Lines that will appear in the proceedings

Lines that will appear in the program

5C-4 Design methodology for IRA Codes
F. Kienle, N. Wehn (Univ. of Kaiserslautern, Germany)

Design methodology for IRA Codes

Frank Kienle, Norbert Wehn
Microelectronic Systems Design Research Group
University of Kaiserslautern
Erwin-Schrödinger-Straße
67663 Kaiserslautern, Germany
{kienle, wehn}@eit.uni-kl.de

Abstract— Channel coding is an important building block in communication systems since it ensures the quality of service. Irregular repeat-accumulate (IRA) codes belong to the class of Low-Density Parity-Ceck (LDPC) codes and even outperform the recently introduced Turbo-Codes of current communication standards. IRA codes can be represented by a Tanner graph with arbitrary connections between nodes of given degrees. The implementation complexity of an IRA decoders is dominated by the randomness of these connections.

In this paper we present for the first time an IRA decoder architecture which can process any given IRA code. We developed a joint graph-decoder design methodology to construct the Tanner graph of a given IRA code which can be efficiently processed by this decoder architecture without any RAM access conflicts. We show that these constructed IRA codes can outperform the UMTS Turbo-Codes.

I. Introduction

Sophisticated channel coding becomes an increasingly important building block of communication systems. Current communication standards already feature Turbo-Codes while for future standards the decision whether to use Turbo- or Low-Density Parity-Check (LDPC) Codes is still open. Implementation issues that determine how expensive it will be to deploy the channel coding schemes will have a major impact on the decision of standardization committees.

LDPC codes were introduced by Gallager 1963 [1] and were re-discovered in 1996 by MacKay and Neal [5]. With the extension to so called irregular LDPC codes [4], these codes closely approach the Shannon limit and are thus the best known codes. The major drawback of LDPC codes is the encoding complexity. Even though an efficient *encoding* algorithm exists [7], its *hardware* realization is still an obstacle.

The irregular repeat-accumulate (IRA) codes were introduced in 2000 by Khandekar and McEliece [2]. These codes have a linear-time encoding complexity with a straight forward hardware realization, and outperform the currently deployed Turbo-Codes. Like LDPC codes, IRA codes can be represented by a Tanner graph with arbitrary connections between nodes. An IRA code is defined by the degree distribution of the nodes.

A fully parallel architecture corresponds to a one to one instantiation of the Tanner graph in hardware. However for large blocksizes this becomes infeasible due to wire routing problems. Therefore partly parallel architectures become mandatory. Their complexity strongly depends on the randomness of the Tanner graph. Regularity in the Tanner graph simplifies the implementation but degrades the communications performance.

In this paper we present for the first time an IRA decoder architecture which can process any IRA code. One important building block of the decoder architecture is the permutation network which realizes the 'arbitrary connection' between the two types of nodes in the graph. Therefore the underlying Tanner graph of an IRA code should be designed under the constraints of a dedicated permutation network.

We present a three step design methodology based on a **joint graph - decoder** design:

1. Find an architecture template which can process all IRA code ensembles within the defined code parameters.
2. For a given permutation network define the constraints for the Tanner graph.
3. Find an IRA code within these constraints with a good communications performance.

We show that IRA codes designed with this methodology can outperform Turbo-Codes by 0.2dB.

The paper is structured as follows. IRA codes are explained in Section II. In Section III a general IRA decoder architecture is presented. The design method is explained in Section IV. Section V gives some results and Section VI concludes the paper.

II. IRA Codes

An IRA code [2] can be represented by a Tanner graph (Fig. 1), with N variable nodes (open circles) and M check nodes (filled squares). The variable nodes (VN) can be partitioned in $K = N - M$ information nodes (IN) and M parity nodes (PN). The INs are of varying degree. f_i denotes the fraction of INs with degree i, with $\sum_i f_i = 1$. Each check node is connected to a information nodes. These $a * M$ connections between CN and IN are 'arbitrary permutations' (Π_1). The check nodes are furthermore connected to parity nodes in

