HIGH THROUGHPUT, PARALLEL, SCALABLE LDPC ENCODER/DECODER ARCHITECTURE FOR OFDM SYSTEMS

Yang Sun, Marjan Karkooti and Joseph R. Cavallaro
Department of Electrical and Computer Engineering, Rice University, Houston, TX, 77005
{ysun, marjan, cavallar}@rice.edu

ABSTRACT

This paper presents a high throughput, parallel, scalable and irregular LDPC coding and decoding system hardware implementation that supports twelve combinations of block lengths 648, 1296, 1944 *bits and code rates* $1/2$, $2/3$, $3/4$, $5/6$ *based on* $IEEE$ $802.11n$ *standard. Based on architecture-aware LDPC codes, we propose an efficient joint LDPC coding and decoding hardware architecture. The prototype architecture is being implemented on FPGA and tested over the air on our wireless OFDM testbed, which is a highly capable, scalable and extensible platform for advanced wireless research. The ASIC resource requirements of the decoder are reported and a trade-off between pipelined and non-pipelined implementation is described.*

1. INTRODUCTION

Low density parity check (LDPC) codes have received major attention in the research community in recent years because of their excellent error correction capability and performance. Several architectures have been proposed for LDPC decoders [2][4]. Most of these architectures support just one specific type of LDPC codes or a family of codes with a fixed block length and code rate. At Rice University, a flexible high throughput LDPC decoder that supports a family of codes with different block lengths and code rates has been proposed in [5]. In this work we propose a joint LDPC encoder and decoder based on the $IEEE$ $802.11n$ draft specification [1]. The encoder/decoder pair supports twelve combinations of block lengths 648, 1296, 1944 bits and code rates $1/2$, $2/3$, $3/4$, $5/6$. The parity check matrices of these codes are irregular block-structured codes as defined in [1]. The layered belief propagation algorithm [7] is used in our design because it converges twice as fast as the standard belief propagation algorithm resulting in twice the throughput. A prototype of the encoder/decoder architecture is implemented in Verilog HDL and tested on FPGA and also synthesized on 0.13 μm ASIC. The logic synthesis report shows a better performance in terms of area efficiency and throughput than the currently reported works on $IEEE$ $802.11n$ LDPC decoder [5][6].

2. EFFICIENT LDPC ENCODER

An LDPC code is a linear block code specified by a very sparse parity check matrix (PCM). LDPC codes are usually represented by a bi-partite graph in which a variable node corresponds to a 'coded bit' or a PCM column, and a check node corresponds to a parity check equation or a PCM row. There is an edge between each pair of nodes if there is a 'one' in the corresponding PCM entry. In a general analysis an (n, k) LDPC code has k information bits and n coded bits with code rate $r = k/n$. The parity-check matrix H is of dimension $(n - k) \times n$, and it defines a set of equations.

$$H \cdot v^T = 0 \ (mod\ 2) \tag{1}$$

Denote H = [H_1 H_2], where H_1 and H_2 have dimensions $(n - k) \times k$ and $(n - k) \times (n - k)$, respectively. Denote codeword v=[s p], where s is the k information bits and p is the $n - k$ parity bits. From (1), we have

$$H_1 \cdot s^T + H_2 \cdot p^T = 0 \ (mod\ 2) \tag{2}$$

$$p^T = H_2^{-1} H_1 \cdot s^T \ (mod\ 2) \tag{3}$$

High encoding complexity arises from the high density of H_2^{-1} [8]. However, for the $IEEE$ $802.11n$ proposed check matrix, H_2 has a simple deterministic structure, and encoding can be performed recursively. As shown in Fig. 1, H_2 is an $m \times m$ array of $S \times S$ sub-matrices, where S could be 27, 54 or 81 depending on the different code lengths. The first column $h = [h_0, h_1, ..., h_{m-1}]^T$ satisfies that 1) h_i is either a $S \times S$ zero matrix or a $S \times S$ shifted identity matrix and 2) $\sum_{i=0}^{m-1} h_i = I_{s \times s}$ $(mod\ 2)$.

