JOINT GRAPH-DECODER DESIGN OF IRA CODES ON SCALABLE ARCHITECTURES

Frank Kienle, Norbert Wehn

Institute of Microelectronic Systems, 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. The advantage of IRA codes over LDPC codes is that they come with a linear-time encoding complexity.

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 a scalable partly parallel IRA decoder architecture. We present a joint graph-decoder design to parallalize IRA codes which can be efficiently processed by this decoder without any RAM access conflicts. We show design examples of these IRA codes which outperform the UMTS Turbo-Code by 0.2dB.

1. INTRODUCTION

Sophisticated channel coding schemes become increasingly important in 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.

LDPC codes were introduced by Gallager 1963 [2] and were re-discovered in 1996 by MacKay and Neal [5]. The so called irregular LDPC codes are approaching the Shannon limit really close and are thus the best known codes. The major drawback of LDPC codes is the encoding complexity which is an obstacle for hardware implementation.

Irregular repeat-accumulate (IRA) codes were introduced in 2000 by Khandekar and McEliece [3]. These codes have a linear-time encoding complexity with a straight forward hardware realization, and outperform the actually deployed Turbo-Codes. Like LDPC codes the IRA codes can be represented by a Tanner graph with arbitrary connections between the nodes of different degree. An IRA code is defined by the degree distribution of the nodes.

IRA codes can be decoded by a combination of the Sum-Product (SP) algorithm and the Maximum A Posteriori (MAP) algorithm. A serial decoder architecture comes along with a limited throughput, a fully parallel architecture is not feasible due the wire routing complexity. Therefore, to achieve high throughput requirements a scalable partly parallel decoder architecture becomes mandatory. Its complexity strongly depends on the randomness of the Tanner graph. Regularity in the Tanner graph will simplify the implementation but can degrade the communications performance. A completely random graph leads, in general, to conflicts in storage element accesses. Therefore, we constrain the randomness of the graph such that no conflicts occur for a given architecture. This leads to a joint graph-decoder design methodology for IRA codes.

We present a partly parallel architectures which uses a shuffling network and an elaborate addressing scheme to provide the randomness of the graph. We show that IRA codes constructed for such an architecture can outperform Turbo-Codes by 0.2dB. Furthermore we apply an extrinsic scaling factor (ESF) to reduce communication degradation.

The paper is structured as follows. IRA codes and the decoding algorithm are explained in Section 2. In Section 3 the design of parallizable IRA codes is presented. The IRA decoder architecture is explained in Section 4. Section 5 gives some results and Section 6 concludes the paper.

2. IRA CODES

An IRA code [3] can be represented by a Tanner graph (Figure 1), with N variable nodes (open circles) and M check nodes (filled squares). The variable nodes can be partitioned in $K = N - M$ information nodes (IN) and M parity nodes (PN). Each information node is connected to i check nodes (CN). The fraction of information nodes connected to exactly i check nodes is denoted f_i, with $\sum_i f_i = 1$. The number of connections of a node is called degree. Each check node is connected to a information nodes. These $a * M$ connections between CN and IN are 'arbitrary permutations' (Π). The check nodes are furthermore connected to parity nodes in a fixed zigzag pattern.

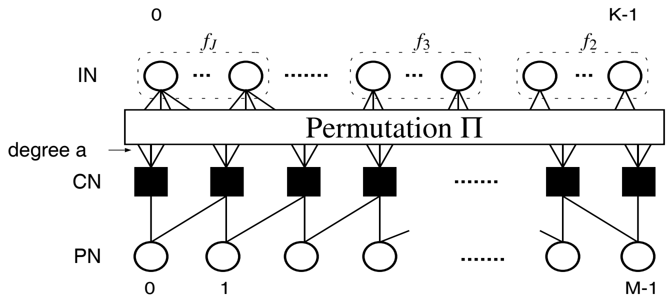

Fig. 1. Tanner graph for an IRA code, with IN (Information Node), CN (Check Node), PN (Parity Node) and f_i (fraction of nodes with degree i)

For a fixed Π the Tanner graph of Figure 1 represents a binary linear code. The code can be described by the degree distribution $f = (f_2, f_3, \ldots, f_J)$ and degree 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 $R = \frac{a}{a+\sum_i i f_i}$.

