Low Complexity Quasi-Cyclic LDPC Decoder Architecture for IEEE 802.11n

Sherif Abou Zied[1], **Ahmed Tarek Sayed**[1], **and Rafik Guindi**[2]
[1]Varkon Semiconductors, Cairo, Egypt
[2]Nile University, Giza, Egypt

Abstract— *In this paper, we present a fully pipelined LDPC decoder for 802.11n standard that supports variable block sizes and multiple code rates. The proposed architecture utilizes features of Quasi-Cyclic LDPC codes and layered decoding to reduce memory bits and interconnection complexity through efficient utilization of permutation network for forward and backward interconnection routing. Permutation network reorganization reduced the overall resources required for routing, thus reducing the overall decoder dynamic power consumption. Proposed architecture has been synthesized using Virtex-6 FPGA and achieved 19% reduction in dynamic power consumption, 5% less logic resources and 12% increase in throughput.*

Keywords: LDPC, Iterative decoder, Error correction, Min-sum, Low power, Low complexity

1. Introduction

Low Density Parity Check (LDPC) codes, a class of error correction codes that can perform close to the Shannon limit proposed by Gallager in his 1962 PhD thesis [5], [8], [9].LDPC has been considered in different wireless standards due to its superior error correction performance, including digital video broadcasting (DVB-S2, DVB-T2) for satellite and terrestrial digital television, 802.11ad, 10 Gigabit ethernet (10GBASE-T), broadband wireless access (WiMax), wireless LAN (802.11n), deep space communications and magnetic storage in hard disk drives.

Reduced interconnect complexities, smaller die areas, lower power dissipation, and design reconfigurability (runtime) to support multiple code lengths and code rates are the main optimization areas required for efficient LDPC decoder [11]. Recently Quasi-Cyclic (QC) LDPC codes, a kind of architecture aware codes were adopted by several modern wireless communication standards such as IEEE 802.11n, IEEE 802.16e and IEEE 802.15.3c. On the one hand, QC-LDPC codes facilitate efficient high-speed decoding due to the regularity of their parity check matrices. On the other hand, random-like LDPC codes require complex routing for VLSI implementation, which not only consumes a large amount of chip area, but also significantly increases the computation delay and dynamic power consumption.

In this paper we present a low complexity fully pipelined QC-LDPC decoder based on layered minimum-sum algorithm with much less memory bits used and reduced routing overhead targeting wireless LAN 802.11n standard. Layered decoding is proved to converge approximately twice as fast as classical message passing with a flooding schedule [12]. Architecture of the proposed decoder reduces hardware overhead by utilizing only one permutation network rather than pre-processing and post-processing permutation network as in [2], [14].

2. Background

2.1 LDPC Codes and Decoding Algorithms

Low density parity check codes are a class of linear block codes defined by a sparse $M \times N$ parity check matrix H. N represents the number of bits in the code, called the block length, and M represents the number of parity checks.The information length K is $K = N - M$ for full-rank matrices, otherwise $K = N - rank$. The rate of the code R is defined as $R = K/N$ and gives the fraction of information bits in each codeword. Number of ones per column is called column weight W_c and same number of ones per row is called row weight W_r. A code is said to be regular if all rows of parity check matrix have same weight W_r and all columns have same W_c otherwise, it is irregular code.

LDPC codes can be described graphically using Tanner graph. A Tanner graph consists of check nodes representing check equations, variable nodes representing soft-bits and between each variable node i and check node j there exists an edge if and only if $H(j, i) = 1$. Connected pairs of variable node and check node are called neighbors. Figure 1 shows Tanner graph for generic irregular LDPC code where variable nodes (VN) drawn as circles and check nodes (CN) represented by squares. A path that can be traced from one node in the graph back to the same node in the graph while not passing through any other node more than once is called a cycle. The communication performance of an LDPC code is determined by the girth of its factor graph, where girth is the size of the smallest cycle in the graph.

Low-density parity-check codes are decoded iteratively using the belief propagation algorithm, also known as the message-passing algorithm [5]. LDPC codes have been shown to perform very close to the Shannon limit when decoded using the iterative message-passing algorithm.

