EFFICIENT HARDWARE REALIZATION OF IRA CODE DECODERS

Frank Kienle and Norbert Wehn

Microelectronic System Design Research Group, University of Kaiserslautern
Erwin-Schrödinger-Straße, 67663 Kaiserslautern, Germany
Phone: (+49) 631 205 4435, Fax: (+49) 631 205 4437
E-mail: {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-Check (LDPC) codes and can even outperform the recently introduced Turbo-Codes of current communication standards. Implementation complexity like area, achievable throughput of these channel coding schemes will have a major impact on the decision of standardization committees.

In this paper we investigate implementation issues of IRA codes and analyze the strong interdependency of code performance and architectural dependencies, like throughput and area. We present an architecture template which is capable to decode hardware optimized IRA codes which can outperform Turbo-Codes (TC). We demonstrate this new approach through instances synthesized in a 0.13μm technology.

1. INTRODUCTION

Channel coding is an important component in communication systems. Current systems feature Turbo-Codes [1], while the discussion which channel codes should be applied for future standards is still on going. One hot candidate are LDPC codes the best know codes today. They were already introduced 1963 by Dr. Gallager [2] and extended to so called irregular LDPC codes in 1998 [3]. The major drawback of LDPC codes is the encoding complexity. Even though an efficient *encoding* algorithm exists [3], its *hardware* realization is still an obstacle.

The irregular repeat-accumulate (IRA) codes were introduced in 2000 by Khandekar and McEliece [4]. These codes belong to the class of LDPC codes and have a *linear-time encoding complexity* with a straight forward hardware realization. Like LDPC codes, IRA codes can be represented by a Tanner graph [3] with arbitrary connections between nodes. The decoding is an iterative process which exchanges informations between the two types of nodes.

An LDPC code can be implemented by a one-to-one mapping of the Tanner graph in hardware, i.e. each node is implemented and the connections are realized hard-wired. This was shown in [5] using a $N = 1024$, $R = 0.5$ irregular code with a resulting symbol throughput of 1 Gbit/s. Global interconnect problems, however, make such a design infeasible for larger code lengths. Therefore partly parallel architectures become mandatory. An important approach is the decoder-first code design [6]: first a hardware architecture is chosen and afterwards the code is constructed to match the architecture. A related design method is presented in [7] to implement a (3,6)-regular LDPC code on a FPGA. It supports a throughput of 54 Mbit/s with a maximum number of 18 iterations. Due to the use of regular LDPC codes the resulting communications performance can not measure up to Turbo-Codes. A further approach is a time folded architecture [8], which avoids the routing problem by instantiating multiple sequential decoding cores. One problem here is the latency issue and flexibility. An efficient LDPC code decoder architecture which uses a modified decoding algorithm is presented in [9]. But the resulting communications performance is again inferior to that of Turbo-Codes and also the encoding problem is not addressed.

The critical parts of all implementations is the lack of flexibility or the degradation in terms of communications performance. The communications performance of LDPC codes always have to be compared with Turbo-Codes.

In [10] we have presented a new methodology to design IRA codes which can be efficiently implemented.The basic idea is to provide an architecture template which can process all IRA codes. We analyzed this template and extracted the critical part of the architecture. This is the permutation network which has to realize the connectivity of the Tanner graph. The bottleneck can be circumvented by using an appropriated permutation network. The Tanner graph is then designed regarding the constraints of the permutation network.

In this paper we present in depth investigations of the trade-off between implementation complexity and communications performance of IRA codes. We present to the best of our knowledge the first IRA code decoder synthesized in a 0.13μm technology. We show synthesis instances of IRA

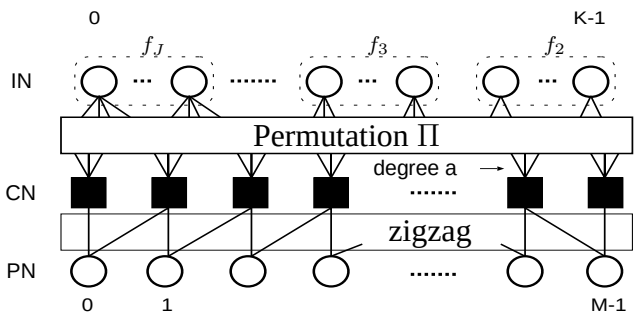

Fig. 1. Tanner graph for IRA code, with IN (Information Node), CN (Check Node), PN (Parity Node) and f_i (fraction of nodes with degree i)

code decoders which can process IRA codes outperforming correctly employed Turbo-Codes of the CDMA2000 standard. The presented decoder can process various IRA codes up to rate of $R = 0.8$ and reaches a decoding throughput of 100Mbit/s at 30 iterations.

