LDPC Decoder Architecture for High-Data Rate Personal-Area Networks

Matthew Weiner and Borivoje Nikolić
EECS Department
University of California, Berkeley
Berkeley, USA
{mgweiner,bora}@eecs.berkeley.edu

Zhengya Zhang
EECS Department
University of Michigan, Ann Arbor
Ann Arbor, USA
zhengya@eecs.umich.edu

***Abstract*—Emerging standards for wireless communications in the 60GHz band, such as WiGig, IEEE 802.11ad, and IEEE 802.15.3c, require throughputs between 1.5 and 6Gb/s and use rate adaptive low-density parity-check (LDPC) codes as the main form of forward error correction. State-of-the-art flexible LDPC decoders cannot simultaneously achieve the high throughput mandated by these standards and the low power needed for mobile applications. This work develops a flexible, fully pipelined architecture for the IEEE 802.11ad standard capable of achieving both goals. Based on a decoder synthesized in a low-power 65nm CMOS technology, the decoder dissipates 42mW at the 1.5Gb/s throughput and 84mW at the 3Gb/s throughput for the worst-case matrix in the standard.**

I. Introduction

Low-density parity-check (LDPC) codes have made their way into virtually every modern high-performance wireless standard because of their powerful error correction capability using near-optimal, highly parallelizable decoding algorithms [1][2]. This includes many emerging standards for local and personal area networks in the 60GHz band, such as IEEE 802.11ad, WiGig, and IEEE 802.15.3c, which use LDPC codes almost exclusively [3][4]. These standards specify throughputs between 1.5 and 6Gb/s and use varying codes for different rates. As these systems target data rates between 1.5 and 3Gb/s for low power applications, the decoder must consume well below 100mW for these throughputs to meet the power budget of mobile platforms.

Currently, no flexible LDPC decoder architecture can simultaneously achieve both the throughput and power required by 60GHz applications. The lowest power flexible decoders dissipate 52mW with a throughput of up to 200Mb/s [5]. Others achieve higher throughput, but dissipate far more power [6].To solve this problem, we develop a fully pipelined architecture targeted for the IEEE 802.11ad standard that has fully parallelized variable nodes and layer serialized check nodes. By exploiting the structure of the lower rate codes, it reduces the number of sub-iterations by up to half.

The rest of the paper is structured as follows. Section II introduces the targeted class of LDPC codes and discusses the desired application. Section III describes the design choices made for the architecture, and Section IV details the functional design of the decoder's major blocks. Section V presents the implementation results, and Section VI concludes.

II. Background

A. LDPC Codes and Decoding Algorithms

LDPC codes are defined by a sparse binary $M \times N$ parity check matrix $\mathbf{H}$, which can be illustrated graphically using a factor graph where each bit has its own variable node (VN) and each parity check has its own factor or check node (CN). An edge connects VN i and a CN j if and only if $\mathbf{H}(j,i) = 1$.

The typical decoding algorithm is the sum-product algorithm, but practical implementations use the less complex offset min-sum algorithm [7]. It begins by initializing all VNs with the corresponding log-likelihood ratio from the channel, $L^{pr}(\mathrm{VN}_i)$, which is passed along the edges of the factor graph to neighboring CNs. Each CN j updates its message to each neighboring VN i according to

$$L(r_{ij}) = \max\left\{\min_{i' \in \mathrm{Row}[j]\backslash\{i\}} \left|L(q_{i'j})\right| - \beta, 0\right\} \prod_{i' \in \mathrm{Row}[j]\backslash\{i\}} \mathrm{sgn}\left(L(q_{i'j})\right) \quad (1)$$

where $L(q_{ij})$ is the previous message that VN i' sent to CN j, $\mathrm{Row}[j]/\{i\}$ is the set of VNs that share an edge with CN j except VN i, and β is an empirical correction factor. Next, each VN i updates its messages to each neighboring CN j according to

$$L(q_{ij}) = \sum_{j' \in \mathrm{Col}[i]\backslash\{j\}} L(r_{ij'}) + L^{pr}(\mathrm{VN}_i) \quad (2)$$

where $\mathrm{Col}[i]/\{j\}$ is the set of CNs that share an edge with VN i except CN j. Updating check and variable messages continues until the decoder reaches a stopping condition, such as finding a valid codeword, or exceeds a maximum number of iterations.