The encoder structure can be directly derived from the Tanner graph. The information sequence is first passed through a repetition unit (Figure 2). The repetition pattern follows the given degree sequence f_i. 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. a values of the interleaved sequence are replaced by their binary sum (modulo 2), which corresponds to a parity check. The resulting sequence is passed through an accumulator, which leads to the zigzag connections between PN and CN. The accumulator corresponds to a RSC-encoder with memory depth one, which yields a two state trellis. The information bits and the parity bits after the accumulator are transmitted. It is evident, that this encoding scheme has linear complexity.

2.1. Decoding Algorithm

A combination of the Sum-Product (SP) algorithm [2] and the Maximum A Posteriori (MAP) algorithm can be used to decode IRA codes[1]. Decoding is an iterative process, with a top down message flow according to Figure 1. One iteration can be divided into several steps:

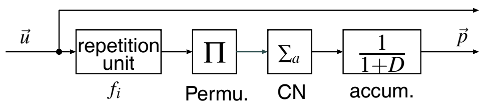

Fig. 2. Encoder for an IRA code

- process the information nodes (repetition code)
- interleave the messages
- calculate the soft values of the parity checks
- MAP processing of the RSC encoder (accumulator)
- update the messages participating the parity checks
- deinterleave the messages

The decoding is finished if all parity check constraints are fulfilled, or if a maximum number of iterations is reached. All calculations are done in the logarithm domain and the exchanged messages are assumed to be log-likelihood ratios $\lambda = log(p(0)/p(1))$. Each information node of degree j calculates an update of message k according to:

$$\lambda_k = \lambda_{ch} + \sum_{i=0, i\neq k}^{j-1} \lambda_i. \qquad (1)$$

λ_{ch} corresponds to the channel evidence of the IN ($\vec{u}$) and λ_i represent the LLRs of the incident edges. The messages are interleaved and passed to the check nodes. a messages are involved in one parity check (see Figure 1). Each check node calculates one 'a priori' messages λ_{ap}:

$$\log(\tanh(\lambda_{ap}/2)) = \sum_{i=0}^{a-1} \log(\tanh(\lambda_i/2)). \qquad (2)$$

This 'a priori' information and the received parity sequence ($\vec{p}$) is fed into a MAP decoder which processes the 2-state trellis of the accumulator. Subtracting λ_{ap} from the the MAP decoder output, leads to a so called extrinsic information λ_{ex}. This extrinsic information together with the a incoming messages of a check node updates the new a messages:

$$\log(\tanh(\lambda_k/2)) = \log(\tanh(\lambda_{ext}/2)) + \sum_{i=0, i\neq k}^{a-1} \log(\tanh(\lambda_i/2)). \qquad (3)$$

These messages are deinterleaved and passed back to the information nodes which completes one iteration.

This decoding results in optimal decoding if the Tanner graph is cycle free [6]. However, for limited block sizes the Tanner graph will contain cycles. A message passed back over a cycle to its origin will give no additional information for this node. Therefore the number of iterations with a gain in communications performance is limited. By selecting Π for a given IRA code, the resulting graph should have cycles as long as possible. Especially cycles of length ≤ 4 should be avoided [6].

The MAP algorithm can be implemented in the logarithm domain, by using the optimal Log-MAP algorithm or the suboptimal Max-Log-MAP algorithm [7]. It is known that the results of the Max-Log-MAP algorithm is too optimistic leading to a lack of communications performance. For decoding IRA codes the degradation can be compensated by using an extrinsic scaling factor (ESF=0.75). This factor is multiplied on the extrinsic information λ_{ex} prior the message update (Equation 3).

3. IRA CODES WITH CONFLICT FREE RAM ACCESSES

A high-throughput IRA decoder requires a partly parallel architecture. The level of parallization P is determined by the number of messages passed concurrently through a permutation network realizing the arbitrary connection Π of the Tanner graph (Figure 1). Two problems come along with the realization of the permutation network. For a high degree of parallization the network is not feasible, due to the wire routing complexity. When accepting arbitrary permutations, concurrent accesses to the same memory can not be avoided.

To tackle these problems we developed a joint **graph - decoder** design method to construct an IRA code. For a given permutation network we define the constraints for the Tanner graph and then find an IRA code within these constraints with a good communications performance. Here, we assume a simple shuffling network as permutation network which can be implemented efficiently. This network ensures that P input data are shuffled to P distinct target memories. However, due to the regularity of this permutation the resulting graph contains many cycles of length ≤ 4 and hence decreases the communications performance of the code. Therefore an elaborated addressing scheme of the target RAMs is applied to ensure a Tanner graph with cycles of length ≥ 6. The shuffling and the addressing scheme finally determines the interleaver Π for the encoding. This design methodology for IRA codes is explained in detail in [4].