Based on $IEEE$ $802.11n$[1], $H1$ consists of a $m \times b$ array of $S \times S$ sub-matrices, which are either zero or shifted identity matrices. Given a block of information bits s, if we decompose s into $1 \times b$ array of $1 \times S$ sub-matrices,

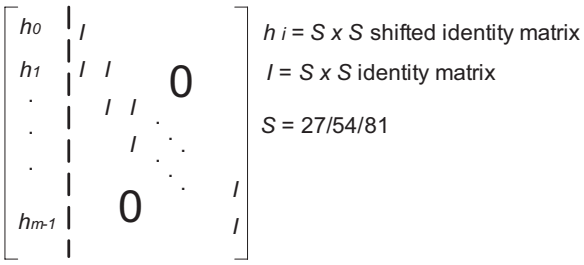

Figure 1. Parity matrix H2

and also decompose parity bits p into $1 \times m$ array of $1 \times S$ sub-matrices, from (2) and H_2, we can prove that

$$p_0^T = \sum_{i=0}^{m-1} H_1^{(row\ i)} \cdot s^T \ (mod\ 2)$$
$$p_1^T = H_1^{(row\ 1)} \cdot s^T + h_0 \cdot p_0^T \ (mod\ 2)$$
$$p_2^T = H_1^{(row\ 2)} \cdot s^T + h_1 \cdot p_0^T + p_1^T \ (mod\ 2)$$
$$...$$
$$p_{m-1}^T = H_1^{(row\ m-1)} \cdot s^T + h_{m-2} \cdot p_0^T + p_{m-2}^T \ (mod\ 2),$$

Since H_1 consists of either zero or shifted identity sub-matrices, $H_1^{(row\ i)} s^T$ can be efficiently implemented by a S-bit barrel shifter, a S-bit XOR and a S-bit Register, as shown in Fig. 2. With all the $H_1^{(row\ i)} s^T$ determined, the p_0 through p_{m-1} can be found recursively.

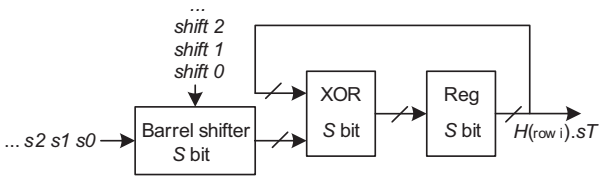

Figure 2. Circuit to calculate $H_1^{(row\ i)} s^T$

3. LDPC SOFT DECODING ALGORITHM

The decoder architecture proposed in this paper utilizes the iterative layered belief propagation (LBP) algorithm as defined in [7]. Fig. 3 shows a block-structured parity-check matrix, which is a $D \times B$ array of $S \times S$ sub-matrices, each sub-matrix is either a zero or a shifted identity matrix with random shift value. In every layer, each column has at most one 1, which satisfies that there are no data dependencies between the variable node messages, so that the messages flow in tandem only between the adjacent layers. The block size S could be 27, 54 or 81 corresponding to the code lengths-648, 1296 and 1944 respectively [1].

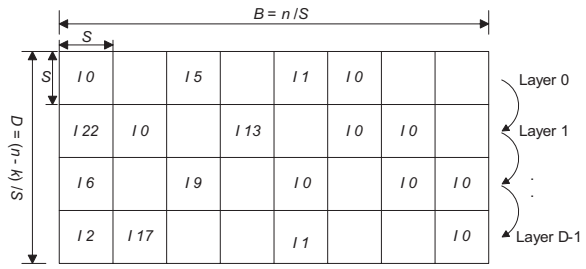

Figure 3. Block-structured irregular matrix, where Ix is an identity sub-matrix right shifted by x

