VISVESVARAYA TECHNOLOGICAL UNIVERSITY

BELAGAVI - 590018

2020-2021

A

Project Report

on

"VLSI Implementation of Turbo codes for LTE Systems"

Submitted in the partial fulfilment of the requirement for the VIII Semester Project – 17ECP85 for the award of degree of

Bachelor of Engineering

in

Electronics and Communication Engineering

by

SINDHU G	1GV17EC039
SPANDANA K. N.	1GV17EC042
V. PRACHI BOHRA	1 GV17EC049
WAJIHA SULTANA	1GV17EC054

Carried at

Dr. T. THIMMAIAH INSTITUTE OF TECHNOLOGY

Under the Guidance of

Prof. VIJAYA BHARATHI M

HOD, Assoc. Prof. Dept. of E & C,

Dr. T. THIMMAIAH INSTITUTE OF TECHNOLOGY

Oorguam Post, K.G.F- 563120

(Approved by AICTE, New Delhi, Affiliated to VTU-Belagavi,

Approved by Govt. Of Karnataka and ISO 21001-2018 Certified)

Dr. T. Thimmaiah Institute of Technology

Oorguam Post, K.G.F-563120

(Approved by AICTE, New Delhi, Affiliated to VTU-Belagavi, Approved by Govt. Of Karnataka and ISO 21001-2018 Certified)

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING.

CERTIFICATE

Certified that the **Project Work** entitled ***"VLSI Implementation of Turbo Codes for LTE Systems"*** is a bonafied work carried out **by Sindhu G.- 1GV17EC039, Spandana K.N.- 1GV17EC042, V. Prachi Bohra- 1GV17EC049 and Wajiha Sultana – 1GV17EC054** in the partial fulfilment for the award of degree of Bachelor of Engineering in **Electronics and Communication Engineering** of the **Visvesvaraya Technological University**, Belagavi during the year 2020-2021. It is certified that all corrections/suggestions indicated for the assessment have been incorporated in the report deposited in the departmental library. The Project report has been approved as it satisfies the academic requirement in respect of **Project Work - 17ECP85** prescribed for the Bachelor of Engineering Degree.

Vijaya Bharathi 2/8/2021 [illegible] 28/05/2021

Signature of HOD and guide	**Signature of Principal**
Prof. Vijaya Bharathi M.	**Dr. Syed Ariff**

Head of the Department
Dept. of Electronics and Communication Engg.
Dr. T.Thimmaiah Institute of Technology
Oorgaum, K.G.F.- 563 120.

Name of Examiner	Signature with Date
1. Dr. Bhuvanendhiran. T	1. [illegible]
2. Rajesh Kumar Kaushal	2. [illegible]

ACKNOWLEDGEMENT

It is with deep feeling of gratitude we would like to express our sincere thanks to our institution **Dr. T. THIMMAIAH INSTITUTE OF TECHNOLOGY, K.G.F** for providing excellent infrastructure for the completion of the Project work.

We wish to express a wholehearted thanks to our **Principal Dr. Syed Ariff** for providing good infrastructure for undertaking this project work in college.

We would like to extend hearty thanks to our **Dean (Administration) Prof. Ruckmani Divakaran,** for being constant support of encouragement to carry out the Project work.

We would like to extend hearty thanks to our **HOD and guide Prof. Vijaya Bharathi M,** for her timely support and guidance in the completion of this Project work.

We would like to thank project coordinator **Mrs. Jenitha A, Associate professor,** for her timely support and co-ordination in the completion of this Project work.

We would like to thank all teaching and non-teaching staff who were directly and indirectly supported for carrying out this project work.

We extend our hearty thanks to our parents, friends for all the moral support provided during the preparation for the project work.

SINDHU G	**1GV17EC039**
SPANDANA K N	**1GV17EC042**
V PRACHI BOHRA	**1GV17EC049**
WAJIHA SULTANA	**1GV17EC054**

SYNOPSIS

Communication is act of transmission of information. Everyone in the world experiences the need to receive information almost continuously. For communication to be successful, it is essential that sender and receiver understands a common language.

When signal is transmitted there are 3 sources of transmission errors, they are: Signal bit errors, burst errors and erasure. Errors in signal may lead to miscommunication between systems. So, error correction is required to retrieve the original message. In order to detect and correct the errors, turbo codes are used.

Turbo encoder and decoder is designed using the Verilog HDL Language. Turbo decoding is time consuming process. So, the SOVA Algorithm gives high throughput and less complexity output.

Xilinx Vivado 2020.2. tool is used which achieves simulation and synthesis of proposed turbo encoder and decoder.

CONTENTS

LIST OF FIGURES AND TABLES

Chapter 1

INTRODUCTION

Communication is a process by which information is exchanged between individuals through a common channel. When data is transferred from source to destination system, errors can be present in signal. So, error correction is required to retrieve the original message.

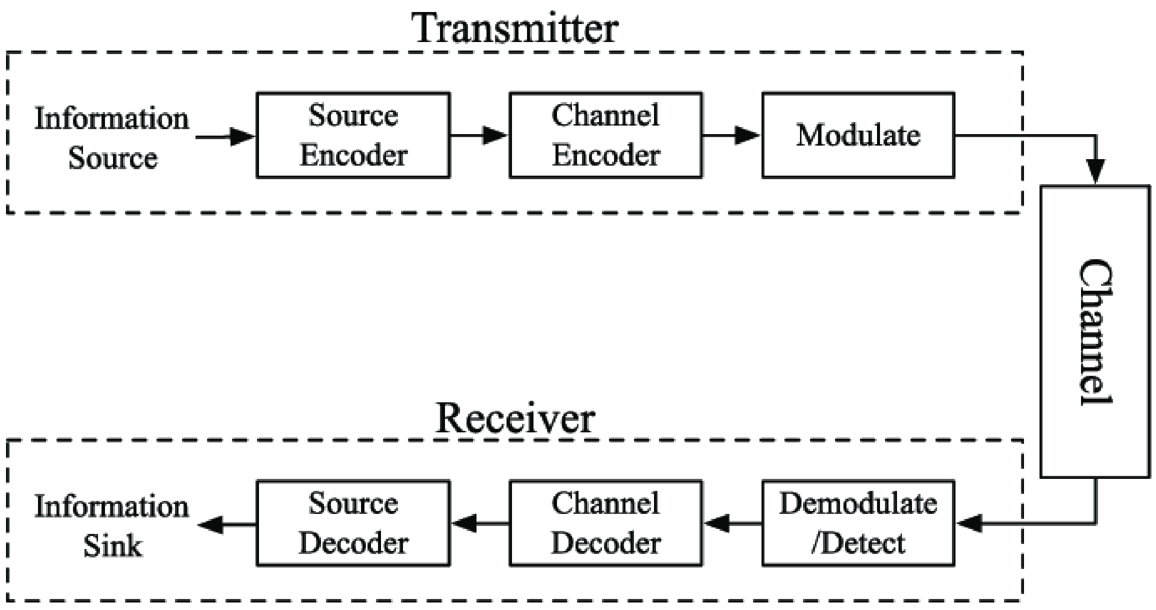


Figure 1.1: Block Diagram of Communication System

Communication is the process of establishing connection or link between two points for information exchange. Communication is simply the basic process of exchanging information.

The electronics equipment’s which are used for communication purpose, are called communication equipment’s. Different communication equipment’s when assembled together form a communication system.

Typical example of communication system are line telephony and line telegraphy, radio telephony and radio telegraphy, radio broadcasting, point-to-point communication and mobile communication, computer communication, radar communication, television broadcasting, radio telemetry, radio aids to navigation, radio aids to aircraft landing etc.

The Communication Process

In the most fundamental sense, communication involves the transmission of information from one point to another through a succession of process as listed below :

1. The generation of a thought pattern or image in the mind of an originator.
2. The description of that image, with a certain measure of precision, by a set of oral visual symbols.
3. The encoding of these symbols in a form that is suitable for transmission over a physical medium of interest.
4. The transmission of the encoded symbols to the desired destination.
5. The decoding and reproduction of the original symbols.
6. The recreation of the original thought pattern or image, with a definable degradation in quality, in the mind of recipient.

