Efficient Construction and Implementation of Short LDPC Codes for Wireless Sensor Networks

James McDonagh*, Massimiliano Sala†, Antoin O'hÁllmhurain*,Vaibhav Katewa‡ and Emanuel Popovici*
*Department of Microelectronic Engineering, University College Cork, Ireland
†Boole Centre for Research in Informatics, University College Cork, Ireland
‡Department of Electrical Engineering, Indian Institute of Technology, Kanpur, India
Email: james.mcdonagh@ue.ucc.ie, msala@bcri.ucc.ie, anthony.ohalloran@ue.ucc.ie, e.popovici@ucc.ie, vkatewa@iitk.ac.in

***Abstract*—Wireless sensor networks gained a lot of attention in recent years due to their widespread applications. Reliability of data communication and power saving are paramount for applications which use wireless sensor network technology. We propose two classes of short quasi-cyclic LDPC codes suitable for implementation on a resource constrained system. The codes we propose are easy to encode and their decoding performance compares well with random LDPC codes with the same parameters. We implement our codes on a 25mm mote platform provided by Tyndall and compare them with Viterbi coding schemes.**

I. Introduction

Wireless Sensor Networks gained a lot of attention in recent years due to their applications, ranging from automotive, avionics, agriculture, security and medicine. The sensor mote platforms are characterized by heavily constrained resource usage such as memory, processing power and cost. Many applications for sensor networks require that data have to be sent reliably over the wireless channel. The reliability of communication can be dramatically improved by using efficient block coding. Traditionally, Viterbi coding schemes are used on this kind of applications due to their relative simplicity of encoding and decoding. A viable alternative would be the celebrated LDPC codes [1]. LDPC codes can be decoded either with hard-decision or with soft-decision algorithms. When decoded with soft-decision algorithms, they are known to outperform equivalent Viterbi coding systems. However, the best-performing LDPC codes are constructed via random-walk processes, which are time and memory consuming. Furthermore, a random LDPC code has a computationally costly encoding.

In this paper we propose two families of short LDPC codes that do not suffer from the limitations of random LDPC codes, but which have a similar performance. Moreover, these families enjoy an algebraic structure that allows an efficient hardware/software implementation on resource constrained systems. Also, when decoded via soft-decoding, their performance is superior to an equivalent Viterbi scheme.

A. LDPC Codes

Any parity-check matrix H for a binary linear block code can be represented graphically, with a graph called a Tanner Graph [3]. The Tanner graph is formed by two disjoint subsets of nodes, called **check nodes** and **bit nodes**, and interconnections placed between them. Each bit node represents a column in the H matrix. The interconnections represent the ones in H, in such a way that check node i is connected to bit node j if and only if entry $h_{i,j} = 1$. To decode, any bit node is assigned a probability, based on the information received from the channel. The information is passed and updated from the bit nodes to the check nodes, and vice-versa (via the connections) by way of a belief-propagation algorithm [5].

A low-density parity-check (LDPC) code is a linear block code where the parity-check matrix is sparsely populated with non-zero entries. A regular LDPC code has a constant amount of non-zero entries per row and a constant amount per column. A parity-check matrix of an LDPC code has very few connections. This benefits decoding since each node has less information coming in, calculations are quicker, and the lack of connections means less complex hardware to implement.

LDPC codes have superior decoding performance with respect to other codes, however there is a trade-off since their generator matrix is densely populated with non-zero entries. Generally speaking, LDPC codes have an inefficient encoding system for both hardware and software implementations.

The decoding performance of LDPC codes is strongly related to the length of the cycles present in the corresponding Tanner graph. The length of the shortest cycle that can be found is called the **girth** of the code. Since short cycles heavily hinder the decoding performance of the LDPC code, the girth becomes a key parameters in describing the efficiency of the decoding process.

B. Quasi-Cyclic LDPC Codes

Quasi-Cyclic LDPC codes form a large class of codes, which have been extensively studied and which possess a nice encoding and decoding, in the sense that their hardware is both cheap and easy to fabricate. This structure has many benefits for hardware and software implementations, due to reduced memory requirements in both decoding and encoding. The memory benefit is provided by being able to describe the matrices using a series of short polynomials.

