Aliazam Abbasfar

Turbo-like Codes

Design for High Speed Decoding

Springer

Turbo-like Codes

Aliazam Abbasfar

Turbo-like Codes

Design for High Speed Decoding

Springer

A C.I.P. Catalogue record for this book is available from the Library of Congress.

ISBN-10 978–1–4020–6390–3
ISBN-13 978–1–4020–6390–9

Published by Springer,
P.O. Box 17, 3300 AA Dordrecht, The Netherlands.

www.springeronline.com

Printed on acid-free paper

Dedicated to my wife

Contents

List of Figures

List of Tables

Acknowledgments

First and foremost, I would like to express my deepest gratitude to my wife for her patience and sacrifices throughout this research. She has been a constant source of assistance, support, and encouragement. My heartfelt thanks go to my parents for their generous love, encouragement, and prayers. Their sacrifices have been my inspiration throughout my career and I am deeply indebted to them in all my successes and accomplishments. Words cannot express the deep feeling of gratitude I have for my family.

There are so many people that I would like to thank for making my experience at UCLA truly one of a kind. I would like to thank my advisor Professor Kung Yao for all the help, support, and opportunities he has provided me over these years. His valuable advice, guidance, and unconditional support helped me overcome all the challenges of the doctoral process. I also want to thank Dr. Flavio Lorenzelli for his support and fruitful discussions throughout my research.

My special thanks go to Dr. Dariush Divsalar who has been the motivation force to go into the specific field of channel coding. I have gained most of my knowledge in the field from discussions with him. He has had a profound effect on my Ph.D. both as a mentor and a colleague.

Finally, I would also like to thank Professor Parviz Jabehdar-Maralani of the University of Tehran for his continued support and encouragement.

Abstract

The advent of turbo codes has sparked tremendous research activities around the theoretical and practical aspects of turbo codes and turbo-like codes. The crucial novelty in these codes is the iterative decoding.

In this work, first a novel high-speed turbo decoder is presented that exploits parallelization. Parallelism is achieved very efficiently by exploiting the message-passing algorithm. It has been shown that very large speed gains can be achieved by this scheme while the efficiency is maintained reasonably high. Memory access, which poses a practical problem for the proposed parallel turbo decoder, is solved by introducing the conflict-free interleaver. The latency is further improved by designing a special kind of conflict-free interleaver. Furthermore, an algorithm to design such an interleaver is presented. Simulation results show that the performance of turbo code is not sacrificed by using the interleaver with the proposed structure.

Although turbo code has near Shannon-capacity performance and the proposed architecture for parallel turbo decoder provides a very efficient and highly regular hardware, the circuit is still very complex and demanding for very high-speed decoding. Therefore, it becomes necessary to find turbo-like codes that not only achieve excellent error correction capability, but are also very simple. As a result, a class of new codes for different rates and block sizes, called Accumulate–Repeat–Accumulate (ARA) codes, was invented during this search. The performance of ARA codes is analyzed; and it has been shown that some ARA codes perform very close to random codes, which achieve the Shannon limit.

The architecture for high-speed ARA decoder is presented and practical issues discussed. This leads us to a general class of turbo-like codes with parallelism capability, i.e. codes with projected graphs. It is shown that parallel turbo decoder, discussed earlier, is in the same class. Projected graph provides a powerful and yet simple method for designing parallelizable turbo-like codes.

Chapter 1
Introduction

Efficient and reliable data communication over noisy channels has been pursued more and more for many decades. Applications such as wire-line modems, wireless and satellite communications, Internet data transfer, digital radio broadcasting, and data storage devices are only a few examples that accelerated the development of data communication systems.

The issue of efficiency and reliability in communication systems was fundamentally addressed by Shannon [30] in 1948. Shannon introduced the capacity (C) for a noisy channel, which determines the maximum data rate (R) that can be transferred over the channel reliably (i.e. without any error). In other words, there exists a coding scheme of rate ($R < C$) with arbitrarily small error probability. The proof of this was done in a nonconstructive way, which means that it does not give any method for construction of capacity-approaching codes.