1. Transmitter:

The Transmitter has some information that it wants to transmit to receiver. The transmitter can also add error correcting codes using channel encoder block which essentially introduces redundancy by adding extra bits.

The function of the transmitter is to process the electrical signal from different aspects.

For example, in radio broadcasting the electrical signal obtained from sound signal, is processed to restrict its range of audio frequencies (up to 5 kHz in amplitude modulation radio broadcast) and is often amplified.

Modulation is the main function of the transmitter. In modulation, the message signal is superimposed upon the high-frequency carrier signal .Inside the transmitter, signal processing's such as restriction of range of audio frequencies, amplification and modulation of signal are achieved . All these processing's of the message signal are done just to ease the transmission of the signal through the channel.

Source Encoder

Any discrete message can be represented with a string of binary numbers. So, transmitter uses source encoder to encode information message as a binary sequence.

Channel Encoder

The transmitter can also add error correcting codes using channel encoder block which essentially introduces redundancy by adding extra bits.

Modulator

Modulator is **a device that produces modulation** (control of the parameters of a high-frequency electromagnetic information carrier in accordance with electrical signals of the transmitted message). A modulator is mainly a component of transmitting apparatus for electrical communication and radio broadcasting.

2. Channel:

The channel may introduce noise in system. A channel can be a wire or air. The term channel means the medium through which the message travels from the transmitter to the receiver. In other words, we can say that the function of the channel is to provide a physical connection between the transmitter and the receiver.
There are two types of channels, namely point-to-point channels and broadcast channels.
Example of point-to-point channels are wire lines, microwave links and optical fibres. Wire-lines operate by guided electromagnetic waves and they are used for local telephone transmission.
In case of microwave links, the transmitted signal is radiated as an electromagnetic wave in free space. Microwave links are used in long distance telephone transmission.
An optical fibre is a low-loss, well-controlled, guided optical medium. Optical fibres are used in optical communications
Although these three channels operate differently, they all provide a physical medium for the transmission of signals from one point to another point. Therefore, for these channels, the term point-to-point is used.
On the other hand, the broadcast channel provides a capability where several receiving stations can be reached simultaneously from a single transmitter.
An example of a broadcast channel is a satellite in geostationary orbit, which covers about one third of the earth’s surface. During the process of transmission and reception the signal gets distorted due to noise introduced in the system.
Noise is an unwanted signal which tend to interfere with the required signal. Noise signal is always random in character. Noise may interfere with signal at any point in a communication system. However, the noise has its greatest effect on the signal in the channel.

3. Receiver:

The main function of the receiver is to reproduce the message signal in electrical form from the distorted received signal. This reproduction of the original signal is accomplished by a process known as the demodulation or detection. Demodulation is the reverse process of modulation carried out in transmitter.

The receiver can use the added redundancy to mitigate errors that may be introduced by channel.

Demodulator/Detector

an electronic receiver that detects and demodulates and amplifies transmitted signals.

Channel Decoder

The estimated channel symbols are decoded using channel decoder, where errors introduced by transmitter, the channel and receiver may be corrected.

Source Decoder

The output of channel decoder goes to source decoder, where receiver estimates what information the transmitter has sent.

If estimation is correct, communication session has been successful. If estimation is wrong, then communication is not faithful and leads to miscommunication.

Turbo codes are error correcting codes that are widely used in communication systems Turbo codes exhibits high error correcting capability as compared with other error correcting codes. Turbo coder architecture comprises of turbo encoder and turbo decoder.

1.1 History of the Project

In 1948, Shannon proved that over a noisy communication channel, an error-free information can be transmitted, if the capacity of the data transmission (rate) is less than or equal to the channel capacity. Subsequently, a lot of efforts and research in the field of communication to find new methods are in progress, which exhibits its transmission capacity close to the channel capacity.

Exhibiting performance approaching the Shannon limit, Turbo Codes (TC) have the TC block set features efficient encoder and decoder designs seen rapid adoption in the design of digital communication systems. Desirable and Designable introduces the basics of turbo codes in their different flavours (more specifically, parallel concatenated convolutional turbo codes and block turbo codes). Through the application of systemic design methodology that considers data transfer and storage a stop priority candidate for optimization, the authors show how turbo codes can be implemented and the attractive performance results that can be achieved in throughput, latency .

To achieve near-capacity, one of the fundamental methods that makes it possible is Channel coding. By adding a structured redundant information to the transmitted information during encoding and exploiting it during decoding at the receiver, a variety of forward error correction and detection techniques can be achieved.

The last ten years have seen the appearance of a new type of correction code -the turbo code. This represents a significant development in the field of error-correcting codes. Turbo codes were introduced in 1993. These codes have very good performance capability and perform very close to channel capacity. The criteria that makes these codes powerful are the concatenated scheme and iterative decoding. At very low signal to noise ratio values, the turbo codes have good performance and hence can be used for applications where high reliable transmission is required.

The first class of turbo codes was parallel concatenated convolutional code (PCCC), many other classes of turbo codes have been discovered, including serial versions serial concatenated convolutional codes and repeat accumulate codes. Turbo equalisation also flowed from the concept of turbo coding.

In addition to turbo codes, Berrou also invented recursive systematic convolutional (RSC) Codes which are used in example implementation of turbo codes described in patent.
The principle of decoding is to be found in an iterative exchange of information between elementary decoders, called extrinsic information, and it is this principle from which the term turbo originates. The turbo concept is now applied to block codes as well as other parts of a digital transmission system, such as detection, demodulation.
The decoding algorithm for Turbo codes is Maximum-a-posteriori (MAP), which is a computationally complex method as it involves non-linear functions, a wide calculation of probabilities and additions and multiplications of these values are required.

Therefore, it is not practical to use MAP algorithm for implementation. Approximations in Logarithmic domain such as Log-MAP and Max-Log-MAP
are derived for MAP algorithms and are used as the practical decoding algorithms for implementation.

Applications that integrate turbo codes into their standards are mobile communications, wireless networks and local radio loops. Future applications could include cable transmission, short-distance communication or data storage includes cable transmission, short-distance communication or data storage. The decoding algorithm for Turbo codes is Maximum-a-posteriori (MAP), which is a computationally complex method as it involves non-linear functions, a wide calculation of probabilities and additions and multiplications of these values are required. Therefore, it is not practical to use MAP algorithm for implementation. Approximations in Logarithmic domain such as Log-MAP and Max-Log-MAP are derived for MAP algorithms and are used as the practical decoding algorithms for implementation. But the SOVA algorithm is preffered.

1.2 Requirements

1) Computer (Operating System: Windows 10)
2) Xilinx Vivado /coding software
    - Xilinx Vivado 2020.2 Version
3) Disk Space: 30 to 40 GB of available disk space with additional of 6 GB during installation.
4) RAM: 2 GB or higher .

Chapter 2

LITERATURE SURVEY

A literature survey or a literature review in a project is a type of review articles. It is a scholarly paper, which includes the current knowledge including substantive findings, as well as theoretical and methodological contributions to a particular topic. Literatures reviews are secondary sources, and do not report new or original experimental work. It is a basis for research in nearly every academic field. Concentrate on the own field of expertise.

2.1 Details of Literature

V. Kavinilavu1, S. Salivahanan, V. S. Kanchana Bhaaskaran2, Samiappa Sakthikumaran, B. Brindha and C. Vinoth "Implementation of Convolutional Encoder and Viterbi Decoder using Verilog HDL", IEEE 3rd International Conference on Electronics Computer Technology,2011

A Viterbi decoder uses the Viterbi algorithm for decoding a bit stream that has been encoded using Forward error correction based on a Convolutional code. The maximum likelihood detection of a digital stream is possible by Viterbi algorithm. Here they present a Convolutional encoder and Viterbi decoder with a constraint length of 7 and code rate of 1/2. Finally, the transmitted sequence is decoded by the Viterbi decoder and the estimated original sequence is produced. This is realized using Verilog HDL. It is simulated and synthesized using Modelsim PE 10.0e and Xilinx 12.4i.

