An Efficient Hardware Implementation of LDPC Decoder

Monazzahalsadat Yasoubi

A Thesis

In the Department

of

Electrical and Computer Engineering

Presented in Partial Fulfillment of the Requirements

for the Degree of Master of Applied Science (Electrical and Computer Engineering) at

Concordia University

Montreal, Quebec, Canada

February 2020

CONCORDIA UNIVERSITY
SCHOOL OF GRADUATE STUDIES

This is to certify that the thesis prepared

By: Monazzahalsadat Yasoubi

Entitled: An Efficient Hardware Implementation of LDPS Decoder

and submitted in partial fulfillment of the requirements for the degree of

Master of Applied Science

Complies with the regulations of this University and meets the accepted standards with respect to originality and quality.

Signed by the final examining committee:

____ Chair
Dr. M. Medraj (MIAE)

____ Examiner, External To the Program
Dr. M. Medraj (MIAE)

____ Examiner
Dr. Y. R. Shayan

____ Supervisor
Dr. M. R. Soleymani

____ Co-Supervisor
Dr.

Approved by: ____
Dr. Yousef R. Shayan, Chair
Department of Electrical and Computer Engineering

_______13/02/2020____

____
Dr. Amir Asif, Dean
Gina Cody School of Engineering and Computer Science

Abstract

An Efficient Hardware Implementation of LDPC Decoder

Monazzahalsadat Yasoubi

Reliable communication over noisy channel is an old but still challenging issues for communication engineers. Low density parity check codes (LDPC) are linear block codes proposed by Robert G. Gallager in 1960. LDPC codes have lesser complexity compared to Turbo-codes. In most recent wireless communication standard, LDPC is used as one of the most popular forward error correction (FEC) codes due to their excellent error-correcting capability. In this thesis we focus on hardware implementation of the LDPC used in Digital Video Broadcasting - Satellite - Second Generation (DVB-S2) standard ratified in 2005. In architecture design of LDPC decoder, because of the structure of DVB-S2, a memory mapping scheme is used that allows 360 functional units implement simultaneously. The functional units are optimized to reduce hardware resource utilization on an FPGA. A novel design of Range addressable look up table (RALUT) for hyperbolic tangent function is proposed that simplifies the LDPC decoding algorithm while the performance remains the same. Commonly, RALUTs are uniformly distributed on input, however, in our proposed method, instead of representing the LUT input uniformly, we use a non-uniform scale assigning more values to those near zero. Zynq XC7Z030, a family of FPGA's, is used for Evaluation of the complexity of the proposed design. Synthesizes result show the speed increase due to use of LUT method, however, LUT demand more memory. Thus, we decrease the usage of resource by applying RALUT method.

Keyword: LDPC code, DVBS2 standard, Hardware implementation, Vivado HLS.

Acknowledgments

I would like to thank my advisors Prof. M. R Soleymani. I value our discussions on different research challenges as well as their insightful guidance. He has been extremely supportive mentors, both professionally and personally.

I could not have come this far without the love of my family. I am always grateful to my parents, my sisters, and my wonderful husband, Ali Mazaheri, for their endless love and support.

Table of Contents

List of Figures

List of Tables

List of Symbols

u: Information sequence

v: Codeword

T: Wave form Duration

r: Received sequence

$\hat{u}$: Estimated sequence

$\mathrm{H(x)}$: Entropy of the source x

C: Capacities of the channel

X: Source

$\widehat{\mathrm{X}}$: Reproduction of source X

$\mathrm{E}\left\{\mathrm{d}(\widehat{\mathrm{X}}, \widehat{\mathrm{X}})\right\}$: Average distortion between source X and the reproduction $\widehat{\mathrm{X}}$

Y : Side information at the decoder

$\mathrm{H(X|Y)}$: Entropy of the source x in presence of side information Y

G: Generator matrix

H: Parity check matrix

n: The length of the codeword

m: The number of the parity bits

R: Rate of the code is

$\lambda(x)$: Distribution polynomial for variable nodes

$\rho(x)$: Distribution polynomial for check nodes

d_v: Maximum degree of variable nodes

d_c: Maximum degree of check nodes

λ_i : The fraction of all edges incident to variable nodes with degree i.

ρ_i: The fraction of all edges incident to check nodes with degree j.

$L(x)$: Likelihood ratio of a binary random variable x

$L(x|y)$: Conditional likelihood ratio of random variable x given y

m_v: The log likelihood of the message node v conditioned on its observed value

m_{vc}^{l}: Message passes from message node v to the check node c at round l.

m_{cv}^{l}: Message passes from check node c to message node at round l.

V_c: The set of variable nodes incident to the check node c.

C_v: The set of check nodes incident to the variable node v.

A: Submatrix with dimensions $(N-K)\times K$

a_{ij}: The elements in the A submatrix

p_I: Parity bits

B: Staircase lower triangular matrix

List of Acronyms

AWGN: Additive White Gaussian Noise

CLB: Configurable logic block

DSP: Digital signal processing

DVB: Digital Video Broadcast

DVB-S2: Digital video Broadcast Second generation

FEC: Forward error correcting

FPGA: Field Programmable Gate Arrays

FF: Flip Flop

HDL: Hardware description language

IC: Integrated Circuit

IRA: Irregular repeat-accumulate codes

LDPC: Low Density Parity Check Code

LUT: Look-up Table

RTL: register-transfer level

Chapter 1

Introduction

1.1. Motivation

Reliable communication over a noisy channel is an old but still challenging issue for communication engineers. Low-density parity-check codes (LDPC) are linear block codes proposed by Robert G. Gallager in 1960. In most modern wireless communication standard, LDPC is used as one of the most popular forward error correction (FEC) code due to its excellent error-correcting capability. In this thesis, we focus on the hardware implementation of the LDPC used in Digital Video Broadcasting - Satellite - Second Generation (DVB-S2) standard ratified in 2005. The structure of the DVB-S2 standard allows a memory mapping scheme in which 360 units implement simultaneously.

Hyperbolic tangent is used in the LDPC decoder algorithm, which is expensive to compute and inexpensive for the cache. Therefore, optimizing hardware implementation of hyperbolic tangent function used in the LDPC decoder algorithm Look Up Tables (LUTs) is an excellent technique. Thus, a precomputing of a function throughout common input is evaluated to find a proper LUT. Indeed, expensive runtime operations can be replaced with inexpensive table lookups [31]. Three main methods for designing Lookup table are used to implement and approximate the function in hardware are as follows:

- Lookup table (LUT) approximation [32],
- Piece-Wise Linear (PWL) approximation [33],
- Hybrid methods, which are essentially a combination of the former two [34].

Our approach is motivated by the fact that among the three aforementioned methods used for approximation of hyperbolic tangent, i.e., LUT, PWL, and hybrid method, LUT is the fastest approach but requires more resource that other two. Therefore, we have used RALUT to compensate for this. A novel design of Range addressable look-up table (RALUT) for the hyperbolic tangent function is proposed that simplifies the LDPC decoding algorithm while the performance remains the same. Commonly, RALUTs are uniformly distributed on input; however, in our proposed method, instead of representing the LUT input uniformly, we use a non-uniform scale assigning more values to those near zero. Zynq XC7Z030, a family of FPGA's, is used for Evaluation of the complexity of the proposed design.

1.2. Related work

The emergence of large scale and high-speed data networks for processing, storage, and exchange of digital information in the military, government, and private spheres resulted in demand for efficient and reliable data storage and transmission networks. It is necessary to control the errors so that reliable transmission could be possible [40]. According to Shannon's theorem, if the transmission rate is less than the channel capacity, there is always an error correction code that can make the probability of error arbitrarily small. Besides, the application of error-correcting codes for data compression is investigated by Shannon due to the duality between source coding and channel coding. Indeed, a good channel code has the capability of being a good source code as a result of duality. The area of channel coding has achieved a state of the art where robust error-correcting codes have been designed, which can approach the capacity of different communication channels. Figure (1.1) shows a block diagram of a generic data transmission storage System. Each block is briefly described as follows [40];

Information source: It can be a person or a machine such as a computer. The output of the source can be a sequence of discrete symbols or a continuous waveform [40].

Source encoder: The source output is transformed into a sequence of binary digits called the information sequence u. It is important that the source output can be regenerated from the information sequence without any ambiguity [40].

Channel encoder: Discrete encoded sequence, called codeword v, is generated from information sequence. The goal of channel encoder is to overcome the noisy environment in which the code-word requires to be stored or transmitted [40].

Modulator: Since discrete symbols are not suitable for transmission over channel or recording on a storage device, modulator transforms each output symbol of channel encoder to a waveform of duration which is suitable for transmission [40].

Channel: The waveform generated by a modulator enters the channels or storage device and corrupts by a noisy environment.

Demodulator: Each received waveform of duration T is processed and produces an output that is discrete or continuous. The output of the demodulator is called received sequence r [40].

Channel decoder: The received sequence r is transformed into binary sequence $\hat{u}$ called an estimated sequence. The goal in channel decoder is to minimize the probability of decoding error. The difference between u and $\hat{u}$ is considered as decoding error, which causes through the noisy environment of data storage or transmission [40].

Source decoder: The estimate sequence is delivered to the destination [40].

Destination: In a well-designed system, the estimation is an exact reproduction of the source output [40].

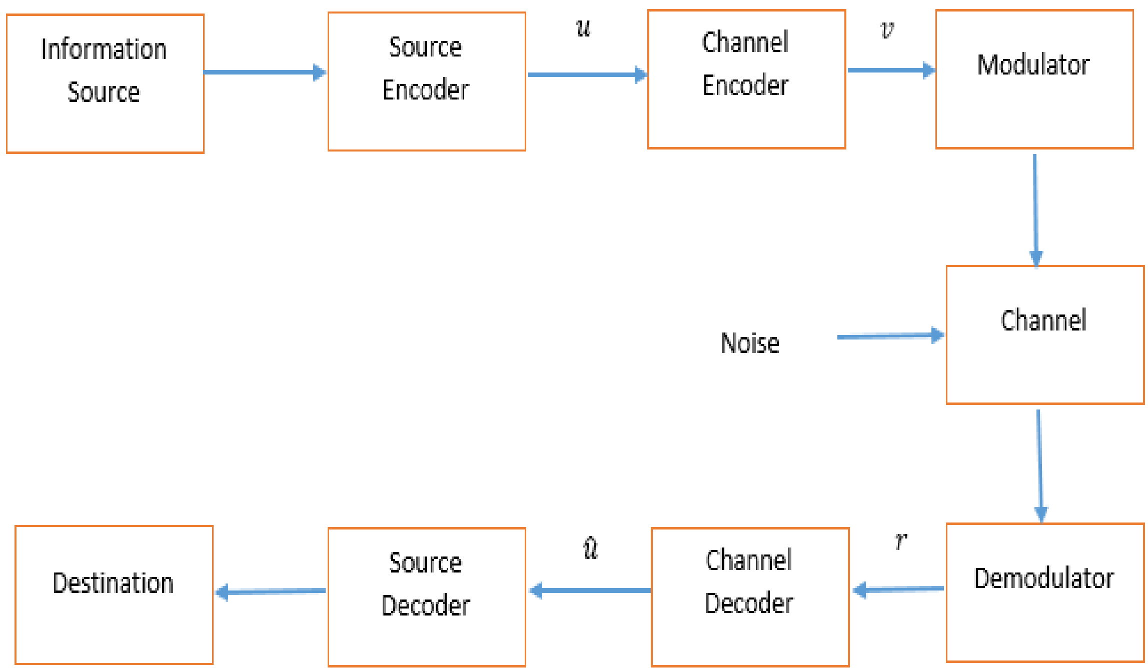


Figure (1.1). Block Diagram of a general data transmission or storage system.

1.2.A. LDPC code and data compression

LDPC is used as one of the error control techniques in different standards in digital communication, Digital Video Broadcasting, and satellite communications. Good error performance near Shannon capacity and also fast decoding are some advantages of LDPC codes. LDPC code was firstly introduced by Gallager in his Ph.D. thesis in the early 1960s [1-3]. The LDPC codes were rediscovered by MacKay and Neal [4] and Wiberg [5] independently from each other for different purposes. In 1997, Luby, Mitzenmacher, Shokrollahi, Spielman, and, Stemann, proposed Cascade constructions for the more straightforward encoding of LDPC code [6-7]. Besides, for linear encoding, the lower triangular restriction on the shape of the parity-check matrix was suggested by MacKay, Wilson, and Davey in 1998 [8].

The duality of channel coding and source coding motivates the application of powerful channel codes in source coding applications, which is used to optimize the usage of limited storage space to save time and help optimize resources in the caching method. Two different types of source coding are lossless and lossy data compression [9]. Data compression by error-correcting code is especially good when the data is transmitted over the noisy channel. Since standard data compression techniques such as Huffman code are not designed for error correction, therefore, it is reasonable that one uses error-correcting code for both data compression and error correction purposes. Moreover, Data compression based on error-correcting code design could be based on a syndrome based approach or parity-based approach. Besides, according to the channel coding theory of Shannon, the source can be reconstructed with small error probability if the rate of the data sequence is less than the capacity of the transmission channel [12]. Consider the model with two independent channels operating in parallel, and the reliable transmission is possible if the entropy of the source is less than the sum of capacities of the two channels $\mathrm{H(x)} \leq \mathrm{C}_1 + \mathrm{C}_2$. However, if the source entropy is above $\mathrm{C}_1 + \mathrm{C}_2$, the reliable transmission is not possible. If one of the channels has an uncoded version of the source as side information at the decoder, known as systematic communication, there are two approaches for error protection of noisy transmission. One is based on Slepian Wolf [13], and the other is based on Wyner Ziv [14].

Wyner Ziv examines the question of how many bits are needed to encode source X under the constraint that the average distortion between X and the reproduction $\widehat{X}$ satisfies $E\{d(\widehat{X},\widehat{X})\} \leq D$. Slepian Wolf coding is actually a channel coding problem that considered the question of how many bits per source character are required for the two correlated encoded message sequences to be decoded accurately by the joint decoder [15]. In LDPC decoding of Slepian Wolf, when we have side information at the decoder (Y), instead of transmitting the whole length of the original message, only the syndrome or check nodes are transmitted ($H(X|Y)$). The Slepian wolf decoding algorithm of LDPC code is almost the same as the channel decoding algorithm of LDPC code with some differences, which is explained more in Chapter two.

1.2.B. Reason for using high level synthesizing

High-level synthesis (HLS) is a designing algorithm that describes the desired behavior of the process results in hardware implementation. HLS is also referred to as C synthesis, electronic system-level (ESL) synthesis, algorithmic synthesis, or behavioral synthesis [23]. Synthesis begins with a high-level specification of the problem, where behavior is generally decoupled from, for example, clock-level timing. Although earlier introduced HLS accepted considerable variation for input specification languages, recent commercial applications and research generally prefer to accept synthesizable subsets of ANSI C, C++, SystemC, and MATLAB [24]. Generally, in HLS synthesis, first, the code is analyzed and architecturally constrained. Then, it is scheduled to trans-compile into a register-transfer level (RTL) design in a hardware description language (HDL). Finally, it is generally synthesized to the gate level by applying the logic synthesis tool.

The RTL tool implementation or reliable logic synthesis tool allows designers to describe their designs at a high level of abstraction. The general usage of abstraction is gate level, register-transfer level (RTL), and algorithmic level**.**

By using the tool implementation of the RTL, the designers have better control over the optimization of their design architecture. The module functionality and the interconnect protocol are usually developed by hardware designers. Thus, the actual goal of using HLS is to permit hardware designers to efficiently build and verify hardware where the tool does the RTL implementation.

The HLS tools create cycle-by-cycle detail for hardware implementation automatically [25]. They transform untimed or partially timed functional code into fully timed RTL implementations. Finally, at the end of the synthesis process, it is essential to verify the RTL implementation. Then, to create a gate-level implementation, the RTL implementations are used directly in a conventional logic synthesis flow.

Following is a list of compilers that are currently available in the market for high-level synthesis.

- Xilinx Vivado HLS
- Xilinx System Generator for DSP
- Intel HLS Compiler
- LabVIEW FPGA
- Mathworks HDL Coder
- Cadence Stratus
- Mentor Graphics Catapult

- Synopsys Synphony C Compiler
- Panda Bamboo
- LegUp

We have chosen to Vivado HLS because of the following features:

- Vivado HLS provides an easier way to implement DSP algorithms
- Less code = fewer bugs.
- Code is more readable, wider audience
- Quality of Result is comparable with hand-coded logic
- Provides easy migration between different FPGA families
- Requires some knowledge of FPGA architecture
- Xilinx FPGAs are heavily used at DESY and on MicroTCA (MicroTCA is an open standard embedded computing specification) AMC boards.
- The created IP integrates nicely with the rest of the IPs in the Xilinx ecosystem.
- Vivado HLS is significantly cheaper than other HLS software suites; therefore, it is very likely that industrial partners will have access to it.

1.2.C. Caching method

In Chapter three, an example of the application of the Slepian Wolf decoding algorithm of LDPC code by DVBS2 standard in a caching scheme is presented [16]. Caching is a reliable solution for communication during busy periods by taking advantage of memory across the network, which leads to more smooth communication network systems [17-22]. The caching method has two phases. The first phase is called the placement phase, where the data is stored in the cache across the network. The main limitation of this phase is the size of the cache memory. In the second phase, which is called the Delivery phase, the users' requests can be partially served through caches near the users. Examples of application of the caching method are streaming media and distributed database, which results in a decreasing delivery rate.

In media Streaming user requests time is most likely at night rather than early in the morning. During congestion periods, the bandwidth-hungry features of media result in more congestion, high latency, and a poor experience for users. Caching is an applicable solution during off-peak hour time. Some examples of the distributed database are meteorological conditions measurement information of the globe, information of traffic sensors spread across several countries, information of the shopping history of the customers, information on the mobility pattern of the mobile devices in cellular networks. Since the database is extensive, it might need several different network calls to load the requested data of the memory before the requested data can be transmitted to the users. These network calls cause latency or stalls the process. In the modern database, it is handled by storing the most common queries in fast memory**.** As an example of caching, consider that a user more probably demands the weather measurement of his hometown rather than of a remote area. Therefore, the information of weather measurement of the user hometown is cached in memory close to it. To the best of our knowledge, little attention was given to the source coding problem in the presence of caching; however, compressing information can highly ease the traffic. We proposed a general approach to decreases the delivery rate by applying source coding using LDPC code to the correlated binary source**.** In the delivery phase, we applied the DVBS2 standard which is adopted a several standards due to its powerful features such as transmission rate close to the theoretical Shannon limit [19]. The results show that there is a direct relation between correlated coefficient α and delivery rate.

1.3. Thesis contributions

In this thesis, a new hardware implementation of the LDPC code used in DVB-S2 is presented. We have used a Range addressable LUT scheme to approximate the Hyperbolic Tangent function. Our approach is motivated by the fact that among the three methods used for approximation of hyperbolic tangent, i.e., LUT, PWL, and hybrid method, LUT is the fastest approach but requires more resources than other two. Therefore, we have used RALUT in order to compensate for this.

In addition, in Chapter three, an example of the application of the Slepian Wolf decoding algorithm of LDPC code by DVBS2 standard in the caching method is presented [16]. The results show that there is a direct relationship between the delivery rate and correlated coefficient of the source and its side information available at the decoder. In the following, the thesis contribution is listed:

- Hardware implementation of LDPC code.
- Range addressable LUT scheme is used to approximate hyperbolic tangent function.
- Example of application of Slepian Wolf decoding algorithm of LDPC code by DVBS2 standard in caching method.

1.4. Thesis outline

Chapter two presents background information related to this thesis. First, Low-Density Parity Check (LDPC) code is described. After that, different methods of LDPC code representation, such as bipartite graph representation, matrix representation, and degree distribution polynomial representation of the LDPC H matrix are presented. Then, the Slepian Wolf coding theorem and Wyner Ziv theorem coding are presented used in the Well-designed Caching example of Chapter three. Finally, some background information for hardware implementation of LDPC code is presented from Section 2.6 to the end of Chapter two, including FPGA, Xilinx FPGA architecture, and three primary methods for designing Lookup table.

In Chapter three, reliable communication over the noisy channel is considered to be implemented by the hardware of one standard of LDPC codes called DVB-S2 [16]. The design and architecture of FPGA implementation of an LDPC decoder are presented. Besides, the hardware implementation of the LDPC decoder is simplified using Range Addressable Look Up Tables. In Section 3.4, Range addressable Lookup Table approximation is applied to update variable nodes in the LDPC decoder. Because of undesired results, a new Range addressable Lookup Table approximation is proposed in order to update variable nodes in the LDPC decoder. Finally, in chapter three, data compression with side information at the decoder is used as a caching solution in a Well-designed Caching example. Chapter four presents conclusion and future direction for the thesis.

In appendix A, the basic measures of information theory proposed by Shannon are described [41]. In appendix B, the values from Annex B and C of the DVB-S2 standard [27] are reproduced. Appendix C shows the RALUT, which is used for calculation of $\tanh x$ for updating variable nodes messages of LDPC decoder [31-34].

Chapter 2

Background information

Summary

Chapter two presents background information related to this thesis. First, Low-Density Parity Check (LDPC) code is described. After that, different methods of LDPC code representation, such as bipartite graph representation, matrix representation, and degree distribution polynomial representation of the LDPC H matrix are presented. Then, the Slepian Wolf coding theorem and Wyner Ziv theorem coding are presented that are used in the Well-designed Caching example of Chapter three. Finally, some background information for hardware implementation of LDPC code is presented from Section 2.6 to the end of Chapter two, including FPGA, Xilinx FPGA architecture, and three main methods for designing Lookup table.

2.1. Low Density Parity Check (LDPC) code

In recent years, because of their near Shannon capacity performance and fast decoding, LDPC code has been approved by many standards as forward error correcting (FEC) technique, these include Digital Video Broadcasting for Satellite Second Generation and Long-Term Evolution

(LTE). Low Density Parity Check (LDPC) codes were first introduced by Gallager in his Ph.D. thesis in the early 1960s [1-3]. Gallegar's introduction of iterative decoding algorithms (or message-passing decoder) was the essential novelty of his discovery. His outstanding innovation was ignored for almost 20 years due to the complexity of encoding. Finally, LDPC codes were rediscovered by MacKay and Neal [4] and Wiberg [5] independently from each other for different purposes. The result of their research showed that long LDPC codes with iterative decoding have an error performance, which is only a fraction of decibel away from the Shannon limit, which made it practical in many communication and digital storage systems with high reliability. Besides, the low density of LDPC codes is a result of their sparse parity check matrix. The characteristic, as mentioned earlier, means that the parity check matrix contains only a few 1's in comparison to the number of 0's.

In data communication systems, the message bits are encoded at the encoder by adding redundancy to the message. However, in practical implementation, the encoding of LDPC codes is ambiguous; i.e., it has high complexity. Thus, several researchers proposed different solutions for reducing the LDPC encoding complexity. In 1997, Luby, Mitzenmacher, Shokrollahi, Spielman, and Stemann, proposed Cascade constructions [6-7] instead of a bipartite graph, the drawback of the case-cade method is a reduction in the performance compared to the standard LDPC codes. The lower triangular restriction on the shape of the parity-check matrix was suggested by MacKay, Wilson, and Davey in 1998, which guarantees linear encoding complexity [8].