2. IRA CODES

An IRA code [4] can be represented by a Tanner graph (Figure 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 IN nodes are of varying degree. f_i denotes the fraction of IN nodes with degree i, with $\sum_i f_i = 1$. Each check node is connected to a information nodes. These $E_\Pi = a * M$ connections between CN and IN are 'arbitrary permutations' (Π). The check nodes are furthermore connected to parity nodes in a fixed zigzag pattern. The code **ensemble** can be described by the degree distribution $\mathbf{f} = (f_2, f_3, \ldots, f_J)$ and a. For each Π we get one code out of this ensemble. The parameters $(\mathbf{f}, a)$ determines the convergence, while Π determines the error floor [3].

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 $R = \frac{a}{a+\sum_i i f_i}$.

2.1. Encoding

The Encoder, shown in Figure 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. a values of the interleaved sequence contribute to one parity check, i.e. a values are replaced by their binary sum

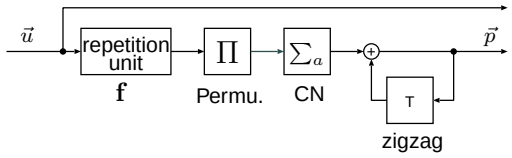

Fig. 2. Encoder for an IRA code

(modulo 2). The resulting sequence is passed through an accumulator which leads to the zigzag connections between PN and CN. $\vec{u}$ and $\vec{p}$ are transmitted. It is evident, that this encoding scheme has linear complexity.

2.2. Decoding

IRA codes can be decoded using the message passing algorithm [2]. It exchanges soft-information iteratively between variable and check nodes. The update of the nodes can be done with a canonical scheduling [2]: In the first step all variable nodes must be updated, in the second step all check nodes respectively. 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). 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

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

The message passing results in optimal decoding if the Tanner graph is cycle free [3]. However, for finite block length the Tanner graph will contain cycles. Once a message has completed a cycle, the node updates according to Equation 1 and 2 become suboptimal. Therefore, the longer the cycles the more you gain in communications performance from an increased number of iterations. By selecting Π for a given IRA code, the resulting graph should contain cycles as long as possible. The shortest cycle is called girth, it should be > 4 [3].

For fixed-point implementations it was shown in [11] that the total quantization loss is ≤ 0.1db when using a 6 bit message quantization compared to infinite precision. For a 5 bit message quantization the loss is 0.1-0.2 dB [3].

3. DECODER ARCHITECTURE

The goal is to have an IRA code decoder which is capable to process a huge variety of codes, regarding block length,