B. *High-Data Rate Wireless Communications*

There is a growing interest in using the wide unlicensed band around 60GHz for very high data rate communications, targeting synchronization of mobile terminals, cameras, and camcorders. To enable low-power dissipation the intention is to use single carrier modulation with complex equalization and LDPC decoding employed in the baseband along with optional beamforming. The IEEE 802.11ad single-carrier standard has three classes of operation with coded throughputs of 1.54, 3.08, 6.16Gb/s, although only the first two modes are used for low power operation [3]. It defines four irregular submatrix-based LDPC codes of rates 1/2, 5/8, 3/4, and 13/16 with a common block length of 672 bits. Figure 1 defines the base matrix for the rate 1/2 code. A numerical entry in the matrix represents a cyclically shifted 42×42 identity matrix with the shift amount given by the number, and blank entries represent all-zero submatrices. Figure 2 shows the bit error rate (BER) curves for all code rates with no quantization and with a quantization to five integer bits and zero fractional bits. The lower rate codes have a small quantization loss (<0.2dB), but the higher rate codes do not.

The matrices have another structural feature that can be exploited in the architectural design; the rate 5/8 and 1/2 codes have two and four pairs of non-overlapping submatrix layers, respectively, as shown in Figure 1. A highly parallelized decoder can process these layers simultaneously, reducing the number of effective layers for both codes to four. This decreases the number of sub-iterations, lowering the decoder's minimum operating frequency, and increases hardware utilization when processing these codes.

40		38		13		5		18						
34		35		27			30	2	1					
	36		31		7		34		10	41				
	27		18		12	20				15	6			
35		41		40		39		28			3	28		
29		0			22		4		28		27		23	
	31		23		21		20			12			0	13
	22		34	31		14		4				13		22	24

Figure 1. Rate 1/2 matrix with pairs of non-overlapping layers shaded in the same color

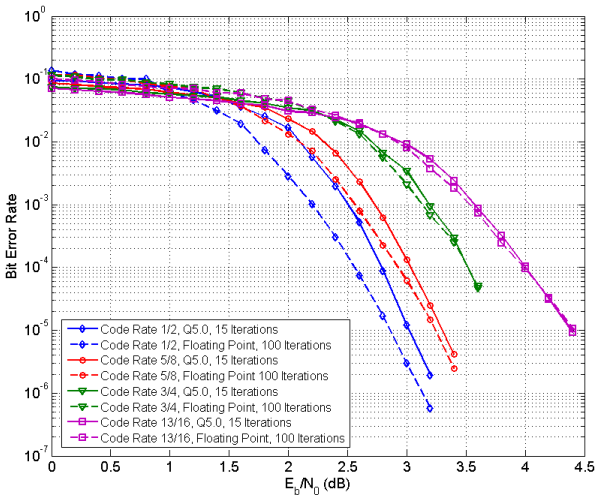

Figure 2. Floating point and 5-bit quantized BER results for 802.11ad codes

III. Architectural Design

Based on [8], the energy-area optimal architecture for Gb/s data rates is highly parallel. In this regime memory-based designs are suboptimal because there are a large number of small, inefficient memory banks. On the other hand, fully pipelined architectures store messages in pipeline registers, which have no issue with overhead or bandwidth restrictions. Although the registers consume significant power, they are more efficient than memory blocks at the targeted rates. For this reason, only fully pipelined designs are considered.

The architectural space for fully pipelined LDPC decoders consists of the parallelism level of the variable nodes and check nodes, the number of independent frames that the decoder processes simultaneously, and the pipeline depth. An exhaustive search for the optimal architecture is unrealistic because of the time needed to design, simulate, synthesize, and place and route an entire decoder for each combination of the parameters. However, a sensitivity-based heuristic search can yield a near-optimal design.

The search starts by listing the combinations of design parameters to explore. Regions of the design space that will yield high power designs are not considered, such as low parallelizations or excessively large pipeline depths because they require extremely large operating frequencies.

The next step creates the unique worst-case pipeline associated with each combination of parameters and uses it to estimate the number of cycles per iteration. The estimation is insensitive to the actual logic in each stage since only the number of cycles is of interest. From here, the minimum operating frequency f_{min} to achieve the desired throughput is calculated according to

$$f_{\min} = \frac{TI_{avg}C}{NF} \tag{3}$$

where T is the throughput, I_{avg} is the average number of iterations, C is the cycles per iteration, N is the block length of the code, and F is the number of frames processed at once.