After channel encoding, the codeword is transmitted to the receiver. The destination receives a noisy version of the codeword. The decoder corrects the errors resulted from noise in order to retrieve the original message. According to Shannon's theorem, if the transmission rate is less than the capacity, there is always an error correction code that can make the probability of error arbitrarily small. In addition, the application of error-correcting codes in data compression is investigated by Shannon due to duality between source coding and channel coding [8]. Indeed, a channel code that provides high rate has the capability to provide high rate in source coding application as a result of this duality. The area of channel coding has achieved a state of maturity where powerful error-correcting codes have been designed, which can approach the capacity of different communication channels. For example, the rate of an appropriately designed low density parity check code reaches close to the capacity of additive white Gaussian noise (AWGN) channel. Thus, the duality of channel coding and source coding motivates the application of powerful channel coding schemes in source coding applications, which is used to optimize the usage of limited storage space to optimize resources. Two different types of source coding are lossless and lossy data compression. In Lossless data compression, the data after decompression is exactly the same as the original data. In fact, redundant data is removed in compression and added during decompression. Run-length, Huffman, Lampel Ziv are some examples of lossless data compression [9]. Lossless methods are used when we can't afford to lose any data, such as medical documents and computer programs and legal documents. Lossy data compression methods are used for compressing images and video files since our eyes cannot distinguish subtle changes; therefore, lossy data is acceptable. These methods are cheaper, which needs less time and space as well. MP3, for compressing audio, MPEG (video compression), and JPEG (pictures and graphics compression) are several methods using lossy data compression.

Standard data compression techniques such as Huffman code are not designed for error correction. When the data is transmitted over the noisy channel, it is reasonable to apply a code which is useful for both compression and error correction purpose. Therefore, the error-correcting code can be used for both data compression and error correction purposes. Data compression based on error correcting code design could be based on a syndrome based approach or parity-based approach. For the syndrome based approach, the bins are indexed by syndrome bits. In parity based approach the containers are indexed by parity bits. The parity based approach can protect compressed data against noise while the syndrome based approach just only does data compression. However, the parity based approach has more calculation complexity than a syndrome based approach.

Considering that LDPC codes have an error performance only a fraction of decibel away from the Shannon limit, the features of LDPC codes, including representation, encoding, and decoding is explained in the following.

2.2. LDPC code Representation

There are different methods to represent LDPC codes; one is matrix representation, which is similar to other linear block codes. Furthermore, there are a polynomial representation and graphical representation through a bipartite graph. These representation helps to design and to analyze the code [10 -11].

2.2. A. Bipartite graph representation

The graph representation of LDPC code was initially introduced by Tanner in [21]. The Tanner graph or bipartite graph is used to explain the iterative decoding algorithm for LDPC code. The bipartite graph has two sets of nodes, including a set of variable nodes and set of check nodes. When two nodes are connected, there is an edge between these two nodes, which is called an incident between two nodes. Besides, the number of edges that are incident to a node is the degree of the node. The tanner graph can be derived from the parity check matrix. The graph can be induced by using the following rules:

1- The n columns of parity check matrix corresponding to the number of bits in a codeword represents by $v_1, v_2, \dots, v_n$. Besides, the m rows of parity check matrix correspond to parity check constraint represents by $c_1, c_2, \dots, c_m$. It means that in that the degree of a variable node (or check node) is equal to the corresponding column (or row) weight.
2- There exist an edge or incident between one variable node and one check node if and only if the corresponding entry in the parity check matrix is equal to one. It means that at most, there is one edge between any two nodes.

2.2.B. Matrix representation

Linear channel codes are usually expressed by generator matrix G and the parity check matrix H. The multiplication of these two matrixes, must be equal to zero.

$$G.H^T = 0 \qquad \text{Eq (2.1)}$$

However, the LDPC code is just defined by parity check matrix H. The parity check matrix of LDPC codes is sparse. However, the generator matrix can have a lot of ones as its entries, which causes a high complexity of computation. Consider the dimension of the parity check matrix H is $\times n$. Where n is the length of the codeword, and m is the number of the parity bits. If the parity check matrix has n columns and m rows, the rate of the code is

$$R = \frac{n-m}{n} \qquad \text{Eq(2.2)}$$

In this thesis, the field is a Galois field. Thus, the elements of the LDPC parity check matrix are either 0's or 1's. If the received message is the null space of the parity check matrix of a linear code $\vec{v}H^T = 0$, then it is an actual codeword of the aforementioned linear code. Where $\vec{v} = [v_1, v_2, \dots, v_n]$ is an-tuplee codeword and $v_i \in \{0, 1\}$. In every Galleger LDPC code, the parity check matrix H has the following structure:

1- Each row consists of ρ ones.
2- Each column consists of λ ones. Properties 1 and 2 determine the degree distribution of LDPC codes.
3- The number of ones in common between any two columns is no more than one, which guarantees the cycle free of the parity check matrix.
4- The length of LDPC codes is much larger than ρ and λ, ensuring the sparsity of the parity check matrix.

As an example, the Tanner graph of the following H matrix is shown in Figure (2.1) in which the variable nodes and the check nodes are shown by blue circles and green circles, respectively.

$$H = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix} \quad \text{Eq(2.3)}$$

The Tanner graph of the above example is a Galleger LDPC code. The number of ones in each row and column are four and two, respectively. So, the code is a regular LDPC code since the number of ones in each row and column is the same as other rows and columns. The code rate is half. Besides, in Figure (2.1), the four blue edges indicate a cycle. Indeed, the cycle is four, which is the shortest cycle. And so, the grith of the graph is four.

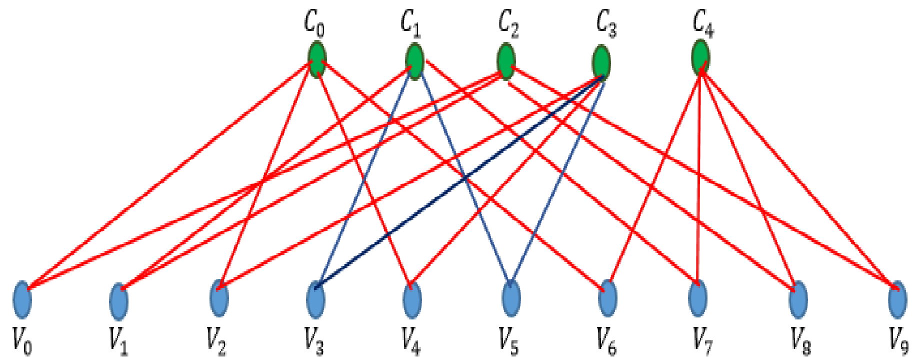


Figure (2.1). Graphical representation of LDPC code.

2.2. C. Degree distribution polynomial representation of LDPC H matrix

Based on a pair degree distribution polynomial and a given code length, we can calculate some parameters of the given LDPC code. Indeed, a pair degree distribution polynomial describes an ensemble of LDPC code but not a specific LDPC code. However, the parity check matrix and the tanner graph define a specific LDPC code. The degree distribution polynomial initially was introduced by Richardson to represent an ensemble of LDPC codes [31]. The degree distribution polynomial is used to specify the degree distribution of the variable nodes and check nodes in the Tanner bipartite graph or the parity check matrix. Equation (2.4) and Equation (2.5) represent the formulation of degree distribution polynomial for variable nodes and for check nodes, respectively.

$$\lambda(x) = \sum_{i=2}^{d_v} \lambda_i . x^{i-1} \qquad \text{Eq(2.4)}$$

$$\rho(x) = \sum_{i=2}^{d_c} \rho_i . x^{i-1} \qquad \text{Eq(2.5)}$$

Where d_v and d_c are the maximum degree of variable nodes and check nodes, respectively. λ_i is the fraction of all edges incident to variable nodes with a degree i. Also, ρ_i is the fraction of all edges incident to check nodes with degree j.

There are two types of LDPC codes: one is regular, and the other is called irregular. The performance of irregular LDPC codes is usually better than that of a regular LDPC code. In the regular LDPC code, the degree of each row is constant and equal to ρ. Also, the degree of each variable node or column is constant and equal to λ. The total number of ones in the parity check matrix is equal to $.\lambda = m.\rho \rightarrow m = n.\lambda/\rho$. Thus, by substituting $m = n.\lambda/\rho$ in $R = 1 - m/n$, the code rate can be computed as $R = 1 - \lambda/\rho$, which is called the design rate. However, the actual rate is usually lower than the design rate due to the dependencies among rows of the parity check matrix. The ensemble of a regular LDPC code is (n, λ, ρ), where n is referred to the length of the LDPC code and λ, ρ is the column and row weight of the parity check matrix, respectively. In irregular LDPC code, the degree of check nodes and variable nodes are not constant. An ensemble of irregular LDPC code is defined by the degree distribution of its variable nodes $\{\lambda_1, \lambda_2, \dots, \lambda_{d_v}\}$ and the degree distribution of its check nodes $\{\rho_1, \rho_2, \dots, \rho_{d_c}\}$, where λ_i is the fraction of edges incident on variable nodes of degree i and ρ_j denotes the fraction of edges incident on check nodes of degree j. Consider E as the number of one's in the parity check matrix of LDPC code or the number of edges in Tanner graph, Similar to regular LDPC code we have

$$n = E \sum_i i \frac{\lambda_i}{i} = E \int_0^1 \lambda(x) dx \qquad \text{Eq(2.6)}$$

$$m = E\sum_i i\frac{\rho_i}{i} = E\int_0^1 \rho(x)dx \qquad \text{Eq(2.7)}$$

Thus, the design rate of an irregular LDPC code is shown below

$$R = 1 - \frac{m}{n} = 1 - \frac{\int_0^1 \lambda(x)dx}{\int_0^1 \rho(x)dx} \qquad \text{Eq(2.8)}$$

2.3. Decoding Algorithm of LDPC code

The decoding of LDPC codes is based on message-passing iterative decoding algorithms [3]. In message-passing iterative algorithms, messages are exchanged between the variable nodes and check nodes. There are two ways of decoding LDPC codes. The first one is hard decision decoding, such as Majority-logic decoding and bit-flipping (BF) decoding. The second one is soft decision decoding, such as weighted bit-flipping decoding and a posteriori probability (APP) decoding algorithms. In the following, the Bit-Flip decoding algorithm of the LDPC code and Belief propagation decoding algorithm based on Log-likelihood is covered in detail.

2.3. A. Bit-Flip decoding algorithm (Hard Decoding)

This method was devised by Gallager in the early 1960s [1-2]. The steps of the Bit-Flip algorithms are as follow:

Step 1: Compute syndrome by $r.H^T = s$ in which is the received bits. If all parity checksums are zero, stop the decoding algorithm.

Step 2: Find the number of failed parity check equations for each node. Determine the number of failed check node for each message node by $f_i, i = 1, 2, \dots, n-1$

Step 3: Identify the set S of the variable node for which f_i is the largest.

Step 4: Flip bits in set S.

Step 5: Repeat steps 1 to 4 until the parity checksums are zero (decoding success) or a maximum number of iterations reaches (decoding failure).

If the syndrome or the value of the check nodes are all zero, it means that there is no error, but if detectable error pattern occurs there will be parity check failure in the syndrome $(s_1, s_2, \ldots, s_j)$, and some of the syndrome bits will be equal to 1. In the above-mentioned decoding algorithm, the decoder continues computing the parity checksums, and the process is repeated until all the parity checksums become equal to zero or a present maximum number of iterations is reached (decoding failure).

2.3. B. Belief propagation decoding algorithm based on Log-likelihood

The algorithm originally presents in Gallager's work, which is an important subclass of message passing algorithms. In the Belief propagation algorithm, the messages passed through the edges are probabilities or beliefs. One important feature related to the belief propagation decoding algorithm of LDPC code is its running time. The algorithm moves from the variable nodes to the check nodes and vice versa. The sparse parity check matrix leading to a sparse graph resulted in a small number of movements. Furthermore, the algorithm itself is completely independent of the channel, but the messages passed through the algorithm are entirely dependent on the channel. Indeed, the messages sent from the variable nodes to a check node c is the probability which that node received from check nodes in the previous iteration except the one it wants to send the message to. The same is true for message passing from check node c to the variable node v. Likelihood ratio of a binary random variable x is represented in Equation (2.9). Also, the conditional likelihood ratio of random variable x given y is expressed in Equation (2.10). Consider that x is an equiprobable random variable then $\mathrm{L}(x|y) = \mathrm{L}(y|x)$ By Bayes' rule.

$$L(x) = \frac{\mathrm{p}(x = 0)}{\mathrm{p}(x = 1)} \qquad \text{Eq(2.9)}$$

$$L(x|y) = \frac{\mathrm{p}(x = 0|y)}{\mathrm{p}(x = 1|y)} \qquad \text{Eq(2.10)}$$

The inputs of the LDPC decoder (l_i) are a log-likelihood ratio (LLR) values. Let the transmitted codeword be v = $v_0, v_1, v_2, \dots, v_{N-1}$ and the soft-decision received sequence be y, then λ_i for each code bit is given by

$$l_i = \log\left(\frac{P_c(m_v = 0|m_y)}{P_c(m_v = 1|m_y)}\right) \qquad \text{Eq(2.11)}$$

Where m_v is the log-likelihood of the message node v conditioned on its observed value m_y, which is independent of check node c. P_c is the cross over the probability of the BSC. It is obvious that if the observed node's value is zero ($m_y = 0$), the message sends to all adjacent check node $\ln P_c \,/\, \ln(1 - P_c)$ value. While, if the node value is one ($m_y = 1$), the message sends to all adjacent check node, the negative value of when the node's value is 0 ($\ln(1 - P_c) \,/\, \ln P_c$). Indeed, the LLR value indicates that the given received value is more probable to be zero or one. In the simulation of LDPC code, initially, a sequence of random bits of length K is generated. The K bits are considered as the message bits. Then, parity bits of length $N - K$ are produced by LDPC encoder based on the message bits. The codeword of length N is transmitted. Then, the output of the channel is the input to the decoder. According to [3] the belief propagation decoding algorithm of LDPC code has the following steps as follow:

Step1: Find the value of syndromes. If the value of syndrome bits are all zero, it means that the received bits are an actual codeword, and the channel does not cause any effect on the transmitted codeword during the transmitting process. If the syndrome or parity check bits are not zero, then go to step two.

Step 2:

Round 0: Find LLR.

Round 1: Update Variable nodes

In round one of step 2, find the messages which send from parity check node c to the adjacent message node v. The update equation of variable nodes is given in Equation (2.12).

$$m_{cv}^{l} = \frac{1 + \prod_{\acute{v} \in V_c \backslash \{v\}} \tanh(m_{\acute{v}c}^{l-1}/2)}{1 - \prod_{\acute{v} \in V_c \backslash \{v\}} \tanh\left(m_{\acute{v}c}^{l-1}/2\right)} \qquad \text{Eq(2.12)}$$

Where m_{vc}^{l} is a message which passed from message node v to the check node c at round l. Similarly, m_{cv}^{l} is the message which passed from check node c to the message node at round l. Where V_c is the set of variable nodes incident to the check node c.

Round 2: Update Check nodes

In round two of step 2, find the messages which send from each message node v to the all adjacent parity check nodes c. The Equation (2.13) shows update check nodes equation.

$$m_{vc} = \begin{cases} m_v & if\ l = 1 \\ m_v + \sum_{\acute{c} \in C_v \backslash \{c\}} m_{\acute{c}v}^{l-1} & if\ l \geq 1 \end{cases} \qquad \text{Eq(2.13)}$$

Where C_v is the set of check nodes incident to the variable node v.

Step 3: Hard decision making

If the value of the m_{vc} is positive, the value of variable node v is considered as zero. Similarly, if the value of the m_{vc} is negative the value of variable node v is considered as one.

Step 4: In this step, the value of check nodes are calculated by $(c_0, \dots, c_{N-K} = (v_0, v_1, v_2, \dots, v_{N-1})H^T$.

Step 5: Stop conditions

Similar to the first step, if the value of all calculated check nodes is zero, it means it is an actual code-word. Therefore, the decoding process is finished. Besides, the other stop condition is when the number of iteration of the decoding algorithm reaches the max number of iteration. Otherwise, go to step 2 and repeat until the stop condition of the decoding algorithm reaches.

2.4. Slepian Wolf coding theorem

Slepian Wolf coding is a channel coding problem that considered the question of how many bits per source character are required for the two correlated encoded message sequences to be decoded accurately by the joint decoder. The two sources do not communicate with each other. However, they are correlated and decoded jointly while encoded separately [15]. Let (X_1, Y_1), (X_2, Y_2), … be an i.i.d sequence of jointly distributed random variables X and Y with joint distribution function $p(x, y)$. Assume that X^n and Y^n are encoded separately without knowledge of each other, and the compressed output is sent to a joint decoder for reconstruction. The explained problem is called Distributed Source Coding (DSC) problem. Indeed, compression of the outputs of two or more physically separated correlated sources while they do not communicate with each other is known as distributed source coding, which could be lossless or lossy. These sources send their compressed outputs to a joint decoder for joint decoding. The final goal in communication is to minimize the energy required by the sources to achieve reliable communications. Figure (2.2) shows a distributed source coding problem.

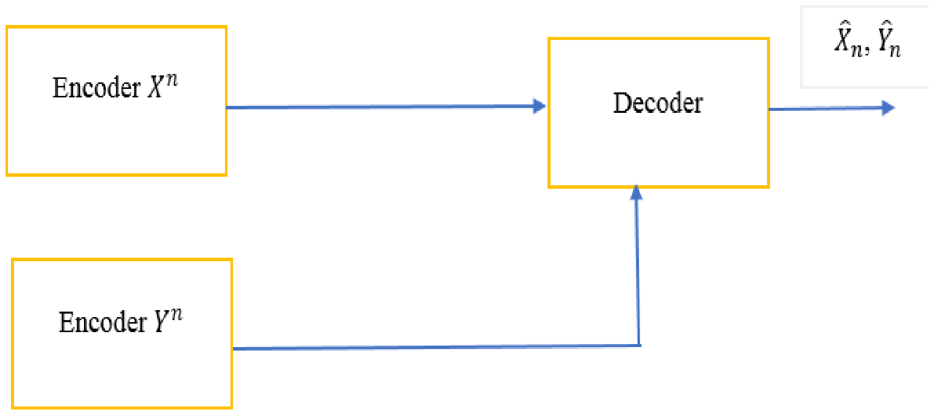


Figure (2.2). Distributed source coding problem with two sources.

Slepian Wolf coding theorem: For the distributed source coding problem of the source (X, Y) the achievable rate region is given by

$$R_X \geq H(X|Y) \qquad \text{Eq(2.14)}$$

$$R_Y \geq H(Y|X)$$

$$R_X + R_Y \geq H(X, Y)$$

According to the separation theorem in Slepian Wolf, where the user has access to the side information Y, the entropy of the source H(X) is replaced by H(X|Y). $\mathrm{H(x) = H(X|Y) + I(X; Z)}$, where $\mathrm{H(X|Y) \leq C_1}$ [15]. Cover proved that this theorem also holds for stationary and ergodic source if we replace entropies with entropy rates and conditional entropies with the conditional entropy rates [12].

2.5. Wyner Ziv Coding

Wyner Ziv examines the question of how many bits are needed to encode source X under the constraint that the average distortion between X and the reproduction $\widehat{X}$ satisfies $E\{d(\widehat{X},\widehat{X})\} \leq D$. Two possible questions related to source coding with side information were proposed in Wyner Ziv approach. The first one is when both encoder and decoder have access to side information, and the second question is when just the decoder has access to side information. Let $R^*_{X|Y}(D)$ as the smallest rate-distortion function of coding with side information Y available at the encoder (the former one) and R^*_{wz} as the achievable lower bound of the bit rate for an expected distortion D when just the decoder has access to side information (the latter one). In general, based on Wyner-Ziv $R^*_{wz} \geq R^*_{X|Y}(D)$ which means that allowable rate distortion function is decreased while both the encoder and decoder have access to side information. That is in contrast to the Slepian and Wolf situation that knowledge of the side information at the encoder does not have any rate reduction of accurate source reconstruction. One interesting case in Wyner Ziv is when sources are jointly Gaussian. In this case, $R^*_{wz} = R^*_{X|Y}(D)$ which is a similar case to the lossless data compression of Slepian-Wolf. Thus, the transmission rate of lossy compression of a Gaussian source with side information cannot be lowered even if the encoder has access to the side information.

2.6. What is an FPGA?

Field Programmable Gate Arrays (FPGAs) are semiconductor devices. FPGA consists of configurable logic blocks (CLBs). The interconnections between CLBs are programmable. Therefore, after manufacturing FPGAs can be reprogrammed to desired application or functionality. All these elements together make the basic architecture of an FPGA which is represented in Figure (2.3).

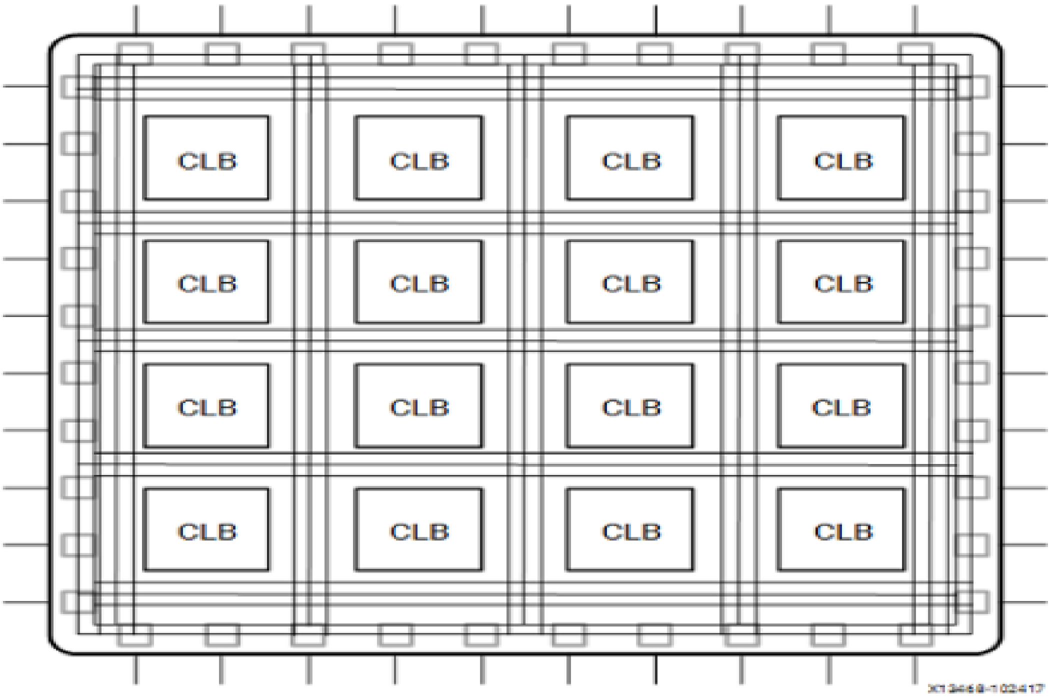


Figure (2.3). Basic FPGA Architecture [39].

