INDIAN INSTITUTE OF TECHNOLOGY
KANPUR , UTTAR PRADESH

Implementation of a LDPC-Decoder using Field-Programmable Gate Array

E-CLUB PROJECT

TEAM MEMBERS

TUTOR

VAIBHAV THAKKAR

MEMBERS

HARSH VARDHAN ARYA

MALKURTHI SREEKAR

NAIZA SINGLA

PRAKHAR MAHESHWARI

PRANAB PANDEY

PRATEEK GUPTA

SHIVAM MALHOTRA

SHUBHAM KORDE

COORDINATORS

ANSHUL RAI

AFZAL RAO

NETRAVAT PENDSEY

UTKARSH GUPTA

Contents

1. Introduction

In this project, we plan to apply a joint code and decoder design methodology to develop a high-speed (3,k)-regular LDPC code partly parallel decoder architecture based on which we implement a 9216 bit, rate-1/2 (3,6)-regular LDPC code decoder on a FPGA device.

LDPC codes are linear codes obtained from sparse bipartite graphs. LDPC have received a lot of attention and have been widely considered as next-generation error-correcting codes for telecommunication and magnetic storage as they provide a performance which is very close to the capacity for a lot of different channels and linear time complex algorithms for decoding.

The high-speed decoder hardware implementation is obviously one of the most crucial issues determining the extent of LDPC applications in the real world. Since the direct implementation of BP algorithm will incur too high hardware complexity due to the large number of multiplications, we introduce some logarithmic quantities to convert these complicated multiplications into additions, which lead to the Log-BP algorithm.

2. Theory

When bits are transmitted over the computer network, they are subject to corruption due to interference and network problems. The corrupted bits lead to spurious data being received by the receiver and are called errors.

For error detection, the sender needs to send some additional bits along with the data bits. The receiver performs necessary checks based upon the additional redundant bits. If it finds that the data is free from errors, it removes the redundant bits before passing the message to the upper layers.

In information theory, a low-density parity-check (LDPC) code is a linear error correcting code, a method of transmitting a message over a noisy transmission channel. LDPC codes functionally are defined by a sparse parity-check matrix. This sparse matrix is often randomly generated.

An LDPC code is typically represented by a bipartite graph, usually called Tanner graph, in which one set of N variable nodes corresponds to the set of code word, another set of M check nodes corresponds to the set of parity-check constraints and each edge corresponds to a nonzero entry in the parity-check matrix H. (A bipartite graph is one in which the nodes can be partitioned into two sets, X and Y, so that the only edges of the graph are between the nodes in X and the nodes in Y.)

An LDPC code is known as (j, k)-regular LDPC code if each variable node has the degree of j and each check node has the degree of k, or in its parity-check matrix each column and each row have j and k nonzero entries, respectively. The code rate of a (j, k)-regular LDPC code is $1 - \mathrm{j/k}$ provided that the parity check matrix has full rank.

2.1 Decoding Algorithm

LDPC codes can be effectively decoded by the iterative belief-propagation (BP) algorithm. This algorithm is called message passing algorithm, and is an iterative algorithm. The reason for their name is that at each round of the algorithm messages are passed from message nodes to check nodes, and from check nodes back to message nodes. The messages from message nodes to check nodes are computed based on the observed value of the message node and some of the messages passed from the neighboring check nodes to that message node.

The messages passed along the edges in this algorithm are probabilities, or beliefs. More precisely, the message passed from a message node v to a check node c is the probability that v has a certain value given the observed value of that message node, and all the values communicated to v in the prior round from check nodes incident to v other than c. On the other hand, the message passed from c to v is the probability that v has a certain value given all the messages passed to c in the previous round from message nodes other than v.

One very important aspect of belief-propagation is its running time. Since the algorithm traverses the edges in the graph, and the graph is sparse, the number of edges traversed is small. Moreover, if the algorithm runs for a constant number of times, then each edge is traversed a constant number of times, and the algorithm uses a number of operations that is linear in the number of message nodes. Another important note about belief propagation is that the algorithm itself is entirely independent of the channel used, though the messages passed during the algorithm are completely dependent on the channel.