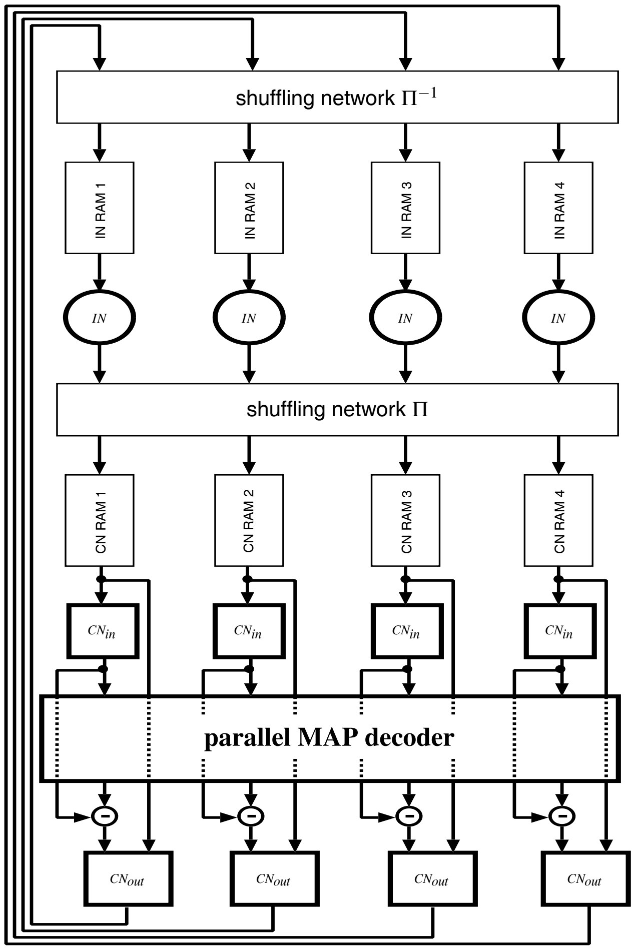

Fig. 3. Architecture of a partly parallel IRA decoder

4. IRA DECODER ARCHITECTURE

Figure 3 shows the architecture for an IRA decoder with a parallization $P = 4$. It consists of 4 functional units for IN and $CN_{in/out}$, a shuffling network Π, a parallel MAP decoder and sufficient memory banks to save the messages. The RAMs for the received channel values and the output values are neglected here. The partly parallel decoder architecture allows a wide range of the code parameters (f, a). Only the amount of the highest degree f_{max} and a_{max} is limited. For practical applications also the blocksize is limited due to latency problems.

All functional units (IN,$CN_{in/out}$) are simple arithmetic units and can be pipelined easily due to the independence of successive data. Hence, the processing units are not in the critical path. Each functional unit has one input and one output. To process an IN of degree j, $2 \cdot j$ cycles are required to complete the computation of this node. The messages of a given node are read sequentially and have to be located in the same memory at successive positions. Therefore, the correct sequence for message retrieval must be provided during storage. By using the IRA code construction of Section 3 the permutation can be done by a simple shuffling network. The code construction also determines the address for storage of each message. At each clock cycle $P = 4$ messages are stored in the CN RAMs1-4 until all IN nodes are processed.

Now the $P = 4$ CN_{in} functional units can start to calculate the a priori information (Equation 2) for the MAP decoder. Always a successive data in a CN RAM determines one a prior message.

The MAP decoder receives $P = 4$ messages per clock cycle in parallel and produces as much output messages. Parallel processing within the MAP decoder is done exploiting a sophisticated windowing technique [8]. The obtained extrinsic information and the corresponding a messages are fed to each CN_{out} functional unit (Equation 3). These produce a new messages which are deinterleaved by the shuffling network and stored in the IN RAMs 1-4. When all messages are stored in the IN RAMs, the next decoding iteration can start.