Let $L(q_{mj})$ denote the variable node log likelihood ratio (LLR) message sent from variable node j to the check node m, then:

$$L(q_{mj}) = L(q_j) - R_{mj}, \tag{4}$$

$$R_{mj} = \prod_{j\prime \in N(m)\backslash\{j\}} \text{sign}\left(L(q_{mj\prime})\right) \Psi\left[\sum_{j\prime \in N(m)\backslash\{j\}} \Psi\left(L(q_{mj\prime})\right)\right], \tag{5}$$

$$\Psi(x) = -\log\left[\tanh\left(\frac{|x|}{2}\right)\right],$$

$$L(q_j) = L(q_{mj}) + R_{mj}, \tag{6}$$

in which R_{mj} is the check node LLR message sent from the check node m to the variable node j and $L(q_j)$ ($j = 1, \ldots, N$) represent the *a posteriori* probability ratio (APP) for all variable nodes. The APP messages are initialized with the channel reliability values of the coded bits. $N(m)$ is the set of all variable nodes connected to the check node m. To simplify the hardware implementation of the non-linear function $\psi(x)$, updating of the check node messages in (5) is replaced with the modified min-sum approximation [3]. According to this solution, the updating of check node messages in the mth row of the PCM is determined as:

$$R_{mj} \approx \prod_{j\prime \in N(m)\backslash\{j\}} \text{sign}\left(L(q_{mj\prime})\right) \times \max\left(\min_{j\prime \in N(m)\backslash\{j\}} |L(q_{mj\prime})| - \beta, 0\right).$$

where β is a correcting offset equal to a positive constant. With a properly chosen β, the modified min-sum approximation exhibit only about 0.1 dB of degradation in performance [3].

Hard decisions can be made after every horizontal layer based on the sign of $L(q_j)$. If all parity-check equations are satisfied or the pre-determined maximum number of iterations is reached, then the decoding algorithm stops. Otherwise, the algorithm repeats from Eq. (4) for the next horizontal layer.

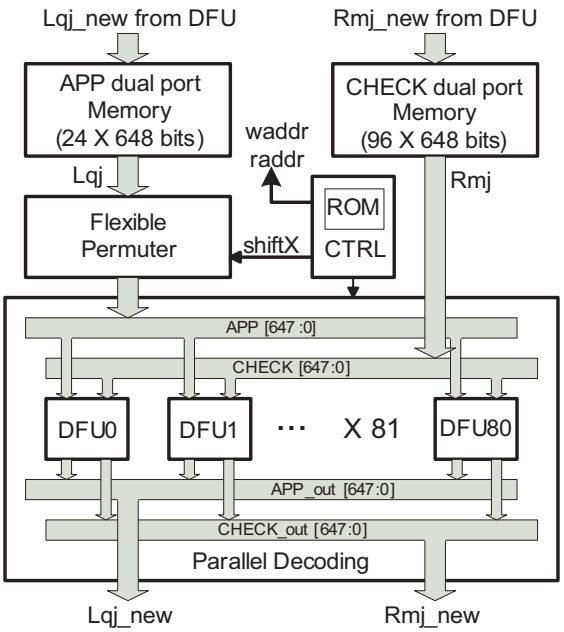

Figure 4. LDPC decoder architecture

4. PARALLEL DECODER

Fig. 4 shows the block diagram of the LDPC decoder based on the layered BP decoding algorithm. Depending on the block lengths, $S(27/54/81)$ parallel variable node messages are read out from APP memory and passed through a flexible permuter to be routed to the correct decoding function unit (DFU) for calculating new variable node messages. The shift values of the matrices are fetched from ROMs to generate appropriate addresses to the APP and Check memories. The DFUs are the central processing unit of the architecture since they calculate and update APP and Check memories based on equations (4)-(6). For the code size of 1944, all the 81 DFUs are used, for the code size of 1296, 54 DFUs are used, and for the code size of 648 only 27 DFUs are used. A pseudo-code for this DFU is shown in Algorithm 1.The decoding algorithm will require Memory Read + Memory Write + Processing = $2 * Wr + 1$ clock cycles to finish one layer of decoding. (Wr is equal to the number of non-zero sub-matrices in a layer).