Tepoju Vivek Vardhan, Bandi Neeraja, Boya Pradeep Kumar, Chandra Sekhar Paidimarry "Implementation of Turbo Codes Using Verilog HDL and Estimation of Its Error Correction Capability", IEEE Asia Pacific Conference on Postgraduate Research in Microelectronics and Electronics (Prime Asia),2015

This paper presents the implementation of Turbo codec for designing the Turbo encoder and decoder. The Decoder is developed based on Viterbi algorithm that incorporates hard-input and hard-output values. The errors are purposefully introduced in the encoded data to estimate error correction capability. These encoded bits are transmitted to the Turbo

Decoder through channel. At the Turbo Decoder, the received data may contain errors, which are decoded using Viterbi algorithm to obtain the original information. In such case, developed Turbo decoder is able to correct two-bit error in the encoded data. The Encoder and Decoder of Turbo codes are implemented using Verilog-HDL. The code is ported in FPGA for real time verification.

Cagri Tanriover, Bahram Honary, Jun Xu, and Shu Lin, "Improving turbo code error performance by multifold coding", IEEE Communication letters, VOL. 6, NO. 5, MAY 2002

This letter presents a simple turbo coding technique to improve the error performance of a conventional rate-1/3 turbo code by shaping its weight spectrum closer to the binomial weight distribution of a random code. This technique can be applied to both symmetric and asymmetric rate 1/3 turbo codes to achieve additional coding gain.

Yaqin Shi, Ming Zhan, Jie Zeng , "FPGA Implementation and Power Estimation of a Memory-Reduced LTE-Advanced Turbo Decoder", IEEE International Conference on Internet of Things (I Things) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData), 2018.

In this paper, they present the design and implementation on field programmable gate array (FPGA) of a memory-reduced Turbo decoder in the LTE-Advanced standard. By inserting a reverse recalculation module in the conventional Turbo decoding architecture, the state metric cache (SMC) capacity is reduced by 50%.

The power estimation demonstrates that, compared with the conventional Turbo decoder, which adopts the

Max-Log-MAP algorithm, the overall power dissipation of the proposed decoding architecture is decreased by 24%, 29.7% and 31% at the operating frequency of 50MHz,75MHz and 100MHz, respectively.

Additionally, in terms of BER performance, the reverse calculation decoding technique has only a slight degradation in comparison with the optimal Log-MAP algorithm.

Arash Ardakani, Mahdi Shabany, "An Efficient Max-Log MAP Algorithm For VLSI Implementation of Turbo Decoders", IEEE International Symposium on Circuits and Systems (ISCAS), 2015

In this paper, a novel form of computation of the max-log MAP core is proposed to highly improve the hardware implementation parameters of a turbo decoder for the LTE and LTE-advanced standards. The proposed method can be applied to any max-log MAP architecture. In order to illustrate the impact of using the proposed method, a max-log MAP, as a case study, is implemented based on both the proposed form of computation and conventional one for a fair comparison. They came a conclusion that the max-log MAP based on the proposed method not only reduces the area and power consumption of each sub-blocks but also increases the throughput of the core without any performance (BER) degradation compared to the conventional method.

Claude Berrou, Ramesh Pyndiah, Patrick Adde, Catherine Douillard and Raphaël Bidan, "Application of turbo codes", IEEE 2005

Turbo Codes are now a mature technology that has been rapidly adopted for application in many commercial transmissions systems. This paper provides an overview of the basic concepts employed in Convolutional and Block Turbo Codes, and review the major evolutions in the field with an emphasis on practical issues such as implementation complexity and high-rate circuit architectures. We address the use of these technologies in existing standards and also discuss future potential applications for this error-control coding technology.

Aswathy Narayanan; Senthil Murugan; Ramesh Bhakthavatchalu, "Low Latency Max Log MAP based Turbo Decoder", International Conference on Communication and Signal Processing (ICCSP),2019

The framework of convolutional coding persisting today incorporates decoder designs whose performance varies with the underlying algorithm's efficiency. As process and technology advances from the basic fixed-point multiplier to the present Booth multiplier, the decoding performance varies.

This paper focuses on bringing out the performance variations of a Max log MAP algorithm-based turbo decoder on implemented with fixed point, Vedic and Booth multipliers.

Upon implementation, fixed point multiplier consumes a large amount of power, also since Max Log MAP algorithm must be power efficient, so it is not implemented. They reached a conclusion such that for lesser delay, Vedic multiplier-based implementation can be preferred, which consumes more area. Whereas, in cases of highly scaled designs, where accommodation of high number of resources is not affordable, Booth algorithm-based implementation is preferred.

Wang Huahua, Liu Wenwen, "Analysis of turbo decoding algorithm in LTE systems", Project of the National Science and Technology Major Special: "LTE wireless comprehensive test instrument development 2012 IEEE

Turbo code, also named Parallel Concatenated Convolutional Code (PCCC), is used for LTE (long term evolution) system for its good error correction ability and anti-interference ability. This paper presents the algorithm of MAP (Maximum A Posteriori probability) and improved MAP algorithm which based on SISO (soft input soft output).The ability of MAP algorithm and improved MAP algorithm are analyzed over the length of data for code, iterative times and complexity of decoding. Finally, the simulation results of each algorithm will be shown.

Moeed Israr and Tad Kwasniewski, "Digital IC design of turbo codes", 9th International Database Engineering & Application Symposium (IDEAS'05) 2005 IEEE

This paper presents a Digital ASIC implementation of Turbo Encoder and simulation of the Encoder and the Decoder. The simulations determine the impact of decoder iterations, encoder transfer function and block size on latency, throughput and bit error rate. The paper proposes a design of Turbo Encoding hardware with dual-port on-chip memory targeting 0.18µm CMOS technology that reduces the memory requirements to half. Architectural and block level consideration are made to reduce power and area requirements while achieving a latency of "packet size + 5" clock cycles. Estimated power and area are 54.19mW and 12.0 mm2 respectively.

Bashar Tahir , Stefan Schwarz , and Markus Rupp " BER Comparison Between Convolutional, Turbo, LDPC, and Polar Codes ", IEEE 24th International Conference on Telecommunications (ICT), 2017

Channel coding is a fundamental building block in any communications system. High performance codes, with low complexity encoding and decoding are a must-have for future wireless systems, with requirements ranging from the operation in highly reliable scenarios, utilizing short information messages and low code rates, to high throughput scenarios, working with long messages, and high code rates. In this paper they investigate the performance of Convolutional, Turbo, Low-Density Parity-Check (LDPC), and Polar codes, in terms of the Bit-Error-Ratio (BER) for different information block lengths and code rates, spanning the multiple scenarios of reliability and high throughput.

They further investigate their convergence behaviour with respect to the number of iterations (turbo and LDPC), and list size (polar), as well as how their performance is impacted by the approximate decoding algorithms. Finally, they concluded that, except convolutional code, the other schemes perform close to each other.

Aldrin Claytus Vaz, C Gurudas Nayak and Dayananda Nayak "Performance Comparison between Turbo and Polar Codes", 3rd International Conference on Electronics, communication and Aerospace Technology (ICECA), 2019

Channel coding is a vital unit of any communications system. Future wireless systems are necessary to be equipped with high performance codes having low complexity encoder and decoder design. The requirements of these systems range from providing high throughput, operating in highly reliable scenarios, to working with short and long information messages, low and high code rates. To obtain maximum coding gain in wireless communication systems, Turbo codes and Polar Codes are used as error correcting codes. Satellite communication and 4G/5G services widely use these codes. Two recursive systematic convolution encoders separated by an interleaver are used to build a turbo encoder. Polar Codes are constructed on the concept of channel polarization. This work focusses on the Bit Error Rate evaluation and analysis of the codes for varying block lengths and code rates. They conclude that, both the codes have good efficiency and achieve results close to the capacity. Turbo codes has high error correcting capability compared to polar codes.