A **circulant** matrix is a square matrix, such that each row is obtained by a cyclic shift of the previous. The parity-check matrix of a QC code is broken up into $m \times m$ blocks of circulant matrices, so that its dimensions are $\alpha m \times \beta m$, where β is the number of circulants in a (submatrix) column and α is the number of circulants in a (submatrix) row. When creating

1-4244-1342-7/07/$25.00 ©2007 IEEE

a QC-LDPC matrix there are four types of circulant that can be used: a weight $t \geq 3$ circulant, a weight-two circulant, a weight-one circulant and a zero circulant. Since any circulant with weight $t \geq 3$ contains internal cycles of length six, they are not used and in the sequel we will consider only identities (with weight one) and weight-2 circulants.

Any circulant matrix can be described in polynomial form, by denoting the positions of the 1's in the first row of the circulant (the first row clearly defines the whole matrix). In the case when the weight is 2, the polynomial form of a circulant is:

$$P(x) = x^a + x^b .$$

When creating our parity-check matrices we need to choose the values a and b in order to remove short cycles and hence to make the decoding as efficient as possible. This is accomplished by identifying a list of rules. These rules involve two variables $S(p)$ and $\epsilon(p)$, which are immediately computed from the exponents a, b, and the circulant dimension m ([6]). The **separation** $S(p)$ is defined by $S(p) = \min\{(b-a), m-(b-a)\}$. The value $\epsilon(p)$ is used in some logical statements to express simultaneously a and b. To be more precise, let f be any function $f : \mathbb{N} \longmapsto \mathbb{Z}$, let $\equiv$ denote equivalence modulo m, then we write

$$f(\epsilon(p)) \equiv l \iff f(a) \equiv l \text{ or } \mathrm{f(b)} \equiv \mathrm{l}$$

and also

$$f(\epsilon(p)) \not\equiv l \iff f(a) \not\equiv l \text{ and } \mathrm{f(b)} \not\equiv \mathrm{l}$$

Letting $p_1(x) = x^{a_1} + x^{b_1}$, $p_2(x) = x^{a_2} + x^{b_2}$, we also write

$$f(\epsilon(p_1), \epsilon(p_2)) \equiv l$$

meaning

$$f(a_1, a_2) \equiv l \text{ or } \mathrm{f(a_1, b_2)} \equiv \mathrm{l} \text{ or } \mathrm{f(b_1, a_2)} \equiv \mathrm{l} \text{ or } \mathrm{f(b_1, b_2)} \equiv \mathrm{l}\,.$$

Similarly, we write

$$f(\epsilon(p_1), \epsilon(p_2)) \not\equiv l$$

for

$$f(a_1, a_2) \not\equiv l \text{ and } \mathrm{f(a_1, b_2)} \not\equiv \mathrm{l} \text{ and } \mathrm{f(b_1, a_2)} \not\equiv \mathrm{l} \text{ and } \mathrm{f(b_1, b_2)} \not\equiv \mathrm{l}\,.$$

For example, a circulant matrix may have a polynomial form $P(x) = x^1 + x^4$, so that $a = 1$ and $b = 4$. The matrix dimension may be $m = 7$ and in this case the separation is $S(p) = 3$ ($\epsilon(p) = 1, 4$).

C. Proposed code constructions

We propose two new families of QC LDPC codes, as follows. The first is constituted by 0.5 rate codes, as in Fig. 1. The second is constituted by 0.66 rate codes, as in Fig. 2. Both are similar to the Bresnan codes [4], but slightly denser and more flexible ([6]). Any parity-check matrix has 5 non-zero entries per column, but the 1/2 rate matrix has 10 non-zero entries per row and the 2/3 rate matrix has 15 non-zero entries per row.

$$\left[\begin{array}{cccc|cccc} I & 0 & H_6 & H_7 & I & H_{12} & 0 & H_{15} \\ H_1 & I & 0 & H_8 & H_9 & I & H_{14} & 0 \\ H_2 & H_3 & I & 0 & 0 & H_{11} & I & H_{16} \\ 0 & H_4 & H_5 & I & H_{10} & 0 & H_{13} & I \end{array}\right]$$

Fig. 1. A 1/2 rate parity-check matrix using the proposed construction

II. Theoretical and Simulation Results

We have formally identified conditions on the parity-check matrix to ensure girth at least 6. The rules to describe these conditions are divided into categories, depending on the placement of the circulant in relation to the others. As for the Bresnan codes, the following rules involve either $S(p)$ or $\epsilon(p)$.

