t)}) \tag{7}$$

$$L_{i+1,j+N/2}^{(t+1)} = L_{i,2j+1}^{(t)} + f(L_{i,2j}^{(t)}, R_{i+1,j}^{(t)}) \tag{8}$$

$$R_{i,2j}^{(t+1)} = f(R_{i+1,j}^{(t)}, L_{i,2j+1}^{(t)} + R_{i+1,j+N/2}^{(t)}) \tag{9}$$

$$R_{i,2j+1}^{(t+1)} = R_{i+1,j+N/2}^{(t)} + f(R_{i+1,j}^{(t)}, L_{i,2j}^{(t)}) \tag{10}$$

where $f(x, y) = sign(x)sign(y)min(|x|, |y|)$.

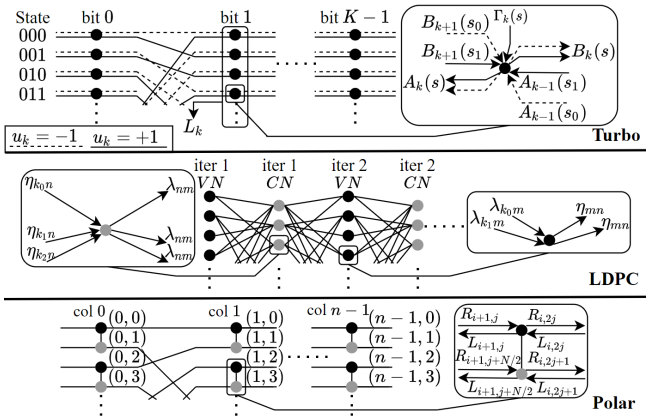

Figure 1: Algorithm visualizations for Turbo, LDPC and Polar decoding

As shown in Fig. 1, all Turbo, LDPC, and Polar decoding algorithms employ similar iterative metric update structures using a code-dependent dataflow graph (DFG) that involve parallel metric computing units and memory units. The message exchange patterns on the DFG are deterministic and regular while a finite set of interconnection configurations is sufficient to realize all aforementioned decoding algorithms. This allows a unified architecture that shares hardware resources for all codes without significantly degrading the energy efficiency.

3 ALGORITHM-ARCHITECTURE CO-OPTIMIZATIONS

Despite DFG similarities (Fig. 1), decoding algorithms require optimizations to enable an efficient unified hardware architecture for all code types. In this section, we propose algorithm-architecture co-optimizations to improve the energy- and area-efficiency of an unified multi-mode FEC decoder.

3.1 LDPC Decoding Optimizations

Consider an $\mathbf{H}$ that has A variable node sub-matrices, B check node sub-matrices, and lifting size Z. The layered algorithm in [11]

requires simultaneously accessing and processing $A \times Z$ or $B \times Z$ data points. These parameters need to be adjustable for unifying multiple decoding modes. We address this issue by dividing **H** into multiple rectangular windows, each containing $C \times D$ sub-matrices as shown in Fig. 2. These windows are processed sequentially due to the limitation of hardware resources. During Min step, the windows are processed row-first in the left-to-right order. The information exchanged between adjacent windows in the same row contains only the two minima and the first minimum index. In Sum step, the windows are processed column-first in the top-to-down order. Only the partial sums of previous windows are exchanged. Zero-padding is used when A (or B) is not an integer multiple of C (or D).

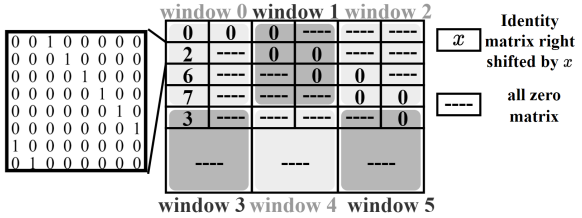

Figure 2: Example window division of an LDPC parity-check matrix H when $A = 5$, $B = 2$, $C = 4$, and $D = 2$.