The traditional design flow of an FPGA is more similar to Integrated Circuit (IC) rather than a processor. However, the architecture of an FPGA is more cost efficient than an IC, while the efficiency of them are the same in most case. Another benefit of the FPGA in comparison to the IC is that FPGA has a dynamic reconfiguration ability. The dynamic reconfiguration ability of the FPGA is the same as loading a program in a processor that is convenient to implement any kind of algorithm. However, the dynamic reconfiguration affects the availability of the resource in the FPGA fabric partially or totally. Therefore, computational throughput, required resources, and achievable clock frequency affect the efficiency of the resulting implementation.

2.7. Xilinx FPGA Architecture

Xilinx FPGAs are heterogeneous compute platforms that include Block RAMs, DSP Slices, PCI Express support, and programmable fabric. They enable parallelism and pipelining of applications across the entire platform as all of these compute resources can be used simultaneously. SDAccel is the tool provided by Xilinx to object and assist these compute resources for OpenCL programs.

The basic structure of an FPGA is composed of the following elements:

- Look-up table (LUT) – LUT performs logic operations.
- Flip-Flop (FF) – This register element stores the result of the LUT.
- Wires – Wires connect elements.
- Input/Output (I/O) pads – These physical ports get data in and out of the FPGA.

The combination of the above-mentioned elements, including LUT, FF, wires, and I/O pads, results in the basic FPGA architecture. The mentioned elements, LUT and FF are described briefly in the following pages.

LUT

The Look-up table LUT is the basic building block of an FPGA. By using LUT, which is a small memory, we can implement any logic function of M Boolean variables. Essentially, LUT is a truth table. Therefore, in LUT, different arrangements of the inputs resulted in various functions, which is generated output values. M represents the number of inputs to the LUT, which is the limit on the size of the truth table. Indeed, the number of memory locations accessed by the table for a LUT with N inputs is 2^N. This permits the table to implement 2^{N^N}functions. Note that a typical value for M in Xilinx FPGAs is 6. Figure (2.4) shows the functional Representation of a LUT as a Collection of Memory Cells.

Here, we try to explain the hardware implementation of a LUT. It can be considered as a collection of memory cells that is linked to a set of multiplexers. The inputs to the LUT can be regarded as a selector bits on the multiplexer so that the outcome can be selected at a given point in time. This representation of LUT makes it easier to consider LUT as a compute engine function and a data storage element.

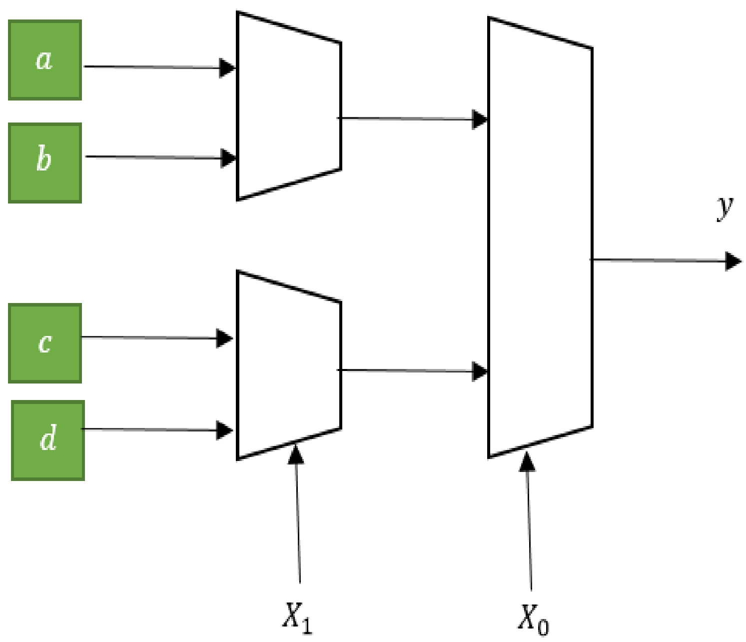


Figure (2.4). Illustration of a functional LUT as a collection of memory cells.

Flip Flop

The basic structure of a flip-flop is shown in Figure (2.5) which represents a data input (d_in), clock input (clk), clock enables (clk_en), reset, and data output (d_out). In each clock pulse, the input value.

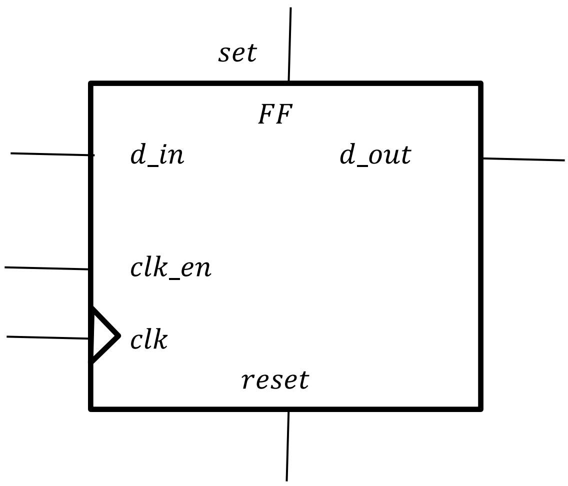


Figure (2.5). Structure of a Flip-Flop.

During normal operation, any value at the data input port is latched and passed to the output on every pulse of the clock. The clock enables pin permits the flip-flop to hold specific value for more than one clock pulse. New data inputs are only latched and passed to the data output port when both clock and clock enable are equal to one.

Present FPGA architectures have the basic elements along with additional computational and data storage blocks. The extra elements added to contemporary FPGA is shown in Figure (3.4). These additional elements, which rise the computational density and efficiency of the device, are discussed in the following sections,

- Embedded memories for distributed data storage
- Phase-locked loops (PLLs) for driving the FPGA fabric at different clock rates
- High-speed serial transceivers

- Off-chip memory controllers
- Multiply-accumulate blocks

Figure (2.6) shows the combination of these elements on a recent FPGA architecture. This provides the FPGA with the flexibility to implement any software algorithm running on a processor. Note that all of these elements across the entire FPGA can be used concurrently.

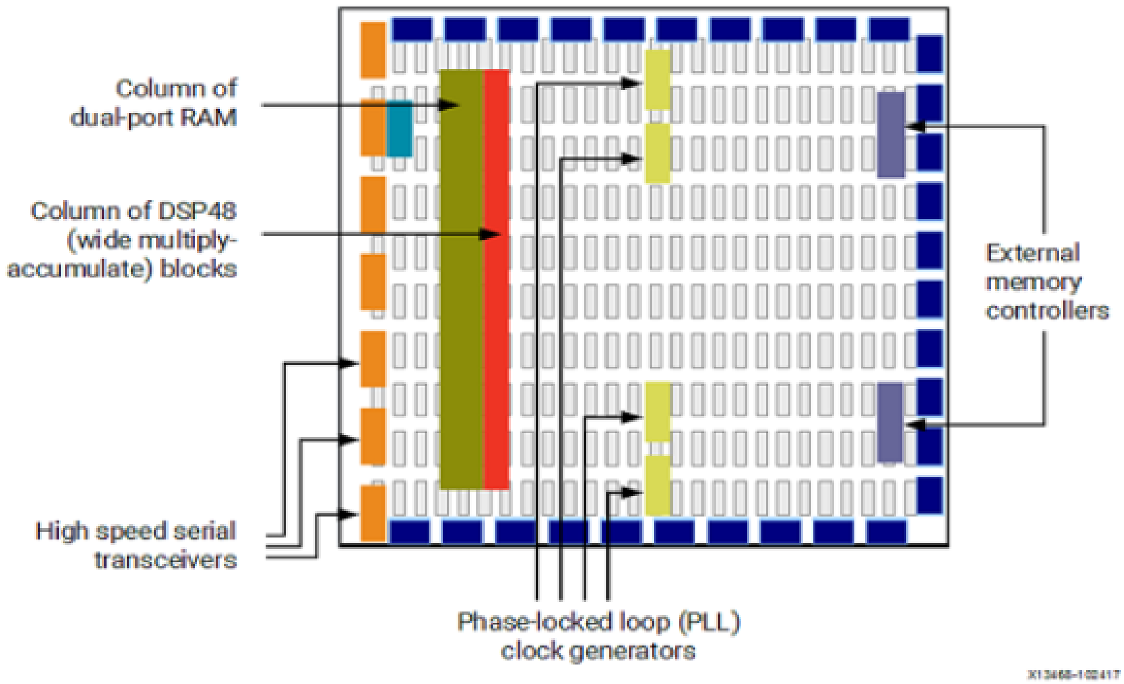


Figure (2.6). Contemporary FPGA Architecture [39].

DSP48 Block

DSP48 block, which is shown below, is the most complex computational block available in a Xilinx FPGA.

The DSP48 block, which is embedded in the fabric of the FPGA, is an arithmetic logic unit. It is composed of a chain of three different blocks, including add/subtract unit, multiplier, and final add/subtract/accumulate engine.

The computational chain in the DSP48 holds an add/subtract unit, which is linked to a multiplier. The multiplier is linked to a final add/subtract/accumulate engine. This chain allows a single DSP48 unit to implement functions of the form, which is represented in Figure (2.7):

$P = B \times (A + D) + C$ or $P += B \times (A + D)$

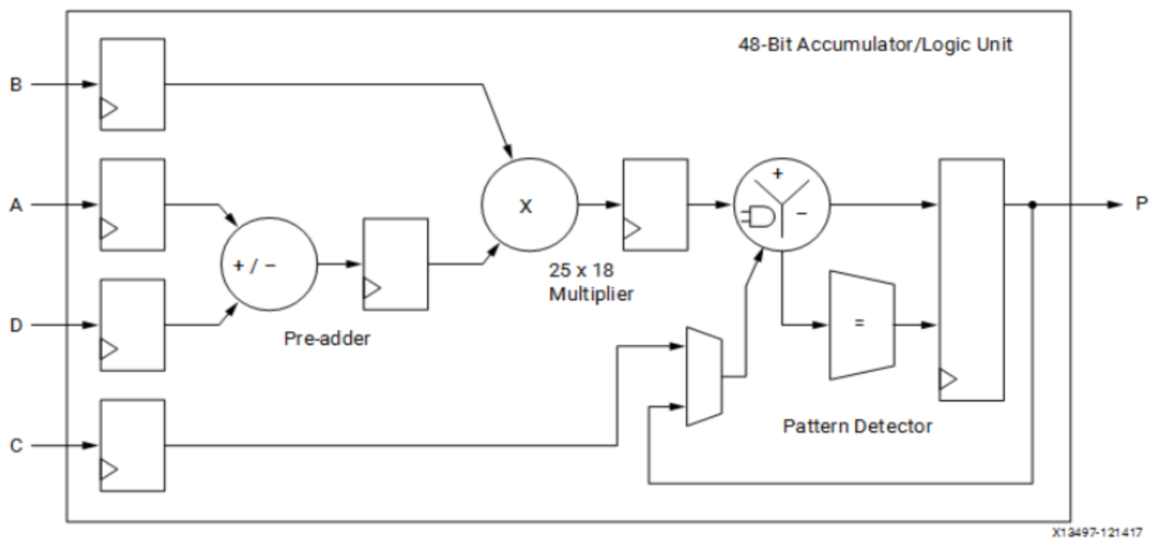


Figure (2.7). A DSP48 Block structure [39].

BRAM and Other Memories

Embedded memory elements in FPGA fabric are random-access memory (RAM), read-only memory (ROM), or shift registers. These elements are block RAMs (BRAMs), LUTs, and shift registers.

The BRAM is a dual-port RAM module available on the FPGA fabric to provide on-chip storage for a relatively large set of data (**18k or 36k bits**). Two types of BRAM memories that can hold either 18k or 36k bits are based on device specific. The dual-port BRAM has parallel, same-clock-cycle access to different locations.

In a RAM configuration, the data can be read and written at any time during the runtime of the circuit. In contrast, in a ROM configuration, data can only be read during the runtime of the circuit. The data of the ROM is written as part of the FPGA configuration and cannot be modified in any way.

As discussed in the LUT section, the contents of a truth table of LUT are written during device configuration. Due to the flexible structure of LUT in Xilinx FPGAs, these blocks can be used as 64-bit memories. LUT is commonly referred to as distributed memories, which is the fastest kind of memory available on the FPGA. Therefore, LUT can be used in any part of the fabric in order to improve the performance of the implemented circuit.

The shift register is a chain of registers connected to each other. Figure (2.8) shows the structure of an Addressable Shift Register. The purpose of this structure is to provide data to be reused along a computational path, such as with a filter.

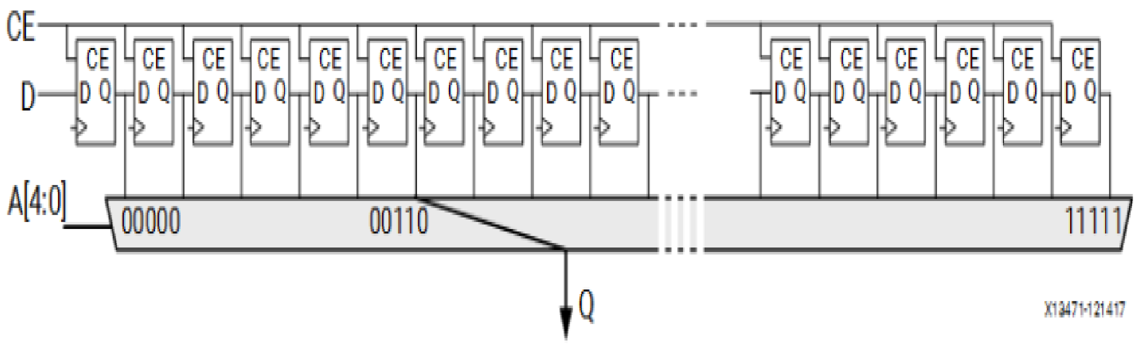


Figure (2.8). Structure of an Addressable Shift Register.

Clock cycle

The speed of a computer processor, is determined by the clock cycle. Clock cycle is the amount of time between two pulses of an oscillator. the higher number of pulses per second, the faster the computer processor can to process information. The clock speed is measured in Hz, often either megahertz (MHz) or gigahertz (GHz). For example, a 3 GHz processor performs 3,000,000,000 clock cycles per second.

2.8. Three main methods for designing Lookup table

To simplify the hardware implementation of tanh x, which is used for variable node update in LDPC decoder, we proposed to apply lookup table. In the literature review, hardware implementations for the hyperbolic tangent function are performed based on the approximation of the function rather than calculating it. Three main methods for designing Lookup table are used to implement and approximate the hyperbolic tangent function in hardware are as follows:

- Lookup table (LUT) approximation [32],
- Piece Wise Linear (PWL) approximation [33],

- Hybrid methods, which are essentially a combination of the former two [34].

2.8. A . Piecewise Linear (PWL) Approximation

A series of linear segments is used in PWL method to approximate a function [32]. The goal in PWL method is to minimize the error, processing time, and area depending on the number and location of the segments. The PWL method which is usually requires multiplier take several clock cycles. And also, multipliers are expensive in terms of resource usage, so, PWL methods are expensive while taking several clock cycles.

2.8.B. Lookup Table (LUT) Approximation

In the Look Up Table method, the number of points, which is uniformly distributed over the input period, is limited [33]. The number of points should be enough to minimize the approximation error of the function since there is a direct relation between the number of bits used to represent the address (input) and output.

2.8.c. Hybrid Methods

In Hybrid methods, look-up tables and other hardware are required to generate the goal function [34]. In the hybrid method, typically, multipliers are not used. However, they take several clock cycles to perform. The speed increases significantly since there is no usage of multipliers.

We choose the LUT method because, according to the literature between current hardware synthesizers, LUTs need less area than PWL methods, and also LUT is faster than the other two. In addition, in [35], it is shown that the range addressable lookup table method performs significantly quicker with the same amount of error while using less area compared to LUT, PWL, and Hybrid. Therefore, based on simulation results, range addressable lookup tables are proposed as a solution that offers partially simplifying hardware implementation of LDPC decoder in terms of speed and resource utilization.

In RALUT, limited number of points are used to approximate the function. The limited number of points are uniformly distributed across the entire input range [33]. Indeed, the size of look up table is diminished by addressing x in a bigger range. However, the answer is not desired. Therefore, another RALUT is proposed in which the range of points are uniformly distributed across the entire output range. The performance of the proposed method to update variable node of LDPC decoder for the DVBS2 standard is the same as the standard while the proposed method is faster and using fewer resources. Finally, Appendix C shows the RALUT, which is used for calculation of $\tanh x$ in update variable node messages of LDPC decoder.

Chapter 3

Decoder Hardware Implementation and Slepian-Wolf compression using DVB-s2 LDPC code

Summary

In Chapter three, reliable communication over the noisy channel is considered to be implemented by the hardware of one standard of LDPC codes called DVB-S2. The design and architecture of FPGA implementation of an LDPC decoder are presented. Besides, the hardware implementation of the LDPC decoder is simplified using Range Addressable Look Up Tables. In Section 3.4, Range addressable Lookup Table approximation is applied to update variable nodes in the LDPC decoder. Because of undesired results, a new Range addressable Lookup Table approximation is proposed in order to update variable nodes in the LDPC decoder. Finally, in chapter three, data compression with side information at the decoder is used as a caching solution in a Well-designed Caching example. Chapter four presents the conclusion and future direction for the thesis.

3.1. LDPC Codes in DVB-S2 Standard

One of the improvements of the DVB-S2 standard from the original DVB-S standard is that instead of convolutional and Reed-Solomon codes, LDPC codes are concatenated with BCH codes for forward error correcting encoding and decoding. However, in this thesis, our main focus is only on the LDPC codes in the DVB-S2 standard. Therefore, the discussion of the BCH codes of the DVBS-2 standard is beyond the scope of this thesis. In this section, an overview of the LDPC codes in the DVB-S2 standard is presented. The LDPC codes in the DVB-S2 standard have two block lengths. Normal frames have block length $N = 64800$, and short frames have $N = 16200$. Eleven code rates are specified in the normal frames and ten in short frames. Table (3.1) shows different code rates used in the normal frames and in short frames.

According to the standard, even though the parity check matrices, H, chosen by the standard are sparse, their corresponding generator matrices are not. Thus, the DVB-S2 standard adopts a special structure of the H matrix in order to reduce the memory requirement and the complexity of the encoder. The special structure of the LDPC code is called Irregular Repeat-Accumulate (IRA) [26]. The H matrix consists of two matrices A and B are shown in Equation (3.1), as follows:

$$H_{(N-K)\times N} = \left[A_{(N-K)\times N} | B_{(N-K)\times N}\right] \tag{3.1}$$

Where B is a staircase lower triangular matrix, as shown in Equation (3.2).

$$B_{(N-K)\times N} = \begin{bmatrix} 1 & 0 & \cdots & \cdots & \cdots & \cdots & 0 \\ 1 & 1 & 0 & \cdots & \cdots & \cdots & 0 \\ 0 & 1 & 1 & 0 & \cdots & \cdots & 0 \\ 0 & 0 & 1 & 1 & 0 & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & 1 & 1 & 0 & 0 \\ 0 & \cdots & \cdots & 0 & 1 & 1 & 0 \\ 0 & \cdots & \cdots & \cdots & 0 & 1 & 1 \end{bmatrix} \quad (3.2)$$

Matrix A is a sparse matrix, where the locations of the non-zero elements are specified in Appendix C of the DVBS2 standard [27]. Furthermore, the standard also introduces a periodicity of $M = 360$ to the submatrix A in order to reduce storage requirements. The periodicity condition divides the A matrix into groups of $M = 360$ columns. For each group, the locations of the non-zero elements of the first column are given in Appendix B. Let the set of non-zero locations on first, or leftmost, column of a group be $c_0, c_1, c_2, \dots, c_{db-1}$ where db is the number of non-zero elements in that first column. For each of the $M - 1 = 359$, other columns, the locations of the non-zero elements of the ith column of the group are given by $(c_0 + (i-1)p)mod(N-K)$, $(c_1 + (i-1)p)mod(N-K)$, … , $(c_l + (i-1)p)mod(N-K)$. Where $N-K$ is the number of parity-check bits and $p = \frac{N-K}{M}$ code dependent constant, as shown in Table (3.1), where the values are obtained from the user guidelines of the standard [28].

$N = 64800$	Rate	$^1/_4$	$^1/_3$	$^2/_5$	$^1/_2$	$^3/_5$	$^2/_3$	$^3/_4$	$^4/_5$	$^5/_6$	$^8/_9$	$^9/_{10}$
	p	135	120	108	90	72	60	45	36	30	20	18
$N = 16200$	Rate	$^1/_5$	$^1/_3$	$^2/_5$	$^4/_9$	$^3/_5$	$^2/_3$	$^{11}/_{15}$	$^7/_9$	$^{37}/_{49}$	$^8/_9$	-
	p	36	30	27	25	18	15	12	10	8	5	-

Table (3.1). The values of p values in DVB-S2 LDPC code.

Since the LDPC codes in the DVB-S2 standard are systematic, the encoding of message bits simply can be found by calculating the parity bits through the parity-check equations. Using the structure of the codes as mentioned above, the A submatrix with dimensions $(N-K) \times K$ can be generated. Let a_{ij} denote the elements in the A submatrix, where $i = 0, 1, \dots, N-K-1$and $j = 0, 1, \dots, K-1$. In order to encode the message, $u = u_0, u_1, \dots, u_{K-1}$, the parity bits are found using the following parity-check equations, as shown in Gomes et al. [29]:

$$p_0 = a_{0,0}u_0 \oplus a_{0,1}u_1 \oplus \dots \oplus a_{0,K-1}u_{K-1}$$

$$p_1 = a_{1,0}u_0 \oplus a_{1,1}u_1 \oplus \dots \oplus a_{1,K-1}u_{K-1}$$

$$p_2 = a_{2,0}u_0 \oplus a_{2,1}u_1 \oplus \dots \oplus a_{2,K-1}u_{K-1}$$

$$\vdots$$

$$p_{N-K-1} = a_{N-K-1,0}u_0 \oplus a_{N-K-1,1}u_1 \oplus \dots \oplus a_{N-K-1,K-1}u_{K-1}$$

The encoded codeword is the concatenation of the message bits and the parity bits. Thus, the resultant N-bit codeword has the following form:

$$u_0, u_1, u_2, \dots, u_{K-1}, p_0, p_1, \dots, p_{N-K-1}$$

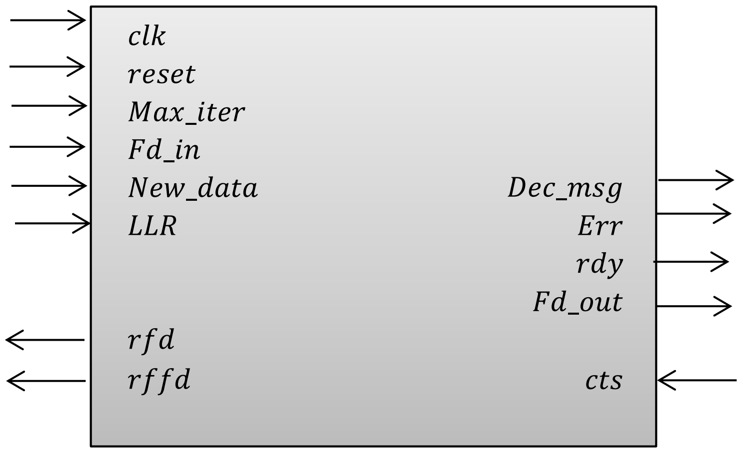


Figure (3.1). Inputs and Outputs of the LDPC decoder.