Parameter combinations with a large f_{min} are eliminated since they cannot be significantly voltage-frequency scaled. The basic blocks of the remaining architectures are designed and synthesized, and the overall power and area for the decoder is estimated. A decoder with balanced area and power is chosen as the final design since it balances wiring overhead and logic power (initial synthesis does not take wiring overhead into account).

For this design the heuristic search returned a decoder with fully parallelized VNs, layer serialized CNs, and a pipeline depth of five that processes two data frames simultaneously.

IV. Functional Design

A. *Overall System*

The decoder has five major blocks: variable nodes, check nodes, barrel shifters, pre-routers, and post-routers. Figure 3 shows the high-level connection of all the blocks. The VNs are combined into 16 groups of 42VNs, called a variable node group (VNG). Each VN within a VNG connects to one port of

a 42-input barrel shifter to implement the submatrix shifts. The outputs of the barrel shifter are further routed by the pre-routers, which connect to one of the 16 inputs of the 42 CNs. The outputs of each CN go through inverse shifting using post-routers and another set of barrel shifters. The design has several levels of hierarchy in order to keep irregular wiring local and make the global wires as regular as possible.

Since the design layer serializes the CNs, they are time-multiplexed to act as different CNs in each cycle. For example, in one cycle, each VN sends a single message that has been marginalized for the CNs in the first layer. The CN has all of the information it needs to compute the new check to variable (C2V) message from all CNs in the first layer, and, after it has finished processing the inputs, it sends back a single message to all neighboring VNs. In the next cycle, the VNs send another single message that has been marginalized for the CNs in the second layer, so the same CNs can compute the C2V message from the CNs of the second layer. This continues for all the layers in the matrix, with serial messages being passed back and forth from VNs to CNs.

The decoder uses the flooding schedule because layered decoding has too many dependencies to be used effectively in a highly parallel, fully pipelined design. It processes one layer per pipeline stage, and the VNs accumulate one frame's messages while sending out the other's.

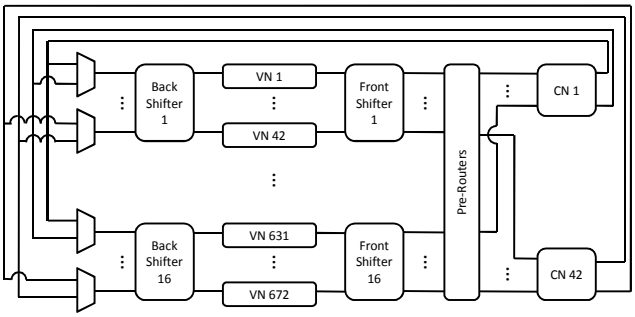

Figure 3. Decoder block diagram

B. Variable Node

The VN implements (1) and performs both the CN and VN marginalizations, which keeps all memory within the VN. When starting to decode a new data frame, the VN loads in a prior value to either the first or second frame's prior and accumulation registers. Over the next four cycles (for the rate 13/16 code, the fourth cycle is unused) it sends variable to check (V2C) messages to its neighboring CNs. Since the VN performs the C2V marginalization, it locally stores the V2C messages it just sent out in a shift register. Once the unmarginalized C2V message returns, the VN compares the first minimum magnitude to the V2C magnitude from the shift register. If they match, the VN uses the second minimum; otherwise, it uses the first minimum. It marginalizes the sign by taking the XOR of the new and stored V2C signs.

The VN accumulates the marginalized C2Vs, with the prior value added in when the first C2V arrives, and it also stores them individually in another shift register. After all C2V messages have been accumulated, the VN calculates the new V2C messages by subtracting out the saved C2V values from the accumulated value, which is the V2C marginalization. Also, the sign bit of the accumulated value is used as the hard decision and for early termination. Figure 4 shows the block diagram for the VN.