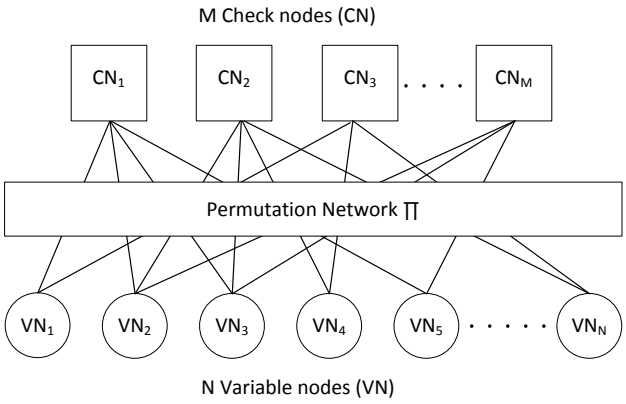

Fig. 1: Tanner graph representation for irregular LDPC codes

The iterative message-passing algorithm is the most widely used method for practical decoding .After receiving the corrupted information, algorithm begins by processing it and then iteratively corrects the received data. Message-passing algorithm can usually reach convergence within a small number of iterations when operating on graphs containing no short cycles. Message passing algorithm can be realised by Sum-Product (SPA) [8] or Min-Sum (MS) [4] algorithms which are near-optimum decoding algorithms and widely used in LDPC decoders.

2.1.1 Sum-Product Decoding Algorithm

The sum-product algorithm (SPA) is the traditional realization of the message passing algorithm. Assuming code word $(x_1, x_2, x_3, \cdots, x_n)$ was sent over a channel and received as $(y_1, y_2, y_3, \cdots, y_n)$. Sum-Product algorithm can be summarized as follows:

- Variable nodes are first initialized with soft information from channel. After initialization, each variable node updates its corresponding check nodes with variable to check messages.

$$\beta_{ij} = L_i = log\frac{Pr(x_i = 0|y_i)}{Pr(x_i = 1|y_i)} \tag{1}$$

  where β_{ij} is message from variable node $V(i)$ to check node $C(j)$.
- Check node update phase: check node messages are calculated using received β messages from variable node as follows:

$$\alpha_{ij} = \prod_{j' \in V(i)\backslash j} sign(\beta_{ij'}) \times \Phi(\sum_{j' \in V(i)\backslash j} \Phi(|\beta_{ij'}|)) \tag{2}$$

$$\Phi(x) = -log(tanh\frac{|x|}{2}) \tag{3}$$

  where α_{ij} is the message from check node $C(i)$ to variable node $V(j)$. Message from check node $C(i)$ to variable node $V(j)$ is calculated using all β messages from neighbouring variable nodes excluding β message from variable node $V(j)$. Φ is a non-linear function represented at equation 3 which indicates magnitude part of equation 2.
- Variable node update phase: Variable nodes calculates variable to check messages β_{ij} using its current value and received α_{ij} messages from neighbour check nodes as follows:

$$\beta_{ij_{new}} = \beta_{ij_{old}} + \sum_{i' \in C(j)\backslash i} \alpha_{i'j} \tag{4}$$

  same as in check to variable messages, Message from variable node $V(i)$ to variable node $C(j)$ is calculated using all α messages from neighbouring check nodes excluding α message from check node $C(j)$.
- Once variable nodes update is finished, reliability values for each bit is calculated same as in equation 4 but with all α messages received from all neighbouring check nodes as in equation 5.

$$z_i = \beta_{ij_{old}} + \sum_{i \in C(j)} \alpha_{ij} \tag{5}$$

  From estimated vector $Z = (z_1, z_2, z_3, \cdots, z_n)$ bit values are calculated by:

$$b_i = \begin{cases} 1, & \text{if } z_i \leqslant 0 \\ 0, & \text{if } z_i > 0 \end{cases} \tag{6}$$

  If $H.B^T = 0$, then B is a valid code word and therefore the iterative process has converged and decoding stops, this is called early termination. Otherwise the decoding repeats until a valid code word is obtained or the number of iterations reaches a maximum number, I_{max} , which terminates the decoding process.