Individual circulants

This rule is applied to all circulants regardless of their position: no circulant can have a separation equal to half its dimension.

$$S(p) \neq \frac{m}{2}$$

Circulants on the same Row or Column

This rule applies to any two circulants on the same row or column: their separations cannot be equal.

$$S(p) \neq S(p')$$

Circulants related Diagonally

These rules involve four circulants, which may include identities, forming the corners of a 'square' or 'rectangle'.

- Four weight-two circulants.

$$\left[\begin{array}{ccc} H_{i,j} & \cdots & H_{i,j+y} \\ \vdots & & \vdots \\ H_{i+x,j} & \cdots & H_{i+x,j+y} \end{array}\right]$$

$$\epsilon(p_{i,j}) - \epsilon(p_{i,j+y}) + \epsilon(p_{i+x,j+y}) - \epsilon(p_{i+x,j}) \equiv 0$$

- Two identities *not* on the same row or column.

$$\left[\begin{array}{ccc} H_{i,j} & \cdots & I_{i,j+y} \\ \vdots & & \vdots \\ I_{i+x,j} & \cdots & H_{i+x,j+y} \end{array}\right]$$

$$\epsilon(p_{i,j}) + \epsilon(p_{i+x,j+y}) \equiv 0$$

- Two identities on the same row or column.

$$\left[\begin{array}{ccc} H_{i,j} & \cdots & I_{i,j+y} \\ \vdots & & \vdots \\ H_{i+x,j} & \cdots & I_{i+x,j+y} \end{array}\right]$$

$$\epsilon(p_{i,j}) - \epsilon(p_{i,j+y}) \equiv 0$$

- One identity.

$$\left[\begin{array}{ccc} H_{i,j} & \cdots & H_{i,j+y} \\ \vdots & & \vdots \\ I_{i+x,j} & \cdots & H_{i+x,j+y} \end{array}\right]$$

$$\epsilon(p_{i,j}) + \epsilon(p_{i+x,j+y}) - \epsilon(p_{i+x,j}) \equiv 0$$

The following graphs show the AWGN simulation results of two code groups, each containing 25 QC-LDPC codes and 5 random codes. The codes in the first group(Fig. 3, Fig. 5)

$$\begin{bmatrix} I & 0 & H_6 & H_7 & I & H_{12} & 0 & H_{15} & I & H_{20} & H_{21} & 0 \\ H_1 & I & 0 & H_8 & H_9 & I & H_{14} & 0 & 0 & I & H_{22} & H_{23} \\ H_2 & H_3 & I & 0 & 0 & H_{11} & I & H_{16} & H_{17} & 0 & I & H_{24} \\ 0 & H_4 & H_5 & I & H_{10} & 0 & H_{13} & I & H_{18} & H_{19} & 0 & I \end{bmatrix}$$

Fig. 2. A 2/3 rate parity-check matrix using the proposed construction

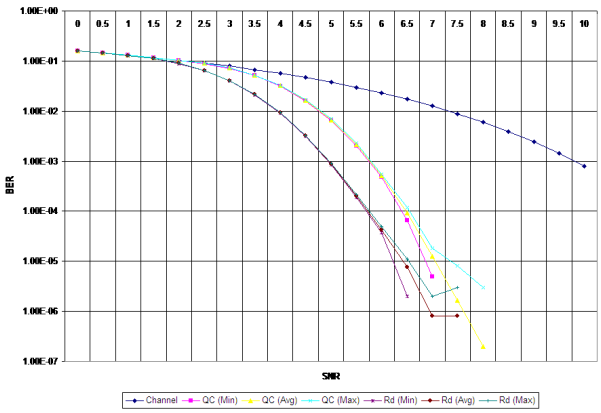

Fig. 3. Results for 1/2 LDPC rate using soft decoding

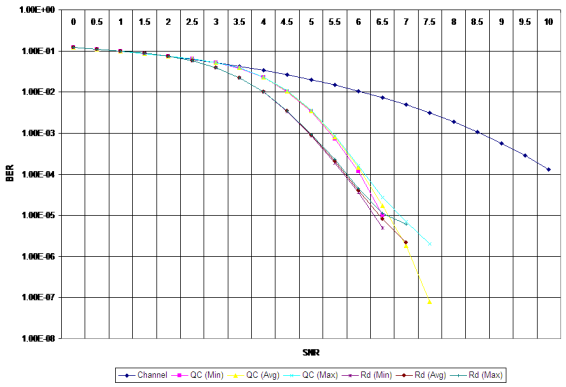

Fig. 4. Results for 2/3 rate LDPC using soft decoding

have rate of 1/2, the others(Fig. 4, Fig. 6) have a rate of 2/3, but all have girth at least 6. Also, the code length for the 1/2 rate is 192 bits and 288 bits for the 2/3 rate code. The algorithms used to generate the random matrix were "optimal permutation" for the 1/2 rate and "McKay 2B" for the 2/3 rate. The QC-LDPC codes were generated one circulant at a time, the first being completely random and the following according to the presented rules.