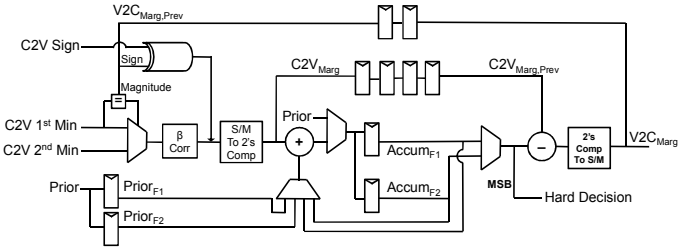

Figure 4. Variable node schematic

C. Check Node

Since the VN performs the marginalization in (2), the CN only needs to find the first and second minima of all the incoming V2C message magnitudes and the product of the V2Cs' signs. The CN computes the former with a compare-select tree. A sorting block at the beginning arranges pairs of inputs in ascending order, and subsequent stages of the tree select the minimum two out of four inputs. A separate XOR tree finds the product of the signs.

Using the matrix properties from Section II.B, the CN is modified to process either a single high-weight layer or two non-overlapping layers. To process a single layer, it takes the output of the entire tree to find the first and second minima of all 16 inputs. For non-overlapping layers, the weight is at most 8, so the top half of a tree can process one layer and the bottom half can process the other. The outputs are taken at the second to last stage to obtain first and second minima for each layer. This partitions a 16-input CN into two 8-input CNs, giving the CN two levels of granularity. Because the layers do not overlap, no read before write conflicts occur. Figure 5 shows the final CN architecture for the magnitude computation. The sign's XOR tree is partitioned similarly.

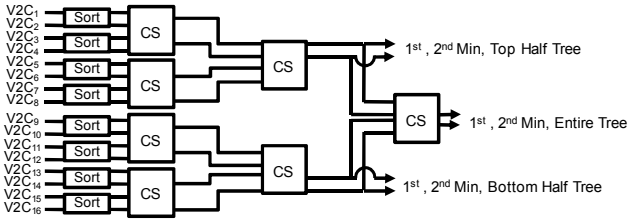

Figure 5. Check node magnitude computation schematic

D. Pre- and Post-Routing

When processing two non-overlapping layers at once, barrel shifters alone cannot guarantee that the first layer's inputs will go to the top half of each CN and the second layer's inputs will go to the bottom half. Granular CNs need two extra sets of routing to allow this, as depicted in Figure 6. Pre-routing comes before the CN and selects which VNGs go to the top half or bottom half of each CN. Post-routing comes after the CN and, for each VNG, selects whether to send the top tree's or bottom tree's result. Both types of routers are

implemented with a small number of muxes. In the case of processing one layer, the routers do not shift the messages.

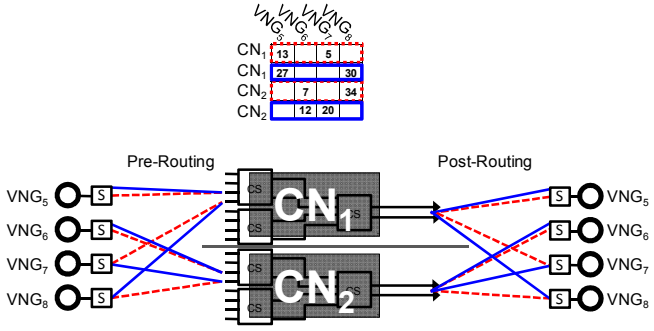

Figure 6. Operation of pre- and post-routers

E. *Pipeline*

The decoder has five pipeline stages and processes two independent data frames simultaneously. Each full iteration takes three sub-iterations for the rate 13/16 code and four for the rest, one for each time-multiplexed use of the check nodes.

In the first stage, the VN outputs the marginalized V2C and the barrel shifter reorders the messages. The second stage consists of the pre-routing and global wiring to the check node. The CNs process their inputs in the third stage and route the messages back to the VNs across the global wires in the fourth stage. In the fifth stage, the VN accumulates the serial messages over three or four cycles. The accumulation would normally cause a four cycle bubble in the pipeline due to the dependency between accumulating all the C2Vs and sending out the next V2Cs. The bubble is just large enough to accommodate processing a second frame. This reduces the bubble to one cycle for the rate 13/16 code and removes it completely for the other rates because no dependencies exist between the two frames. Figure 7 gives the pipeline diagram for the decoder with four sub-iterations.