The operations for a sub-matrix are sequentially executed while multiple sub-matrices in a window are processed in parallel. This approach targets flexible and high utilization of the available memory bandwidth. At time $t \equiv k \pmod{Z}$ of Min step, all data at row offset k of all sub-matrices in the current window are fed into processing elements (PE) in parallel. In Sum step, a similar schedule takes place on data with column offset k at time $t \equiv k \pmod{Z}$.

The proposed parameterized **H** partitioning and memory access patterns enable a very flexible and efficient control for LDPC decoding. Each simultaneous data access is limited to $C \times D$ words whereas C and D are determined by the target parallelism, performance, and area constraint.

3.2 Polar Decoding Optimizations

Naive decoding of a Polar code of length N in parallel requires $6N$ concurrent memory accesses and data processing. This poses excessively high resource and bandwidth requirements that are unseen in Turbo and LDPC decoding algorithms. To resolve this difference, we propose a Polar Folded (PF) schedule for a large N, and Polar Parallel (PP) schedule for a small N.

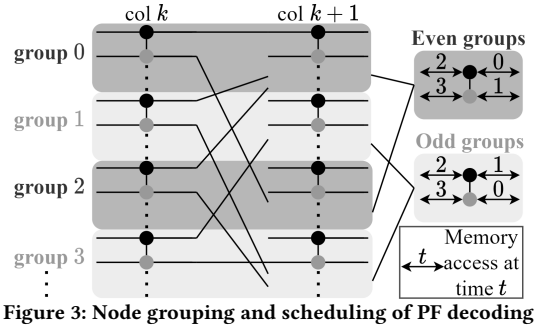

Figure 3: Node grouping and scheduling of PF decoding

Fig. 3 depicts the PF schedule which operates with G groups of nodes. Nodes in each group are processed sequentially from top to bottom while multiple groups are processing in parallel. Intermediate results are read out and written back to the metric memory. Due to the concurrent memory accesses given the DFG structure, parallel memory accesses create collisions among different groups. This problem persists even if nodes are sequentially processed in each group. To mitigate this collision, we divide all nodes into even and odd sets with two non-colliding data access patterns as shown in Fig. 3 for a left-propagating example. The pattern for the right-propagating is similar. This scheme allows full hardware utilization regardless of the code length.

For a shorter code that inherently require low utilization of computation and memory bandwidth resources, we apply the PP schedule. It consists of a single group and all intermediate results can be forwarded to the next column in parallel for a better throughput and energy efficiency.

4 PROPOSED UNIFIED DECODER ARCHITECTURE

Based on the selected algorithms and optimizations in Section II/III, we propose a unified architecture shown in Fig. 4. This architecture aims at maximizing hardware (memories, interconnect, PEs, etc.) sharing between Turbo, LDPC, and Polar. We discuss several essential architectural features for the shared memory and PE logic in subsequent sections.

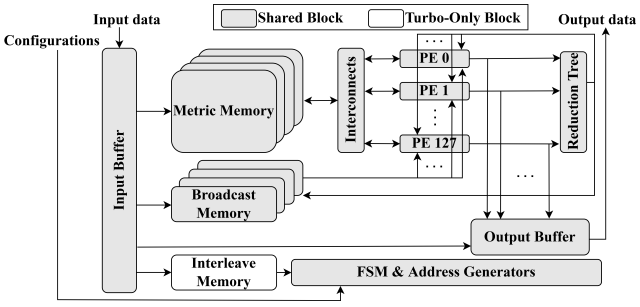

Figure 4: Top-level architecture of the unified decoder

4.1 Unified Metric Memory

Metric memory is the storage of intermediate metrics such as $A_k, B_k, \eta, \lambda, L, R$ in eq. (1) – (10). We propose a novel memory mapping and scheduling scheme for the unified metric memory system to improve utilization and avoid conflicts of memory accesses for parallel execution in all codes. LDPC and Polar codes tend to have larger bandwidth requirements while the storage depth (i.e., the number of stored entries in the memory) is relatively shallow. Hence we need to design the memory system using small banks for simultaneous/parallel data accesses. Turbo codes, on the other hand, tend to have larger storage capacity requirements whereas the access bandwidth requirements are relatively relaxed. Thus, to share the memory system for all codes, a large number of small banks need to be reconfigurable to a smaller number of larger logical memory units with more capacity and reduced bandwidth.