2.2 Objective of the Project

The main objective of this project is to Implement Turbo encoder and decoder.
To correct the errors and retrieve the original message and to reduce the number of iterations required to decode the information bits being transmitted.

Chapter 3

METHODOLOGY

Methodology is the systematic, theoretical analysis of the methods applied to field of study. It comprises the theoretical analysis of the body of methods and principles associated with a branch of knowledge. Typically , it encompasses concepts such as theoretical model, phases and qualitative or quantitative techniques.

3.1 Introduction To Methodology

Methodology is “a contextual framework for research, a coherent and logical scheme based on views, beliefs, and values, that guides the choices researchers make.”
It comprises the theoretical analysis of the body of methods and principles associated with a branch of knowledge such that the methodologies employed from differing disciplines vary depending on their historical development.

Methodology may be visualized as a spectrum from a predominantly quantitative approach towards a predominantly qualitative approach. Although a methodology may conventionally sit specifically within one of these approaches, researchers may blend approaches in answering their research objectives and so have methodologies that are multimethod and or interdisciplinary.

Overall, a methodology does not set out to provide solutions- it is therefore, not the same as a method.
A methodology offers a theoretical perspective for understanding which method, set of methods, or best practices can be applied to research questions at hand.

Some of the definitions of methodology includes:

1. “ The analysis of principles of methods, rules, and postulates employed by a discipline.”
2. “ The study or description of methods.”

3. “ The systematic study of methods that are, can be, or have been applied within a discipline.”

Methodology has several related concepts : Paradigm, algorithm, and method.
The methodology is the general research strategy that outlines the way in which research is to be undertaken and among other things, identifies the methods to be used in it.

These methods, described in the methodology, define the means or modes of data collection or, sometimes, how a specific result is to be calculated.
Methodology does not define specific methods, even though much attention is given to nature and kinds of processes to be followed in a particular procedure or to attain an objective.

The methodology of the project is as follows , Error correction is required to retrieve the original message, which is sent in the communication systems. Turbo coder is designed. Both Turbo Encoder and Turbo Decoder are to be implemented .Therefore, we are trying to implement a decoder which successfully corrects the error and retrieves the original message and also reduces the number of iterations. The decoder is developed based on MAP Algorithm.

3.2 Basic Block Diagram of Turbo Coder

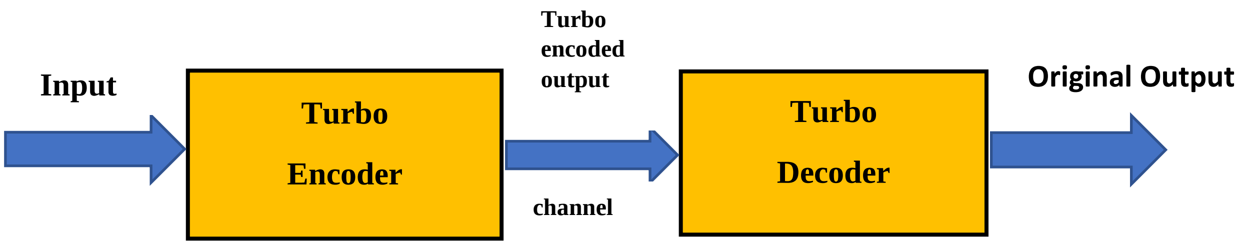


Figure 3.1: Basic Block Diagram of Turbo Coder

The Basic Block Diagram of Turbo Coder consists of Turbo Encoder and Turbo Decoder. A channel for transmitting data from encoder to decoder. Turbo encoder produces an encoded output which is input to turbo decoder, that produces a decoded output by removing the errors and reducing the number of iterations by using MAP Algorithm and get back the original output.

The Encoder produces a codeword with randomlike properties. Two identical Recursive convolutional encoder(RSC) and a psuedorandom interleaver constitutes the turbo encoder. The Decoder makes use of soft-output values and iterative decoding. It consists of two modules of SISO decoders together with two psuedorandom interleaver and a psuedorandom deinterleaver.

3.3 Turbo Encoder

Turbo encoder produces an encoded output which is an input to turbo decoder. A turbo code is formed from the parallel concatenation of two codes separated by an interleaver. The two encoders normally used are identical . Encoders are recursive systematic convolutional codes(RSC). The interleaver reads the bits ina psuedo-random order. The fundamental turbo code encoder is built using two recursive systematic convolutional (RSC) codes with parallel concatenation.

LTE employs a 1/3 rate parallel concatenated turbo code. Each RSC works on two different data. Original data is provided to first encoder, while the second encoder receives the interleaved version of input data. A specified algorithm is used to scramble the data bits and the method is called interleaving.

- **Recursive Convolutional Encoders(RSC)**

Each RSC works on two different data. Original data is provided to the first encoder, while the second encoder receives the interleaved version of the input data. A specified algorithm is used to scramble the data bits and the method is called Interleaving. An appreciable impact on the performance of a decoder is seen with the interleaving algorithm when used. The RSC1 and RSC2 encoder outputs along with systematic input comprises the output of turbo encoder, that is, a 24-bit output is generated. This will be transmitted through the channel to the Turbo decoder.

- **Interleaver**

Here, pseudo-random interleaver is used, due to which the interleaved version of the code tends to be long and scrambled, that gives good performance of random codes.

Interleavers scrambles the data in a pseudorandom order to lessen the resemblance between adjacent bits at the input of the encoder. The interleaver is used on both the encoder part and the decoder part. It produces a long block of data on the encoder side, while it compares two SISO decoders output in the decoder portion and helps to fix the error.

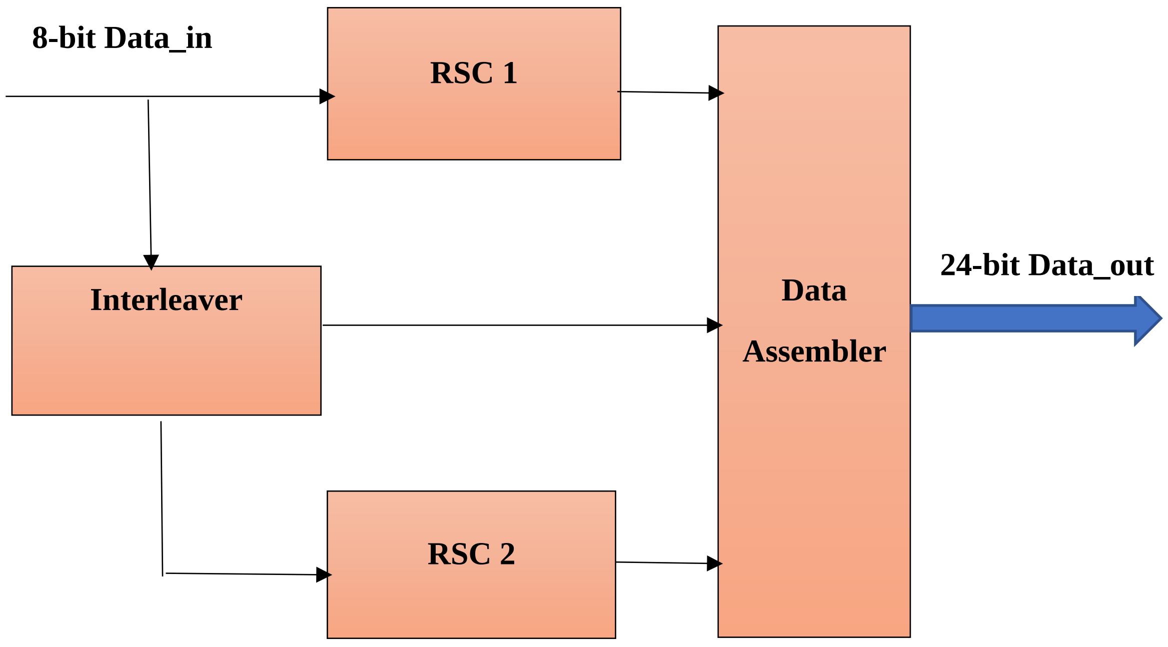


Figure 3.2: Block Diagram of Turbo Encoder

To achieve performance close to Shannon limit, the information block length (interleaver size) is chosen to be very large, usually at least several thousand bits. RSC codes, generated by systematic feedback encoders, give much better performance than non-recursive systematic convolutional codes, that is, feedforward encoders.