0-7803-8175-0/04/$17.00

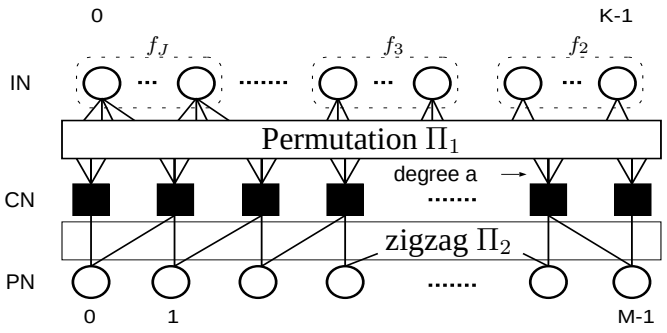

Fig. 1. Tanner graph for IRA code, with IN (Information Node), CN (Check Node), PN (Parity Node) and f_i (fraction of nodes with degree i)

a fixed zigzag pattern (Π_2). The code can be described by the degree distribution $\mathbf{f} = (f_2, f_3, \ldots, f_J)$ and a.

Only systematic codes are treated here, with the codeword $(\vec{u}, \vec{p})$ of length $N = K + M$. The information sequence $\vec{u} = (u_0, \ldots, u_{K-1})$ is associated with the information nodes $(IN_0, \ldots, IN_{K-1})$ and each parity bit $\vec{p} = (p_0, \ldots, p_{M-1})$ is associated with one of the parity nodes $(PN_0, \ldots, PN_{M-1})$. The resulting code rate is $R = \frac{a}{a+\sum_i if_i}$.

A. Encoding

The Encoder, shown in Fig. 2, can be directly derived from the graph. The information sequence is first passed through a repetition unit. The repetition pattern follows the given degree distribution $\mathbf{f}$. The highest degree comes first, i.e. u_0 is repeated J times, u_{K-1} two times.

The expanded information sequence is interleaved by Π_1. a values of the interleaved sequence participate in one parity check, which means a values are replaced by their binary sum (modulo 2). The resulting sequence is passed through an accumulator which leads to the zigzag connections between PN and ZN. $\vec{u}$ and $\vec{p}$ are transmitted. It is evident, that this encoding scheme has linear complexity.

B. Decoding

IRA codes can be decoded using the message passing algorithm [1]. It exchanges soft-information iteratively between variable and check nodes. The update of the nodes can be done in a canonical scheduling [1]: In the first step all variable nodes must be updated and in the second step all check nodes. The processing of individual nodes within one step is independent, and can thus be parallelized.

The exchanged messages are assumed to be log-likelihood ratios (LLR) $\lambda = log(p(0)/p(1))$. Each variable node of degree i calculates an update of message k according to:

$$\lambda_k = \lambda_{ch} + \sum_{l=0, l\neq k}^{i-1} \lambda_l, \tag{1}$$

with λ_{ch} the corresponding channel LLR of the VN and λ_i the LLRs of the incident edges. The check node LLR updates are calculated with

$$\tanh(\lambda_k/2) = \prod_{l=0, l\neq k}^{i-1} \tanh(\lambda_l/2). \tag{2}$$

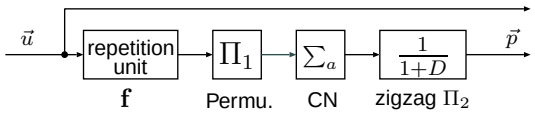

Fig. 2. Encoder for an IRA code

The message passing results in optimal decoding if the Tanner graph is cycle free [8]. However, for finite block sizes the Tanner graph will contain cycles. Once a message has completed a cycle, the node updates Equation 1 and 2 becomes suboptimal. Therefore, the longer the cycles the more you gain from an increased number of iterations. By selecting Π_1 for a given IRA code, the resulting graph should have cycles as long as possible. The shortest cycle is called girth, it should be > 4.

III. Decoder Architecture

A decoder architecture for an IRA code should allow a wide range of code parameters $(\mathbf{f}, a)$. For practical applications the blocksize should be limited due to latency issues, also it is reasonable to limit the highest degrees f_{max} and a_{max}. For a partly parallel approach a fixed number of functional units for variable and check nodes is instantiated.