Table (3.2). Description of the Inputs and Outputs of the Decoder.

Input/ Output	Bit width	Name	Description
Input	1	Clk	Clock
Input	1	$reset$	Reset
Input	8	Max_iter	Sets the maximum number of iterations the decoder will perform
Input	1	nd	New data indicates that input LLR values are incoming
Input	6	llr	Serial 6-bit wide input LLR values
Input	1	Fd_in	First data input marks the beginning of an input frame
Input	1	cts	Clear to send informs the decoder as to whether or not to output the decoded message
output	1	rfd	Ready for data indicates that the decoder is ready for more LLR values
output	1	$rffd$	Ready for first data indicates that the decoder is ready for a new frame
output	1	$decmsg$	Serial hard decoded message output
output	1	err	Indicates whether or not a decoding error has occurred
output	1	rdy	Ready indicates the output data is ready to stream out
Output	1	fd	out First data output marks the beginning of an output frame

3.2. The architecture of the hardware implementation of DVB-S2 LDPC

Here, the details of the architecture of the hardware implementation of the DVB-S2 LDPC decoder are presented. Figure (3.1) shows the inputs and outputs of the decoder. Table (3.2) describes each input and output of the decoder in more detail.

The finite state machine of the LDPC decoder

The finite state machine controls the data flow of the LDPC decoder. Therefore, the finite state machine has connections to all available components. The state transition diagram of the controller is shown in Figure (3.2).

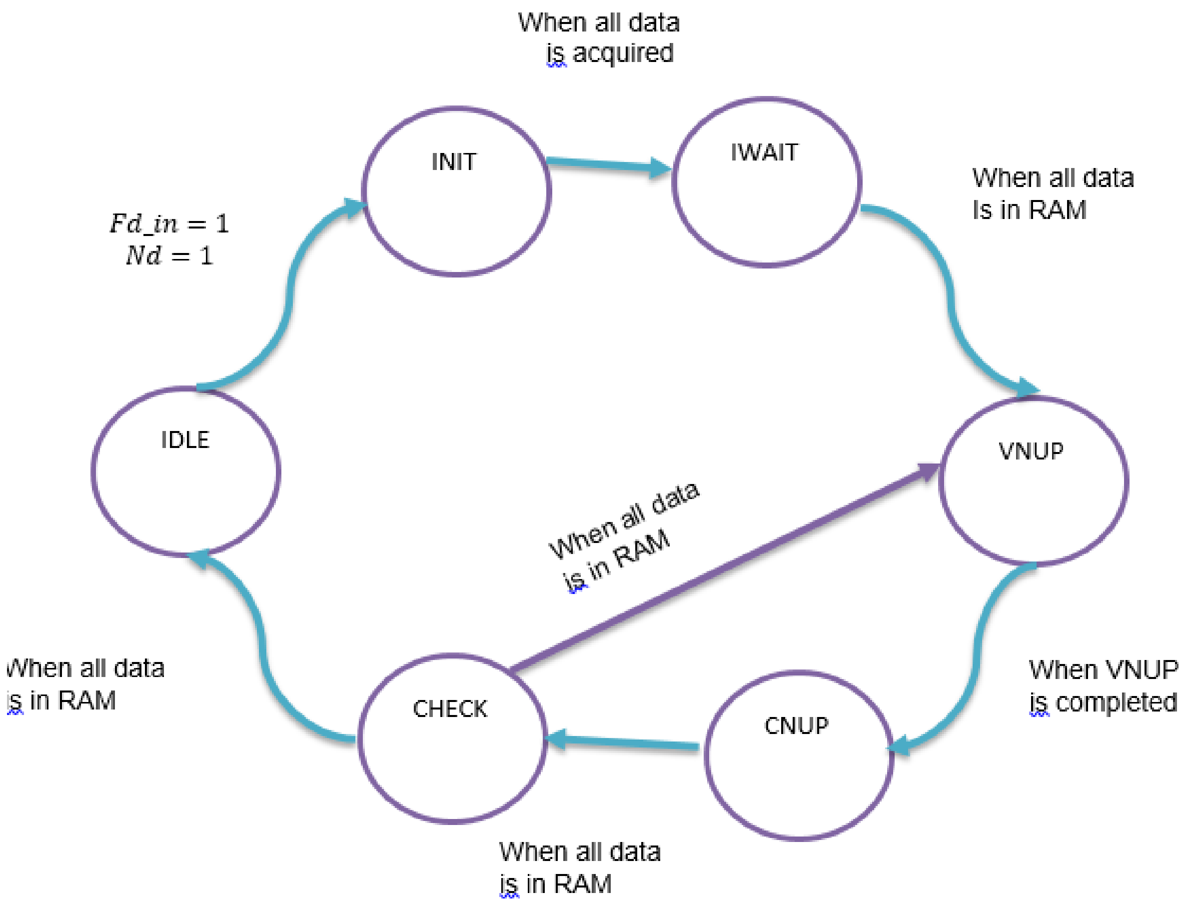


Figure (3.2). The finite state machine of LDPC decoder.

***IDLE* State:** *the IDLE* state is the initial step of the decoder's control flow.

***INIT* State:** When both inputs nd and fd in are active, the controller enters the $INIT$ state. It remains in this state until all the 64800 LLR values input into the decoder. Then the controller moves to the next state.

***IWAIT* State:** It is a transition state where the decoder is received all the 64800 LLR values, but some LLR values are still being written into the RAM through the FUs. So it is not ready for calculation.

***VNUP* State:** Once all the RAM values are ready, the controller goes into the $VNUP$ state where the Variable Node Update based on the message which is sent from the parity check node to the adjacent message variable node (Equation (2.12)).

***CNUP* State:** Once the Variable Node Update step is complete; the controller goes into the $CNUP$ state where the Check node value is updated based on Equation (2.13).

CHECK State: After all the Check Node Update calculations are performed, the controller enters the CHECK state. During the CHECK state, the parity-check equations are verified. If all parity check equations are satisfied, error = 0, then the controller enters the IDLE state and waits for the next frame of LLR values while outputting the decoded message. Otherwise, error =1, and the controller returns to the VNUP state to repeat the VNUP, CNUP, and CHECK states. If the maximum number of iterations is reached during the CHECK state, the controller also moves to the IDLE state and outputs the decoded message with the output err set to 1.

3.3. Hyperbolic Tangent Function Implementation

The citation that follows from Pharr and Fernando [31] describes the concept of using the Look up table briefly:

For optimizing a function that is expensive to compute and inexpensive for the cache using Look Up Tables (LUTs) is an excellent technique. So, a precomputing of a function over a range of common input is evaluated in order to find a proper LUT. Indeed, expensive runtime operations can be replaced with inexpensive table lookups. If the computation run time is much longer than the read time of Look up table, then the use of a lookup table will result in a significant performance gain. Besides, an interpolation algorithm by nearby averaging samples can be used for the data, which is between the data sample's so that the result will be reasonable approximations [31].

To simplify and increase the speed of the hardware implementation of $\tanh x$, which is used to update variable node in the LDPC decoder, we proposed to apply the Lookup table. Usage of Lookup table for the hyperbolic tangent function is essential for increasing the efficiency of designing and implementing the hardware. Indeed, expensive runtime operations of hyperbolic tangent function can be replaced by inexpensive table lookup. Therefore, if the computation run time is much slower than the computation by the Lookup table, then the usage of the lookup table will result in a significant performance gain. The hyperbolic tangent function graph is a sigmoid curve with the shape of S in which the variation of the hyperbolic tangent function is limited outside the period of (-2, 2).

Three main approaches are LUT, PWL, and Hybrid, which are used to approximate the hyperbolic tangent function in hardware. Figure (3.3) shows the hyperbolic tangent function S shape curve. Figure (3.4) presents Lookup Table Approximation of the hyperbolic tangent function with Eight Points. Figure (3.5) shows $tanh(x)$ approximation with piecewise linear approximation by five Segments.

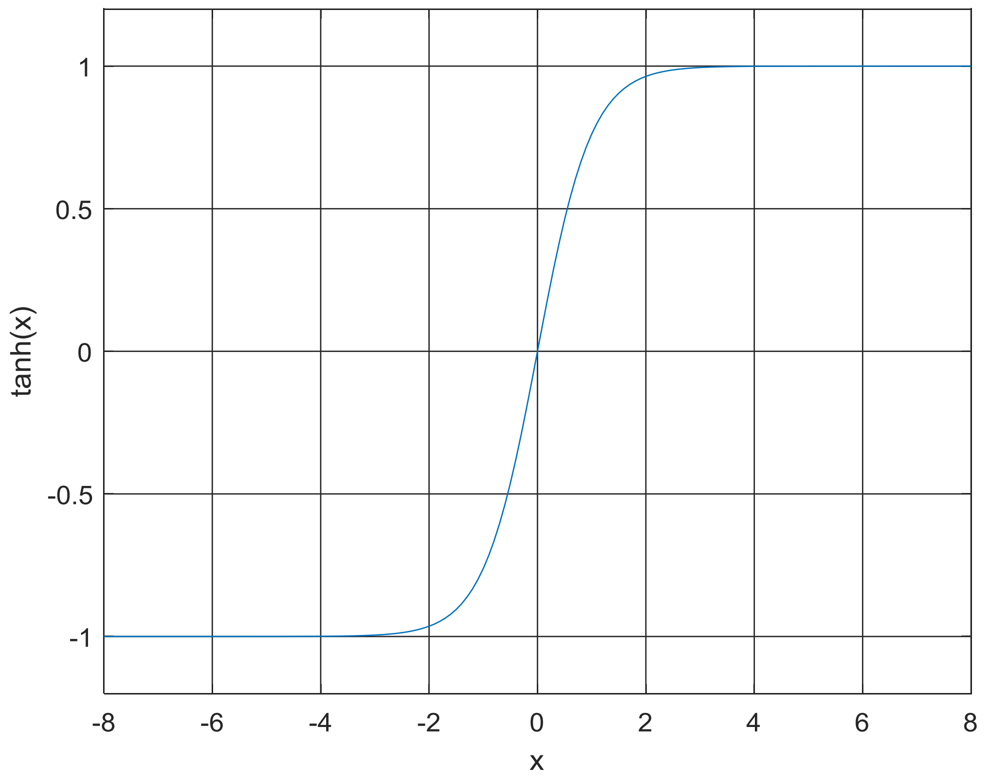


Figure (3.3). The Hyperbolic Tangent Function S shape curve.

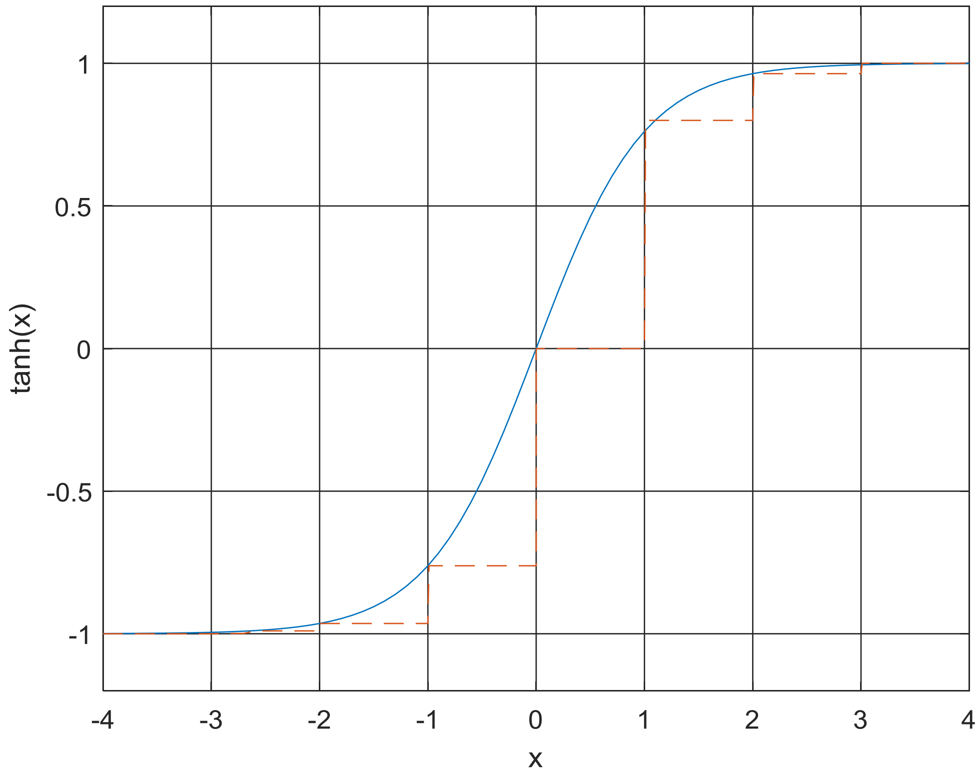


Figure (3.4) Lookup Table Approximation of $tanh(x)$ which is represented by Eight Points.

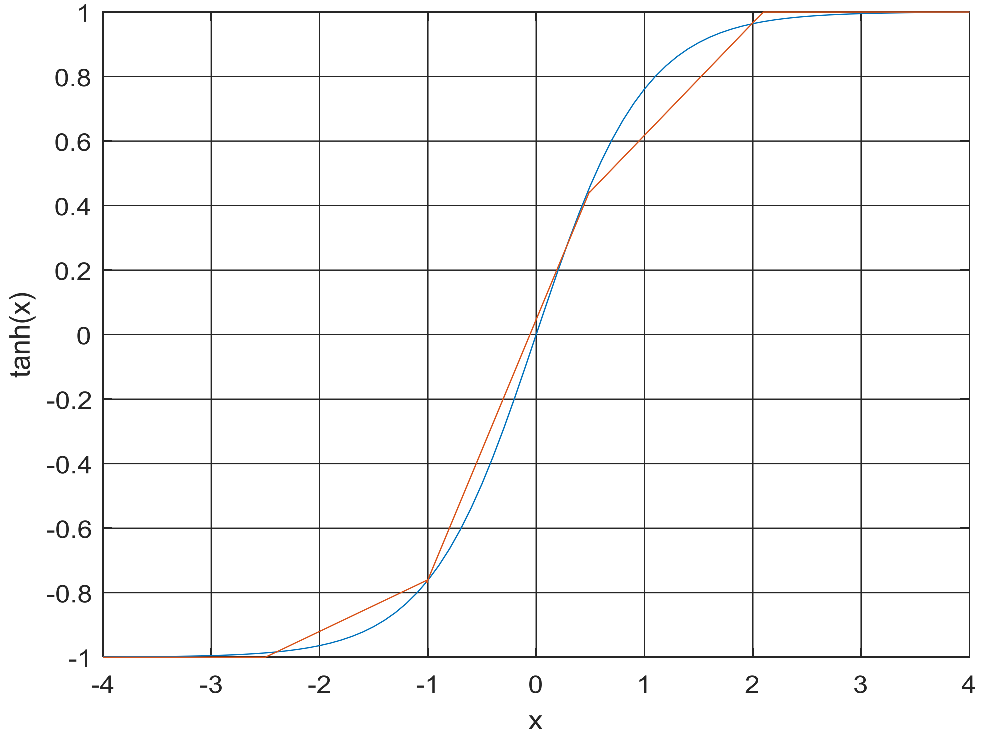
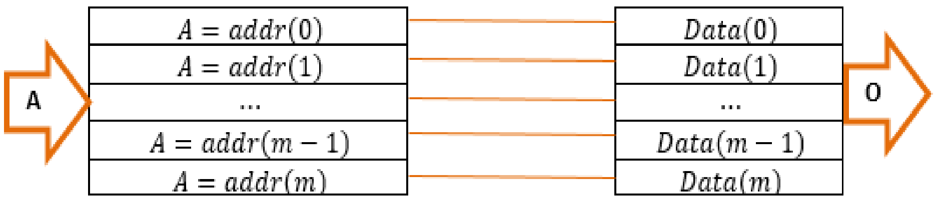


Figure (3.5). Piecewise Linear Approximation of $tanh(x)$ by five Segments.

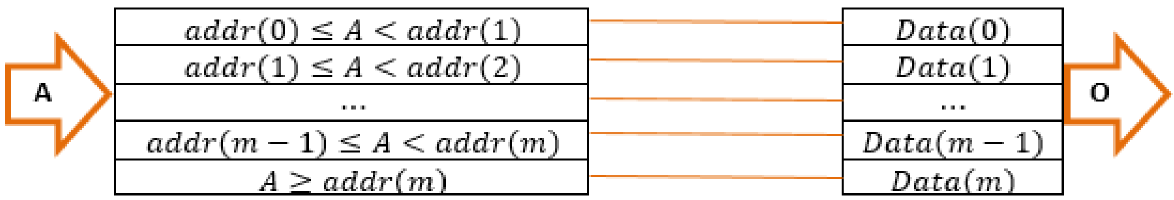
3.4 Implementation of the Hyperbolic Tangent Function by Range Addressable Lookup Table

Three main methods for designing a Lookup table are used to implement and approximate the hyperbolic tangent function in hardware, including LUT, PWL, and Hybrid approximation methods. According to the literature among the currently available hardware synthesizers, LUTs need less area than PWL methods, and also LUT is faster than the other two.

Furthermore, it is shown that the range addressable lookup table method performs significantly faster with the same amount of error while it uses less area compared to LUT. Range addressable LUT was originally proposed in [37] so that highly nonlinear, discontinuous functions are implemented. RALUT is similar to regular LUT in which the memory is only readable. However, there are a few notable differences between LUT and RALUT. A lookup table uses a classic decoding scheme. However, a range addressable lookup table decoding scheme is designed in a way, decreasing the size of LUT. In LUTs, each output belongs to a unique input address, while RALUTs output belongs to a range of addresses, as shown in Figure (3.6). This difference between LUT and RALUT results in a large reduction in data points in the RALUT method, especially when the output is non-changeable over a significant period of input. In the hyperbolic tangent function, the output changes a little outside the period of $(-2, 2)$. Therefore, the RALUT method is an efficient and optimized method for approximating $tanh(x)$ function compared to LUT. This is due to the fact that in LUT, every individual input point is represented by an output while in RALUT, a range of input points are represented by an output. Figure (3.7) represents the RALUT approximation of $tanh(x)$ with Eight Points [36].



a. Lookup Table Architecture



b. Range addressable Lookup Table Architecture

Figure (3.6). Comparison between LUT and RALUT Addressing methods.

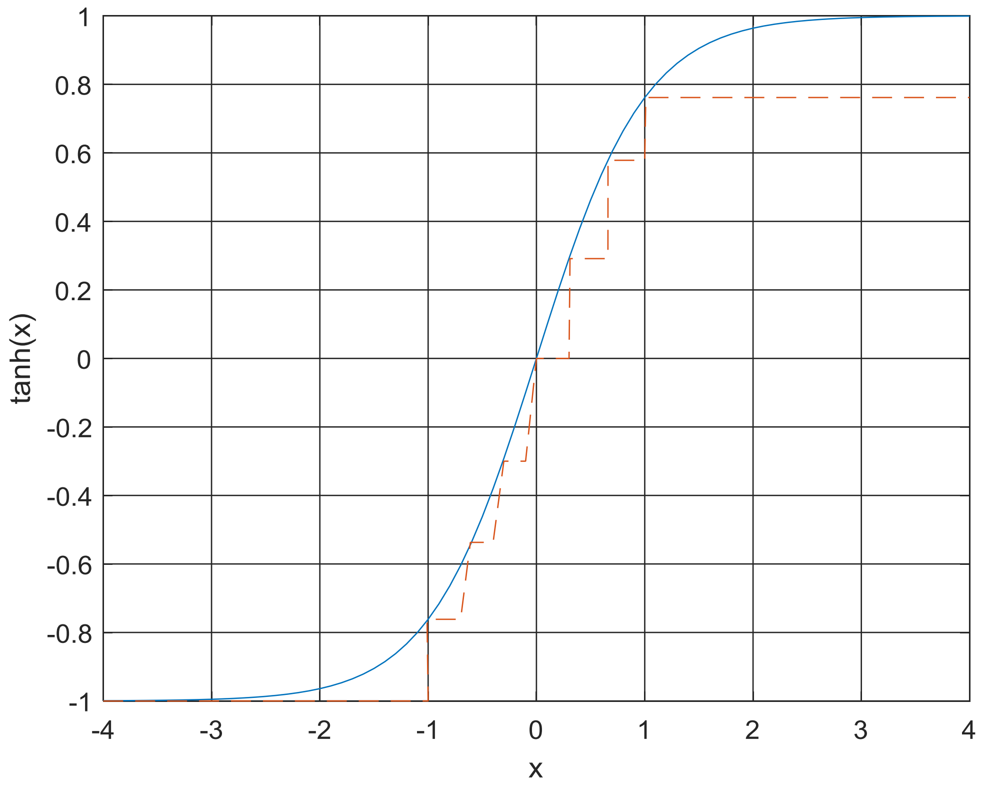


Figure (3.7). RALUT Approximation of $tanh(x)$ with eight points.

3.4 A. Applying Range addressable Lookup Table Approximation to update variable nodes in the LDPC decoder

In RALUT, the function is approximated with a limited number of points uniformly distributed across the entire input range [33]. Applying the RALUT results in an LDPC decoder, which will never reach zero BER. The results are shown in Figure (3.8), showing the BER of DVBS2 rate half for different values of SNR when $tanh(x)$ is approximated by RALUT. Indeed, the size of the lookup table is diminished by addressing x in a wider range of input. The decoding result is not desired. Therefore, a novel design of RALUT for the hyperbolic tangent function is proposed that simplifies the LDPC decoding algorithm while the performance remains the same.

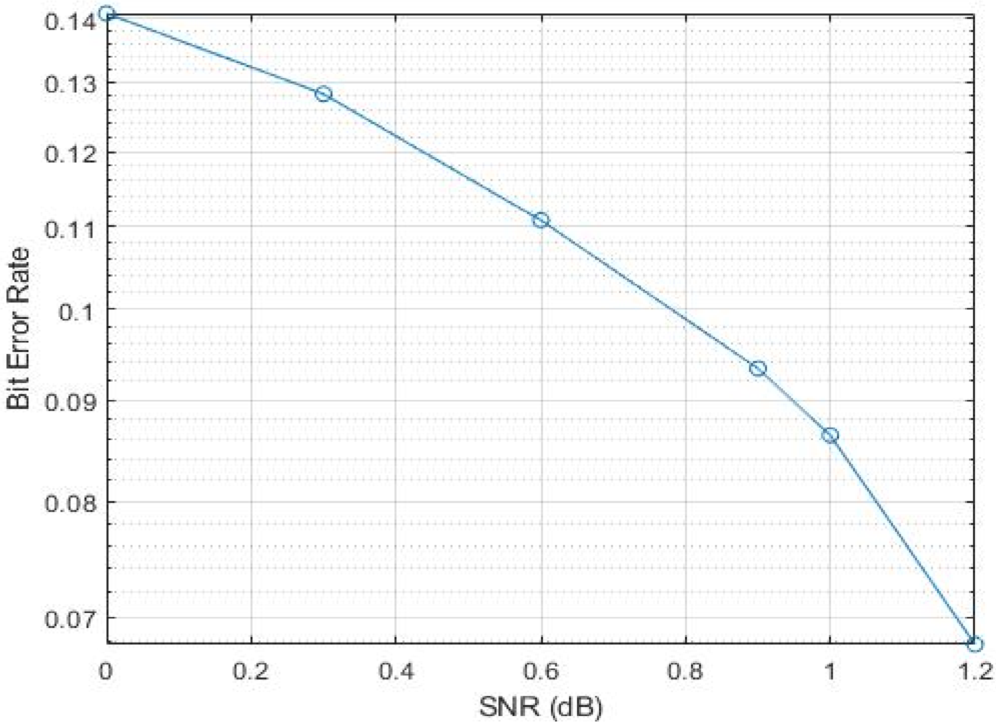


Figure (3.8). BER of LDPC code (Rate= $1/2$) by Applying rstaRALUT approximation to updandate variable node of LDPC decoder.