Details of memory mappings are presented in Fig. 5 where each memory instance is a 512×8b SRAM to store 8b-wide metrics. For Turbo codes, we map the code bit index k to memory address k, and map trellis state s to memory group s. To provide enough storage, a memory group is formed to contain 8 memory banks to hold 8×512 8b entries. Memory accesses are sequential in Turbo mode. The path metric of state s and bit location k is stored at memory group s

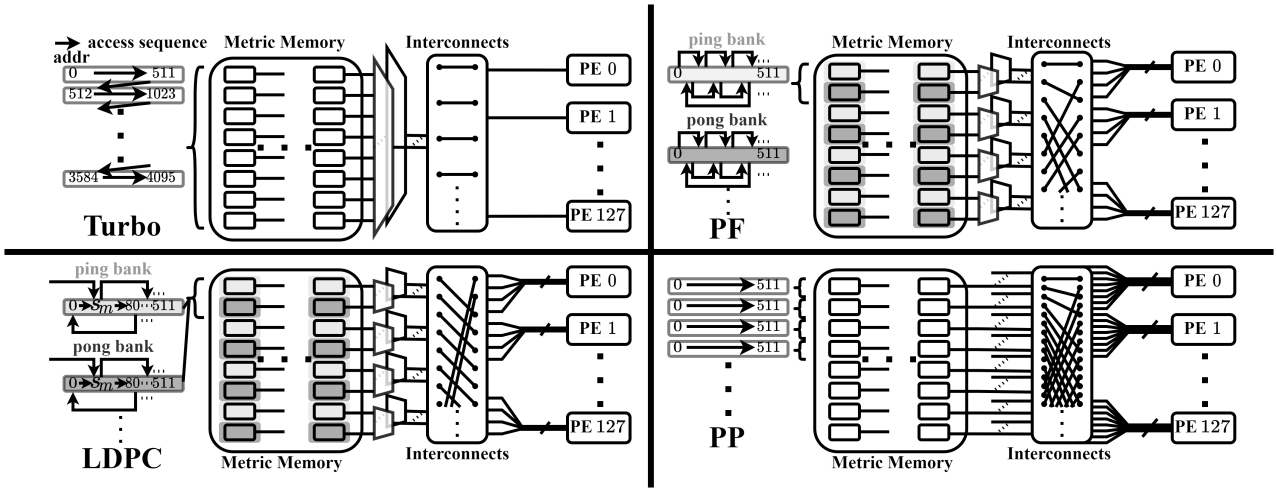

Figure 5: Metric memory grouping and mapping, and example memory-PE interconnection configurations for different codes.

and address k. It is accessed at time $t_k = k$ in forward propagation and $t_k = N - 1 - k$ in backward propagation.

An LDPC code exhibits a different mapping of metric data to the memories. Each sub-matrix inside a window is mapped to a memory group depending on its position in the window. Because of the QC nature of LDPC codes, each column in a sub-matrix m has at most one valid data. Therefore a column c in a sub-matrix of a window w_m is mapped to address $k_{m,c} = c + w_m Z_{max}$. Denoting the shift amount of an identity matrix of this sub-matrix by s_m, the access time $t_{k_{m,c}}$ of entry $k_{m,c}$ in a memory is obtained by $t_{k_{m,c}} = w_m Z + (k_{m,c} - s_m \pmod Z)$ for Min step and by $t_{k_{m,c}} = w_m Z + c$ for Sum step. Each memory group consists of 2 banks acting as ping-pong buffers for simultaneous read and write accesses. As discussed in Section III, LDPC decoding is sequential between windows but sub-matrices within a window are accessed in parallel. The proposed mapping can guarantee collision-free memory accesses. The sub-matrix shift amounts s_m are stored in a register file nearby the metric memory, serving as an offset for address generation to accommodate arbitrary **H** patterns.