Fig. 3 shows an architecture template for a parallization $P = 4$. It features 4 functional units for CNs, INs and PNs, the permutation network Π_1, the zigzag network Π_2 and sufficient memory banks for messages storage. The IN and PN processing can be done separately due to the disjunctive of Π_1 and Π_2, denoted as IN and PN branch in the following (dotted line in Fig. 3). Due to the simple structure of Π_2 we focus on the IN branch which has to realize 'arbitrary' connections between INs and CNs.

The IN, PN functional units are implemented using a single input and output. Thus all incoming edges of one node are processed sequentially. The units can easily be pipelined. Hence, these processing units are not in the critical path. All messages for a node must reside in the same memory at consecutive positions. Therefore, the correct sequence for message retrieval must be provided during storage. The costs to determine the right location for storage depends on the underlying Tanner graph of the code. These dependencies are explained in detail in Section IV.

The number of messages handled by the IN branch is $M \cdot a$ and by the PN branch $M \cdot 2$. With $a > 2$ the IN branch requires more cycles to process all data than the PN branch. The CN functional units have to process $M \cdot a + M \cdot 2$ messages. Each CN unit accepts two data in parallel, one from the IN and one from the PN branch. It has to be ensured that the messages provided by both branches belong to the same check node. The results of the CN nodes are passed back deinterleaved (Π_1^{-1},Π_2^{-1}) to the IN and PN branches.

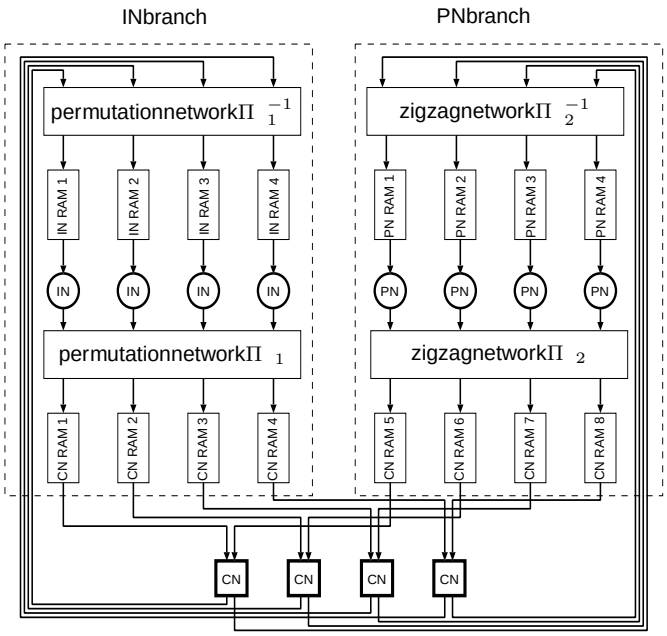

Fig. 3. Basic architecture of an IRA code decoder

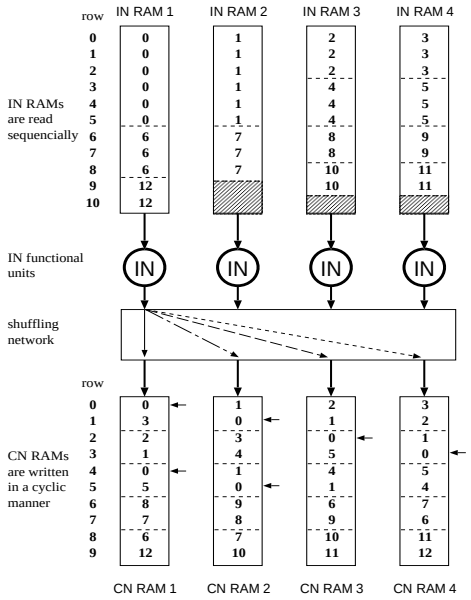

Fig. 4. Mapping of information nodes to the IN RAMs and the cycle write pattern to the CN RAMs

IV. Joint Graph-Decoder Design

The design of Π_1 is essential for code performance *and* architectural efficiency. A given permutation network defines constraints on the Tanner graph. If the permutation networks accepts any permutation, the decoder architecture can process any IRA code. However when accepting arbitrary permutations, concurrent accesses to the same memory can not be avoided. These access conflicts can be resolved at run-time using dedicated communication networks [3] or crossbars. But, for a high degree of parallization this solution is not feasible due to the wire routing complexity.