3.4. B. Applying the Proposed Range addressable Lookup Table Approximation to update variable node of eLDPC decoder

In Normal RALUT, the function d is approximated with a limited number of points uniformly distributed across the entire input range. However, we proposed a RALUT where the function is approximated with a limited number of points where more values are assigned to the points near zero.

Consider the output in a range of $y_1 \leq y = \tanh x \leq y_2$ would be $\frac{y_1+y_2}{2}$, so that $y = \tanh x$, $y_1 = \tanh x_1$, and $y_2 = \tanh x_2$, the input range must be $\tanh^{-1} y_1 \leq x \leq \tanh^{-1} y_2$. Besides, Appendix C shows the RALUT, which is used for updating the variable node messages of the LDPC decoder. The decoder presented in Chapter 2 is verified using a code which is coded in

MATLAB and C++. The code begins by generating a random sequence of bits. Every frame of the sequence is encoded ad decoded by an LDPC encoder and decoder implemented by us, i.e., we have not used the LDPC decoder function of Matlab. Frames of N=64800 bits long subsequently modulated using the BPSK modulation scheme. The BCH outer encoding specified in the DVB-S2 standard is not used because only the performance of the LDPC decoder is evaluated. The DVB-S2 standard also uses quadrature phase-shift keying (QPSK), 8 phase-shift keying (8PSK), 16 amplitude and phase-shift keying (16APSK) and 32 amplitude and phase-shift keying (32APSK) modulation schemes, but for simplifying the simulation test bench uses BPSK modulation scheme to modulate the encoded sequence. Subsequently, the modulated signal passes through a transmission channel, which is simulated by adding AWGN. The receiving side of the test bench demodulates the transmitted signal and producing the initial LLR values. These LLR values are divided into frames of N values, and each frame is inputted into the LDPC decoder. Finally, after decoding the input of the channel, decoded codeword for each frame is compared to the frames of the original random sequence generated. If the two sequences are identical, then the decoding is correct. Otherwise, decoding error has failed for that particular frame. Here, because we want to test and evaluate the results, the decoded codeword for each frame is compared with the original random sequence; however, in reality, the original random sequence is not available at the receiver. Therefore, when the syndrome became zero or when the number of iteration reaches to maximum, the LDPC decoder stops.

The SNR is the characteristic of the AWGN channel in units of decibels (dB), which is defined by the power of the signal received divided by the power of the noise in the channel. For normal frames, 100 frames are used. The result of the proposed RALUT for the positive part of $\tanh x$

function is shown in Figure (3.9). The performance of the proposed method to update variable nodes of the LDPC decoder for the DVBS2 standard is the same as the standard.

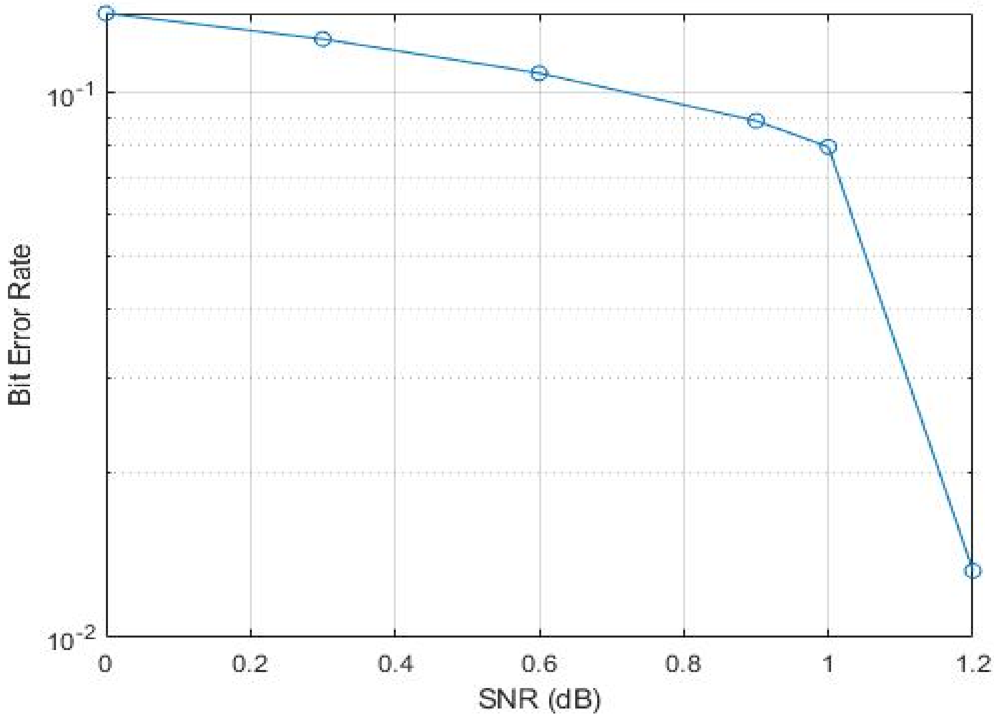


Figure (3.9). BER of LDPC code (Rate= 1/2) by applying proposed RALUT approximation to update the variable node of the LDPC decoder.

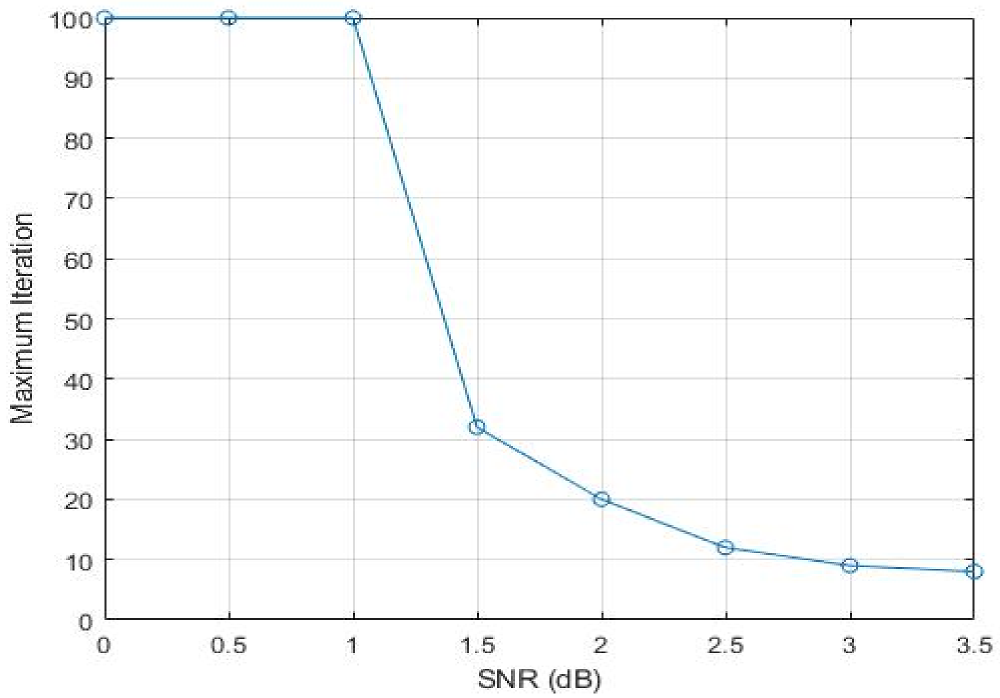


Figure (3.10). The maximum number of iteration for 30 packets for different values of SNR.

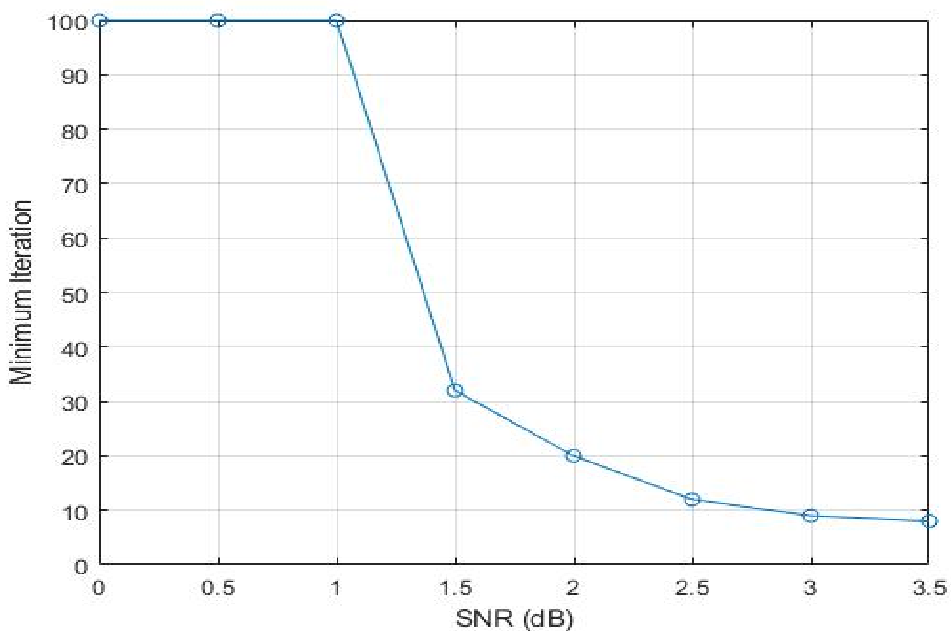


Figure (3.11). The minimum number of iteration for 30 packets for different values of SNR.

Figure (3.10) and Figure (3.11) represent the maximum and the minimum number of iteration for 30 packets for different values of SNR where the code rate is half, respectively. The number of maximum and the minimum number of iteration is precisely the same for a specific value of SNR when the thirty packets are produced randomly. There is a possibility to use this feature in hardware implementation to simplify and decrease the usage of resources. It means that for a specific value of SNR, the maximum number of iteration can be considered based on the maximum number of iteration of Figure (3.11). For example, for the value of SNR equal to two, the maximum number of iteration is 20. Thus after 20 iterations, the algorithm will decide to finish the decoding process and use the resources for other out coming packets.

XC6VLX240T, a family of FPGAs, is used for evaluation of the complexity of the proposed design by [30]. The synthesis result shows the speed increase due to the use of the RALUT method. Finally, Table (3.3) shown the hardware implementation results compared to [30]. Besides, Since Vivado HLS synthesis is available for Zynq XC7Z030, therefore, the result is presented for evaluation.

Table (3.3). Hardware implementation results for code rate half, $N = 64800$, compared by [30].

FPGA	*BRAM*	*FF*	*LUT*	*Clock cycle(MHz)*
XC6VLX240T [30]	31%	17%	60%	214.5
XC6VLX240T	30.5%	19.5%	47%	225
ZINC XC7Z030	25%	18.5%	54%	238.5

Results show the speed increase due to the use of the LUT method. However, LUT demands more memory resources. Thus, we decrease the usage of memory resources by applying the RALUT method. Commonly, RALUTs are uniformly distributed on input; however, in our proposed method, instead of representing the LUT input uniformly, we use a non-uniform scale assigning more values to those near zero.

3.5. Data compression with side information

According to the channel coding theory of Shannon, the source can be reconstructed with small error probability if the rate of the data sequence is less than the capacity of the transmission channel, which means that the problem of channel coding can be isolated into source coding problem [12]. Consider the model with two independent channels operating in parallel. According to Shanon's coding theorem, if the input to both channels were allowed to be encoded, the reliable transmission is possible if the entropy of the source is below the sum of capacities of the two channels $H(x) \leq H(X|Y) + H(Y)$. However, if the source entropy is above $H(X|Y) + H(Y)$ the reliable transmission is not possible. If one of the channels has an uncoded version of the source as side information at the decoder, known as systematic communication, there are two approaches for error protection of noisy transmission. One is based on Slepian Wolf [13], and the other is based on Wyner Ziv [14]. In this section, some basic concepts, including the Slepian Wolf Coding theorem, Wyner-Ziv Coding, Source channel with decoder side information, are presented.

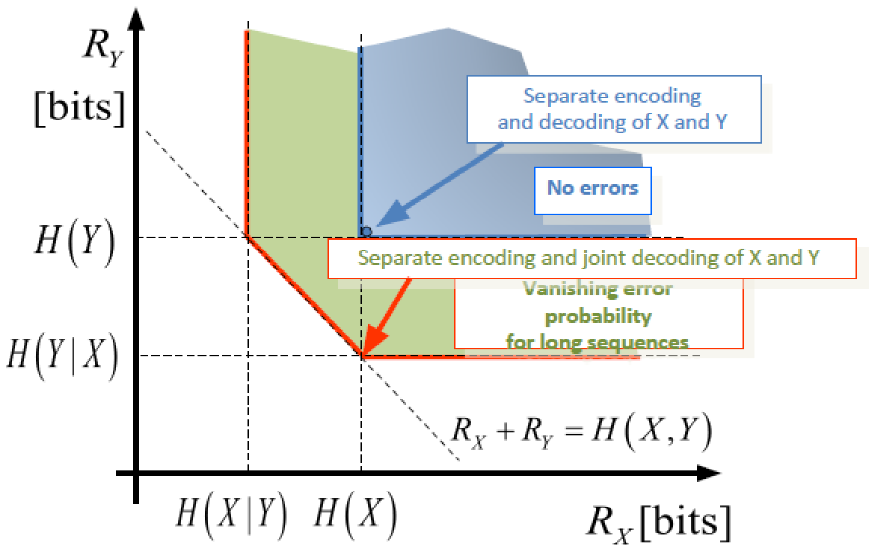


Figure (3.12). Achievable two-dimensional rate region.

It is figured that if vector X and Y with length of n bits are compressed into sequence of length of nR_X and nR_Y , respectively, where $R_X \geq H(X|Y)$, $R_Y \geq H(Y|X)$, and $R_X +, R_Y \geq H(X, Y)$, then the joint decoder can have a highly reliable reconstruction of X and Y. The result is shown as an achievable two-dimensional rate region in Figure (3.12).

Slepian Wolf decoding algorithm of LDPC code

In LDPC decoding of Slepian Wolf, when we have side information at the decoder, instead of transmitting the whole length of the original message, only the syndrome or check nodes are transmitted (H(X|Y)). The Slepian wolf decoding algorithm of LDPC code is almost the same with the channel decoding algorithm of LDPC code with some differences. In the following, the differences are explained.

Step 0: Instead of transmitting the complete message of length N, the check node bits value of the original message with a length of $N - K$ bits are transmitted. Besides, the correlated version of original message X with the length of N bits, which is called side information Y is available at the Slepian Wolf decoder.

Step 3 (round 1): Not zero positions of check node bits are marked. So in round 1 of step 3 or in **the update equation of variable nodes**, an extra sign is applied for the marked position. It means that whenever the message, which is the ratio of the probabilities in the log domain, passed from the marked check node to all adjacent variable nodes, an extra change of sign is applied. Therefore Equation (2.13) is changed to Equation (2.15)

$$m_{vc} = \tag{3.1}$$

$$\begin{cases} m_v & if\ l = 1 \\ m_v + \sum_{\acute{c} \in C_v \backslash \{c\}} \begin{cases} -m_{\acute{c}v}^{l-1} & \acute{c} \in Set\ of\ marked\ check\ nodes \\ m_{\acute{c}v}^{l-1} & else \end{cases} & if\ l \geq 1 \end{cases}$$

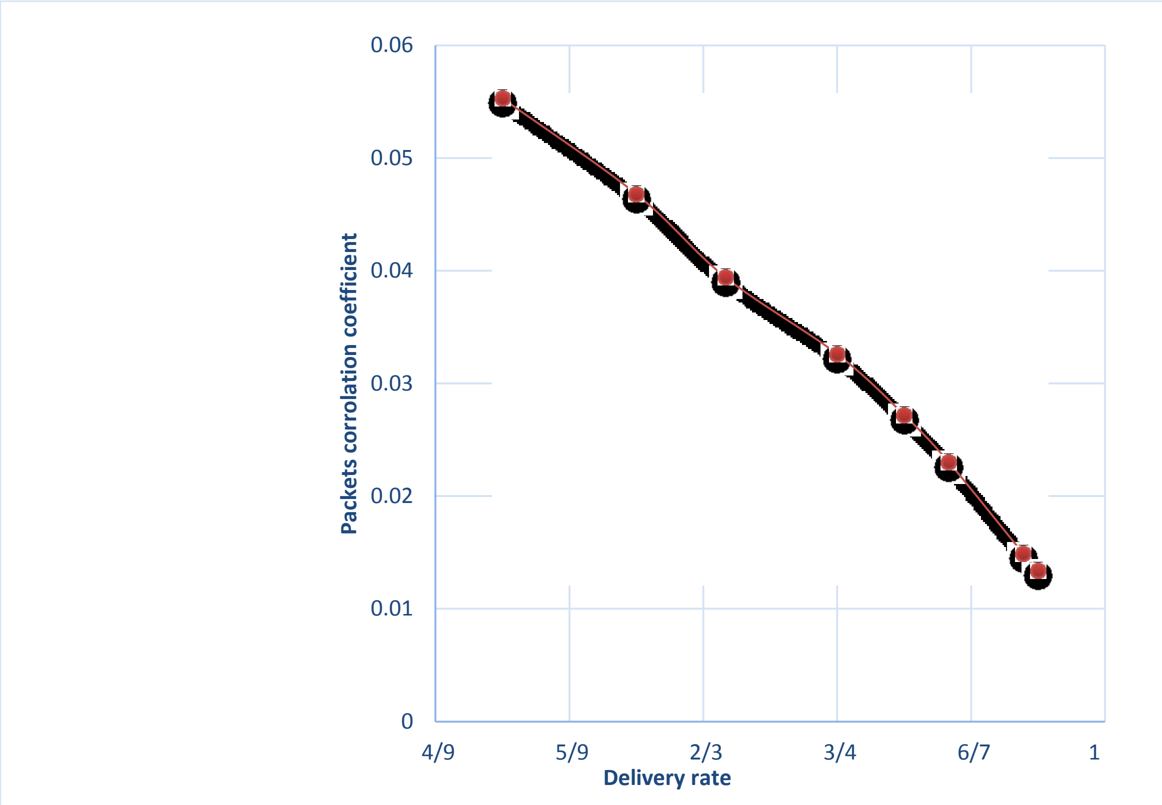
Step 5 (Stop conditions): The first stop condition is when the value of check nodes is the same as the check nodes value of the original message. The other stop condition is when the number of iteration of the decoding algorithm reaches the max number of iteration.

3.6. Well-designed Caching by DVBS2 Standard

Here is an example of the application of the Slepian Wolf decoding algorithm of LDPC code by the DVBS2 standard in the caching method [16]. Caching is a reliable solution for communication during a busy period by taking advantage of memory across the network, which leads to more smooth communication network systems [17-22]. The caching method has two-phase. The first phase is called the placement phase, where the data is stored in the cache across the network. The main limitation of this phase is the size of the cache memory**.** In the second phase, which is called the Delivery phase, the user's request can be partially served through caches near the users**.** Examples of application of the caching method are streaming media and distributed database**s,** which results in decreasing the delivery rate.

Streaming media: User requests time is most likely at night rather than early in the morning. During congestion periods, the bandwidth-hungry features of media result in more congestion, high latency, and a poor experience for users. One applicable solution is caching during off-peak hour time.

Distributed database: Some examples of the distributed database are meteorological conditions measurement information of the globe, information of traffic sensors spread across several countries, information on the shopping history of the customers, information on the mobility pattern of the mobile devices in cellular networks. Since the database is extensive, it might need several different network calls to load the requested data of the memory before the requested data can be transmitted to the users. These network calls cause latency or stalls the process. In the modern database, it is handled by storing the most common queries in fast memory**.** For instant, consider that a user more probably demands the weather measurement of his hometown rather than of a remote area. Therefore, the information of weather measurement of the user hometown is cached in memory close to it. To our best knowledge, little attention was given to the source coding problem in the presence of caching; however, Compressing information can highly mitigate the traffic.



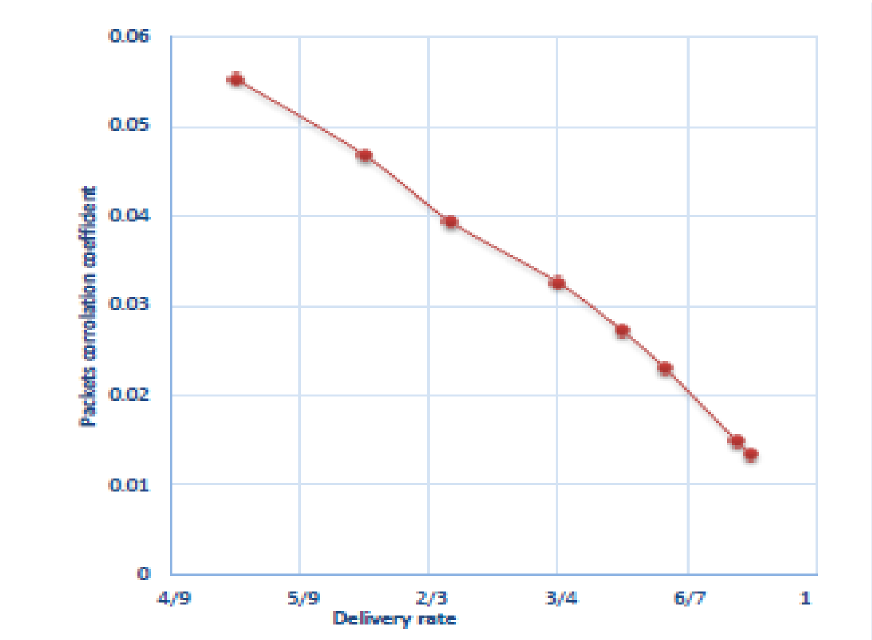


Fig (3.13). The tradeoff between packet correlation and delivery rate.

We proposed a general approach to –decreases the delivery rate by applying source coding of LDPC code to the correlated binary source**.** Consider that we have a source with correlated packets. The original packets are put in the cache in the placement phase. The rest of the packets that are correlated to the original packets with the coefficient of α will be sent during the delivery phase with rate $1-k/n$**.** For the delivery phase, we applied DVBS2 standard, which is adopted by many numbers of standards because of having powerful features such as transmission rate close to the theoretical Shannon limit [19]. For flexible configuration DVBS2 standard has several code rate including $R = 1/4$, $1/3$, $2/5$, $3/5, 2/3,\ 3/4, 4/5, 5/6, 8/9,$ $9/10$. Code rates $1/4,\ 1/3$,and $2/5$ have been introduced for exceptionally poor reception conditions. In this example, we focus on 64800 length bits of a codeword of rates $R = 1/4$, $1/3$, $2/5$, $3/5,\ 2/3,\ \ 3/4,\ 4/5,\ 5/6,\ 8/9,\ 9/10$, which is logical for our source coding purposes. The LDPC codes, as defined in the DVB-S2 standard, have straight forward encoder realization since the DVBS2 standard has a lower triangular shape for its parity check matrix. Figure (3.13) shows the tradeoff between packet correlation and delivery rate. It is obvious that there is an inverse relation between correlated coefficient α and delivery rate. Thus, when the correlation between packets is high, the delivery rate is low.