The first two graphs(Fig. 3 and Fig. 4) show the results for the 'Soft' decoding and the final two(Fig. 5 and Fig. 6) show the results for the 'Hard' decoding. For each simulation the SNR range was 0 - 10db for soft decoding and 0-9dB for hard decoding. The soft decoding simulations had an SNR increment of 0.5dB while the hard decoding simulations had an increment of 1dB. At each value of SNR 50,000 simulations were performed and an average recorded (while decoding the maximum allowed number of passes was 40). Each of the graphs has seven trend lines, one trend line represents the channel or undecoded data, the remaining six are placed into two set of three. One set represents the simulated random code while the other set represents the simulated QC-LDPC code. Each trend line in both sets represents the best, worst and average performance. Figure 3 shows that random codes outperform the QC-LDPC codes by roughly $0.5 - 1$dB when using soft decoding at a rate of 1/2. Simulations for both random and QC-LDPC start to show different performances at roughly 6dB in SNR. Signals with SNR of 8dB and above can be fully recovered using both codes and 7.5dB and above for random.

Figure 4 reveals random outperforming QC-LDPC by approximately 0.5dB with using soft decoding and a rate of 2/3. The difference in performance start to show at roughly 5dB but are not significant until about 6dB for both random and QC. Signals can be fully recovered for both at 7.5dB and above, and 7dB for the random only.

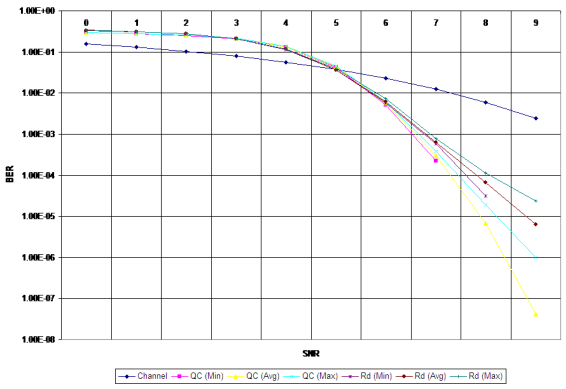

Fig. 5. Results for 1/2 rate LDPC using hard decoding

Figure 5 shows QC-LDPC outperforming random by 0.5-1dB when hard decoding is used and the rate is 1/2. Differences between the decoding for both become apparent at 6dB but not significantly until roughly 7.5dB. Due to 9dB being the maximum SNR, we cannot tell for sure where error free decoding begins on average, however best case is 7dB and 8dB for QC and random, respectively.

Figure 6 also demonstrates QC-LDPC outperforming random at 2/3 rate, while using hard decoding. Compared with 1/2 rate, the margin of difference is significantly larger, which can be over 1dB. Like in the rate-$1/2$ case, for signals with low SNR errors are induced rather the recovered, although unlike