The pursuit of capacity-approaching codes took almost 50 years. The introduction of turbo codes by Berrou, Glavieux, and Thitimajshima [9] was a major breakthrough in the world of practical capacity-approaching codes causing a revolution in the field of error-correcting codes. As a result, several other classes of capacity-approaching codes were rediscovered and invented including Low-Density Parity-Check (LDPC) codes [14], Repeat–Accumulate (RA) codes [12], and product codes.

The trend in data communications is towards high data rate applications which require high-speed decoding. Although very excellent codes have been proposed and their efficient decoding algorithms are known, design of codes that are suitable for high-speed applications is still a challenging task. This includes design of high-speed decoding architectures, as well as low complexity codes, which are naturally more suitable for parallelism.

1.1 Outline

The advent of turbo codes has sparked tremendous research activities around the theoretical and practical aspects of turbo codes and turbo-like codes. This study introduces different types of turbo codes and turbo-like codes. These codes include the Parallel Concatenated Convolutional Code (PCCC), originally introduced by Berrou [9], Serial Concatenated Convolutional Codes (SCCC) later introduced in [7], RA codes [12], and product codes.

Aliazam Abbasfar, *Turbo-Like Codes*, 1–3.

The common property among turbo-like code is that they consist of very simple constituent codes that are connected to each other with random or pseudorandom interleavers. The crucial novelty in these codes is the iterative decoding. This means that the constituent codes are decoded separately, which is efficient and practically feasible since they are very simple codes. Then, they pass new information to each other in a course of a few iterations.

It has been shown that iterative decoding is a generalization of the well-known probability or belief propagation algorithm. The belief propagation algorithm that has been essential for development of new ideas throughout this work is described in the context of coding. The basic theorems for this algorithm are explained and proven in the following paragraphs. Thisis then followed by a description of the computational algorithm. The probability propagation algorithm is proven in conjunction with a tree-structured graph – graphs without any cycle. In fact, the graphical representation of any problem solved by this algorithm is the centerpiece of the algorithm. The generalization of the algorithm for graphs with cycles is presented later on.

Representation of codes on graph is the next step towards characterization of the iterative decoding as an example of the probability propagation algorithm. The graph representations are presented for a few codes that are commonly used in turbo-like codes.

In Chapter 3, first the traditional turbo decoder is introduced. Second, a novel high-seed turbo decoder is presented that exploits parallelization. Parallelism is achieved very efficiently by exploiting the message-passing algorithm. Finally, two characterization factors for the proposed parallel turbo decoder are investigated: speed gain and efficiency. It has been shown by simulations that very large speed gains can be achieved by this scheme while the efficiency is maintained reasonably high.

Memory access poses a practical problem for the proposed parallel turbo decoder. This problem is solved in the next section. Conflict-free interleaver is introduced to address the memory access problem. The latency is further improved by designing a special kind of conflict-free interleaver. Lastly, an algorithm to design such an interleaver is presented. Simulation results show that the performance of turbo code is not sacrificed by using this interleaver.

Hardware complexity, which is of major concern, is investigated and the overall architecture of the hardware for high-speed turbo decoder is presented.

Although turbo code has near Shannon-capacity performance and the proposed architecture for parallel turbo decoder gives a very efficient and highly regular hardware, the circuit is still very complex and demanding for very high-speed decoding. Therefore, Chapter 4 examines simple turbo-like codes that can achieve excellent error correction capability. In order to investigate the performance of the turbo-like codes some useful analysis tools are used. Some maximum likelihood (ML) performance bounds are briefly explained which evaluate the performance of codes using ML decoding. The density evolution (DE) method, which analyzes the behavior of turbo-like codes using iterative decoding, is also described.

The RA code provided valuable insight in the search for low-complexity turbo-like codes. A class of new codes for different rates and block sizes, called ARA code, was invented during this search. The performance of ARA codes is analyzed and some are shown to perform very close to random codes, which achieve the Shannon limit. The simplicity of ARA codes allows us to build a high-speed ARA decoder with very low complexity hardware.

Chapter 5 first presents the architecture for high-speed ARA decoder and then discusses practical issues. This leads us to a general structure for turbo-like codes with parallelization capability. The concept of projected graph is presented. In fact, codes with projected graph comprise a class of turbo-like code. The parallel turbo decoder discussed earlier is in the same class. Projected graph provides a powerful and yet simple method for designing parallelizable turbo-like codes, which is used in future research.