Because only the ordering of the bits changed by the interleaver, the sequence that enters the second RSC encoder has the same weight as the sequence x that enters the first encoder. An interleaver is often between the encoders to improve burst error correction capacity or to increase the randomness of the code.

Turbo codes use the parallel concatenated encoding scheme. However, the turbo code decoder is based on the serial concatenated decoding scheme. The serial concatenated decoders are used because they perform better than the parallel concatenated decoding scheme due to the fact that the serial concatenation scheme has the ability to share information between the concatenated decoders whereas the decoders for the parallel concatenation scheme are primarily decoding independently.

The random interleaver uses a fixed random permutation and maps the input sequence according to the permutation order.The length of the input sequence is assumed to be L.The best interleaver reorder the bits in a pseudo-random manner. Conventional block (row-column) interleavers do not perform well in turbo codes, except at relatively short block lengths.

3.4 Turbo Decoder

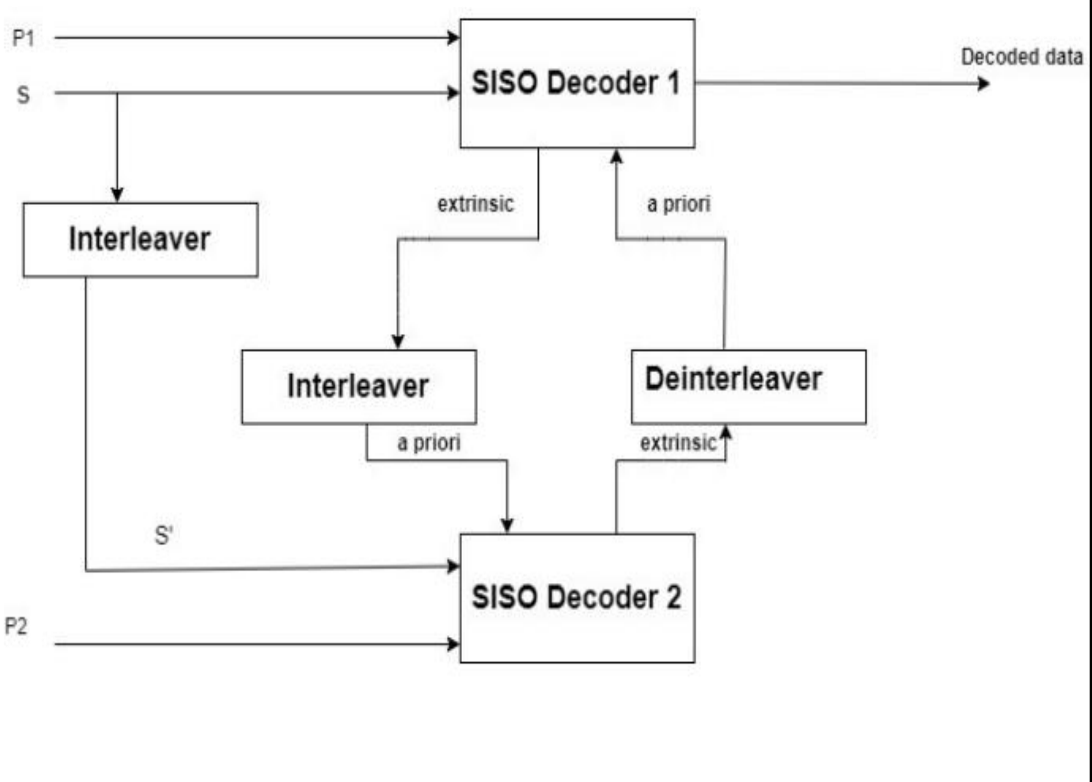


Figure 3.3 : Block Diagram of Turbo Decoder

A Turbo decoder consists of two single soft-in softout (SISO) decoders that work iteratively.The output of the first (upper decoder) feeds into the second to form a Turbo decoding iteration.Turbo codes are decoded using a method called the Maximum Likelihood Detection or MLD.Filtered signal is fed to the decoders, and the decoders work on the signal amplitude to output a soft "decision" The a priori probabilities of the input symbols is used, and a soft output indicating the reliability of the decision is calculated which is then iterated

between the two decoders. The form of MLD decoding used by turbo codes is called the Maximum a-posteriori Probability or MAP.However, ML decoder is often too complex to be implemented for turbo decoding because of the very complex trellis structure caused by the interleaver between the two constituent codes(CCs).In iterative decoding algorithm the two constituent decoders are used to perform SISO decoding over the coded sequences generated by the two CCs respectively, where the reliability information is exchanged between them during the decoding iterations.

INTERLEAVER:

The interleaver is a very important constituent of the turbo encoder. It spreads the burst error pattern and also increases the free distance .Thus, it allows the decoders to make uncorrelated estimates of the soft output values .The convergence of the iterative decoding algorithm improves as correlation of the

estimates decreases. Turbo encoder consists of interleaver unit which can be used to increase the BER performance by varying the interleaving size .In turbo code, interleaver unit is a random block that is used to rearrange the input data bits with no repetition. Interleaver unit is used in both encoder and decoder part. At the encoder side it generates along block of data, whereas in decoder part it correlates the two SISO (soft in soft out) decoder and helps to correct the error. At the decoder side after passing the encoded data from first decoder some of the errors may get corrected, then we again interleaver, this first decoded data and pass through the second decoder. There are certain types of interleavers "row column" interleaver in which data is written row wise and read column wise. While very simple, it also provides little randomness. In helical interleaver data is written row-wise and read diagonally. In an odd-even interleaver first, the bits are left uninterleaved and encoded, but only the odd-positioned coded bits are stored.

SISO: Soft in Soft out decoder

A soft-in soft-out (SISO) decoder is a type of soft-decision decoder used with error correcting codes.

"Soft-in" refers to the fact that the incoming data may take on values other than 0 or 1, in order to indicate reliability. "Soft-out" refers to the fact that each bit in the decoded output also takes on a value indicating reliability. Typically, the soft output is used as the soft input to an outer decoder in a system using concatenated codes, or to modify the input to a further decoding iteration such as in the decoding of turbo codes. The maximum a-posteriori (MAP) algorithm is used in the turbo-decoder under consideration for the SISO component decoder.

DEINTERLEAVER:

The deinterleaver is used to convert from an interleaved format.

It is a process of separation of combined data. Pseudo-random deinterleaver functions in a complimentary manner of pseudo-random interleaver.

3.5 Method to achieve the Objective

Our main objective is to implement the turbo coder. The objective is achieved by designing a Turbo Encoder and a Turbo Decoder using Verilog HDL. The decoder is developed based on SOVA algorithm.

The SOVA algorithm identifies the most probable bit of information that was sent. The algorithm helps in reducing the number of iterations at the decoding process. The SOVA algorithm is preferred , as it highly improves the hardware implementation parameters of a Turbo Decoders for LTE and LTE Advanced Standards also helps in reducing the area and power consumption of each block but also increases the throughput of the core.

Chapter 4

IMPLEMENTATION

To implement a project means to carry out activities proposed in the application form with the aim to achieve project objectives and deliver results and outputs. Its success depends on many internal and external factors. Project implementation consists of carrying out the activities with the aim of delivering the outputs and monitoring the progress compared to work plan.