Algorithm 1: Pseudo code for layered LDPC decoding:

for iteration = 0 *to* P
  for layer = 0 *to* $M-1$
    for $j = 0 : n-1$
      Read $L(q_j)$ *and* R_{mj} *from memory*
      Calculate equation $(4)-(6)$, *update new* $L(q_j)$ *and* R_{mj} *to memory*
    end
  end
end

4.1. Pipelined decoder architecture

In this section, we propose a pipelined decoding algorithm as well as a hardware implementation. Fig. 5 shows a two-stage pipeline decoding. Instead of waiting for layer i to update all the APP node messages, the next layer $i+1$ can start to read APP node messages slightly after layer i begins to update new APP node messages. Due to the random locations of non-zero sub-matrices in each layer, it might have a pipeline hazard which is shown in Fig. 6 as an example. The cross signs in Fig. 6(a) indicate non-zero sub-matrices. As shown in Fig. 6(b), at clock cycle 6, layer $i+1$ is trying to read memory location 3, which will not be updated until clock cycle 8 (we assume 1 clock cycle SRAM write latency). In order to avoid memory conflicts, two memory-read stalls are inserted at clock cycle 6 and cycle 7.

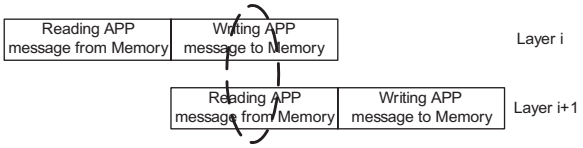

Figure 5. Two stage pipelining decoding

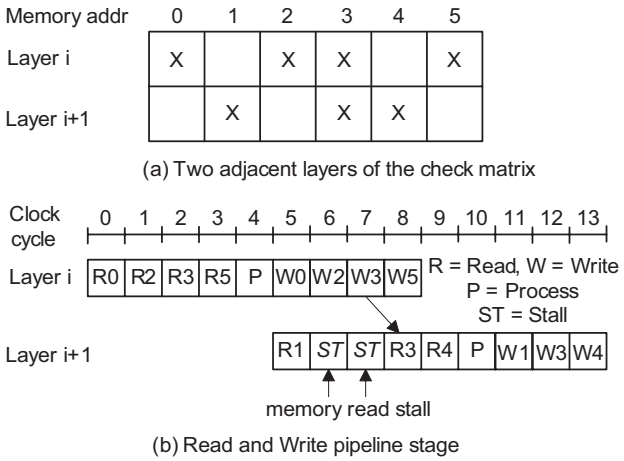

Figure 6. Pipelining hazard

A hardware implementation of this pipelined DFU is shown in Fig. 7. A local FIFO will be needed to buffer the next layer's data while processing the current layer's data. The proposed pipeline decoding can increase the overall throughput by about 1.5 to 2 X, depending on the code rates.

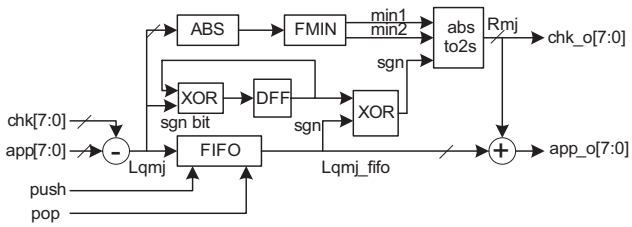

Figure 7. Pipelined DFU

5. DECODER ASIC IMPLEMENTATION AND PERFORMANCE ANALYSIS

The LDPC Decoder architecture was implemented in Verilog HDL and synthesized on a TSMC $0.13\mu m$ standard cell library. Table 1 shows a summary of synthesis results. Complexity is measured in equivalent gates for logic and in bits for memories. An overall complexity of 90 K logic gates is measured for non-pipelined implementation, plus $77,760$ bits RAM. While 195 K logic gates is measured for pipelined implementation, plus $77,760$ bits memories.

A Verilog RTL simulation model was used to measure average throughput v.s. SNR level. For instance, at a rather low SNR 1.0 dB, the pipelined decoder can achieve 150 Mbps. While at SNR 2.2 dB, the pipelined decoder can achieve about 1 Gbps.

Table 1. LDPC decoder design statistics.