Finally, future research directions are discussed in light of this study's findings.

Chapter 2
Turbo Concept

2.1 Turbo Codes and Turbo-like Codes

2.1.1 Turbo Codes

Turbo code introduced by Berrou [9] is a PCCC, which consists of two or more convolutional codes encoding different permuted versions of a block of information. A block diagram of a PCCC turbo encoder is shown in Figure 1.

Each convolutional code is called a constituent code. Constituent codes may be same or different, systematic or nonsystematic. However, they should be of recursive type in order to have good performance. In most cases the C_0 is a systematic code, which means that the input sequence is transmitted along with the coded sequences; the overall code is systematic too. If the code rate of the constituent codes is r_0, $r_1, \ldots, r_n$, then the overall rate is $r_0||r_1|| \cdots ||r_n$; like parallel resistors.

Later on SCCC were introduced in [7]. A block diagram of a SCCC is drawn in Figure 2.

The block of information is encoded by the first constituent code. Then the output is interleaved and encoded by the second constituent code, and so on and so forth. The output of the last stage is sent over the communication channel. If the code rate of the constituent codes are $r_1, r_2, \ldots, r_n$, the overall rate is $r_1 \times r_2 \times \cdots \times r_n$. To obtain a systematic code all constituent codes should be systematic.

We can combine the PCCC and SCCC codes to come up with various Hybrid Concatenated Convolutional Codes (HCCCs). Such a HCCC is shown in Figure 3.

We can extend the above-described turbo codes to obtain more generalized codes. In the above codes all the constituent codes are convolutional. If we remove this limitation and let them be arbitrary block codes, then we will have a broader class of codes called turbo-like codes. Some examples of turbo-like codes are given in the sequel.

2.1.2 Repeat–Accumulate Codes

Perhaps the simplest type of turbo-like codes is RA codes; which makes it very attractive for analysis. The general block diagram of this code is given in Figure 4. It is a serial concatenated code of two constituent codes: repetition code and

Aliazam Abbasfar, *Turbo-Like Codes*, 5–17.

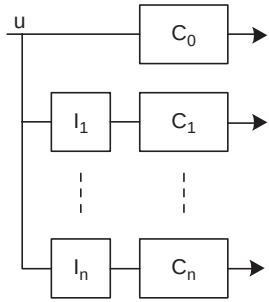

Fig. 1 The block diagram of a PCCC encoder

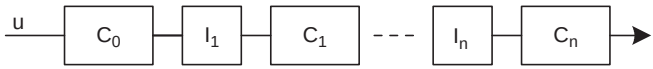

Fig. 2 The block diagram of a SCCC encoder

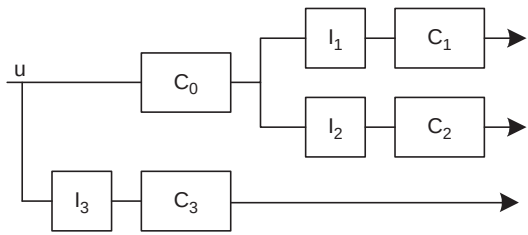

Fig. 3 An example of a HCCC encoder

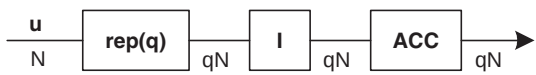

Fig. 4 Repeat–Accumulator code block diagram

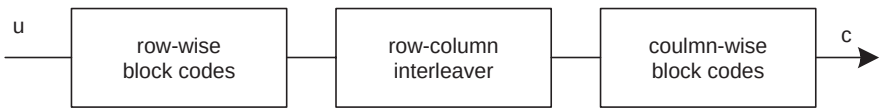

Fig. 5 Block diagram of a product code

accumulate code. An information block of length N is repeated q times and interleaved to make a block of size qN, and then followed by an accumulator. Using rate formula for serial concatenated codes, the code rate for RA codes would be $1/q$.

2.1.3 Product Codes

The serial concatenation of two block codes with a row–column interleaver results in a product code. In fact, each block code consists of several identical smaller block codes that construct rows/columns of the code word. A block diagram of this code is shown in Figure 5.