Chapter 4

Conclusion and future work

In this Chapter conclusion and future direction for the thesis are presented.

4.1. Conclusion summary

The emergence of large scale and high-speed data networks for processing, storage, and exchange of digital information in military, government, and private spheres resulted in demand for efficient and reliable data storage and transmission network systems. According to Shannon's theorem, if the transmission rate is less than the capacity, there is always an error correction code that can make the probability of error arbitrarily small. Besides, the application of error-correcting codes of data compression is investigated by Shannon due to duality between source coding and channel coding. Indeed, a channel code that provides high rates has the capability to be a source code with high rates as a result of duality.

Caching is a reliable solution for communication during a busy period by taking advantage of memory across the network, which leads to more smooth communication network systems [17-

22]. A general approach is proposed to decreases the delivery rate by applying source coding of LDPC code to the correlated binary source. The results show that there is an inverse relation between correlated coefficient α and delivery rate.

In addition, we have presented a new hardware implementation of the LDPC code used in DVB-S2. We have used a Range addressable LUT scheme to approximate the Tangent Hyperbolic function. Our approach is motivated by the fact that among the three methods used for approximation of Hyperbolic Tangent, i.e., LUT, PWL, and hybrid method, LUT is the fastest approach but requires more resources than other two. Therefore, we have used RALUT in order to compensate for this. Synthesis results on Xilinx, XC7Z030, family of FPGA's shows that our method is faster than another implementation [30].

4.2. Future direction

Some proposed work as a progress of this thesis are as follows;

- Apply hardware implementation on a new standard such as ATSC 3.
- Expand the idea of Range addressable lookup table for hardware implementation for other applications.

References

[1]. R. G. Gallager,"Low density parity check codes," IRE Trans. Inform. Theory, IT-8: 21-28, January 1962.

[2]. R. G. Gallager, Low density parity check codes, MIT press, Cambridge, 1963.

[3] A. Shokrollahi, ,"LDPC codes: An Introduction," April, 2003.

[4] D. J. C. MacKay and R. M. Neal, "Near Shannon limit performance of low density parity check codes," Electron. Lett., vol. 32, pp. 1645–1646, Aug. 1996.

[5] N. Wiberg,"Codes and decoding on general graphs," Dissertation no. 440, Dept. Elect. Eng. Linkping Univ., Linkping , Sweden, 1996.

[6] M. Luby, M. Mitzenmacher, A. Shokrollahi, D. Spielman, and V. Stemann,"Practical loss-resilient codes," in Proc. 29th Annual ACM Symp. Theory of Computing, 1997, pp. 150–159.

[7] M. Sipser and D. Spielman, "Expander codes," IEEE Trans. Inform.Theory, vol. 42, pp. 1710–1722, Nov. 1996.

[8] D. J. C. MacKay, S. T. Wilson, and M. C. Davey, "Comparison of constructions of irregular Gallager codes," in Proc. 36th Allerton Conf. Communication, Control, and Computing, Sept. 1998.

[9]. El-Sherbini, Ahmed M. "Method and apparatus for differential run-length coding." U.S. Patent No. 4,631,521. 23 Dec. 1986.

[10].S. Lin and D. J. Costello, Error Control Coding: Fundamentals and Applications, 2nd Edition, Prentice-Hall, 2005. Chapter 17.

[11]. T. J. Richardson and R. L. Urbanke, “Efficient encoding of low density parity check codes,” IEEE Trans. Inform. Theory, February 2001.

[12] Cover, Thomas M., and Joy A. Thomas. *Elements of information theory*. John Wiley & Sons, 2012.

[13] Slepian, David, and Jack Wolf. "Noiseless coding of correlated information sources." *IEEE Transactions on information Theory* 19.4 (1973): 471-480.

[14] Wyner, Aaron, and Jacob Ziv. "The rate-distortion function for source coding with side information at the decoder." *IEEE Transactions on information theory* 22.1 (1976): 1-10.

[15] Shamai, Shlomo, and Sergio Verdú. "Capacity of channels with uncoded side information." *European Transactions on Telecommunications* 6.5 (1995): 587-600.

[16] European Telecommunications Standards Institute (ETSI). Digital Video Broadcasting (DVB) Second generation framing structure for broadband satellite applications; EN 302 307 V1.1.1. www.dvb.org

[17]. Maddah-Ali, Mohammad Ali, and Urs Niesen. "Fundamental limits of caching." *IEEE Transactions on Information Theory*60.5 (2014): 2856-2867.

[18]. Niesen, Urs, and Mohammad Ali Maddah-Ali. "Coded caching with non-uniform demands." *IEEE Transactions on Information Theory* 63.2 (2017): 1146-1158.

[19]. M. A. Maddah-Ali and U. Niesen, “Decentralized coded caching attains order-optimal memory-rate tradeoff,” arXiv:1301.5848 [cs.IT], Jan. 2013.

[20]. Pedarsani, Ramtin, Mohammad Ali Maddah-Ali, and Urs Niesen. "Online coded caching." *IEEE/ACM Transactions on Networking (TON)* 24.2 (2016): 836-845.

[21]. Timo, Roy, et al. "A rate-distortion approach to caching." *IEEE transactions on information theory* 64.3 (2018): 1957-1976.

[22] C.-Y. Wang, S. H. Lim, and M. Gastpar, “Information-theoretic caching: Sequential coding for computing,” IEEE Trans. Inf. Theory, vol. 62, no. 11, pp. 6393–6406, Nov. 2016.

[23]. Martin, Grant, and Gary Smith. "High-level synthesis: Past, present, and future." *IEEE Design & Test of Computers* 26.4 (2009): 18-25.

[24]. Casseau, Emmanuel, et al. "C-based rapid prototyping for digital signal processing." *2005 13th European Signal Processing Conference*. IEEE, 2005.

[25]. Lai, Yung-Te, Massoud Pedram, and Sarma BK Vrudhula. "BDD based decomposition of logic functions with application to FPGA synthesis." *30th ACM/IEEE Design Automation Conference*. IEEE, 1993.

[26]. H. Jin, A. Khandekar, and R. McEliece, "Irregular repeat-accumulate codes," in 2nd International Symposium on Turbo Codes and Related Topics, September 2000.

[27]. ETSI, Digital Video Broadcasting (DVB): Second generation framing structure, channel coding and modulation systems for Broadcasting, Interactive Services, News Gathering and other broadband satellite applications (DVB-S2), EN 302 307 V 1.2.1." August 2009.

[28]. ETSI, Digital Video Broadcasting (DVB): User guidelines for the second generation system for Broadcasting, Interactive Services, News Gathering and other broadband satellite applications (DVB-S2), TR 102 376 V 1.1.1." February 2005.

[29]. M. Gomes, G. Falcao, A. Sengo, V. Ferreira, V. Silva, and M. Falcao, High throughput encoder architecture for DVB-S2 LDPC-IRA codes," in International Conference on Microelectronics, 2007. ICM 2007, December 2007, pp. 271{274.

[30]. Loi, Kung Chi Cinnati. *Field-programmable Gate-array (FPGA) Implementation of Low-density Parity-check (LDPC) Decoder in Digital Video Broadcasting-Second Generation Satellite (DVB-S2)*. Diss. University of Saskatchewan, 2010.

[31] Matt Pharr and Randima Fernando. GPU Gems 2: Programming Techniques for High-Performance Graphics and General-Purpose Computation. Addison-Wesley Professional, Boston, MA, USA, 2005.

[32] K. Basterretxea, J. Tarela, and I. D. Campo. Approximation of sigmoid function and the derivative for hardware implementation of arti_cial neurons. *IEE Proceedings - Circuits, Devices and Systems*, 151(1):18.24, February 2004.

[33] F. Piazza, A. Uncini, and M. Zenobi. Neural networks with digital lut activation functions. *IEE Proceedings - Circuits, Devices and Systems*, 151(1):18.24, February 2004.

[34] S. Vassiliadis, M. Zhang, and J. Delgado-Frias. Elementary function generators for neural-network emulators. *IEEE Transactions on Neural Networks*, 11(6):1438.1449, November 2000.

[35]. Leboeuf, Karl, et al. "High speed VLSI implementation of the hyperbolic tangent sigmoid function." *2008 Third International Conference on Convergence and Hybrid Information Technology*. Vol. 1. IEEE, 2008.

[36]. Leboeuf, Karl, et al. "High speed VLSI implementation of the hyperbolic tangent sigmoid function." *2008 Third International Conference on Convergence and Hybrid Information Technology*. Vol. 1. IEEE, 2008.

[37]. R. Muscedere, V. Dimitrov, G. Jullien, and W. Miller. Efficient techniques for binary-to multi digit multi-dimensional logarithmic number system conversion using range addressable look-up tables. *IEEE Transactions on Computers*, 54(3):257.271, March 2005.

[38] S. Lin and D. J. C. Jr., Error Control Coding, 2nd ed. Upper Saddle River, NJ, USA: Pearson Prentice Hall, 2004.

[39] https://www.xilinx.com/support/documentation/sw_manuals/ug998-vivado-intro-fpga-design-hls.pdf

[40]. Viterbi, Andrew J., and Jim K. Omura. *Principles of digital communication and coding*. Courier Corporation, 2013.

[41] R. Gray and A. Wyner, “Source coding for a simple network,” Bell Systems Technical Journal, vol. 53, no. 9, pp. 1681 – 1721, 1974

Appendix A

Basic measures of information proposed by Shannon

In Appendix A, the primary measures of information proposed by Shannon are presented. In Appendix A, the logarithm is considered as base two otherwise specified [41].

Definition 1: The entropy of a random variable X with discrete alphabet χ and probability distribution $p(x) = \Pr(X = x)$ is given by

$$H(X) = -\sum_{x \epsilon \chi} p(x) \log p(x)$$

Definition 2: Let X, Y be two discrete random variables with joint probability distribution $p(x, y)$, then the joint entropy of X given Y is given by

$$H(X, Y) = -\sum_{x \epsilon \chi} \sum_{y \epsilon \Upsilon} p(x, y) \log p(x, y)$$

Definition 3: Let X, Y be two discrete random variables with joint probability distribution $p(x, y)$, then the conditional entropy of X given Y is given by

$$H(X|\,Y) = -\sum_{x\epsilon\chi}\sum_{y\epsilon\Upsilon} p(x,y)\log p(x|\,y)$$

Definition 4: The mutual information between random variables X and Y defined over alphabet χ and Υ, respectively, is defined by

$$I(X\,;\,Y) = -\sum_{x\epsilon\chi}\sum_{y\epsilon\Upsilon} p(x,y)\log\frac{p(x,y)}{p(x)p(y)}$$

The concept can be expended to random process.

Definition 5: The entropy rate of the random process $\{X_i\}_{i=1}^{\infty}$is given by

$$H(X) = \lim_{n\to\infty}\frac{1}{n}H(X_1, X_2, \dots, X_n)$$

The entropy rate may not exist for all random processes, but for a stationary source $\{X_i\}_{i=1}^{\infty}$, its entropy rate $H(X)$ and is equal to H($X_n|\,X_{n-1},\ X_{n-2}, \dots, X_1$)

Definition 6: A $(2^{nR_1},\ 2^{nR_2}, n)$ distributed source code for the joint source (X, Y) consists of two encoder maps,

$$f_1\colon \chi^n \to \{1, 2, \dots, 2^{nR_1}\}$$

$$f_2\colon \mathcal{Y}^n \to \{1, 2, \dots, 2^{nR_2}\}$$

And a decoder map

$$g : \{1, 2, \dots, 2^{nR_1}\} \times \{1, 2, \dots, 2^{nR_2}\} \to \chi^n \times \mathcal{Y}^n$$

Where (R_1, R_2) is called the rate pair of the code.

Definition 7: The probability of error for a distributed source code is defined as

$$P_e^{(n)} = P(g(f_1(X^n), f_2(Y^n)) \neq (X^n, Y^n))$$

Definition 8: A rate pair (R_1, R_2) is said to be achievable for a source pair $\{(X_i, Y_i)\}_{i=1}^{\infty}$ if there exists a sequence of $(2^{nR_1},\ 2^{nR_2}, n)$ distributed source code with $P_e^{(n)} \to 0$. The achievable region is closer to the set of achievable rates.

Source coding of Gray Wyner network

The problem of source coding subject to fidelity criterion for a simple network connecting a single source with two receivers via a common channel and two private channels. The region of available rates is formulated as an information-theoretic minimization. Let consider $\{X_k, Y_k\}_{k=1}^{\infty}$ be a sequence of independent drawing of a pair of random variables (X, Y), $X \epsilon x$, $Y \epsilon x \mathcal{Y}$. x and $\mathcal{Y}$ are finite sets and $Pr\{X = x, Y = y\} = Q(x, y)$, $x \epsilon x$, $y \epsilon x \mathcal{Y}$. The marginal distributions are $Q_X(x) = \sum_{y \epsilon x \mathcal{Y}} Q(x, y)$, and $Q_Y(y) = \sum_{x \epsilon x} Q(x, y)$. When the random variables are clear from the context, we write $Q_X(x)$ as $Q(x)$, etc. Define for $m = 1,2, \dots, m - 1$, the set $I_{m=}\{0,1,2, \dots, m - 1\}$

An encoder $(n, M_0, M_1, M_2)$with parameters is mapping

$$f_E: x^n \times \mathcal{y}^n \to I_{M_0} \times I_{M_1} \times I_{M_2}$$

Given an encoder, the decoder is a pair of mappings

$$f_D^{(X)}: I_{M_0} \times I_{M_1} \to x^n$$

$$f_D^{(Y)}: I_{M_0} \times I_{M_2} \to \mathcal{y}^n$$

An encoder with parameters (n, M_0, M_1, M_2) is applied as follows. Let $f_E(X, Y) = (S_0, S_1, S_2\ )$ where $X = (X_1\ , X_2\ , \dots, X_n\ )$ and $Y = (Y_1\ , Y_2\ , \dots, Y_n\ )$. Then let $\hat{X} = f_D^{(X)}(S_0, S_1\ )$ and $\hat{Y} =$

$f_D^{(Y)}(S_0, S_2)$. The resulting error rate is $\Delta = \max(\Delta_x, \Delta_y)$ where $\Delta_x = E\frac{1}{n}\sum_{k=1}^{n} \mathrm{d}_H(X_K, \hat{X}_K)$ and $\Delta_y = E\frac{1}{n}\sum_{k=1}^{n} \mathrm{d}_H(Y_K, \hat{Y}_K)$. $\mathrm{D}_H(.,.)$ is defined as follows.

$$\mathrm{D}_H(u, \hat{u}) = \begin{cases} 0 & u = \hat{u} \\ 1 & u \neq \hat{u} \end{cases}$$

The Hamming distance $D_H(u, v)$ between the n −vectors u and v is the number of positions in which u and v differ. Thus, $\Delta_x = E(\frac{1}{n})D_H(X, \hat{X})$ and $\Delta_y = E(\frac{1}{n})D_H(Y, \hat{Y})$. The communication system of the correspondence defined encoder and decoder is shown in Figure (B.1).

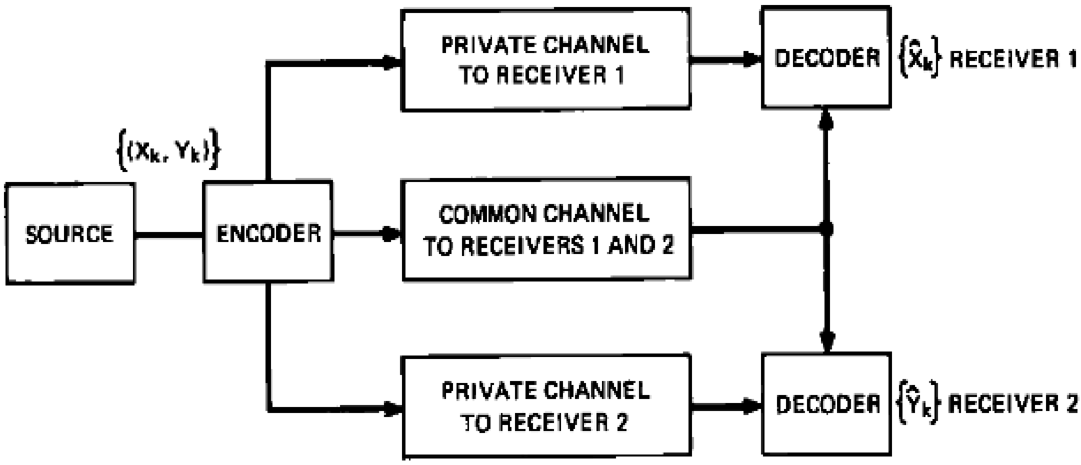


Figure (B.1). Source coding for a simple Gray Wyner network [40].

The capacity of the channels must be at least $c_i = (1/n)\log_2 M_i$ $(i = 0,1,2)$. The achievable rate region is $\mathcal{R}$ in which a triple of (R_0, R_1, R_2) exit. (R_0, R_1, R_2) is achievable if, for arbitrary $\epsilon > 0$, there exists a code with parameters (n, M_0, M_1, M_2) with $M_i \leq 2^{n(R_i+\epsilon)}$, $i = 0,1,2$, and error rate $\Delta < \epsilon$.

The property of $(R_0, R_1, R_2) \in \mathcal{R} \rightarrow (R_0 + \epsilon_0, R_1 + \epsilon_1, R_2 + \epsilon_2) \in \mathcal{R}$ causes by the fact that $\mathcal{R}$ is a closed subset of Euclidean three-space**.** Therefore, the lower boundary of the $\mathcal{R}$ region is defined as follow;

$$\bar{\mathcal{R}} \triangleq \left\{(R_0, R_1, R_2) \in \mathcal{R} : (\hat{R}_0, \hat{R}_1, \hat{R}_2) \in \mathcal{R}, \hat{R}_i \leq R_i (i = 0,1,2) \rightarrow \hat{R}_i = R_i (i = 0,1,2)\right\}$$

Because of the convexity of $\mathcal{R}$, we can define $T(\alpha_0, \alpha_1, \alpha_2) = \min_{(R_0, R_1, R_2) \in \mathcal{R}} (R_0\alpha_0 + R_1\alpha_1 + R_2\alpha_2)$. Indeed, the lower boundary $\bar{\mathcal{R}}$ is the upper envelope of the family of planes $T(\alpha_0, \alpha_1, \alpha_2) = \sum_0^2 \alpha_i R_i. T(\alpha_0, \alpha_1, \alpha_2)$ as the minimum cost of transmitting, using a code with rate-triple (R_0, R_1, R_2) over the network of Figure (B.1), when the cost of transmitting a bit per second over the common channel is α_0 and the costs of transmitting a bit per second over the private channels to receivers 1 and 2 are α_1 and α_2, respectively. Now, since information sent over the common channel can also alternatively be sent over *both* private channels, it is never necessary to consider the case where the sum of the costs of a bit per second on the private channels $\alpha_1 + \alpha_2 < \alpha_0$ the cost of a bit per second on the common channel. Similarly, we need never consider the cases where $\alpha_1 > \alpha_0$, or $\alpha_2 > \alpha_0$, since information transmitted over a private channel can alternatively be sent over the common channel. Since we can normalize α_0 as unity, the following theorem should be plausible. Thus, for $R = (R_0, R_1, R_2)$ satisfying $R_i \geq 0$ and $\alpha = (\alpha_1, \alpha_2)$ arbitrary, let the cost defined by

$$\mathrm{c}(\alpha, \mathrm{R}) = \mathrm{R}_0 + \mathrm{R}_1\alpha_1 + \mathrm{R}_2\alpha_2$$

$$T(\alpha) = \min_{R \in \mathcal{R}} (\alpha, R).$$

The following theorem[], give the lower bound to the region $\mathcal{R}$.

Theorem: If $(R_0, R_1, R_2) \in \mathcal{R}$ then, the lower bound to the region $\mathcal{R}$ is as follow

a) $R_0 + R_1 + R_2 \geq H(X, Y)$

b) $R_0 + R_1 \geq H(X)$

c) $R_0 + R_2 \geq H(Y)$

Appendix B

Values from Annex B and C of the DVB-S2 Standard

According to the standard, even though the parity check matrices, H, chosen by the standard are sparse, their corresponding generator matrices are not. Thus, the DVB-S2 standard adopts a special structure of the H matrix in order to reduce the memory requirement and the complexity of the encoder. In this Appendix, the values from Annex B and C of the DVB-S2 standard [27] are reproduced.

The standard introduces a periodicity of $M = 360$ to the submatrix A in order to further reduce storage requirements. The periodicity condition divides the A matrix into groups of $M = 360$ columns. For each group, the locations of the non-zero elements of the first column are given in the following. Let the set of non-zero locations on first, or leftmost, column of a group be $c_0, c_1, c_2, \dots, c_{db-1}$ where db is the number of non-zero elements in that first column. For each of the , $M - 1 = 359$, other columns, the locations of the non-zero elements of the ith column of the group are given by $(c_0 + (i-1)p) mod(N-K)$, $(c_1 + (i-1)p) mod(N-K)$, … , $(c_l + (i-1)p) mod(N-K)$. Where $N - K$ is the number of parity-check bits and $p = \frac{N-K}{M}$ code dependent constant.

The values for the normal frames are shown first, followed by the values for short frames.