2.1.2 Minimum Sum Algorithm

Minimum sum algorithm [6], [4] is a simplified version of sum product algorithm. Minimum sum algorithm simplifies the calculation of equation 2 even further by recognizing that the term corresponding to the smallest β_{ij} dominates the product term and so the product can be approximated by a minimum:

$$\alpha_{ij} = \prod_{j' \in V(i)\backslash j} sign(\beta_{ij'}) \times min|\beta_{ij'}| \tag{7}$$

Because check node processing requires the exclusion of $V(j)$ while calculating the $min|\beta_{ij'}|$ for α_{ij} , it necessitates finding both the first and second minimums (Min1 and Min2, respectively). In this case $min|\beta_{ij'}|$ is more precisely defined as follows:

$$min_{j' \in V(i)\backslash j}|\beta_{ij'}| = \begin{cases} Min1, & \text{if } j \neq argmin(Min1_i) \\ Min2, & \text{if } j = argmin(Min1_i) \end{cases} \tag{8}$$

13	48	80	66	4	74	7	30	76	52	37	60		49	73	31	74	73	23		1	0		
69	63	74	56	64	77	57	65	6	16	51		64		68	9	48	62	54	27		0	0	
51	15	0	80	24	25	42	54	44	71	71	9	67	35		58		29		53	0		0	0
16	29	36	41	44	56	59	37	50	24		65	4	65	52		4		73	52	1			0

Fig. 2: Base matrix for the rate 5/6 code with codeword length 1944

Table 1: IEEE 802.11n LDPC parameters

Codeword size	Rates	Z
648	1/2 , 2/3, 3/4, 5/6	27
1296	1/2 , 2/3, 3/4, 5/6	54
1944	1/2 , 2/3, 3/4, 5/6	81

Then, minimum sum algorithm goes through same steps of sum-product algorithm till reaching a valid codeword or reaching maximum number of iterations.

2.2 *IEEE 802.11n* LDPC Code

The IEEE 802.11n defines four irregular LDPC codes of rates 1/2, 2/3, 3/4, and 5/6 with three different codeword lengths 648, 1296 and 1944 bits. Figure 2 defines the base H-matrix for the rate 5/6 code with codeword length 1944. In general, H-matrices for QC-LDPC codes are divided into number of macro layers and each macro layer is further divided into number of sub-matrices which are cyclic-permutations of the identity matrix or null sub-matrices. For the 802.11n standard, the base matrices for all codes consists of 24 columns and $R \times 24$ rows of sub-matrices. Each entry in the base matrices defines a $Z \times Z$ sub-matrix. The first $R \times 24$ columns correspond to information bits, the second $(1-R) \times 24$ columns for the parity information which have a fixed structure required by the encoder design. The sub-block size Z of the sub-matrices is variable according to codeword size. Table 1 summarizes different parameters for 802.11n LDPC.

All code rates are suitable for layered decoding. This layered decoding approach which requires a special message scheduling is explained in Section 2.3.

2.3 LDPC Decoding Schedule

Decoding of sum-product algorithm was based on flooding algorithm where in one iteration, all check nodes compute check-to-variable messages and send the results to their neighbouring variable nodes. Then, all variable nodes compute variable-to-check message and send them to their neighbouring check nodes.

Flexible message scheduling was then introduced which is called layered decoding. Layered decoding is done by dividing parity check matrix into L layers and decoding is done row by row and this is called row-layered scheduling investigated in [10], [7] or column by column which is column-layered scheduling investigated in [15]. Layered message scheduling is able to significantly speed up the decoding convergence as each layer passes already updated messages to next layer [10]. Due to its fast convergence speed and implementation regularity , layered message passing has attracted extensive attention for VLSI implementation and it is widely utilized in recent LDPC decoder designs [13].

3. LDPC Decoder Architecture

Decoder architecture must be designed flexible enough to support different code rates and codeword sizes. In general, higher decoder parallelism results in large area and capacitance but low operating frequency and high routing congestion. Fully parallel LDPC decoders maps each CN and VN in tanner graph to a single processing element. Total number of CNs is $M \times W_r$ and number of VNs is $N \times W_c$ while partially parallel LDPC decoders divides the parity check matrix into less number of CNs and VNs such that a set of CN and VN updates can be done at a time. On the one hand, the advantage of high parallel decoders is increased throughput and energy efficiency defined as energy consumed to decode one codeword compared to other less parallel architectures due to fewer clock cycles per iteration [3]. On the other hand, less parallel decoders achieve lower throughput and provide less hardware area and less routing congestion problems also they provide shorter wires than high parallel architectures.