2.2 Iterative Decoding

The crucial novelty in turbo codes is the introduction of iterative decoding. The other key component is random or pseudorandom interleaver, which is discussed later. Having exploited these concepts, turbo codes achieve excellent performance with a moderate complexity.

The iterative decoding algorithm is based on maximum a posteriori (MAP) estimation of the input sequence. However, since it is difficult to find the MAP solution by considering all the observations at the same time, the MAP decoding is performed on the observations of each constituent code separately. This is explained for a PCCC with two constituent codes. Since two codes have been produced from one input sequence, the a posteriori probability (APP) of data bits coming from the first constituent decoder can be used by the second decoder and vice versa. Therefore the decoding process is carried out iteratively. At the beginning we do not have any information about input sequence. The MAP decoding of the first constituent code is performed without any prior knowledge of the input sequence. This process generates APP of the input sequence bits that can be exploited in the second decoder. The information passed to the other constituent decoder is called extrinsic information.

BCJR algorithm [5] is an efficient algorithm that recursively computes the APPs for a convolutional code. In [6] a general unit, called SISO, is introduced that generates the APPs in the most general case. Chapter 2 explains the BCJR algorithm based on decoding on graphs with tree structure.

Since the second constituent code is using the permuted version of the input sequence, therefore, extrinsic information should also be permuted before being used by the second decoder. Likewise, the extrinsic information of the second decoder is to be permuted in reverse order for the next iteration of the first decoder. The iterative decoding block diagram is shown in Figure 6.

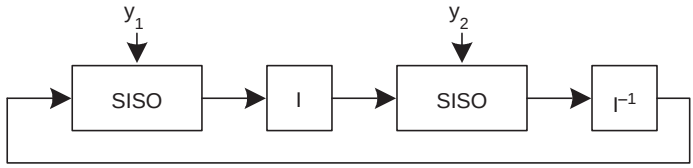

Fig. 6 The iterative turbo decoding block diagram

2.3 Probability Propagation Algorithms

It has been shown that the iterative decoding is the generalization of the well-known probability or belief propagation algorithm. This algorithm has been developed in the artificial intelligence and expert system literature, most notably by Pearl [25] and Lauritzen and Spiegelhalter [19]. Connection between the Pearl's belief propagation algorithm with coding was first discovered by MacKay and Neal [22, 23], who showed the Gallager algorithm [14] for decoding LDPC codes is essentially an instance of belief propagation algorithm. McEliece et al. [24] independently showed that turbo decoding is also an instance of belief propagation algorithm. We describe this algorithm in a way that is more suitable for coding applications.

If we have some variables that are related to each other, there is a bipartite graph representation showing their dependence, which is called the Tanner graph [31]. The nodes are divided between variable nodes (circles) and constraint nodes (squares). Two examples of such graphs are shown in Figure 7.

If we have some noisy observation of the variables, then it becomes a probabilistic graph. In some communication systems some of the variables are not sent through the channel, hence, there is no observation available for them in the receiver side. Therefore, we should distinguish between observed and unobserved variables. Two examples of probabilistic graphs are shown in Figure 8.

We denote all variables with vector **x** and their observations with vector **y**. Some useful definitions are listed in Table I.

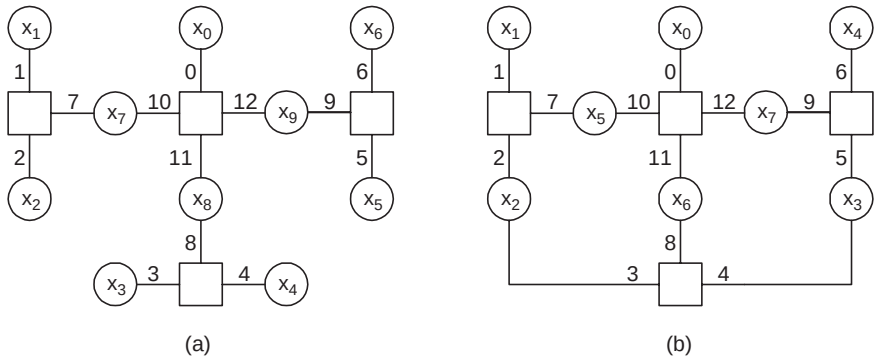

Fig. 7 Examples of Tanner graphs: (a) tree, (b) with cycles