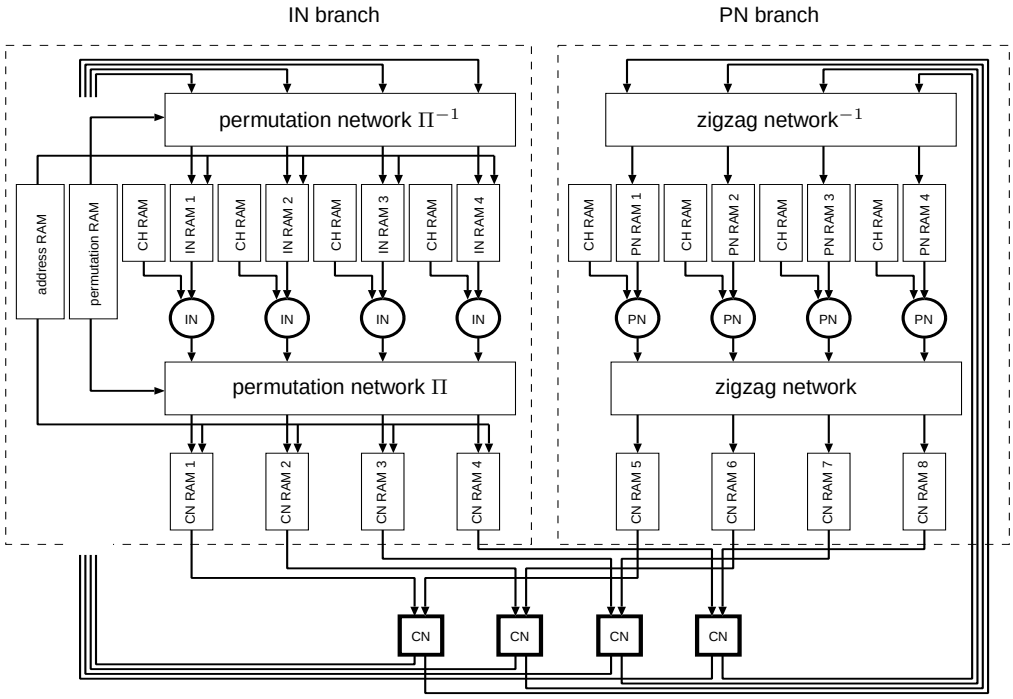

Fig. 3. Basic architecture of our IRA code decoder

code rate and also communications performance (i.e. varying $(\mathbf{f}, a)$ and Π). As mentioned in the introduction partly parallel decoder architectures are mandatory. By using a partly parallel architecture we have to store all messages exchanged during the process of iterative decoding. Furthermore the connectivity of the Tanner graph has to be stored and we have to provide networks which realizes these connectivities.

The basic architecture of our decoder is shown in Figure 3 for a parallization $P = 4$. It features 4 functional units for CNs, INs and PNs, the permutation network Π, the zigzag network, memory banks for message storage, address and permutation information. The IN and PN processing can be done independently due to the disjunctive of Π and the zigzag connections, denoted as IN and PN branch in the following (dotted line in Figure 3). The IN and PN functional units are implemented as single input and output modules. Thus all incoming edges of one node are processed sequentially. Each CN unit accepts two data in parallel, one from the IN and one from the PN branch and calculates at most two data which are passed back to the IN and PN branches respectively. It has to be ensured that the messages provided by both branches belong to the same check node. All messages for a node should be stored in the same memory at consecutive positions. This allows a simple reading scheme without any conflicts. Therefore, the correct sequence for message retrieval must be provided during storage. This is achieved by an appropriated network and memories keeping the desired storage location. The costs to realize the network, address and permutation RAMs (see Figure 3) depends on the underlying Tanner graph of the code. To minimize these costs we developed a joint **graph-decoder** design methodology [10], which is summarized in Section 4.

The number of messages processed by the IN branch are $E_\Pi = M \cdot a$ and by the PN branch $M \cdot 2$. With $a \geq 2$ the IN branch requires more cycles to process all data compared to the PN branch. Though, we define the level of parallization P of our architecture, as the number of simultaneously processed edges of E_Π.

4. JOINT GRAPH-DECODER DESIGN

A high-throughput IRA code decoder requires a high degree of parallization P. We have to pass P messages concurrently through a permutation network realizing the connection Π of the Tanner graph (Figure 1). We do not focus on the zigzag network because of its simple structure. The realization of the permutation network causes some problems: as already mentioned, 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. Furthermore, the storage of the required connectivity can dominate the overall memory space [12].

In our joint **graph - decoder** design method [10] we de-

fine the constraints for the Tanner graph for a given permutation network. Then within an IRA code ensemble (Section 2) we determine one code which fulfills the network constraints without degrading the communications performance. One example of a permutation network is a simple shuffling 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 we apply an elaborated addressing scheme to ensure a Tanner graph with cycles of length ≥ 6. The shuffling and the addressing scheme finally determines the whole connectivity between IN and PN.

In [10] we did not present implementation results and the important evaluation of the trade-off between communications performance and architectural efficiency. This analysis is done in the following chapter.

5. COMMUNICATIONS PERFORMANCE VS. ARCHITECTURAL EFFICIENCY

The communications performance is measured in terms of frame error rate (FER) dependant on the required signal to noise ratio (E_b/N_0). The architectural efficiency are fixed in terms of throughput and area. First we analyze the throughput which is then put in relation to the communications performance and implications for the area. All simulations presented in this chapter are based on codes of rate $R = 0.5$ and I=1600 information bits using a floating point representation. The clock frequency is $f_{\text{cycle}} = 334MHz$, which is determined by the memory access time.