For Layered decoding, layers are processed sequentially since each layer depends on output messages of upper layer. Dependencies between layers impose less parallelism flexibility for layered decoders compared to non-layered ones at which maximum parallelism can be achieved by processing a whole macro layer at a time since all check equations in a macro layer are non-overlapped and independent. Processing of a macro layer can be made more parallel by processing more than one sub-matrix at a time which indicates more variable node processing elements.

Check node parallelism level of $P = 81$ was selected which indicates the maximum number of check equations processed at a time. This size P was chosen from the maximum sub-matrix size presented in the 802.11n standard. Each CN can process various number of input soft bits which indicates number of sub-matrices that can be processed per clock cycle. Each sub-matrix of the simultaneously processed sub-matrices requires its own permutation network. While increasing number of processed sub-matrices per clock increases decoder's complexity and degrades frequency, it increases throughput by reducing number of clock cycles per iteration. To allow flexibility for processing different codeword sizes, CNs are grouped into three groups each group contains 27 CNs which corresponds to minimum sub-block size. For codeword size 648, only first CN group will be active and others will be idle, these idle groups can be utilized to increase throughput by accepting other codewords while processing previous one and this is valid for this codeword size only. Also power reduction can be

done using clock-gating by shutting down idle CN groups in codeword sizes 648 and 1296.

Figure 3 shows overall architecture for proposed decoder. In order to reduce decoder complexity hardware implemented and routing overhead, CNs are designed to process one input at a time. Serialization in CN processing degrades throughput as processing time for one layer will take number of clock cycles equal to number of non-zero sub-matrices per layer but this is still a reasonable trade-off as throughput requirements for decoder can be achieved at a reasonable frequency. CN block functionality will be further illustrated in section 3.1. VNs are used to compute equations 4 and 5. To reduce routing and calculation overhead of VNs, equation 5 can be simply calculated by adding input soft bits of CN block with current CN output value while exclusion of CN previous value that is in equation 4 is done by subtraction of CN previous output value which is stored in internal RAM from current input within CN block . This concept for variable node calculation was introduced in different decoders as in [1].

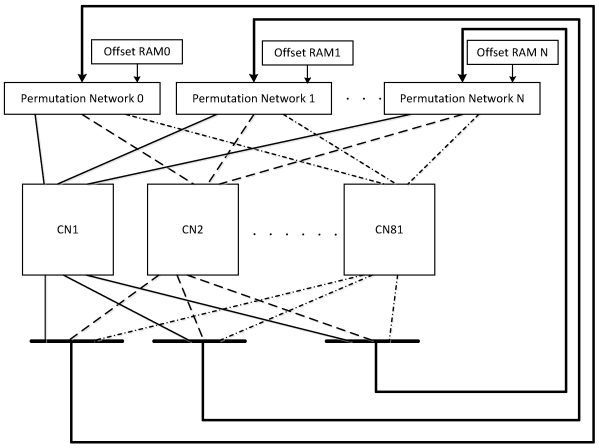

Fig. 3: Overall architecture for partially parallel LDPC decoder

3.1 Check Node Block

Check node block shown in figure 4 is considered the core processor of the decoder. In order to lower number of clock cycles per iteration, layer processing is done in a pipelined fashion inside the CN block. CN performs three main operations divided into three pipeline stages as shown in figure 5. First stage is the marginalization of reliability bits stage. Marginalization is used to do the exclusion of CN's own message resulted from previous iteration from input reliability bits which resembles $\alpha_{i'j}$ in equation 4. Second pipeline stage is detecting first and second minimums of input soft bits and calculating sign bit multiplication which is done using simple XOR gate. Finally, CN to VN message is calculated according to equation 7.