Polar Folded (PF) mode maps each node group shown in Fig. 3 to a memory group that consists of a ping-pong memory pair. In a Polar decoding trellis (Fig. 1), the right- and left-propagating messages R and L of different columns are mapped using ascending address blocks, ordered by their node offset in each node group. The R (L) message at column c and bit location b is stored in memory group $M_{b,R}$ ($M_{b,L}$), where $M_{b,R} = \lfloor \frac{b}{N/G} \rfloor$ holds (and $M_{b,L} = M_{b',R}$ for the connected bits b and b' in the trellis). Each node access is sequential within a node group in the order specified in Fig. 3.

Polar Parallel (PP) mode maps each bit's L message in even columns and R message in odd columns in the first half of all memory banks. The mapping is flipped (R to even and L to odd) in the second half of all memory banks. This guarantees that an even-column node only reads even-column L message and writes odd-column R message, and vice versa, avoiding memory conflicts for the maximum bandwidth utilization when parallel PEs access multiple memory banks.

4.2 Unified Processing Element Array

The Processing Element (PE) array employs 128 copies of PEs to compute mode-dependent decoding metrics in parallel while concurrently accessing the unified metric memory. The 128 PEs and unified metric memory communicate with each other via an interconnect logic that cycles through deterministic interconnection patterns given by the dataflow graph. Each PE consists of 32 Multi-Mode Adders (MMAs) and each MMA unifies 8-bit (auto-saturate) addition, (auto-saturate) subtraction, modulo-max, equality check, and unsigned-min. The PE is capable of performing all necessary operations for Turbo, LDPC and Polar decoding with deliberate logic sharing to minimize the total area. The proposed MMA is synthesized with 21 full adders(FA) and 1 half adder(HA). Fig. 6

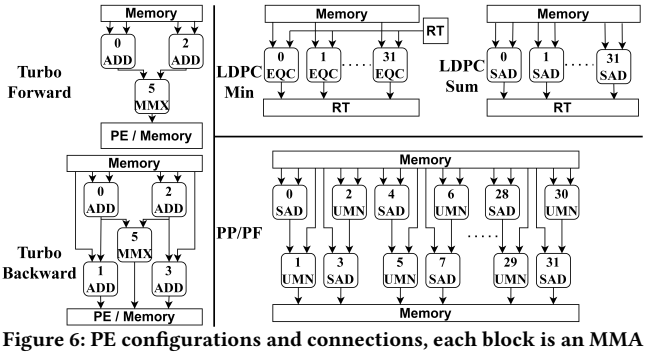

Figure 6: PE configurations and connections, each block is an MMA

shows the PE configuration and MMA connections for different decoding modes. In Turbo mode, only 3 or 5 MMAs are used. MMA0 and 2 compute addition (ADD) and MMA5 computes modulo-max (MMX) in Turbo backward propagation whereas additional MMA1 and 3 compute path metric additions in forward propagation. In LDPC Min step, all MMAs perform equality check (EQC) together with the reduction tree (RT, Section III.C) to find the min data index and write the first and second minimum to memory. In LDPC Sum step, all MMAs compute saturated subtraction (SSB) to extract intrinsic-information from the sum to be written in the metric memory. In Polar modes (PM and PP), MMAs are divided into groups of 4. MMA0 and 3 of each group calculate saturated additions (SAD) of inputs while MMA1 and 2 of each group compute unsigned-mins (UMN) of inputs. Both LDPC and Polar (PF/PP) modes fully utilize all available MMAs in the PE.