Providing only a set of permutations leeds to optimized networks, yielding an efficient hardware architecture. However, the randomness of the Tanner graph is restricted, hence the code performance can degrade. Thus, a trade off between code performance and optimized network design has to be found.

We propose a joint **graph-decoder** design method for IRA codes, which can be efficiently implemented on the partly parallel architecture of Section III. We define two architectural constraints on the structure of the Tanner graph: First, the resulting permutation network should be a simple shuffling network. Second, the collisions when accessing momories should be minimized.

Consider a code with a degree distribution $[f_6 = \frac{2}{13}, f_3 = \frac{6}{13}, f_2 = \frac{5}{13}]$ and $a = 2$. The code rate is $R \approx 0.4$ with $M = 20$ check nodes, $K = 13$ information nodes and 40 edges between IN and CN. An architecture parallization of $P = 4$ is assumed. The construction of Π_1 consists of five steps:

step 1: bin packing problem

Map the values associated with the edges of the INs on the IN RAMs (Fig. 4). For a node of degree j, j successive positions for the messages have to be allocated in one RAM. The IN RAMs are filled with a simple mapping scheme, starting with INs of highest degree. Target is that the RAMs are even filled.

step 2: ensure a shuffling network

Use a cyclic write sequence to the CN RAMs to ensure randomness of the message distribution. The $P = 4$ messages produced from the IN functional units can thus be written conflict free into distinct RAMs. In Fig. 4 the positions of the messages from IN_0 are marked by an arrow. Due to the different number of data in the IN RAMs, conflicts in memory access can occur within the last 2 cycles when writing back to the IN RAMs. But these write conflicts are deterministic and can be avoided through appropriate scheduling. The number of data per CN RAM must be a multiple of a, since a successive entries belong to one CN node (this is indicated by the dashed lines).

step 3: mapping of the check nodes

The messages associated with the CNs are mapped sequentially to the CN RAMs (Fig. 5). This mapping ensures that the PN branch can be processed without write conflicts with a deterministic addressing scheme. The addressing scheme of step 2 and 3 defines the communication between the nodes which are edges in the Tanner graph. However, the message passing algorithm on this graph (fixed by the shuffling network) will lead to bad communications performance, see Section IIB.

step 4: ensure communications performance

To enhance the randomness of the resulting graph a row permutations of the IN entries can be applied. This permutation comply with the architectural constraints of the previous steps. To ensure communications performance the resulting graph has to fulfill certain constraints. Remove double edges between nodese removed. Furthermore, two successive CN must not have connections to the same IN node, otherwise a

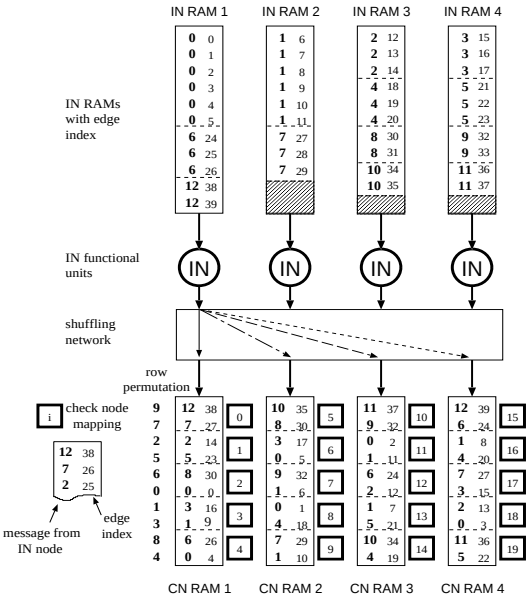

Fig. 5. Mapping of the check nodes to the CN RAMs and the resulting row permutation