Table B.1: N = 64800, Code Rate = ¼

23606 36098 1140 28859 18148 18510 6226 540 42014
20879 23802 47088
16419 24928 16609 17248 7693 24997 42587 16858
34921 21042 37024 20692
1874 40094 18704 14474 14004 11519 13106 28826
38669 22363 30255 31105
22254 40564 22645 22532 6134 9176 39998 23892 8937
15608 16854 31009
8037 40401 13550 19526 41902 28782 13304 32796
24679 27140 45980 10021
40540 44498 13911 22435 32701 18405 39929 25521
12497 9851 39223 34823
15233 45333 5041 44979 45710 42150 19416 1892 23121
15860 8832 10308
10468 44296 3611 1480 37581 32254 13817 6883 32892
40258 46538 11940
6705 21634 28150 43757 895 6547 20970 28914 30117
25736 41734 11392 22002 5739 27210 27828 34192
37992 10915 6998 3824 42130 4494 35739
8515 1191 13642 30950 25943 12673 16726 34261 31828
3340 8747 39225
18979 17058 43130 4246 4793 44030 19454 29511 47929
15174 24333 19354
16694 8381 29642 46516 32224 26344 9405 18292 12437
27316 35466 41992
15642 5871 46489 26723 23396 7257 8974 3156 37420
44823 35423 13541
42858 32008 41282 38773 26570 2702 27260 46974 1469
20887 27426 38553
22152 24261 8297
19347 9978 27802
34991 6354 33561
29782 30875 29523
9278 48512 14349
38061 4165 43878
8548 33172 34410
22535 28811 23950
20439 4027 24186
38618 8187 30947
35538 43880 21459
7091 45616 15063
5505 9315 21908
36046 32914 11836
7304 39782 33721
16905 29962 12980
11171 23709 22460
34541 9937 44500
14035 47316 8815
15057 45482 24461
30518 36877 879
7583 13364 24332
448 27056 4682
12083 31378 21670
1159 18031 2221
17028 38715 9350
17343 24530 29574
46128 31039 32818
20373 36967 18345
46685 20622 32806

Table B.2: N = 64800, Code Rate = 1/3

34903 20927 32093 1052 25611 16093 16454 5520 506 37399
18518 21120
11636 14594 22158 14763 15333 6838 22222 37856 14985
31041 18704 32910
17449 1665 35639 16624 12867 12449 10241 11650 25622
34372 19878 26894
29235 19780 36056 20129 20029 5457 8157 35554 21237 7943
13873 14980
9912 7143 35911 12043 17360 37253 25588 11827 29152
21936 24125 40870
40701 36035 39556 12366 19946 29072 16365 35495 22686
11106 8756 34863
19165 15702 13536 40238 4465 40034 40590 37540 17162
1712 20577 14138
31338 19342 9301 39375 3211 1316 33409 28670 12282 6118
29236 35787
11504 30506 19558 5100 24188 24738 30397 33775 9699 6215
3397 37451
34689 23126 7571 1058 12127 27518 23064 11265 14867
30451 28289 2966
11660 15334 16867 15160 38343 3778 4265 39139 17293
26229 42604 13486
31497 1365 14828 7453 26350 41346 28643 23421 8354 16255
11055 24279
15687 12467 13906 5215 41328 23755 20800 6447 7970 2803
33262 39843
5363 22469 38091 28457 36696 34471 23619 2404 24229
41754 1297 18563
3673 39070 14480 30279 37483 7580 29519 30519 39831
20252 18132 20010
34386 7252 27526 12950 6875 43020 31566 39069 18985
15541 40020 16715
1721 37332 39953 17430 32134 29162 10490 12971 28581
29331 6489 35383
736 7022 42349 8783 6767 11871 21675 10325 11548 25978
431 24085
1925 10602 28585 12170 15156 34404 8351 13273 20208 5800
15367 21764
16279 37832 34792 21250 34192 7406 41488 18346 29227
26127 25493 7048
39948 28229 24899 38788 27081 7936
17408 14274 38993 4368 26148 10578
29094 5357 19224
9562 24436 28637
40177 2326 13504
6834 21583 42516
40651 42810 25709
31557 32138 38142
18624 41867 39296
37560 14295 16245
6821 21679 31570
25339 25083 22081
8047 697 35268
9884 17073 19995
26848 35245 8390
18658 16134 14807
12201 32944 5035
25236 1216 38986
42994 24782 8681
28321 4932 34249
4107 29382 32124
22157 2624 14468

38774 15968 28459 25353 4122 39751
41404 27249 27425
41229 6082 43114
13957 4979 40654
3093 3438 34992
34082 6172 28760
42210 34141 41021
14705 17783 10134
41755 39884 22773
14615 15593 1642
29111 37061 39860
9579 33552 633
12951 21137 39608
38244 27361 29417
2939 10172 36479

Table B.3: N = 64800, Code Rate = 2/5

31413 18834 28884 947 23050 14484 14809 4968 455
33659 16666 19008
13172 19939 13354 13719 6132 20086 34040 13442
27958 16813 29619 16553
1499 32075 14962 11578 11204 9217 10485 23062
30936 17892 24204 24885
32490 18086 18007 4957 7285 32073 19038 7152 12486
13483 24808 21759
32321 10839 15620 33521 23030 10646 26236 19744
21713 36784 8016 12869
35597 11129 17948 26160 14729 31943 20416 10000
7882 31380 27858 33356
14125 12131 36199 4058 35992 36594 33698 15475
1566 18498 12725 7067
17406 8372 35437 2888 1184 30068 25802 11056 5507
26313 32205 37232
15254 5365 17308 22519 35009 718 5240 16778 23131
24092 20587 33385
27455 17602 4590 21767 22266 27357 30400 8732 5596
3060 33703 3596
6882 873 10997 24738 20770 10067 13379 27409 25463
2673 6998 31378
15181 13645 34501 3393 3840 35227 15562 23615
38342 12139 19471 15483
13350 6707 23709 37204 25778 21082 7511 14588
10010 21854 28375 33591
12514 4695 37190 21379 18723 5802 7182 2529 29936
35860 28338 10835
25796 31795
12152 12184
35088 31226
38263 33386
24892
23114 37995
29796
34336 10551
36245
35407 175
7203
14654 38201
22605
28404 6595
1018
19932 3524
29305
31749 20247
8128
18026 36357
26735
7543 29767
13588
13333 25965
8463
14504 36796
19710
4528 25299
7318
28229 31684
30160
15293 8483
28002
14880 13334
12584
28646 2558
19687
6259 4499
26336
11952 28386
8405
10609 961
7582
10423 13191
26818
15922 36654
21450
10492 1532
1205
30551 36482
22153
5156 11330
34243
28616 35369
13322
8962 1485
21186

34283 25610 33026 31017 21259 2165 21807 37578
1175 16710 21939 30841
27292 33730 6836 26476 27539 35784 18245 16394
17939 23094 19216 17432
11655 6183 38708 28408 35157 17089 13998 36029
15052 16617 5638 36464
15693 28923 26245 9432 11675 25720 26405 5838
31851 26898 8090 37037
24418 27583 7959 35562 37771 17784 11382 11156
37855 7073 21685 34515
10977 13633 30969 7516 11943 18199 5231 13825
19589 23661 11150 35602
19124 30774 6670 37344 16510 26317 23518 22957
6348 34069 8845 20175
34985 14441 25668 4116 3019 21049 37308 24551
24727 20104 24850 12114
38187 28527 13108 13985 1425 21477 30807 8613
26241 33368 35913 32477
5903 34390 24641 26556 23007 27305 38247 2621 9122
32806 21554 18685
17287 27292 19033

35091 25550
14798
7824 215 1248
30848 5362
17291
28932 30249
27073 13062
2103 16206
7129 32062
19612
9512 21936
38833
35849 33754
23450
18705 28656
18111
22749 27456
32187

23541 17445
35561
33133 11593
19895
33917 7863
33651
20063 28331
10702
13195 21107
21859
4364 31137
4804
5585 2037
4830
30672 16927
14800

Table B.4: N = 64800, Code Rate = ½

54 9318 14392 27561 26909 10219 2534 8597
55 7263 4635 2530 28130 3033 23830 3651
56 24731 23583 26036 17299 5750 792 9169
57 5811 26154 18653 11551 15447 13685 16264
58 12610 11347 28768 2792 3174 29371 12997
59 16789 16018 21449 6165 21202 15850 3186
60 31016 21449 17618 6213 12166 8334 18212
61 22836 14213 11327 5896 718 11727 9308
62 2091 24941 29966 23634 9013 15587 5444
63 22207 3983 16904 28534 21415 27524 25912
64 25687 4501 22193 14665 14798 16158 5491
65 4520 17094 23397 4264 22370 16941 21526
66 10490 6182 32370 9597 30841 25954 2762
67 22120 22865 29870 15147 13668 14955 19235
68 6689 18408 18346 9918 25746 5443 20645
69 29982 12529 13858 4746 30370 10023 24828
70 1262 28032 29888 13063 24033 21951 7863
71 6594 29642 31451 14831 9509 9335 31552
72 1358 6454 16633 20354 24598 624 5265
73 19529 295 18011 3080 13364 8032 15323
74 11981 1510 7960 21462 9129 11370 25741

40 30051 30426
41 1335 15424
42 6865 17742
43 31779 12489
44 32120 21001
45 14508 6996
46 979 25024
47 4554 21896
48 7989 21777
49 4972 20661
50 6612 2730
51 12742 4418
52 29194 595
53 19267 20113

75 9276 29656 4543 30699 20646 21921 28050
76 15975 25634 5520 31119 13715 21949 19605
77 18688 4608 31755 30165 13103 10706 29224
78 21514 23117 12245 26035 31656 25631 30699
79 9674 24966 31285 29908 17042 24588 31857
80 21856 27777 29919 27000 14897 11409 7122
81 29773 23310 263 4877 28622 20545 22092
82 15605 5651 21864 3967 14419 22757 15896
83 30145 1759 10139 29223 26086 10556 5098
84 18815 16575 2936 24457 26738 6030 505
85 30326 22298 27562 20131 26390 6247 24791
86 928 29246 21246 12400 15311 32309 18608
87 20314 6025 26689 16302 2296 3244 19613
88 6237 11943 22851 15642 23857 15112 20947
89 26403 25168 19038 18384 8882 12719 7093
0 14567 24965
1 3908 100
2 10279 240
3 24102 764
4 12383 4173
5 13861 15918
6 21327 1046
7 5288 14579
8 28158 8069
9 16583 11098
10 16681 28363
11 13980 24725
12 32169 17989
13 10907 2767
14 21557 3818
15 26676 12422
16 7676 8754
17 14905 20232
18 15719 24646
19 31942 8589
20 19978 27197
21 27060 15071
22 6071 26649
23 10393 11176
24 9597 13370
25 7081 17677
26 1433 19513
27 26925 9014
28 19202 8900
29 18152 30647
30 20803 1737

31 11804 25221
32 31683 17783
33 29694 9345
34 12280 26611
35 6526 26122
36 26165 11241
37 7666 26962
38 16290 8480
39 11774 10120

Table B.5: N = 64800, Code Rate = 3/5

22422 10282 11626 19997 11161 2922 3122 99 5625 17064 8270 179
25087 16218 17015 828 20041 25656 4186 11629 22599 17305 22515 6463
11049 22853 25706 14388 5500 19245 8732 2177 13555 11346 17265 3069
16581 22225 12563 19717 23577 11555 25496 6853 25403 5218 15925 21766
16529 14487 7643 10715 17442 11119 5679 14155 24213 21000 1116 15620
5340 8636 16693 1434 5635 6516 9482 20189 1066 15013 25361 14243
18506 22236 20912 8952 5421 15691 6126 21595 500 6904 13059 6802
8433 4694 5524 14216 3685 19721 25420 9937 23813 9047 25651 16826
21500 24814 6344 17382 7064 13929 4004 16552 12818 8720 5286 2206
22517 2429 19065 2921 21611 1873 7507 5661 23006 23128 20543 19777
1770 4636 20900 14931 9247 12340 11008 12966 4471 2731 16445 791
6635 14556 18865 22421 22124 12697 9803 25485 7744 18254 11313 9004
19982 23963 18912 7206 12500 4382 20067 6177 21007 1195 23547 24837
756 11158 14646 20534 3647 17728 11676 11843 12937 4402 8261 22944
9306 24009 10012 11081 3746 24325 8060 19826 842 8836 2898 5019
7575 7455 25244 4736 14400 22981 5543 8006 24203 13053 1120 5128
3482 9270 13059 15825 7453 23747 3656 24585 16542 17507 22462 14670
25 6393 3725
26 597 19968
27 5743 8084
28 6770 9548
29 4285 17542
30 13568 22599
31 1786 4617
32 23238 11648
33 19627 2030
34 13601 13458
35 13740 17328
36 25012 13944
37 22513 6687
38 4934 12587
39 21197 5133
40 22705 6938
41 7534 24633
42 24400 12797
43 21911 25712
44 12039 1140
45 24306 1021
46 14012 20747
47 11265 15219
48 4670 15531
49 9417 14359
50 2415 6504
51 24964 24690
52 14443 8816
53 6926 1291
54 6209 20806
55 13915 4079
56 24410 13196
57 13505 6117

15627 15290 4198 22748 5842 13395 23918 16985 14929 3726
25350 24157
24896 16365 16423 13461 16615 8107 24741 3604 25904 8716 9604
20365
3729 17245 18448 9862 20831 25326 20517 24618 13282 5099
14183 8804
16455 17646 15376 18194 25528 1777 6066 21855 14372 12517
4488 17490
1400 8135 23375 20879 8476 4084 12936 25536 22309 16582 6402
24360
25119 23586 128 4761 10443 22536 8607 9752 25446 15053 1856
4040
377 21160 13474 5451 17170 5938 10256 11972 24210 17833 22047
16108
13075 9648 24546 13150 23867 7309 19798 2988 16858 4825 23950
15125
20526 3553 11525 23366 2452 17626 19265 20172 18060 24593
13255 1552
18839 21132 20119 15214 14705 7096 10174 5663 18651 19700
12524 14033
4127 2971 17499 16287 22368 21463 7943 18880 5567 8047 23363
6797
10651 24471 14325 4081 7258 4949 7044 1078 797 22910 20474
4318
21374 13231 22985 5056 3821 23718 14178 9978 19030 23594 8895
25358
6199 22056 7749 13310 3999 23697 16445 22636 5225 22437 24153
9442
7978 12177 2893 20778 3175 8645 11863 24623 10311 25767 17057
3691
20473 11294 9914 22815 2574 8439 3699 5431 24840 21908 16088
18244
8208 5755 19059 8541 24924 6454 11234 10492 16406 10831 11436
9649
16264 11275 24953 2347 12667 19190 7257 7174 24819 2938 2522
11749
3627 5969 13862 1538 23176 6353 2855 17720 2472 7428 573 15036
0 18539 18661 61 20671 24913
1 10502 3002 62 24558 20591
2 9368 10761 63 12402 3702
3 12299 7828 64 8314 1357
4 15048 13362 65 20071 14616
5 18444 24640 66 17014 3688
6 20775 19175 67 19837 946
7 18970 10971 68 15195 12136
8 5329 19982 69 7758 22808

58 9869 8220
59 1570 6044
60 25780 17387

9 11296 18655 70 3564 2925
10 15046 20659 71 3434 7769
11 7300 22140
12 22029 14477
13 11129 742
14 13254 13813
15 19234 13273
16 6079 21122
17 22782 5828
18 19775 4247
19 1660 19413
20 4403 3649
21 13371 25851
22 22770 21784
23 10757 14131
24 16071 21617

Table B.6: N = 64800, Code Rate = 2/3

0 10491 16043 506 12826 8065 8226 2767 240 18673 9279 10579 20928
1 17819 8313 6433 6224 5120 5824 12812 17187 9940 13447 13825 18483
2 17957 6024 8681 18628 12794 5915 14576 10970 12064 20437 4455 7151
3 19777 6183 9972 14536 8182 17749 11341 5556 4379 17434 15477 18532
4 4651 19689 1608 659 16707 14335 6143 3058 14618 17894 20684 5306
5 9778 2552 12096 12369 15198 16890 4851 3109 1700 18725 1997 15882
6 486 6111 13743 11537 5591 7433 15227 14145 1483 3887 17431 12430
7 20647 14311 11734 4180 8110 5525 12141 15761 18661 18441 10569 8192
8 3791 14759 15264 19918 10132 9062 10010 12786 10675 9682 19246 5454
9 19525 9485 7777 19999 8378 9209 3163 20232 6690 16518 716 7353
10 4588 6709 20202 10905 915 4317 11073 13576 16433 368 3508 21171
11 14072 4033 19959 12608 631 19494 14160 8249 10223 21504 12395 4322
12 13800 14161
13 2948 9647

4 9161 15642
5 10714 10153
6 11585 9078
7 5359 9418
8 9024 9515
9 1206 16354
10 14994 1102
11 9375 20796
12 15964 6027
13 14789 6452
14 8002 18591
15 14742 14089
16 253 3045
17 1274 19286
18 14777 2044
19 13920 9900
20 452 7374
21 18206 9921
22 6131 5414
23 10077 9726
24 12045 5479
25 4322 7990
26 15616 5550
27 15561 10661
28 20718 7387
29 2518 18804

14 14693 16027
15 20506 11082
16 1143 9020
17 13501 4014
18 1548 2190
19 12216 21556
20 2095 19897
21 4189 7958
22 15940 10048
23 515 12614
24 8501 8450
25 17595 16784
26 5913 8495
27 16394 10423
28 7409 6981
29 6678 15939
30 20344 12987
31 2510 14588
32 17918 6655
33 6703 19451
34 496 4217
35 7290 5766
36 10521 8925
37 20379 11905
38 4090 5838
39 19082 17040
40 20233 12352
41 19365 19546
42 6249 19030
43 11037 19193
44 19760 11772
45 19644 7428
46 16076 3521
47 11779 21062
48 13062 9682
49 8934 5217
50 11087 3319
51 18892 4356
52 7894 3898
53 5963 4360
54 7346 11726
55 5182 5609
56 2412 17295
57 9845 20494
58 6687 1864
59 20564 5216

30 8984 2600
31 6516 17909
32 11148 98
33 20559 3704
34 7510 1569
35 16000 11692
36 9147 10303
37 16650 191
38 15577 18685
39 17167 20917
40 4256 3391
41 20092 17219
42 9218 5056
43 18429 8472
44 12093 20753
45 16345 12748
46 16023 11095
47 5048 17595
48 18995 4817
49 16483 3536
50 1439 16148
51 3661 3039
52 19010 18121
53 8968 11793
54 13427 18003
55 5303 3083
56 531 16668
57 4771 6722
58 5695 7960
59 3589 14630

0 18226 17207
1 9380 8266
2 7073 3065
3 18252 13437

Table B.7: N = 64800, Code Rate = ¾

0 6385 7901 14611 13389 11200 3252 5243 2504 2722 821 7374
1 11359 2698 357 13824 12772 7244 6752 15310 852 2001 11417
2 7862 7977 6321 13612 12197 14449 15137 13860 1708 6399 13444
3 1560 11804 6975 13292 3646 3812 8772 7306 5795 14327 7866
4 7626 11407 14599 9689 1628 2113 10809 9283 1230 15241 4870
5 1610 5699 15876 9446 12515 1400 6303 5411 14181 13925 7358
6 4059 8836 3405 7853 7992 15336 5970 10368 10278 9675 4651
7 4441 3963 9153 2109 12683 7459 12030 12221 629 15212 406
8 6007 8411 5771 3497 543 14202 875 9186 6235 13908 3563
9 3232 6625 4795 546 9781 2071 7312 3399 7250 4932 12652
10 8820 10088 11090 7069 6585 13134 10158 7183 488 7455 9238
11 1903 10818 119 215 7558 11046 10615 11545 14784 7961 15619
12 3655 8736 4917 15874 5129 2134 15944 14768 7150 2692 1469
13 8316 3820 505 8923 6757 806 7957 4216 15589 13244 2622
14 14463 4852 15733 3041 11193 12860 13673 8152 6551 15108
8758
15 3149 11981
16 13416 6906
17 13098 13352
18 2009 14460
19 7207 4314
20 3312 3945
21 4418 6248
22 2669 13975
23 7571 9023
24 14172 2967
25 7271 7138
26 6135 13670
27 7490 14559
28 8657 2466
29 8599 12834
30 3470 3152
31 13917 4365
32 6024 13730
33 10973 14182
34 2464 13167
35 5281 15049
36 1103 1849
24 2655 14957
25 5565 6332
26 4303 12631
27 11653 12236
28 16025 7632
29 4655 14128
30 9584 13123
31 13987 9597
32 15409 12110
33 8754 15490
34 7416 15325
35 2909 15549
36 2995 8257
37 9406 4791
38 11111 4854
39 2812 8521
40 8476 14717
41 7820 15360
42 1179 7939
43 2357 8678
44 7703 6216
0 3477 7067
1 3931 13845
2 7675 12899
3 1754 8187
4 7785 1400
5 9213 5891
6 2494 7703
7 2576 7902
8 4821 15682
9 10426 11935
10 1810 904
11 11332 9264
12 11312 3570
13 14916 2650
14 7679 7842
15 6089 13084
16 3938 2751

37 2058 1069
38 9654 6095
39 14311 7667
40 15617 8146
41 4588 11218
42 13660 6243
43 8578 7874
44 11741 2686
0 1022 1264
1 12604 9965
2 8217 2707
3 3156 11793
4 354 1514
5 6978 14058
6 7922 16079
7 15087 12138
8 5053 6470
9 12687 14932
10 15458 1763
11 8121 1721
12 12431 549
13 4129 7091
14 1426 8415
15 9783 7604
16 6295 11329
17 1409 12061
18 8065 9087
19 2918 8438
20 1293 14115
21 3922 13851
22 3851 4000
23 5865 1768
17 8509 4648
18 12204 8917
19 5749 12443
20 12613 4431
21 1344 4014
22 8488 13850
23 1730 14896
24 14942 7126
25 14983 8863
26 6578 8564
27 4947 396
28 297 12805
29 13878 6692
30 11857 11186
31 14395 11493
32 16145 12251
33 13462 7428
34 14526 13119
35 2535 11243
36 6465 12690
37 6872 9334
38 15371 14023
39 8101 10187
40 11963 4848
41 15125 6119
42 8051 14465
43 11139 5167
44 2883 14521

Table B.8: N = 64800, Code Rate = 4/5

0 149 11212 5575 6360 12559 8108 8505 408 10026 12828
1 5237 490 10677 4998 3869 3734 3092 3509 7703 10305
2 8742 5553 2820 7085 12116 10485 564 7795 2972 2157
3 2699 4304 8350 712 2841 3250 4731 10105 517 7516
4 12067 1351 11992 12191 11267 5161 537 6166 4246 2363
5 6828 7107 2127 3724 5743 11040 10756 4073 1011 3422
6 11259 1216 9526 1466 10816 940 3744 2815 11506 11573
7 4549 11507 1118 1274 11751 5207 7854 12803 4047 6484
8 8430 4115 9440 413 4455 2262 7915 12402 8579 7052
9 3885 9126 5665 4505 2343 253 4707 3742 4166 1556
3 6970 5447
4 3217 5638
5 8972 669
6 5618 12472
7 1457 1280
8 8868 3883
9 8866 1224
10 8371 5972
11 266 4405
12 3706 3244

10 1704 8936 6775 8639 8179 7954 8234 7850 8883 8713
11 11716 4344 9087 11264 2274 8832 9147 11930 6054 5455
12 7323 3970 10329 2170 8262 3854 2087 12899 9497 11700
13 4418 1467 2490 5841 817 11453 533 11217 11962 5251
14 1541 4525 7976 3457 9536 7725 3788 2982 6307 5997
15 11484 2739 4023 12107 6516 551 2572 6628 8150 9852
16 6070 1761 4627 6534 7913 3730 11866 1813 12306 8249
17 12441 5489 8748 7837 7660 2102 11341 2936 6712 11977
18 10155 4210
19 1010 10483
20 8900 10250
21 10243 12278
22 7070 4397
23 12271 3887
24 11980 6836
25 9514 4356
26 7137 10281
27 11881 2526
28 1969 11477
29 3044 10921
30 2236 8724
31 9104 6340
32 7342 8582
33 11675 10405
34 6467 12775
35 3186 12198
0 9621 11445
1 7486 5611
2 4319 4879
3 2196 344
4 7527 6650
5 10693 2440
6 6755 2706
7 5144 5998
8 11043 8033
9 4846 4435
10 4157 9228
11 12270 6562
12 11954 7592
13 7420 2592
14 8810 9636
15 689 5430
16 920 1304
17 1253 11934
18 9559 6016
19 312 7589
13 6039 5844
14 7200 3283
15 1502 11282
16 12318 2202
17 4523 965
18 9587 7011
19 2552 2051
20 12045 10306
21 11070 5104
22 6627 6906
23 9889 2121
24 829 9701
25 2201 1819
26 6689 12925
27 2139 8757
28 12004 5948
29 8704 3191
30 8171 10933
31 6297 7116
32 616 7146
33 5142 9761
34 10377 8138
35 7616 5811
0 7285 9863
1 7764 10867
2 12343 9019
3 4414 8331
4 3464 642
5 6960 2039
6 786 3021
7 710 2086
8 7423 5601
9 8120 4885
10 12385 11990
11 9739 10034
12 424 10162
13 1347 7597
14 1450 112
15 7965 8478
16 8945 7397
17 6590 8316
18 6838 9011
19 6174 9410
20 255 113
21 6197 5835
22 12902 3844