5.1. Throughput

The decoder throughput T is

$$T = \frac{I}{\#\text{cycles}} \cdot f_{\text{cycle}}, \tag{3}$$

with I the number of information bits to be decoded and $\#\text{cycles}$ the number of cycles to decode one block. The number of cycles is calculated as $\frac{C}{P_{IO}} + It \cdot \left(2 \cdot \frac{E_\Pi}{P}\right)$. Thus Equation 3 yields:

$$T = \frac{I}{\frac{C}{P_{IO}} + It \cdot \left(2 \cdot \frac{E_\Pi}{P}\right)} \cdot f_{\text{cycle}}. \tag{4}$$

The part $\frac{C}{P_{IO}}$ is the number of cycles for input/output (I/O) processing. We assume that reading a new codeword (C) and writing the result of the prior processed block can be done in parallel with reading/writing P_{IO} data concurrently. We assume $P_{IO} = 10$ which is a reasonable assumption. Due to the small fraction of $\frac{C}{P_{IO}}$ the throughput can be approximated by $T \sim \frac{I}{E_\Pi} \cdot P \cdot \frac{1}{It}$, i.e. it depends on the number of iterations (It), the number of edges processed in parallel (P) and the ratio of information bits and edges $\frac{I}{E_\Pi}$. Figure 4 plots the throughput of a decoder depending on the level of parallization. Figure 4a shows the throughput for a fixed $\frac{I}{E_\Pi}$, assuming 20, 30 or 40 iterations respectively. Figure 4b depicts the throughput for a fixed number of 30 iterations and different number of edges E_Π. In the following we will analyze the implications for the communications performance and the area.

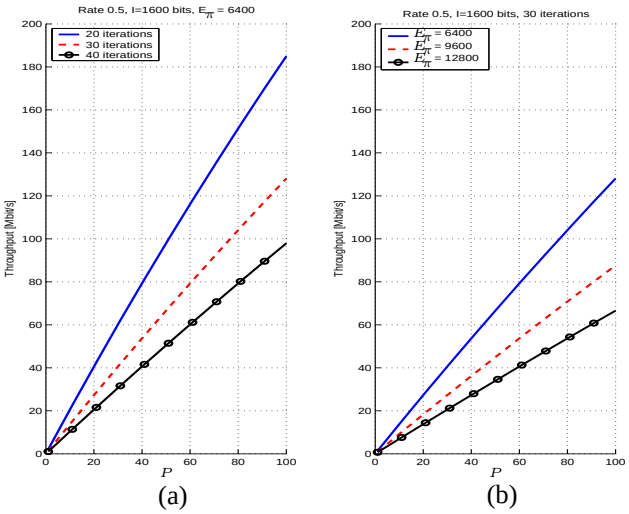

Fig. 4. Decoder throughput depending on the parallization: a) varied number of iterations, b) varied number of E_Π

5.2. Number of Iterations

The easiest way to increase the throughput of a LDPC decoder implementation is to use a small number of iterations. But the number of iterations is of course a crucial parameter for the communications performance. In fact we have to compare the communications performance with Turbo-Codes which are deployed in current standards.

Figure 5 shows the communications performance of an IRA code with $E_\Pi = 9600$ assuming 40, 30, 20 and 10 iterations. It shows also the performance of a CDMA2000 compliant Turbo-Code (TC) with Log-MAP implementation and 10 iterations. It can be seen that the performance of the IRA code with 30 iterations is equivalent to the TC performance with 10 iterations. For 40 iterations the IRA code outperforms the TC, for larger block length I=5000 even by 0.2db [10]. The performance gain of both codes by further increasing the number of iterations is negligible. One important fact is that the performance gain of IRA codes is substantial up to 30 iterations, while further iterations will just gain a little bit. The same fact holds for other (irregular) LDPC codes which are decoded by a belief propagation algorithm.