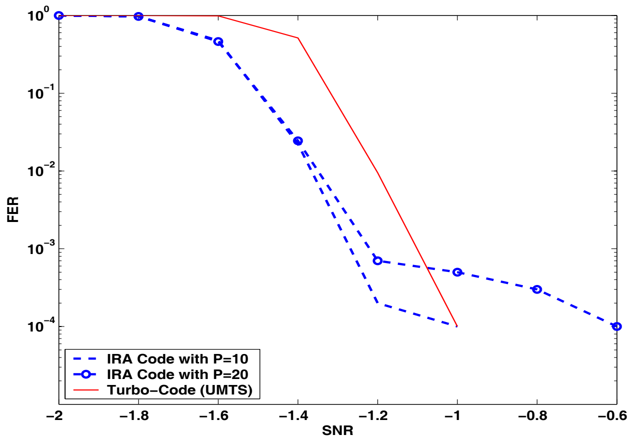

Fig. 4. Communications performance of IRA codes in comparison with Turbo-Codes, Rate=1/3, blocksize=5000, FER (frame error rate) and SNR (signal to noise ratio)

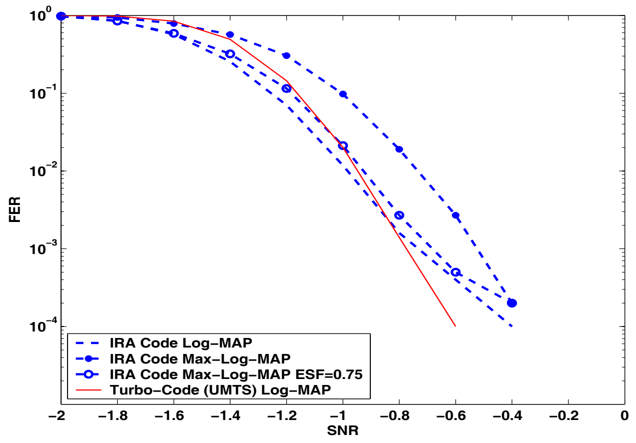

Fig. 5. Communications performance of IRA code (P=10) in comparison with Turbo-Codes, Rate=1/3, blocksize=1000

5. PERFORMANCE RESULTS

To investigate communications performance IRA codes are 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 codes are optimized for an architecture with a parallization of $P = 10$ and $P = 20$.

Figure 4 and Figure 5 show the communications performance of IRA codes with information blocksize of 5000 bits and 1000 bits respectively. Furthermore an UMTS compliant Turbo-Code (TC) with the same code rate and blocksize is plotted. For the IRA codes a maximum number of 40 iterations and for the TC 10 iterations are carried out. The performance gain by increasing the number of iterations is negligible in both cases. For a blocksize of 5000 bits (Figure 4) the IRA codes outperform the Turbo-Code by 0.2 dB at a FER of 10^{-3}. For the IRA code design for a parallization of $P = 20$ an error floor is visible. This is due to the constraints to the code design, and the resulting lower minimum code distance. Even for a smaller blocksize the performance of an IRA code ($P = 10$) is close to the TC performance, see Figure 5. By scaling the extrinsic values by $ESF = 0.75$, the decoding with the suboptimal and less complex Max-Log MAP algorithm approximates to that with the Log-MAP algorithm.

6. 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 a scalable partly parallel decoder architecture which can efficiently process IRA codes which are constructed in respect to the level of parallization. These IRA codes based on a joint graph-decoder design can be processed without RAM access conflicts and can even outperform the commonly deployed Turbo-Codes by 0.2dB. We show that the communications performance with the Max-Log MAP algorithm can be improved by using an extrinsic scaling factor (ESF=0.75).

7. REFERENCES

[1] S. Brink and G. Kramer. Turbo Processing for Scalar and Vector Channels. In *Proc. 3nd International Symposium on Turbo Codes & Related Topics*, pages 23–30, Brest, France, Sept. 2003.

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

[3] 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.

[4] F. Kienle and N. Wehn. Design Methodology for IRA Codes. In *Proc. 2004 Asia South Pacific Design Automation Conference (ASP-DAC '04)*, Yokohama, Japan, Jan. 2004. Accepted for Publication.

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

[6] 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.

[7] P. Robertson, P. Hoeher, and E. Villebrun. Optimal and Sub-Optimal Maximum a Posteriori Algorithms Suitable for Turbo Decoding. *European Transactions on Telecommunications (ETT)*, 8(2):119–125, March–April 1997.

[8] M. J. Thul, F. Gilbert, T. Vogt, G. Kreiselmaier, and N. Wehn. A Scalable System Architecture for High-Throughput Turbo-Decoders. In *Proc. 2002 Workshop on Signal Processing Systems (SiPS '02)*, pages 152–158, San Diego, California, USA, Oct. 2002.