The major steps associated with implementation of Project are:

1. Prepare the infrastructure.
2. Coordinate with the organization involved in implementation.
3. Implement training
4. Install the production solution.
5. Convert the data.
6. Perform final verification in production.
7. Implement new processes and procedures.
8. Monitor the solution.

Implementation is the process that turns strategies and plans into actions in order to accomplish strategic objectives and goals.

In project implementation, we manage implementation of all our project plans, following the triple constraint:

1. Project Scope Management
2. Project Time Management
3. Project Cost Management

Factors Affecting Project Implementation are:

1. Simplicity of Design
2. Careful Preparation and
3. Good Management.

4.1 Architecture of Turbo Coder

Architecture of Turbo Coder consists of designing the Turbo Encoder and Decoder. The Turbo Encoder Consists of two identical Recursive Systematic Convolutional (RSC) Encoders and interleaver. But each RSC encoder Operates on two different data. The first Encoder operates on an original input data, whereas the second encoder operates on an interleaved version of input data. LTE employs a 1/3 rate parallel concatenated turbo code. A specific algorithm is used to scramble the data bits and the method is called interleaving.

Interleaving is the method in which bits are rearranged according to a specified algorithm. The outputs of two RSC encoders and input are appended to get encoded data, whose length is three times the original bit stream. RSC encoders are preferred as it produces low weight parity, which will in turn decrease the transmission power. This will be transmitted through the channel to the Turbo decoder.

The Turbo decoder, on receiving the data, separates the received data to get the three-bit streams: one data stream and two parity streams. A standard turbo decoder block diagram contains two modules of SISO decoders together with a pseudorandom interleavers and a pseudorandom deinterleaver. The decoder is based on Viterbi Algorithm. The Decoders take these bit streams as inputs for producing the original data.

Basically, two algorithms are used namely MAP (Maximum a posteriori) and Soft output Viterbi algorithm (SOVA). Turbo decoders using MAP algorithm performs better in terms of Bit Error Rate (BER), but their complexity and latency is very high. MAP with sliding window can be used to reduce latency problem but still complexity is very high. So, SOVA algorithm is preferred.

SOVA based Turbo decoders can be implemented with high throughput and less complexity. In SOVA ,Soft Decision Decoding is used where received signal is represented as strong or weak based on its signal strength and is effectively determined, whether ‘0’ or‘1’.

4.2 Design of Turbo Encoder

The Turbo Encoder block encodes a binary input signal using a parallel concatenated coding scheme. This coding scheme employs two identical convolutional encoders and one internal interleaver.

4.2.1 Introduction to Encoder

An encoder in digital electronics is a one-hot to binary converter. That is, if there are 2 power n input lines, and at most only one of them will ever be high, the binary code of this ‘hot, line is produced on the n-bit output lines. A binary encoder is the dual of a binary decoder.

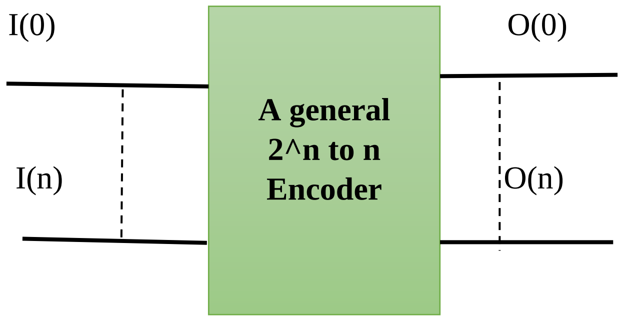


Figure 4.1 : General Encoder

For example, a 4-to-2 simple encoder takes 4 input bits and produces 2 output bits.

A0, A1, A2 and A3 are the 4 inputs and B0 , B1 are 2 outputs of an encoder.

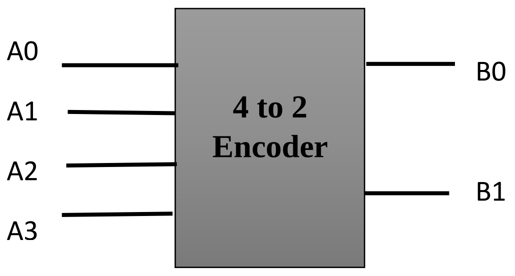


Figure 4.2: 4 to 2 Encoder

If the input circuit can guarantee at most single active input, a simple encoder is a better choice than a priority encoder, since it requires less logic to implement. However, a simple encoder can generate an incorrect output when more than a single input is active, so a priority encoder is required in such cases.

4.2.2 Types of Encoder

Encoder are classified into 4 categories:

1. **Decimal to BCD Encoder**- A decimal to BCD encoder has 10 input lines while 4 output lines. The 10 input lines correspond to decimal values and 4 output lines correspond to BCD code.

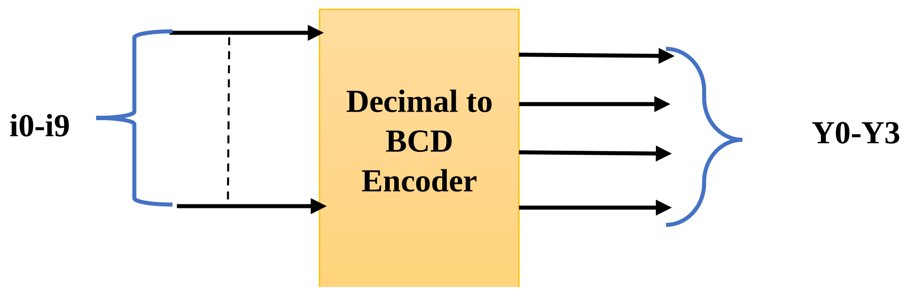


Figure 4.3: Decimal to BCD Encoder

2. **Octal to Binary Encoder-** In octal to binary encoder 8 inputs and 3 output lines are present. The applied input to the encoder corresponds to octal values while the output shows the binary values. Here, i0 to i7 represents the octal input and y0 to y2 shows binary output values.

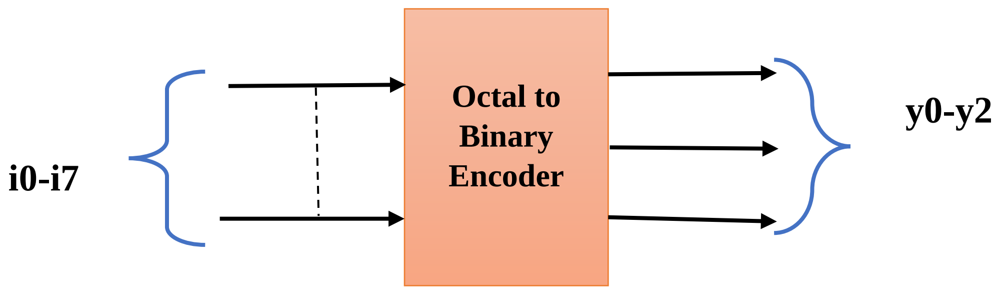


Figure 4.4: Octal to Binary Encoder

3. Hexadecimal to Binary Encoder- the hexadecimal to binary encoder contains 16 input lines as well as 4 output lines. So, the input provided shows the hexadecimal count and output represents the binary values. Here, i0 to iF represents hexadecimal input and y0 to y3 represents binary output.

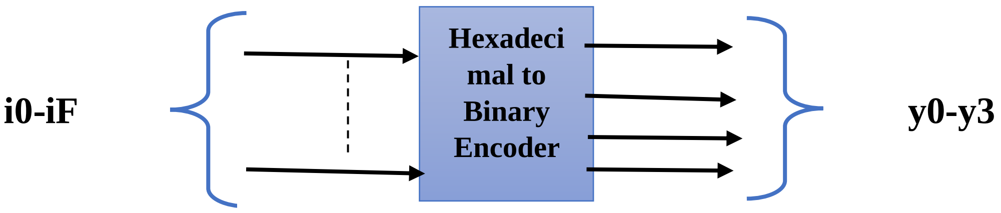


Figure 4.5: Hexadecimal to Binary Encoder

4. Priority Encoder- Priority encoders are a special type of encoders that were developed to eliminate the drawback associated with normal encoders. As when multiple inputs are high then the encoder will not be correctly responding to any one of the inputs. As in this case, an ambiguity will get generated. Due to this, Priority encoders were taken into consideration. In priority encoders, with movement in downward direction, Priority increases.