4.3 Unified Reduction Tree

The Reduction Tree (RT) in Fig. 4 and 8 is an essential component to perform bandwidth-reducing operations such as extrinsic information calculations in Turbo and min/sum evaluation across all variable nodes/check nodes in LDPC decoding. These operations have very similar data movement patterns and unit operations, thus allowing a shared RT architecture for Turbo and LDPC. Polar decoding does not use RT.

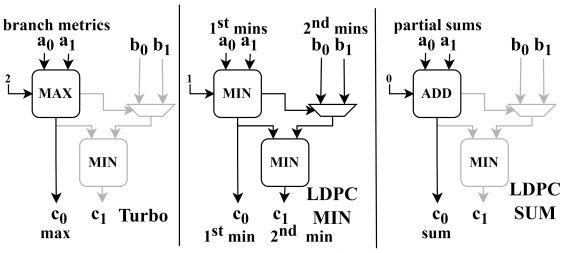

Figure 7: Reduction Unit (RU) mappings

Fig. 7 shows the basic operations of a Reduction Unit (RU) in RT. An RU computes modulo-max in Turbo, two minimum in LDPC Min step, and sum in LDPC Sum step. In Turbo mode, it takes two branch metric values a_0 and a_1 from the previous layer and calculates the modulo-max c_0. For LDPC Min step, an RU takes in two first minimums a_0 and a_1 as well as two corresponding second minimums b_0 and b_1 to calculate the first and second minimums c_0 and c_1 among these four inputs as shown in Fig. 7. In LDPC Sum step, an RU takes two partial sums a_0 and a_1 to compute the sum of them.

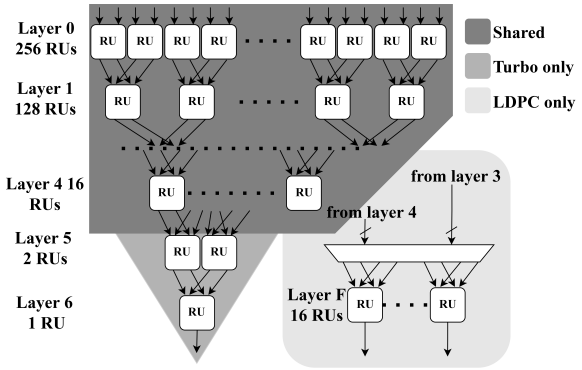

Figure 8: RT connections

As shown in Fig. 8, RT consists of 7 layers of RUs in a tree form. The first layer has 256 RUs to match the PE array throughput of LDPC decoding. As each LDPC window size is 16×8, RT requires 5 and 4 layers of RUs for Min and Sum step, respectively. A layer F is added next to the last layer only for LDPC to merge results from different windows. The connection between layers are identical in LDPC and Turbo modes, which minimizes MUX overhead and simplifies control. Furthermore, RT is reconfigurable to produce output for different (code dependent) numbers of total Turbo states by enabling or bypassing certain layers in RT.

4.4 Other components

The proposed design features an unified Broadcast Memory (Fig. 4) for Turbo extrinsic information broadcasting and LDPC channel LLR broadcasting. Input Buffer uses ping-pong memory to hide irregular and bursty input arrivals. Interleave Memory is used for interleaving (with an arbitrary programmable pattern) required in Turbo operation.

Interconnects in Fig. 4 realizes all required connection patterns (some are illustrated in Fig. 5) between Metric Memory and PE array with a code-dependent configurable schedule. Since the address generation is localized within the metric memory (to handle arbitrary **H**), the number of necessary interconnect configurations for each code type is relatively small (≤ 4). The interconnects include a point-to-point inter-PE interconnect network to provide inter-PE connections in Turbo and PP modes for data forwarding without involving the metric memory. The user-configured MUXes in the interconnects allow decoding of arbitrary Turbo generator polynomials.