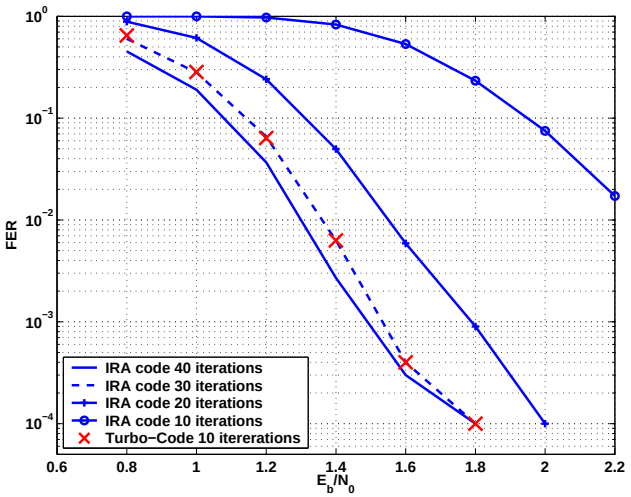

Fig. 5. Communications Performance depending on the number of iterations (TC and IRA code 30 iterations are identical)

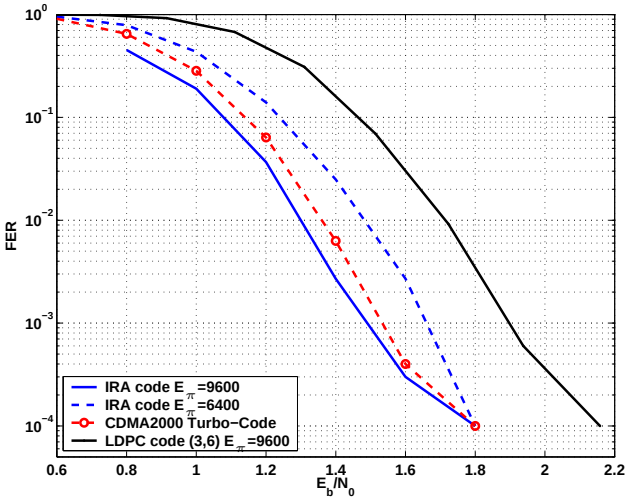

Fig. 6. Communications performance depending on the number of messages E_Π

5.3. Ratio I/E_Π

The ratio of number of information bits I to the number of edges E_Π affects the throughput and the communications performance. Capacity approaching IRA codes ($R = 0.5$) suggested in [4] have code-parameters $a = 8$ and a maximum VN degree of 48. Which yields $E_\Pi = 12800$ for I=1600 bits. Reducing a to 6 or 4 decreases the number of edges to $E_\Pi = 9600$ and $E_\Pi = 6400$ respectively. But reducing E_Π will also reduce the maximum degree of the variable nodes. This degrades the communications performance.

Figure 6 shows the communications performance of IRA codes with $E_\Pi = 9600$ and $E_\Pi = 6400$ in comparison to a Turbo-Code and a regular (3,6) LDPC code. 40 iterations are carried out for the IRA and LDPC codes and 10 iterations for the TC with Log-MAP implementation. The IRA code with $E_\Pi = 9600$ has a maximum VN degree of 15, the IRA code with $E_\Pi = 6400$ a degree of 7. Note that the regular (3,6) LDPC code has also 9600 edges but its performance is 0.4dB worse than the corresponding IRA code. This demonstrates that irregular codes are necessary for a good communications performance. But increasing a and thus the maximum VN degree decreases the throughput for a given parallization as shown in Figure 4.

5.4. Node Area

The node area depends on the node degree. This degree depends on the communications performance **and** code rate which will be shown in the following.

To process a node of degree 15 we have to store all 15 messages in registers before we can process the first output result according to Equation 1 and 2. Thus, the node area depends linearly on the maximum degree which can be processed. This holds for the variable and check nodes.

Table 1 shows examples of the degree and area for VN and CN nodes depending on the code rate. The degrees are chosen with respect to a communications performance comparable to that of Turbo-Codes. For low code rate, e.g. $R = 0.33$, the maximum degree to be processed is dominated by the VN degree. For high rate codes, e.g $R = 0.9$, a high CN degree is mandatory. The area of the CN nodes is larger compared to the VN nodes. This is due the look-up tables for the $tanh$ (Equation 2) and the capability to process data from the PN branch in parallel (Figure 3).