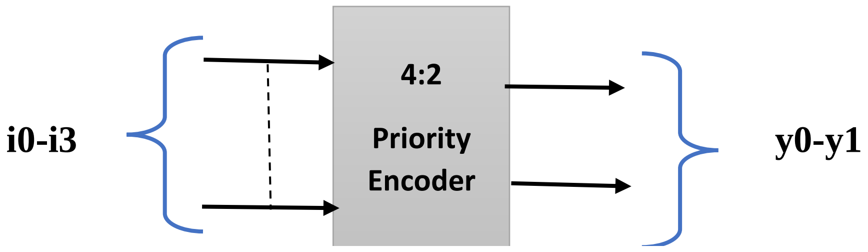


Figure 4.6: 4:2 Priority Encoder

Consider the 4:2 priority encoder where i0-i3 represents the 4 input lines and y0 and y1 shows output lines. Also, the priority encoder consists a 3rd output line V which is termed as a valid bit.

4.2.3 Example of 8-to-3 Encoder (Octal to Binary) with code

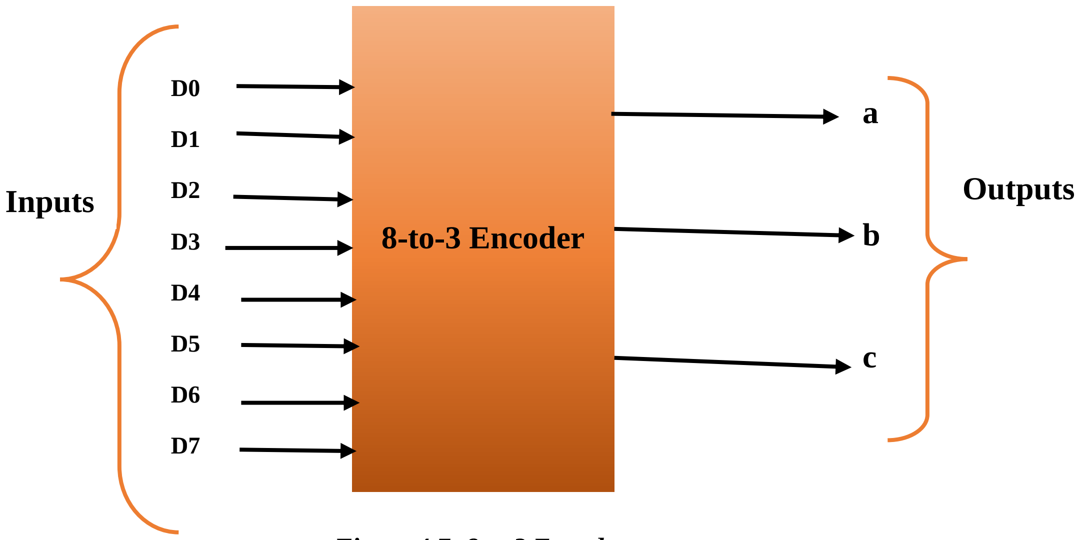


Figure 4.7: 8 to 3 Encoder

An 8-to-3 encoder consists of 8 inputs and 3 outputs. Here, D0 to D7 are inputs and a, b, c are the outputs.

Each input line corresponds to each octal digit and three outputs generate corresponding binary code.

➢ **Logical expression for a, b and c**

a = D0 + D1 + D2 +D3

b= D0 + D1+ D4 +D5

c= D0 + D2 + D4 +D6

➢ **Truth Table for 8 to 3 encoder**

Inputs								**Outputs**		
D0	**D1**	**D2**	**D3**	**D4**	**D5**	**D6**	**D7**	**a**	**b**	**c**
0	**0**	**0**	**0**	**0**	**0**	**0**	**1**	**0**	**0**	**0**
0	**0**	**0**	**0**	**0**	**0**	**1**	**0**	**0**	**0**	**1**
0	**0**	**0**	**0**	**0**	**1**	**0**	**0**	**0**	**1**	**0**
0	**0**	**0**	**0**	**1**	**0**	**0**	**0**	**0**	**1**	**1**
0	**0**	**0**	**1**	**0**	**0**	**0**	**0**	**1**	**0**	**0**
0	**0**	**1**	**0**	**0**	**0**	**0**	**0**	**1**	**0**	**1**
0	**1**	**0**	**0**	**0**	**0**	**0**	**0**	**1**	**1**	**0**
1	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**1**	**1**	**1**

Table 4.1: Truth table of 8 to 3 Encoder

➢ **Circuit Diagram Using OR gates**

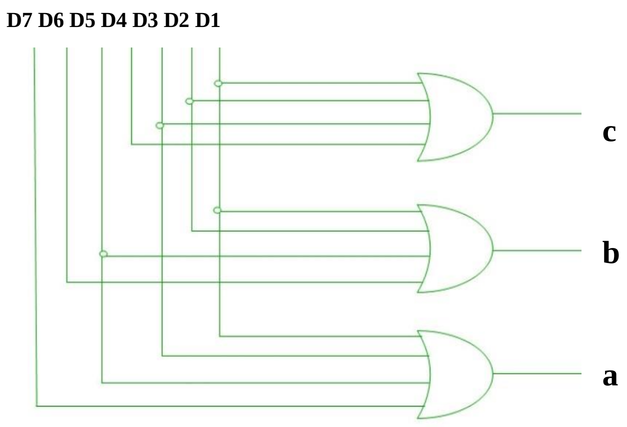


Figure 4.8: circuit diagram of 8 to 3 Encoder

- **Verilog Code**

```
module encoder8_3(a, b, c, D, En);
input [0:7]D;
input En;
output a, b, c;
reg a, b, c;
always@(En or D)
begin
if(En)
begin
case(D)
8’o0: Eo= 3’b000;
8’o1: Eo= 3’b001;
8’o2: Eo= 3’b010;
8’o3: Eo= 3’b011;
8’o4: Eo= 3’b100;
8’o5: Eo= 3’b101;
8’o6: Eo= 3’b110;
8’o7: Eo= 3’b111;
Default: $display(“Error!”);
endcase
end
end
endmodule
```

- **Test Bench Code:**

```
module encount_tb();
reg [0:7]D;
reg En;
wire a, b, c;
encoder u1(.(a(a), .b(b), .c(c), .D(D), .En(En));
initial
begin

D = 8’b00000000;
En = 1’b1;
#10 D = 8’b00000001;
#10 D = 8’b00000010;
#10 D = 8’b00000100;
#10 D = 8’b00001000;
#10 D = 8’b00010000;
#10 D = 8’b00100000;
#10 D = 8’b01000000;
#10 D = 8’b10000000;
end
endmodule
```

4.2.4 Implementation of Turbo Encoder

To Design Turbo Encoder it requires the following modules:

1. Recursive Systematic Convolutional (RSC) Encoder.

2. Interleaver.

3. Data Assembler.

For the Turbo Encoder, 4-bit information is given as input. This input bit stream is applied to various blocks like RSC1 and Interleaver. Interleaved input bit stream is applied to RSC2 block. RSC1 generates partiy-1 and RSC2 generates Parity-2. The systematic data comprising of Parity-1 and Parity-2 is given to Data Assembler, which generates the Turbo encoded output. So, the output of Turbo Encoder is 12-bit data.

4.3 Design of RSC Encoder

Two identical Recursive convolutional encoders(RSC) and a pseudorandom interleaver constitutes the turbo encoder .LTE employs a 1/3 rate parallel concatenated turbo code. Each RSC works on two different data. Original data is provided to the first encoder, while the second encoder receives the interleaved version of the input data. The RSC1 and RSC2 encoder outputs along with systematic input comprises the output of turbo encoder, that is, a 12- bit output is generated.