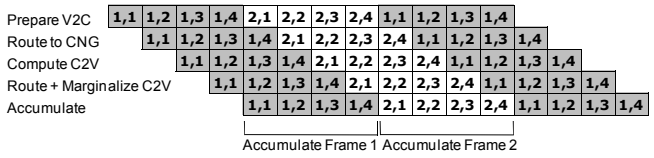

Figure 7. Pipeline diagram for rate 1/2, 5/8, and 3/4 codes where the first number indicates the frame and the second the sub-iteration

V. Implementation Results

To achieve a 3Gb/s throughput in the worst case, the decoder must operate at 150MHz for the rate 1/2 code, which requires the highest frequency since it takes the most iterations to converge. In order to accommodate voltage-frequency scaling, the decoder is synthesized at 200MHz to allow for at least a 25% reduction in supply voltage. An extra slack of 20% allows further voltage-frequency scaling down to 0.8V. Table 1 summarizes the synthesis results.

Almost 60% of the power goes into the variable nodes because they are the most complicated and the most numerous objects in the decoder. Explicit pipeline registers use half of the remaining power, and the front and back shifters and check nodes consume 4-6% of the overall power each. The pre- and post-routers use negligible portion of the decoder's power, which shows the granular check nodes cost little overhead. The relative power consumed by pipeline registers is misleading because the variable nodes contain implicit pipeline registers. When they are taken into account, pipeline registers consume 67.5% of the total power, showing the cost of using a fully pipelined decoder.

Table 1. Synthesis Results

Technology	ST 65nm low power CMOS	
Logic Area	1.3mm^2	
Operating Class	1	2
Coded Throughput	1.54Gb/s	3.08Gb/s
Clock Frequency	75MHz	150MHz
Supply Voltage	0.8V	0.8V
Power Consumption	42mW	84mW

VI. Conclusion

The proposed fully pipelined architecture with granular check nodes supports all matrices and low-power modes of the IEEE 802.11ad standard. It occupies 1.3mm^2 and consumes 42mW at a throughput of 1.5Gb/s and 84mW at 3Gb/s for the worst-case matrix, meeting the constraints set by the target applications. The architecture can be extended to other short block length codes that have the same property of non-overlapping layers, such as the 802.15.3c standard.

Acknowledgments

The authors acknowledge the support from the NSF GFRP and NSF Grant CCF-0635372. They also acknowledge the contributions of the students, faculty and sponsors of the Berkeley Wireless Research Center, particularly Marvell Semiconductor, ST-Microelectronics, and Pascal Urard.

References

[1] R.G. Gallager, *Low-Density Parity-Check Codes*. Cambridge, MA: MIT Press, 1963.

[2] T.J. Richardson and R.L. Urbanke, "The capacity of low-density parity check codes under message-passing decoding," *IEEE Trans. Inf. Theory*, vol. 47, pp. 599-618, Feb 2001.

[3] *IEEE draft standard for information technology-wireless LANs-part 21: mmWave PHY specification,* May 2010, IEEE Std. 802.11ad.

[4] *Draft amendement to IEEE standard for information technology -local and metropolitan area networks-part 12.2: SC mmWave PHY mode,* 2008, IEEE Std. 802.15.3c.

[5] X.-Y. Shih, C.-Z. Zhan, C.-H. Lin, and A.-Y. Wu, "An 8.29 mm^2 52 mW multi-mode LDPC decoder design for mobile WiMAX system in 0.13 µm CMOS process," *IEEE J. of Solid-State Circ.*, vol.43, no.3, pp.672-683, Mar 2008.

[6] S. Yang and J.R. Cavallaro, "A low-power 1-Gbps reconfigurable LDPC decoder design for multiple 4G wireless standards," *IEEE Intl. SOC Conf., 2008*, vol., no., pp.367-370, 17-20 Sept 2008.

[7] J. Chen, A. Dholakia, E. Eleftheriou, M. Fossorier, and X-Y. Hu, "Reduced-complexity decoding of LDPC codes," *IEEE Trans. Comm.*, vol.53, no.7, pp. 1232- 1232, Jul 2005.

[8] Z. Zhang, V. Anantharam, M.J. Wainwright, and B. Nikolic, "An efficient 10GBASE-T ethernet LDPC decoder design with low error floors," *IEEE J. Solid-State Circ.*, vol.45, no.4, pp.843-855, Apr 2010.