20 4439 4197
21 4002 9555
22 12232 7779
23 1494 8782
24 10749 3969
25 4368 3479
26 6316 5342
27 2455 3493
28 12157 7405
29 6598 11495
30 11805 4455
31 9625 2090
32 4731 2321
33 3578 2608
34 8504 1849
35 4027 1151
0 5647 4935
1 4219 1870
2 10968 8054
23 4377 3505
24 5478 8672
25 4453 2132
26 9724 1380
27 12131 11526
28 12323 9511
29 8231 1752
30 497 9022
31 9288 3080
32 2481 7515
33 2696 268
34 4023 12341
35 7108 5553

Table B.9: N = 64800, Code Rate = 5/6

0 4362 416 8909 4156 3216 3112 2560 2912 6405 8593 4969 6723
1 2479 1786 8978 3011 4339 9313 6397 2957 7288 5484 6031 10217
2 10175 9009 9889 3091 4985 7267 4092 8874 5671 2777 2189 8716
3 9052 4795 3924 3370 10058 1128 9996 1016 5 9360 4297 434 5138
4 2379 7834 4835 2327 9843 804 329 8353 7167 3070 1528 7311
5 3435 7871 348 3693 1876 6585 10340 7144 5870 2084 4052 2780
6 3917 3111 3476 1304 10331 5939 5199 1611 1991 699 8316 9960
7 6883 3237 1717 10752 7891 9764 4745 3888 10009 4176 4614 1567
8 10587 2195 1689 2968 5420 2580 2883 6496 111 6023 1024 4449
9 3786 8593 2074 3321 5057 1450 3840 5444 6572 3094 9892 1512
10 8548 1848 10372 4585 7313 6536 6379 1766 9462 2456 5606 9975
11 8204 10593 7935 3636 3882 394 5968 8561 2395 7289 9267 9978
12 7795 74 1633 9542 6867 7352 6417 7568 10623 725 2531 9115
13 7151 2482 4260 5003 10105 7419 9203 6691 8798 2092 8263 3755
14 3600 570 4527 200 9718 6771 1995 8902 5446 768 1103 6520
15 6304 7621
16 6498 9209
17 7293 6786
18 5950 1708
19 8521 1793
14 7067 8878
15 9027 3415
16 1690 3866
17 2854 8469
18 6206 630
19 363 5453
20 4125 7008
21 1612 6702
22 9069 9226
23 5767 4060
24 3743 9237
25 7018 5572
26 8892 4536
27 853 6064
28 8069 5893
29 2051 2885
0 10691 3153
1 3602 4055
2 328 1717
3 2219 9299
4 1939 7898
5 617 206
6 8544 1374

20 6174 7854
21 9773 1190
22 9517 10268
23 2181 9349
24 1949 5560
25 1556 555
26 8600 3827
27 5072 1057
28 7928 3542
29 3226 3762
0 7045 2420
1 9645 2641
2 2774 2452
3 5331 2031
4 9400 7503
5 1850 2338
6 10456 9774
7 1692 9276
8 10037 4038
9 3964 338
10 2640 5087
11 858 3473
12 5582 5683
13 9523 916
14 4107 1559
15 4506 3491
16 8191 4182
17 10192 6157
18 5668 3305
19 3449 1540
20 4766 2697
21 4069 6675
22 1117 1016
23 5619 3085
24 8483 8400
25 8255 394
26 6338 5042
27 6174 5119
28 7203 1989
29 1781 5174
0 1464 3559
1 3376 4214
2 7238 67
3 10595 8831
4 1221 6513
5 5300 4652
7 10676 3240
8 6672 9489
9 3170 7457
10 7868 5731
11 6121 10732
12 4843 9132
13 580 9591
14 6267 9290
15 3009 2268
16 195 2419
17 8016 1557
18 1516 9195
19 8062 9064
20 2095 8968
21 753 7326
22 6291 3833
23 2614 7844
24 2303 646
25 2075 611
26 4687 362
27 8684 9940
28 4830 2065
29 7038 1363
0 1769 7837
1 3801 1689
2 10070 2359
3 3667 9918
4 1914 6920
5 4244 5669
6 10245 7821
7 7648 3944
8 3310 5488
9 6346 9666
10 7088 6122
11 1291 7827
12 10592 8945
13 3609 7120
14 9168 9112
15 6203 8052
16 3330 2895
17 4264 10563
18 10556 6496
19 8807 7645
20 1999 4530
21 9202 6818
22 3403 1734

6 1429 9749
7 7878 5131
8 4435 10284
9 6331 5507
10 6662 4941
11 9614 10238
12 8400 8025
13 9156 5630

23 2106 9023
24 6881 3883
25 3895 2171
26 4062 6424
27 3755 9536

Table B.10: N = 64800, Code Rate = 8/9

0 6235 2848
3222
1 5800 3492
5348
2 2757 927 90
15
3 6961 4516
4739
4 1172 3237
6264
5 1927 2425
3683
6 3714 6309
2495
7 3070 6342
7154
8 2428 613
3761
9 2906 264
5927
10 1716 1950
4273
11 4613 6179
3491
12 4865 3286
6005
13 1343 5923
3529
14 4589 4035
2132
15 1579 3920
6737
16 1644 1191
5998

13 1969 3869
14 3571 2420
4632 981
16 3215 4163
17 973 3117
18 3802 6198
19 3794 3948
0 3196 6126
1 573 1909
2 850 4034
3 5622 1601
4 6005 524
5 5251 5783
6 172 2032
7 1875 2475
8 497 1291
9 2566 3430
10 1249 740
11 2944 1948
12 6528 2899
13 2243 3616
14 867 3733
15 1374 4702
16 4698 2285
17 4760 3917
18 1859 4058
19 6141 3527
0 2148 5066
1 1306 145
2 2319 871
3 3463 1061
4 5554 6647
5 5837 339

6 5821 4932
7 6356 4756
8 3930 418
9 211 3094
10 1007 4928
11 3584 1235
12 6982 2869
13 1612 1013
14 953 4964
15 4555 4410
16 4925 4842
17 5778 600
18 6509 2417
19 1260 4903
0 3369 3031
1 3557 3224
2 3028 583
3 3258 440
4 6226 6655
5 4895 1094
6 1481 6847
7 4433 1932
8 2107 1649
9 2119 2065
10 4003 6388
11 6720 3622
12 3694 4521
13 1164 7050
14 1965 3613
15 4331 66
16 2970 1796
17 4652 3218
18 1762 4777

19 5736 1399
0 970 2572
1 2062 6599
2 4597 4870
3 1228 6913
4 4159 1037
5 2916 2362
6 395 1226
7 6911 4548
8 4618 2241
9 4120 4280
10 5825 474
11 2154 5558
12 3793 5471
13 5707 1595
14 1403 325
15 6601 5183
16 6369 4569
17 4846 896
18 7092 6184
19 6764 7127
0 6358 1951
1 3117 6960
2 2710 7062
3 1133 3604
4 3694 657
5 1355 110
6 3329 6736
7 2505 3407
8 2462 4806
9 4216 214
10 5348 5619
11 6627 6243

12 2644 5073
13 4212 5088
14 3463 3889
15 5306 478
16 4320 6121
17 3961 1125
18 5699 1195
19 6511 792
0 3934 2778
1 3238 6587
2 1111 6596
3 1457 6226
4 1446 3885
5 3907 4043
6 6839 2873
7 1733 5615
8 5202 4269
9 3024 4722
10 5445 6372
11 370 1828
12 4695 1600
13 680 2074
14 1801 6690
15 2669 1377
16 2463 1681
17 5972 5171
18 5728 4284
19 1696 1459

17 1482 2381
4620
18 6791 6014
6596
19 2738 5918
3786
0 5156 6166
1 1504 4356
2 130 1904
3 6027 3187
4 6718 759
5 6240 2870
6 2343 1311
7 1039 5465
8 6617 2513
9 1588 5222
10 6561 535
11 4765 2054
12 5966 6892

Table B.11: N = 64800, Code Rate = 9/10

0 5611 2563
2900
1 5220 3143
4813
2 2481 834 81
3 6265 4064
4265
4 1055 2914
5638
5 1734 2182
3315
6 3342 5678
2246
7 2185 552
3385
8 2615 236
5334
9 1546 1755
3846
10 4154 5561
3142
11 4382 2957
5400
17 3216 2178
0 4165 884
1 2896 3744
2 874 2801
3 3423 5579
4 3404 3552
5 2876 5515
6 516 1719
7 765 3631
8 5059 1441
9 5629 598
10 5405 473
11 4724 5210
12 155 1832
13 1689 2229
14 449 1164
15 2308 3088
16 1122 669
17 2268 5758
0 5878 2609
1 782 3359
2 1231 4231
3 4225 2052
16 6296 2583
17 1457 903
0 855 4475
1 4097 3970
2 4433 4361
3 5198 541
4 1146 4426
5 3202 2902
6 2724 525
7 1083 4124
8 2326 6003
9 5605 5990
10 4376 1579
11 4407 984
12 1332 6163
13 5359 3975
14 1907 1854
15 3601 5748
16 6056 3266
17 3322 4085
0 1768 3244
1 2149 144
2 1589 4291
15 1263 293
16 5949 4665
17 4548 6380
0 3171 4690
1 5204 2114
2 6384 5565
3 5722 1757
4 2805 6264
5 1202 2616
6 1018 3244
7 4018 5289
8 2257 3067
9 2483 3073
10 1196 5329
11 649 3918
12 3791 4581
13 5028 3803
14 3119 3506
15 4779 431
16 3888 5510
17 4387 4084
0 5836 1692
1 5126 1078
14 3267 649
15 6236 593
16 646 2948
17 4213 1442
0 5779 1596
1 2403 1237
2 2217 1514
3 5609 716
4 5155 3858
5 1517 1312
6 2554 3158
7 5280 2643
8 4990 1353
9 5648 1170
10 1152 4366
11 3561 5368
12 3581 1411
13 5647 4661
14 1542 5401
15 5078 2687
16 316 1755
17 3392 1991

12 1209 5329
3179
13 1421 3528
6063
14 1480 1072
5398
15 3843 1777
4369
16 1334 2145
4163
17 2368 5055
260
0 6118 5405
1 2994 4370
2 3405 1669
3 4640 5550
4 1354 3921
5 117 1713
6 5425 2866
7 6047 683
8 5616 2582
9 2108 1179
10 933 4921
11 5953 2261
12 1430 4699
13 5905 480
14 4289 1846
15 5374 6208
16 1775 3476

4 4286 3517
5 5531 3184
6 1935 4560
7 1174 131
8 3115 956
9 3129 1088
10 5238 4440
11 5722 4280
12 3540 375
13 191 2782
14 906 4432
15 3225 1111

3 5154 1252
4 1855 5939
5 4820 2706
6 1475 3360
7 4266 693
8 4156 2018
9 2103 752
10 3710 3853
11 5123 931
12 6146 3323
13 1939 5002
14 5140 1437

2 5721 6165
3 3540 2499
4 2225 6348
5 1044 1484
6 6323 4042
7 1313 5603
8 1303 3496
9 3516 3639
10 5161 2293
11 4682 3845
12 3045 643
13 2818 2616

Table B.12: N = 16200, Code Rate = 1/5

6295 9626 304 7695 4839 4936 1660 144 11203 5567 6347 12557
10691 4988 3859 3734 3071 3494 7687 10313 5964 8069 8296 11090
10774 3613 5208 11177 7676 3549 8746 6583 7239 12265 2674 4292
11869 3708 5981 8718 4908 10650 6805 3334 2627 10461 9285 11120
7844 3079 10773
3385 10854 5747
1360 12010 12202
6189 4241 2343
9840 12726 4977

Table B.13: N = 16200, Code Rate = 1/3

416 8909 4156 3216 3112 2560 2912 6405 8593 4969 6723 6912

8978 3011 4339 9312 6396 2957 7288 5485 6031 10218 2226 3575
3383 10059 1114 10008 10147 9384 4290 434 5139 3536 1965 2291
2797 3693 7615 7077 743 1941 8716 6215 3840 5140 4582 5420
6110 8551 1515 7404 4879 4946 5383 1831 3441 9569 10472 4306
1505 5682 7778
7172 6830 6623
7281 3941 3505
10270 8669 914
3622 7563 9388
9930 5058 4554
4844 9609 2707
6883 3237 1714
4768 3878 10017
10127 3334 8267

Table B.14: N = 16200, Code Rate = 2/5

5650 4143 8750 583 6720 8071 635 1767 1344 6922 738 6658
5696 1685 3207 415 7019 5023 5608 2605 857 6915 1770 8016
3992 771 2190 7258 8970 7792 1802 1866 6137 8841 886 1931
4108 3781 7577 6810 9322 8226 5396 5867 4428 8827 7766 2254
4247 888 4367 8821 9660 324 5864 4774 227 7889 6405 8963
9693 500 2520 2227 1811 9330 1928 5140 4030 4824 806 3134
1652 8171 1435
3366 6543 3745
9286 8509 4645
7397 5790 8972
6597 4422 1799
9276 4041 3847
8683 7378 4946
5348 1993 9186
6724 9015 5646
4502 4439 8474
5107 7342 9442
1387 8910 2660

Table B.15: N = 16200, Code Rate = 4/9

20 712 2386 6354 4061 1062 5045 5158
21 2543 5748 4822 2348 3089 6328 5876
22 926 5701 269 3693 2438 3190 3507
23 2802 4520 3577 5324 1091 4667 4449
24 5140 2003 1263 4742 6497 1185 6202
11 8935 4996
12 3028 764
13 5988 1057
14 7411 3450

0 4046 6934
1 2855 66
2 6694 212
3 3439 1158
4 3850 4422
5 5924 290
6 1467 4049
7 7820 2242
8 4606 3080
9 4633 7877
10 3884 6868

Table B.16: N = 16200, Code Rate = 3/5

2765 5713 6426 3596 1374 4811 2182 544 3394
2840 4310 771
4951 211 2208 723 1246 2928 398 5739 265
5601 5993 2615 210 4730 5777 3096 4282 6238
4939 1119 6463 5298 6320 4016
4167 2063 4757 3157 5664 3956 6045 563 4284
2441 3412 6334
4201 2428 4474 59 1721 736 2997 428 3807
1513 4732 6195 2670 3081 5139 3736 1999 5889
4362 3806 4534 5409 6384 5809
5516 1622 2906 3285 1257 5797 3816 817 875
2311 3543 1205
4244 2184 5415 1705 5642 4886 2333 287 1848
1121 3595 6022 2142 2830 4069 5654 1295 2951
3919 1356 884 1786 396 4738
0 2161 2653
1 1380 1461
2 2502 3707
3 3971 1057
4 5985 6062
5 1733 6028
6 3786 1936
7 4292 956
8 5692 3417
9 266 4878
10 4913 3247
11 4763 3937
12 3590 2903
13 2566 4215
14 5208 4707
15 3940 3388
16 5109 4556
17 4908 4177

Table B.17: N = 16200, Code Rate = 2/3

0 2084 1613 1548 1286 1460 3196 4297 2481
3369 3451 4620 2622
1 2583 1180
2 1542 509

1 122 1516 3448 2880 1407 1847 3799 3529 373
971 4358 3108
2 259 3399 929 2650 864 3996 3833 107 5287
164 3125 2350 3 342 3529
4 4198 2147
5 1880 4836
6 3864 4910 7 243 1542
8 3011 1436
9 2167 2512
10 4606 1003
11 2835 705
12 3426 2365
13 3848 2474
14 1360 1743
0 163 2536
3 4418 1005
4 5212 5117
5 2155 2922
6 347 2696
7 226 4296
8 1560 487
9 3926 1640
10 149 2928
11 2364 563
12 635 688
13 231 1684
14 1129 3894

Table B.18: N = 16200, Code Rate = 11/15

3 3198 478 4207 1481 1009 2616 1924
3437 554 683 1801
4 2681 2135
5 3107 4027
6 2637 3373
7 3830 3449
8 4129 2060
9 4184 2742
10 3946 1070
11 2239 984
0 1458 3031
1 3003 1328
2 1137 1716
3 132 3725
4 1817 638
5 1774 3447
6 3632 1257
7 542 3694
8 1015 1945
9 1948 412
10 995 2238
11 4141 1907
0 2480 3079
1 3021 1088
2 713 1379
3 997 3903
4 2323 3361
5 1110 986
6 2532 142
7 1690 2405
8 1298 1881
9 615 174
10 1648 3112
11 1415 2808

Table B.19: N = 16200, Code Rate = 7/9

5 896 1565
6 2493 184
7 212 3210
8 727 1339
9 3428 612
0 2663 1947
1 230 2695
2 2025 2794
3 3039 283
4 862 2889
5 376 2110
6 2034 2286
7 951 2068
8 3108 3542
9 307 1421
0 2272 1197
1 1800 3280
2 331 2308
3 465 2552
4 1038 2479
5 1383 343
6 94 236
7 2619 121
8 1497 277
9 2116 1855
0 722 1584
1 2767 1881
2 2701 1610
3 3283 1732
4 168 1099
5 3074 243
6 3460 945
7 2049 1746
8 566 1427
9 3545 1168

Table B.20: N = 16200, Code Rate = 37/49

3 2409 499 1481 908 559 716 1270 333
2508 2264 1702 2805
4 2447 1926
5 414 1224
6 2114 842
7 212 573
0 2383 2112
1 2286 2348
2 545 819
3 1264 143
4 1701 2258
5 964 166
6 114 2413
7 2243 81
0 1245 1581
1 775 169
2 1696 1104
3 1914 2831
4 532 1450
5 91 974
6 497 2228
7 2326 1579
0 2482 256
1 1117 1261
2 1257 1658
3 1478 1225
4 2511 980
5 2320 2675
6 435 1278
7 228 503
0 1885 2369
1 57 483
2 838 1050
3 1231 1990
4 1738 68
5 2392 951
6 163 645
7 2644 1704

Table B.21: N = 16200, Code Rate = 8/9

0 1558 712 805
1 1450 873 1337
2 1741 1129 1184
3 294 806 1566
4 482 605 923
0 926 1578
1 777 1374
2 608 151
3 1195 210
4 1484 692
0 427 488
1 828 1124
2 874 1366
3 1500 835
4 1496 502
0 1006 1701
1 1155 97
2 657 1403
3 1453 624
4 429 1495
0 809 385
1 367 151
2 1323 202
3 960 318
4 1451 1039
0 1098 1722
1 1015 1428
2 1261 1564
3 544 1190
4 1472 1246
0 508 630
1 421 1704
2 284 898
3 392 577
4 1155 556
0 631 1000
1 732 1368
2 1328 329
3 1515 506
4 1104 1172

Appendix C

Table (D.1). RALUT approximation of $\tanh^{-1} x$.

Input range which uniformly distributed across the output	Input range	Output
$y_1 \leq y = \tanh x \leq y_2$	$\tanh^{-1} y_1 \leq x < \tanh^{-1} y_2$	$y = \frac{y_1 + y_2}{2}$
$y = \tanh x = 0$	$x = 0$	$y = 0$
$0 \leq y = \tanh x \leq 0.05$	$\tanh^{-1} 0 \leq x < \tanh^{-1} 0.05$	$y = 0.025$
$0.05 \leq y = \tanh x \leq 0.1$	$\tanh^{-1} 0.05 \leq x < \tanh^{-1} 0.1$	$y = 0.075$
$0.01 \leq y = \tanh x \leq 0.15$	$\tanh^{-1} 0.1 \leq x < \tanh^{-1} 0.15$	$y = 0.125$
$0.15 \leq y = \tanh x \leq 0.2$	$\tanh^{-1} 0.15 \leq x < \tanh^{-1} 0.2$	$y = 0.175$
$0.2 \leq y = \tanh x \leq 0.25$	$\tanh^{-1} 0.2 \leq x < \tanh^{-1} 0.25$	$y = 0.225$
$0.25 \leq y = \tanh x \leq 0.3$	$\tanh^{-1} 0.25 \leq x < \tanh^{-1} 0.3$	$y = 0.275$
$0.3 \leq y = \tanh x \leq 0.35$	$\tanh^{-1} 0.3 \leq x < \tanh^{-1} 0.35$	$y = 0.325$
$0.35 \leq y = \tanh x \leq 0.4$	$\tanh^{-1} 0.35 \leq x < \tanh^{-1} 0.4$	$y = 0.375$
$0.4 \leq y = \tanh x \leq 0.45$	$\tanh^{-1} 0.4 \leq x < \tanh^{-1} 0.45$	$y = 0.425$
$0.45 \leq y = \tanh x \leq 0.5$	$\tanh^{-1} 0.45 \leq x < \tanh^{-1} 0.5$	$y = 0.475$
$0.5 \leq y = \tanh x \leq 0.55$	$\tanh^{-1} 0.5 \leq x < \tanh^{-1} 0.55$	$y = 0.525$
$0.55 \leq y = \tanh x \leq 0.6$	$\tanh^{-1} 0.55 \leq x < \tanh^{-1} 0.6$	$y = 0.575$
$0.6 \leq y = \tanh x \leq 0.65$	$\tanh^{-1} 0.6 \leq x < \tanh^{-1} 0.65$	$y = 0.625$
$0.65 \leq y = \tanh x \leq 0.7$	$\tanh^{-1} 0.65 \leq x < \tanh^{-1} 0.7$	$y = 0.675$
$0.7 \leq y = \tanh x \leq 0.75$	$\tanh^{-1} 0.7 \leq x < \tanh^{-1} 0.75$	$y = 0.725$
$0.75 \leq y = \tanh x \leq 0.8$	$\tanh^{-1} 0.75 \leq x < \tanh^{-1} 0.8$	$y = 0.775$
$0.8 \leq y = \tanh x \leq 0.85$	$\tanh^{-1} 0.8 \leq x < \tanh^{-1} 0.85$	$y = 0.825$
$0.85 \leq y = \tanh x \leq 0.9$	$\tanh^{-1} 0.85 \leq x < \tanh^{-1} 0.9$	$y = 0.875$
$0.9 \leq y = \tanh x \leq 0.95$	$\tanh^{-1} 0.9 \leq x < \tanh^{-1} 0.95$	$y = 0.925$
$0.95 \leq y = \tanh x \leq 0.99$	$\tanh^{-1} 0.95 \leq x < \tanh^{-1} 0.99$	$y = 0.975$
$0.99 \leq y = \tanh x$	$x \leq \tanh^{-1} 0.99$	$y = 1$