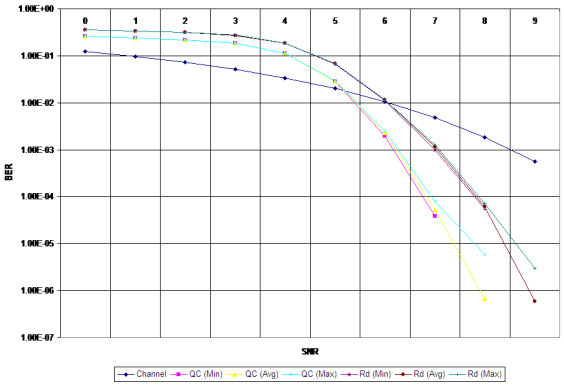

Fig. 6. Results for 2/3 rate LDPC using hard decoding

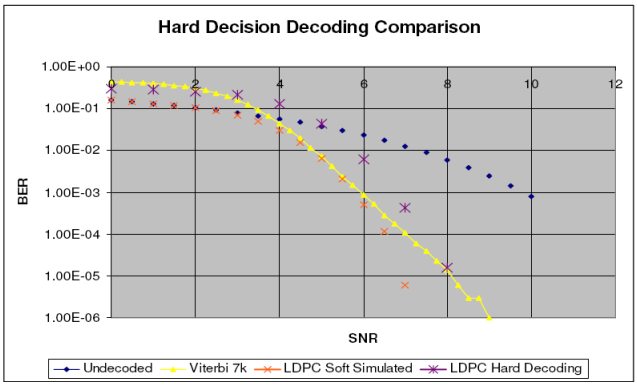

Fig. 7. Viterbi Vs. LDPC at 2/3 rate

1/2 rate the QC-LDPC starts recovering errors at a lower SNR compared to random. Also unlike 1/2 rate, which has both code families performing relatively equally until 6dB, at 2/3 rate the QC-LDPC code outperforms the random code at all values of SNR.

III. Implementation Results and Conclusion

Implementation of the codes was done on the Tyndall 25mm mote [2]. The mote imposes some constraints on the codes to become implementable. One of these constraints was that soft information about the received symbols from the channel is not readily available to the decoder implemented on the 8 bit microcontroller. This forced us to look at various different solutions to the decoding problem.

The three codes that we decided to investigate were:

1) Hard Decision LDPC: This uses Gallagher's Hard Decision Decoding algorithm, which although fairly simple, it is only effective at rates far below the capacity of the channel. If the parity-check sets are small, then the decoding process is reasonable.
2) Soft Simulated LDPC: This algorithm uses the standard LDPC evaluation platform, with some minor modifications. The main modification is that, rather to input the floating point representation of the received symbols into the decoder, to use hard information. In this case, the probabilities of 0.1 and 0.9 are passed to the decoder, when a 0 and a 1 are received, respectively.
3) Hard Viterbi: For the purposes of this comparison, the platform for evaluating the Viterbi Hard Decoding algorithm used a constraint length of 7. In the previous version, this would have entailed increasing the look-up tables (26 states in decoder trellis) by a large margin, cluttering up and slowing down the application. Instead, shift registers were used to manage both addressing for connections between nodes and state transitions between nodes, when finding the Most Likely Path during the Traceback process.

Simulations were carried out in software before a decision was made on which code to implement in hardware. Figure 7 shows a comparison between a hard LDPC, hard Viterbi and soft LDPC. Due to the limited available memory on the mote, it is impossible to store the whole of either of the G and H matrices in the flash memory. The solution to this is to generate the rows and columns dynamically. To generate any row of these matrices, all we need to store is simply the position of the 1s in the first row of each circulant. Then, we just iterate through the rows, right shifting any circulant one bit at a time. The matrix columns can be directly recovered from the rows.

Another limitation of our setup is the maximum data packet that can be transmitted by the transceiver, which is only 24 Bytes long. Thus, only codes of very short length can be used. We implemented a length 192 code on the platform and compared it with a punctured Viterbi code with similar parameters. The mesured results show that LDPC decoding consumes 307mW while the Viterbi decoding consumes 485mW. The encoding of quasi-cyclic LDPC codes compares well with the encoding of Viterbi codes and consumes 260mW. These results show that it is feasible to implement short QC-LDPC codes with good performance on resource constrained sensor networks platforms.

Acknowledgment

The authors would like to thank Enterprise Ireland for financial support to carry out this work. The second author has been also partially supported by ST Microelectronics.

References

[1] M. Sartipi and F. Fekri. Source and channel coding in wireless sensor networks using ldpc codes. *IEEE Proc, IEEE SECON 2004*, pages 309 – 316, 2004.
[2] J. Barton, B. O'Flynn, S. Bellis, A. Lynch, M. Morris, and C. O'Mathuna. A miniaturised modular platform for wireless sensor networks. *IEEE Proc. of ECCTD*, pages 35–38, 2005.
[3] R. M. Tanner. A recursive approach to low complexity codes. *IEEE Trans. Inform. Theory*, 27(5):533–547, 1981.
[4] R. Bresnan. Novel code construction and decoding techniques for LDPC codes. *MEngSc thesis, UCC*, 2004.
[5] R. Gallager. *Low-Density Parity-Check Codes*. PhD thesis, Massachusetts Institute of Technology, 1963.
[6] M. Rossi and M. Sala. On a class of quasi-cyclic codes. Technical Report 50, University College Cork, Cork, Ireland, 2005.