3. Plan of Action

3.1 Verilog/VHDL

Verilog is a Hardware Description Language; a textual format for describing electronic circuits and systems. Applied to electronic design, it is intended to be used for verification through simulation, for timing analysis, for test analysis (test ability analysis and fault grading) and for logic synthesis. Choose a simulation software to work upon and learn the syntax of Verilog or VHDL.

3.2 Introduction to Information Theory

We aim for achieving reliable communication through unreliable channel. To achieve the same using the 'system' solution we learn about basic error correcting codes and understand the need and process of error correction. Learn about repetition codes, Hamming codes and compare their performance and learn about Shannon channel capacity. Learn about information content of an event and implement these in interesting real-life examples.

3.3 Introduction to Graphs

Learn about graphs, nodes and vertices in graphs, representation of graphs by an Adjacency Matrix, indegree and outdegree of nodes, cycles in graphs, Bipartite graphs, modelling real life problems through graphs.

3.4 Introduction to LDPC

Learn about LDPC codes, what are they, how they are encoded, how to represent a parity check matrix as a Tanner graph.

3.5 Shannon's Theorem

A communication channel is usually defined as a triple consisting of an input alphabet, an output alphabet, and for each pair (i, o) of input and output elements a transition probability p(i, o). Semantically, the transition probability is the probability that the symbol o is received given that i was transmitted over the channel. Given a communication channel, Shannon proved that there exists a number, called the capacity of the channel, such that reliable transmission is possible for rates arbitrarily close to the capacity, and reliable transmission is not possible for rates above capacity. To achieve reliable communication, it is thus imperative to send input elements that are correlated. This leads to the concept of a code, defined as a (finite) set of vectors over the input alphabet. We assume that all the vectors have the same length, and call this length the block length of the code. If the number of vectors is K = 2 k , then every vector can be described with k bits. If the length of the vectors is n, then in n times use of the channel k bits have been transmitted. We say then that the code has a rate of k/n bits per channel use, or k/n bpc.

3.6 Data Compression

A file is composed of a sequence of bytes. A byte is composed of 8 bits and can have a decimal value between 0 and 255. A typical text file is composed of the ASCII character set (decimal values 0 to 127). This character set uses only seven of the eight bits in a byte

There are only two ways in which a 'compressor' can actually compress files:

1. A lossy compressor compresses some files, but maps some files to the same encoding. We'll assume that the user requires perfect recovery of the source file, so the occurrence of one of these confusable files leads to a failure (though in applications such as image compression, lossy compression is viewed as satisfactory). We'll denote by δ the probability that the source string is one of the confusable files, so a lossy compressor has a probability δ of failure. If can be made very small then a lossy compressor may be practically useful.

2. A lossless compressor maps all files to different encoding; if it shortens some files, it necessarily makes others longer. We try to design the compressor so that the probability that a file is lengthened is very small, and the probability that it is shortened is large.

4. Progress

As per the plan of action, the initial stage of our project is mostly theoretical. We have introduced ourselves to the various topics of the information theory and made ourselves familiar with a Hardware Description Language which will later facilitate the implementation of LDPC decoder using FPGA.

From the text book 'Information Theory, Inference and Learning Algorithms', by David Mackay, we have covered the introduction to information theory. In that we went through various error correcting codes for binary symmetric channel.

In that we saw repetition codes, block codes (like hamming code) in a bit detail and studied their decoding, performance and their limitations.

Further, we studied about Shannon information content, entropy and data compression to get an intuition as to why LDPC codes are of such importance.

5. Future Course of Action

1) Understanding any general message passing algorithms.
2) Understanding belief Propagation algorithms and deriving the equations.
3) Understanding the architecture in the implementation research paper
4) Implementing individual parts of the architecture in verilog and testing each part.
5) Combining all the parts developed.