For the final output, the decoded soft bits needs to be converted to hard bits then punctured (removing Polar frozen bits) and re-ordered (resolving the decoding order and information bit order mismatch). The output bits are sent to the 64-bit AXI port for interfacing with a host processor and other components in the system. We use Output scheduling Buffer (OB) logic to complete this job. The OB first reads in soft bits, generates hard decisions based on their sign, and stores them in a byte-addressable memory. Next, depending on the decoding mode, the data is read out of the memory in different patterns for re-ordering. Bits then go through puncturing units and are fed into a FIFO for AXI transactions. In Polar modes, an arbitrary puncturing pattern (supporting an arbitrary Polar code rate) can be programmed either in an SRAM or registers depending on the code length. The puncturing units can be reused in LDPC mode to trim unwanted tail bits. Turbo mode bypasses puncturing and re-ordering operations.

5 EXPERIMENTAL RESULTS

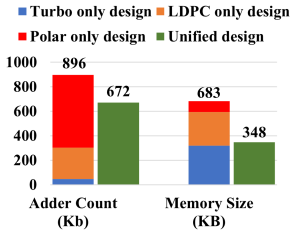

Figure 9: Area savings of memory and logic

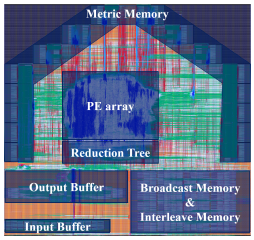

Figure 10: Layout view of proposed architecture

Fig. 9 quantifies the logic and memory area savings from the proposed unified architecture compared to a collection of dedicated single-mode decoders. The total adder count of the unified architecture is similar to that of a Polar decoder and the memory area is similar to that of a Turbo decoder. Consequently the unified design saves 25% of logic and 49% of memory compared to the sum of individual implementations of each code. The design uses 8-bit fixed point metrics. The architecture was synthesized with Synopsys Design Compiler, and the physical design (APR, layout, and DRC/LVS check) is completed with Cadence Innovus in GF 12nm technology. Fig. 10 shows the layout result, and the total chip area is $8.47mm^2$. We estimated the power consumption using Synopsys PTPX at 250 MHz and 0.8V. Full design validation was conducted by enumerating all decoding modes and code parameters. For benchmarking,

Table 1: Performance comparisons with prior designs

Designs	Turbo						LDPC					Polar		
	Ours	[21]	[22]	[18]	[19]	[20]	**Ours**	[23]	[18]	[19]	[20]	**Ours**	[24]	[25]
Algorithm	**ML MAP**	Dual Trellis	2PW logMAP	SW MAP	MAP	SW MAP	**MS**	MS	Turbo-Like	TDMP	N/A	**BP**	SC	BP
Voltage (V)	**0.8**	0.9	1.2	N/A	N/A	1.2	**0.8**	0.9	N/A	N/A	1.2	**0.8**	1.3	1.0
Frequency (GHz)	**0.250**	0.370	0.125	N/A	0.150	0.32	**0.250**	0.550	N/A	0.150	0.32	**0.250**	0.0025	0.300
Technology (nm)	**12**	90	90	90	45	65	**12**	90	90	45	65	**12**	90	65
Metric Precision (bit)	8	6	N/A	N/A	5,7	10	8	4	4	5	10	8	5	N/A
Multi-Code Modes [a]	**T,L,P**	–	–	T,L	T,L	T,L	**T,L,P**	–	T,L	T,L	T,L	**T,L,P**	–	–
Arbitrarily Configurable Parameters[b]	k, P, I, N	–	–	–	–	not reported	Z, R, **H**, N	–	–	–	not reported	F, R, N	–	–
Benchmark	3GPP LTE, $N = 512$						IEEE 802.11n, $N = 1944$					Rate=1/2, $N = 128$		
Throughput (Gb/s/iter)	**0.69**	2.55	2.50	0.05	0.12	0.14	**0.74**	4.50	0.46	0.98	0.64	**0.95**	2.56	30.7
Energy (pJ/b/iter)	**44**	122	38	3883	699	4071	**27.4**	116	435	87.6	2407	**45.8**	74.5	15.5
Energy normalized to 12nm		16	3.45	509	184	740		1.52	57.1	23.0	438		13.5	2.82