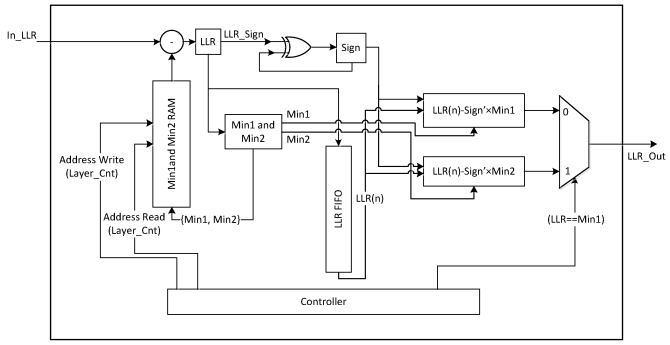

Fig. 4: Check Node block architecture

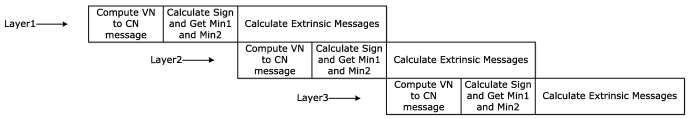

Fig. 5: Check Node pipeline stages

3.2 Permutation Network

For LDPC decoder design, implementation of interconnection between check and variable nodes in the Tanner Graph of the code is always a critical problem, especially in case of high parallelism decoder. A reconfigurable QC-LDPC decoder requires a programmable shift network for different sub-matrix sizes, code rates, and block lengths. Though, an architecture that minimizes the interconnect complexity and, hence, the interconnect delay is more desirable.

Connectivity network between CNs and VNs can be implemented using different scenarios. In [1], only one network was used for routing of messages from VNs to CNs while the route back from CNs to VNs is done by storing these messages inside a RAM and circulation can be done by controlling sequence of memory reads. In [2], [14], one permutation network was used for routing of VNs to CNs messages and another one for reverse direction.

A simple logarithmic barrel shifter composed of modular cells which provide wrap-arounds on 27, 54 and 81 positions for each supported block size was utilized for both routing directions. Since the outputs of CNs are already updated messages, we propose they can be appended directly to their inputs. For the first layer in the first iteration, reliability bits are passed from the wireless channel to the decoder's input directly to permutation network which passes these messages to CNs input permuted by offsets presented at first layer. The output of CNs is then appended back to permutation network which must restore original order of the messages and then perform permutations of layer 2, this is simply

done by rotating these messages by the difference of rotation value of previous layer with respect to the current one. This approach of using one permutation network instead of two reduced routing resources by half also it reduced memory bits required compared to implementation in [1].

4. Results

Figure 6 shows the bit error rate (BER) curves for all code rates with codeword size 648 and six iterations. Due to fixed quantization with 6-bits for internal soft bits inside CNs, a small quantization loss (<0.2dB) can be observed compared to floating point performance for lower rate codes while higher rate codes do not show any significant loss.

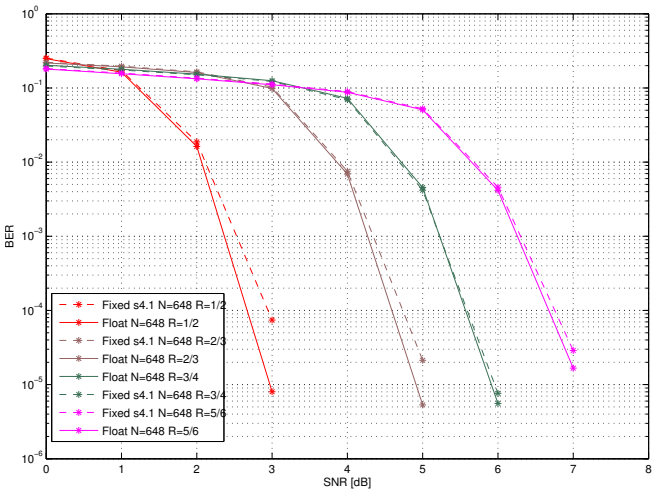

Fig. 6: Fixed-point vs. Floating point simulations for codeword 648-bits

Two designs were implemented in order to verify resources reduction and power optimization of using one permutation network using Xilinx Virtex-6 FPGA. First design uses only one permutation network for both routing directions, second design was implemented with two routing networks same as in [2], [14]. As shown in table 2, logic slices and LUT utilization is reduced by 5% of total FPGA resources. Also as can be seen utilizing one switching network reduced overall switching resources and wirings inside the decoder which reduced dynamic power consumption by 19% and reduced maximum path delay by 1.3ns. Throughput achieved can be calculated as in equation 9.

$$Throughput = \frac{F \times N \times R}{Iter. \times L \times Clk_{layer}} \quad (9)$$

Where F is the achieved frequency , $Iter.$ is number of iterations, L is number of layers and Clk_{layer} is number of clocks per layer which depends on pipeline overhead which is 5 clock cycles (constant for all rates)in addition to number of non-zero H-matrices per layer.