	Non-pipeline	Pipeline
Frequency	400 MHz	400 MHz
Area	1.3 mm^2	1.9 mm^2
Logic gates	90 K	195 K
Total memory	$77,760$ bits	$77,760$ bits
Throughput@2.2dB SNR	500 Mbps	1 Gbps
Throughput@1.0dB SNR	80 Mbps	150 Mbps

6. LDPC ENCODING/DECODING TESTING ON WIRELESS TESTBED

In order to explore LDPC encoding and decoding performance, we have been conducting over the air OFDM experiments using the Rice Wireless Open Access Research Platform. As shown in Fig. 8, the Rice WARP Platform (http://warp.rice.edu) is reconfigurable and consists of FPGA baseband processors along with multiple attached 2.4 GHz radio subsystems, which enables quick prototyping of wireless communication algorithms. Proposed LDPC encoding and decoding is currently being tested on the WARP platform.

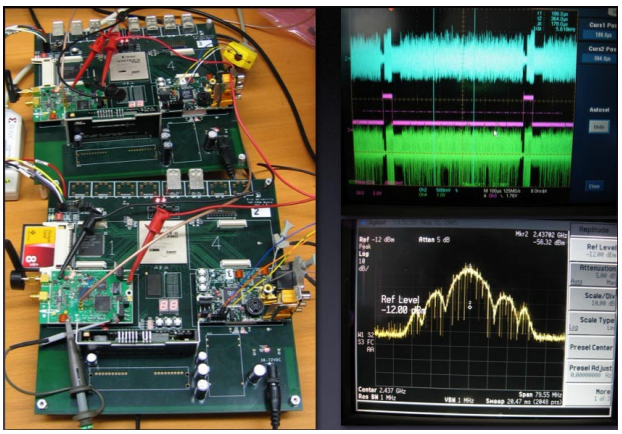

Figure 8. Wireless OFDM testbed

7. CONCLUSION

We have presented a high throughput parallel LDPC decoder and an efficient LDPC encoder based on the $IEEE802.11n$ standard. The encoder/decoder is based on block structured irregular codes that can be extended to support other code lengths and code rates. The LDPC encoder and decoder is being implemented in FPGA and tested on our wireless testbed. Future applications will be LDPC real time encoding and decoding for MIMO OFDM.

REFERENCES

[1] *IEEE 802.11n Wireless LAN Medium Access Control MAC and Physical Layer PHY specifications*. IEEE 802.11n-D1.0, 2006.

[2] A.J. Blanksby and C.J. Howland. A 690-mW 1-Gb/s 1024-b, rate-1/2 low-density parity-check code decoder. *IEEE Journal of Solid-State Circuits*, 37(3):404 – 412, 2002.

[3] J. Chen, A. Dholakai, E. Eleftheriou, M. Fossorier, and X. Hu. Reduced-complexity decoding of LDPC codes. *IEEE Transactions on Communications*, 53:1288 – 1299, Aug 2005.

[4] Y. Chen and D. Hocevar. A FPGA and ASIC implementation of rate 1/2, 8088-b irregular low density parity check decoder. In *IEEE Global Telecommunications Conference, 2003*, Dec. 2003.

[5] M. Karkooti, P. Radosavljevic, and J. R. Cavallaro. Configurable, High Throughput, Irregular LDPC Decoder Architecture:Tradeoff Analysis and Implementation. In *IEEE International Conference on Application-specific Systems, Architectures and Processors (ASAP 06)*, 2006. Accepted.

[6] F. R. M. Rovini, N. L'Insalata and L. Fanucci. VLSI Design of a High-Throughput Multi-Rate Decoder for Structured LDPC Codes. *Proceedings of the 2005 8th Euromicro conference on Digital System Design*, pages 202–209, Aug 2005.

[7] M. M. Mansour and N. R. Shanbhag. High-throughput LDPC decoders. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 11:976–996, Dec. 2003.

[8] T.J. Richardson and R. Urbanke. Efficient Encoding of Low-Density Parity-Check Codes. *Information Theory, IEEE Transactions on,*, 47(2):638 – 656, 2001.