[a] Multi-code types supported by the decoder – T: Turbo, L: LDPC, P: Polar

[b] Arbitrarily-configurable parameters – k: constraint length, P: encoder polynomial, I: interleave pattern, N: code length, Z: sub-matrix size, R: code rate, **H**: parity-check matrix pattern, F: frozen bit locations

Table 2: Reconfigurability summary

Turbo		LDPC		Polar	
Param.	Range	Param.	Range	Param.	Range
N	≤3072	N	≤1944	N	≤4096
R	1/3	R	arbitrary	R	arbitrary
k	3~7	Z	≤ 81	F	arbitrary
P	arbitrary	**H**	QC, arbitrary pattern		
I	arbitrary				

we tested Turbo code for 3GPP LTE standard[2] with rate 1/3 and $N = 512$, LDPC code for IEEE 802.11n [5] with rate 1/2, $Z = 81$ and $N = 1944$, and Polar code with rate $1/2$ and $N = 128$.

We compare our design with other signle-/multi-mode designs in Table 1. To have a fair comparison, we report per-iteration energy consumption numbers scaled to 12nm using an optimistic scaling factor obtained from Spice simulations that compare Fan-Out-of-4 (FO4) inverter chain energy in different process technologies. The result shows our architecture has comparable energy efficiency while supporting much higher flexibility and reconfigurability. Compared to other multi-mode designs, ours exhibits higher throughput and significantly lower energy (except for LDPC in [19]) while ours is the only design that supports Polar modes and various arbitrary parameters for each mode. Reconfigurabile parameters supported by our design are summarized in Table 2. With abundant configurable parameters for Turbo, LDPC and Polar codes, this decoder can support a wide range of existing/future standards as well as proprietary codes tailored for target applications.

6 CONCLUSION

This paper proposes a unified FEC decoder architecture for current and future standards of Turbo, LDPC and Polar codes. Algorithm-architecture co-optimizations are performed to cope with code-dependent datapaths in a unified architecture. Post-synthesis/-layout simulation results indicate large reconfigurability advantages and comparable energy efficiencies compared to dedicated prior single-/multi-mode FEC decoders.

ACKNOWLEDGMENTS

This work was sponsored in part by the U.S. Government under the DARPA DSSoC program, award #FA8650-18-2-7860.

REFERENCES

[1] Y. Hawwar et al., "3G UMTS wireless system physical layer: baseband processing hardware implementation perspective," IEEE Comm. Mag., vol. 44, no. 9, pp. 52-58, Sept. 2006.

[2] Z. Shen, A. Papasakellariou, J. Montojo, D. Gerstenberger and F. Xu, "Overview of 3GPP LTE-advanced carrier aggregation for 4G wireless communications," IEEE Comm. Mag., Feb. 2012.

[3] J. P. Woodard and L. Hanzo, "Comparative study of turbo decoding techniques: an overview," IEEE Trans. Veh. Tech., Nov. 2000.

[4] C. B. Shung, P. H. Siegel, G. Ungerboeck and H. K. Thapar, "VLSI architectures for metric normalization in the Viterbi algorithm," IEEE ICC, 1990, pp. 1723-1728 vol.4.

[5] IEEE P802.11 Wireless LANs WWiSE Proposal: High Throughout Extension to the 802.11 Standard, IEEE 11-04-0886-00-000n, 2005.

[6] 3GPP TS 38.212 version 15.2.0 Release 15 : 5G; NR; Multiplexing and channel coding, ETSI, 2017.

[7] R. M. Tanner, D. Sridhara, A. Sridharan, T. E. Fuja and D. J. Costello, "LDPC block and convolutional codes based on circulant matrices," IEEE Trans. Inf. Theory, vol. 50, no. 12, pp. 2966-2984, Dec. 2004.