Table 2: FPGA implementation results of the multi-rate decoder, Device Xilinx Virtex 6 (xc6vlx240t-3-ff1156)

	Design 1	Design 2	Savings
Slices	13,229(35%)	15,322(40%)	5%
LUT	35,668(23%)	42,572(28%)	5%
Dyn. Power(mW)	1066.70	1324.22	19%
Frequency(MHz)	100	88	12%
Throughput(Mbps)	$37.5 \sim 281.25$	$33 \sim 247$	12%

5. Conclusions

A fully pipelined multi-rate, multi-codes low complexity QC-LDPC decoder was implemented for the 802.11n standard. Interconnection complexity was greatly reduced by implementing reusable permutation network for forward and backward message routing. Decoder was implemented on Xilinx Virtex-6 FPGA and achieved 19% reduction in dynamic power, 5% reduction in resources utilized and 12% increase in throughput compared to architecture designed with same routing approach in [2], [14]. We conclude that the engineering of the permutation network has a very significant effect on the LDPC implementation results without compromising its BER. This effect will be more significant with future processes.

Future work will explore a more parallel decoder to achieve higher throughput and the effect of wiring congestion and high switching activity will be further investigated and optimized to achieve best power consumption.

References

[1] T. Brack, M. Alles, F. Kienle, and N. Wehn. A synthesizable ip core for wimax 802.16e ldpc code decoding. In *Personal, Indoor and Mobile Radio Communications, 2006 IEEE 17th International Symposium on*, pages 1–5. IEEE, 2006.

[2] Z. Chen, X. Peng, X. Zhao, Q. Xie, L. Okamura, D. Zhou, and S. Goto. A macro-layer level fully parallel layered ldpc decoder soc for ieee 802.15. 3c application. In *VLSI Design, Automation and Test (VLSI-DAT), 2011 International Symposium on*, pages 1–4. IEEE, 2011.

[3] A. Darabiha, A. Chan Carusone, and F. Kschischang. Power reduction techniques for ldpc decoders. *Solid-State Circuits, IEEE Journal of*, 43(8):1835–1845, 2008.

[4] M. Fossorier, M. Mihaljevic, and H. Imai. Reduced complexity iterative decoding of low-density parity check codes based on belief propagation. *Communications, IEEE Transactions on*, 47(5):673–680, 1999.

[5] R. Gallager. Low-density parity-check codes. *Information Theory, IRE Transactions on*, 8(1):21–28, 1962.

[6] J. Hagenauer, E. Offer, and L. Papke. Iterative decoding of binary block and convolutional codes. *Information Theory, IEEE Transactions on*, 42(2):429–445, 1996.

[7] D. Hocevar. A reduced complexity decoder architecture via layered decoding of ldpc codes. In *Signal Processing Systems, 2004. SIPS 2004. IEEE Workshop on*, pages 107–112. IEEE, 2004.

[8] D. MacKay. Good error-correcting codes based on very sparse matrices. *Information Theory, IEEE Transactions on*, 45(2):399–431, 1999.

[9] D. MacKay and R. Neal. Near shannon limit performance of low density parity check codes. *Electronics letters*, 32(18):1645, 1996.

[10] M. Mansour and N. Shanbhag. A 640-mb/s 2048-bit programmable ldpc decoder chip. *Solid-State Circuits, IEEE Journal of*, 41(3):684–698, 2006.

[11] T. Mohsenin and B. Baas. Trends and challenges in ldpc hardware decoders. In *Signals, Systems and Computers, 2009 Conference Record of the Forty-Third Asilomar Conference on*, pages 1273–1277. IEEE, 2009.

[12] E. Sharon, S. Litsyn, and J. Goldberger. An efficient message-passing schedule for ldpc decoding. In *Electrical and Electronics Engineers in Israel, 2004. Proceedings. 2004 23rd IEEE Convention of*, pages 223–226. IEEE, 2004.

[13] Z. Wang, Z. Cui, and J. Sha. Vlsi design for low-density parity-check code decoding. *Circuits and Systems Magazine, IEEE*, 11(1):52–69, 2011.

[14] M. Weiner, B. Nikolic, and Z. Zhang. Ldpc decoder architecture for high-data rate personal-area networks. In *Circuits and Systems (ISCAS), 2011 IEEE International Symposium on*, pages 1784–1787. IEEE, 2011.

[15] J. Zhang and M. Fossorier. Shuffled iterative decoding. *Communications, IEEE Transactions on*, 53(2):209–213, 2005.