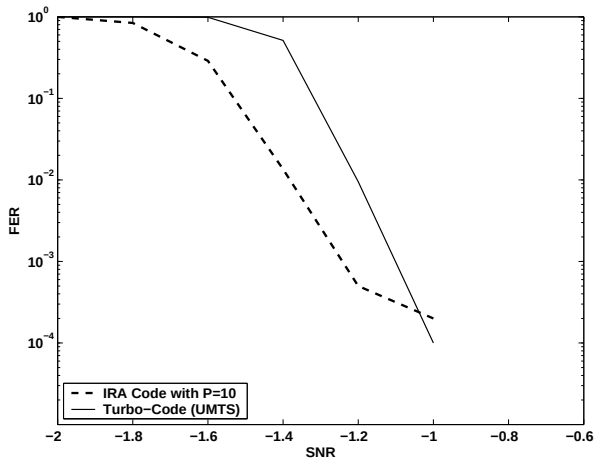

Fig. 6. Communications performance of IRA code in comparison with Turbo-Code, with FER (frame error rate) and SNR (signal to noise ratio)

cycle of length 4 exists, as the CNs are already connected by the zigzag network. The cycles of length 4 inside Π_1 must be eliminated. Heuristic methods are used to obtain a graph with a girth average as high as possible [6]. The resulting row permutation corresponds to the RAM addresses where the IN data are written in each cycle.

step 5: extract interleaver pattern Π_1

Each value in the IN RAMs is associated with a fixed edge index (Fig. 5). The edge index correspond to the position of the messages between the repetition unit of the encoder and the interleaver Π_1. The interleaved positions of the messages can be deduced from the resulting CN mapping. Starting with CN_0 to CN_{19} the corresponding edge indexes give the interleaver pattern.

V. Performance Results

To investigate communications performance an IRA code is constructed, with code parameters $f = (f_{13} = 0.183, f_{12} = 0.051, f_6 = 0.037, f_5 = 0.3, f_3 = 0.429)$ and $a = 3$, with a resulting code rate of $R = 1/3$. The IRA code is optimized for an architecture with a parallization of $P = 10$. Fig. 6 shows the communications performance of this IRA code in comparison with an UMTS compliant Turbo-Code. Both codes have an information blocksize of 5000 bits and the same code rate. The parallized IRA code outperform the Turbo-Code by 0.2 dB at a FER of 10^{-3}.

VI. Conclusions

IRA codes are candidates for future communication systems. The complexity of an IRA decoder depends on the randomness of the Tanner graph. We have presented an partly parallel decoder architecture which can process all IRA codes only restricted by the amount of the highest degree of a node. We have shown a **joint graph-decoder** design methodology which can be applied to IRA codes with any given degree distribution. The resulting IRA code can be efficiently implemented on the decoder architecture without any conflict in RAM accesses. An design example shows, that these IRA codes can outperform currently deployed Turbo-Codes by 0.2dB. This joint graph-decoder design can also be used for irregular LDPC codes with constant check node degree.

References

[1] R. G. Gallager. *Low-Density Parity-Check Codes*. M.I.T. Press, Cambridge,Massachusetts, 1963.

[2] H. Jin, A. Khandekar, and R. McEliece. Irregular Repeat-Accumulate Codes. In *Proc. 2nd International Symposium on Turbo Codes & Related Topics*, pages 1–8, Brest, France, Sept. 2000.

[3] F. Kienle, M. J. Thul, and N. Wehn. Implementation Issues of Scalable LDPC Decoders. In *Proc. 3rd International Symposium on Turbo Codes & Related Topics*, pages 291–294, Brest, France, Sept. 2003.

[4] M. Lubi, M. Mitzenmacher, A. Shokrollahi, and D. Spielmann. Analysis of low-density codes and improved designs using irregular graphs. In *Proc. 30th ACM Symp. on the Theory of Computing*, pages 249–258, 1998.

[5] D. MacKay and R. Neal. Near Shannon limit performance of Low-Density Parity-Check Codes. *Electronic Letters*, 32:1645–1646, 1996.

[6] Y. Mao and A. Banihashemi. A Heuristic Search for Good Low-Density Parity-Check Codes at Short Block Lengths. In *Proc. 2001 International Conference on Communications (ICC '01)*, pages 11–14, June 2001.

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

[8] T. Richardson and R. Urbanke. The Capacity of Low-Density Parity-Check Codes Under Message-Passing Decoding. *IEEE Transaction on Information Theory*, 47(2):599–618, Feb. 2001.