[8] A. Anastasopoulos, "A comparison between the sum-product and the min-sum iterative detection algorithms based on density evolution," GLOBECOM, 2001, pp. 1021-1025 vol. 2.

[9] K. Niu, K. Chen, J. Lin and Q. T. Zhang, "Polar codes: Primary concepts and practical decoding algorithms," IEEE Comm. Mag., July 2014.

[10] S. Sun and Z. Zhang, "Architecture and optimization of high-throughput belief propagation decoding of polar codes," ISCAS, 2016, pp. 165-168.

[11] A. Amaricai, D. Stein and O. Boncalo, "Generalized Very High Throughput Unrolled LDPC Layered Decoder," TELFOR, 2020, pp. 1-4.

[12] J. Li, G. He, H. Hou, Z. Zhang and J. Ma, "Memory efficient layered decoder design with early termination for LDPC codes," ISCAS, 2011.

[13] T. T. Bao Nguyen and H. Lee, "Efficient Four-way Row-splitting Layered QC-LDPC Decoder Architecture," ISOCC, 2018, pp. 210-211.

[14] S. Kim, G. E. Sobelman and H. Lee, "A Reduced-Complexity Architecture for LDPC Layered Decoding Schemes," IEEE TVLSI, vol. 19, no. 6, pp. 1099-1103, June 2011.

[15] Z. Wu and D. Liu, "Memory sharing techniques for multi-standard high-throughput FEC decoder," SAMOS, 2014, pp. 93-98.

[16] M. Y. Zinchenko, A. M. Levadniy and Y. A. Grebenko, "LDPC Decoder Power Consumption Optimization," IEEE REEPE, 2020, pp. 1-5.

[17] F. Maessen et al., "Memory power reduction for the highspeed implementation of turbo codes," IEEE SiPS, 2001, pp. 16-24.

[18] A. Niktash, H. T. Parizi, A. H. Kamalizad and N. Bagherzadeh, "RECFEC: A Reconfigurable FEC Processor for Viterbi, Turbo, Reed-Solomon and LDPC Coding," IEEE WCNC, 2008, pp. 605-610.

[19] G. Gentile, M. Rovini and L. Fanucci, "A multi-standard flexible turbo/LDPC decoder via ASIC design," ISTC, 2010, pp. 294-298.

[20] F. Naessens et al., "A 10.37 mm2 675 mW reconfigurable LDPC and Turbo encoder and decoder for 802.11n, 802.16e and 3GPP-LTE," IEEE VLSIC, 2010, pp. 213-214.

[21] C. -Y. Lin, C. -C. Wong and H. -C. Chang, "An Area Efficient Radix-4 Reciprocal Dual Trellis Architecture for a High-Code-Rate Turbo Decoder," IEEE TCAS-II, vol. 62, no. 1, pp. 65-69, Jan. 2015.

[22] C. -H. Lin, C. -Y. Chen and A. -Y. Wu, "Area-Efficient Scalable MAP Processor Design for High-Throughput Multistandard Convolutional Turbo Decoding," IEEE TVLSI, vol. 19, no. 2, pp. 305-318, Feb. 2011.

[23] I. Tsatsaragkos and V. Paliouras, "A Reconfigurable LDPC Decoder Optimized for 802.11n/ac Applications," IEEE TVLSI, Jan. 2018.

[24] O. Dizdar and E. Arıkan, "A High-Throughput Energy-Efficient Implementation of Successive Cancellation Decoder for Polar Codes Using Combinational Logic," IEEE TCAS-I, vol. 63, no. 3, March 2016.

[25] Youn Sung Park, Yaoyu Tao, Shuanghong Sun and Zhengya Zhang, "A 4.68Gb/s belief propagation polar decoder with bit-splitting register file," IEEE VLSIC, 2014, pp. 1-2.

[26] Ling Cong, Cui Long and Wu Xiaofu, "Further results on the equivalence between SOVA and max-log-MAP decodings," ICCT 2000.