6. RESULTS

Results for different instances of our architecture template are presented taking into account the trade-offs described in the previous chapter. The decoder is modeled in synthesizable VHDL. The results were obtained with the Synopsis

	max degree		**area [um^2]**	
Code Rate	VN	CN (a)	VN	CN
$R = 0.33$	20	4	18227	19087
$R = 0.5$	15	6	14644	20879
$R = 0.8$	4	16	8347	28045
$R = 0.9$	4	26	8347	35127

Table 1. Example of maximum node degrees and area depending on the code rate, regarding a Turbo-Code comparable communications performance

Area [mm^2]		P (Paralliztion)		
		40	**60**	**80**
RAMs	channel LLRs	0.68		
	Messages	4.1		
	Permutation Pattern	0.136	0.163	0.193
Logic	Information Nodes	0.59	0.88	1.17
	Parity Nodes	0.2	0.3	0.4
	Check Nodes	1.12	1.68	2.24
Shuffling Network		0.08	0.153	0.244
Zigzag Network		0.03	0.05	0.07
control logic		0.172	0.226	0.280
Total Area [mm^2]		7.088	8.23	9.349
Throughput @30 iter. [Mbit/s]		53	80	106

Table 2. Synthesis Results for IRA code decoders

Design Compiler on a 0.13 μm technology. The maximum clock frequency is 334MHz. The decoder is capable to process codes of $R = 0.8$ to $R = 0.5$ with a maximum of I=5000 information bits and $E_\Pi = 20000$. The synthesis results are presented in Table 2 for parallization of P=40, 60 and 80 and a 6 bit message quantization. The throughput calculations are done for 30 iterations and $\frac{I}{E_\Pi} = \frac{1}{4}$. The resulting area is divided into 3 parts: memory, logic and the permutation networks including control logic.

Increasing P does not affect the number of messages to be stored. The aspect ratio of the memories, however, changes. The RAMs to store the permutation pattern and addressing information of the shuffling network slightly increases with P. The higher the parallization, the more permutation possibilities of the network Π has to be provided. This is the reason why the shuffling network does not grow linearly with P. The node area depends on the maximum degree which can be processed. The decoder can process codes with a maximum CN degree of $a = 16$ and a maximum VN degree of 15 respectively. For example, a $R = 0.8$ code with I=1600 bits and $E_\Pi = 6400$ reaches for P=80 a throughput of 106 Mbit/s at 30 iterations.

7. CONCLUSION

IRA codes are one of the best know codes and can even outperform Turbo-Codes. In this paper we presented a thorough exploration of the design of IRA codes regarding hardware constraints. We presented an architecture and synthesis examples of IRA code decoders, which can process many different codes up to 5000 information bits ranging from a code rate of $R = 0.5$ to $R = 0.8$. For 30 iterations a decoded throughput of 100 Mbit/s can be reached. The IRA codes processed by the decoder can outperform Turbo-Codes.

8. REFERENCES

[1] C. Berrou, "The Ten-Year-Old Turbo Codes are Entering into Service," *IEEE Communications Magazine*, vol. 41, pp. 110–116, Aug. 2003.

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

[3] T. Richardson and R. Urbanke, "The Renaissance of Gallager's Low-Density Pariy-Check Codes," *IEEE Communications Magazine*, vol. 41, pp. 126–131, Aug. 2003.

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

[5] A.J. Blanksby and C. J. Howland, "A 690-mW 1-Gb/s, Rate-1/2 Low-Density Parity-Check Code Decoder," *IEEE Journal of Solid-State Circuits*, vol. 37, no. 3, pp. 404–412, Mar. 2002.

[6] E. Boutillon, J. Castura, and F.R. Kschischang, "Decoder-first coder design," in *Proc. 2nd International Symposium on Turbo Codes & Related Topics*, Brest, France, Sept. 2000, pp. 459–462.

[7] T. Zhang and K. Parhi, "A 54 MBPS (3,6)-Regular FPGA LDPC Decoder," in *Proc. IEEE Workshop on Signal Processing Systems (SiPS '02)*, San Diego,USA, Sept. 2002, pp. 127–132.

[8] M. Cocco, J. Dielissen, M. Heijligers, A. Hekstra, and J. Huisken, "A Scalable Architecture for LDPC Decoding," in *Proc. 2004 Design, Automation and Test in Europe (DATE '04)*, Paris, France, Mar. 2004.

[9] M. Mansour and N. Shanbhag, "Architecture-Aware Low-Density Parity-Check Codes," in *Proc. 2003 IEEE International Symposium on Circuits and Systems (ISCAS '03)*, Bangkok, Thailand, May 2003.

[10] 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.

[11] T. Zhang, Z. Wang, and K. Parhi, "On finite precision implementation of low-density parity-check codes decoder," in *Proc.International Symposium on Circuits and Systems (ISCAS '01)*, Antwerp,Belgium, May 2001.

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