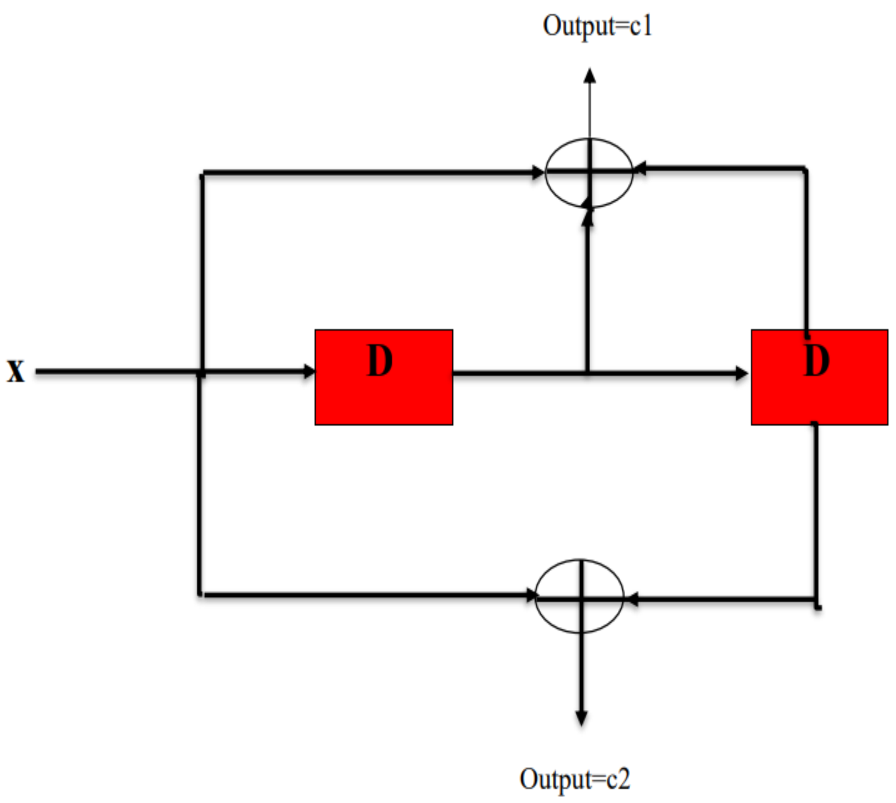


Figure 4.9: Conventional convolutional encoder with r=1/2 and K=3.

The conventional convolutional encoder is represented by the generator sequences g1 =[111] and g2 =[101] and can be equivalently represented in a more compact form as G=[g1, g2].Convolutional codes are commonly described using two parameters: the code rate and the constraint length. The code rate, k/n, is expressed as a ratio of the number of bits into the convolutional encoder (k) to the number of channel symbols output by the convolutional encoder (n) in a given encoder cycle. Constraint length, k, is the size of the shift register in the memory encoder without feedback. It means how many stages are needed for the combinational logic that produces the output bits. The shift register of size k, stores the present bit and the past (k -1) bits. Block codes, information bits are followed by parity bits. Convolution codes, information bits are spread along the sequence. Block codes are memoryless whereas Convolution codes have memory.

4.3.1 Systematic Convolutional Encoder

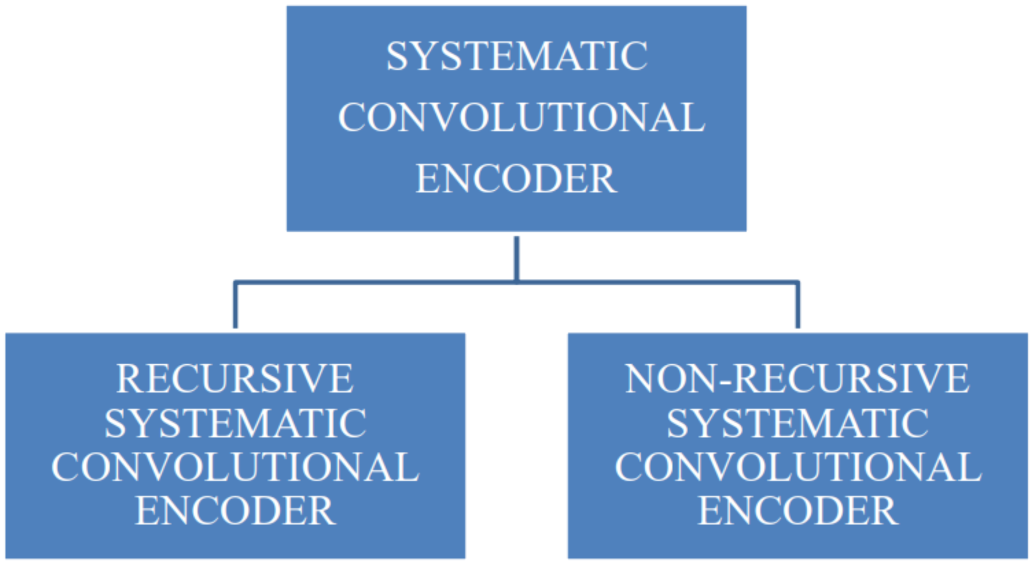

SYSTEMATIC CONVOLUTIONAL ENCODER
RECURSIVE SYSTEMATIC CONVOLUTIONAL ENCODER
NON-RECURSIVE SYSTEMATIC CONVOLUTIONAL ENCODER

Figure 4.10: Types of Systematic Convolutional Encoder

During the last years, recursive systematic convolutional (RSC) encoders have found application in modern telecommunication systems to reduce the bit error rate (BER). In view of the necessity of increasing the throughput of such applications, several approaches using hardware implementations of RSC encoders were explored. During the last years, many applications, like space telecommunications systems , digital TV broadcasting and wireless metropolitan area networks have been exploiting forward error correcting (FEC) codes to reduce the bit error rate (BER) in data transmission. FEC approaches encode data to transmit by using redundant codes which allow to estimate the correct bit stream transmitted at receiver by using a maximum likelihood or a maximum a posteriori decoding. One of the most important FEC techniques is represented by recursive systematic convolutional (RSC) encoders. An RSC encoder can be constructed from a standard convolutional encoder by feeding back the one of the outputs. An RSC encoder has the infinite impulse response. An arbitrary input will cause “good” (high weight)output with high probability. Some inputs will cause “bad”(low weight)outputs. The recursive systematic convolutional (RSC) encoder is obtained from the non-recursive non-systematic (conventional) convolutional encoder by feeding back one of its encoded outputs to its input. RSC encoders are realized through linear feedback shift registers (LSFRs). The latter are devices described by a set of generator polynomials whose coefficients belong to Galois fields of two elements (GF(2)). In particular, in Equation (1) we define the feedback polynomial:

g(x) = N ∑ $gi\ n\ i$=0 · x ^i ------(1)

The unitary coefficients gi in g(x) indicate which present states contribute to determinate the successive ones. The presence of a feedback polynomial makes the encoder recursive. Moreover, for each input bit the RSC produces No outputs, a set of No − 1 feedforward polynomials are also defined as in Equation (2):

hk (x) = N ∑ $(hki)xi,\ k\ N\ i$=0 , k ∈ {1, 2, ..., No − 1}--------- (2)

where, hk (x) is the polynomial producing the (k + 1) the RSC output, whose coefficients are indicated as hki .

Moreover, the input bit is directly reproduced (systematic output) as output together with the other No − 1 redundant bits in the output code. For such reason, the encoder is defined as systematic. One of the merit parameters which describes the RSC encoder is the code rate, defined in Equation (3):

R =K /No -------------(3)

The nature of recursive and non-recursive convolutional encoders is best examined by an example. Figure shows a simple non-recursive convolutional encoder with generator sequences g1=[11] and g2=[10].

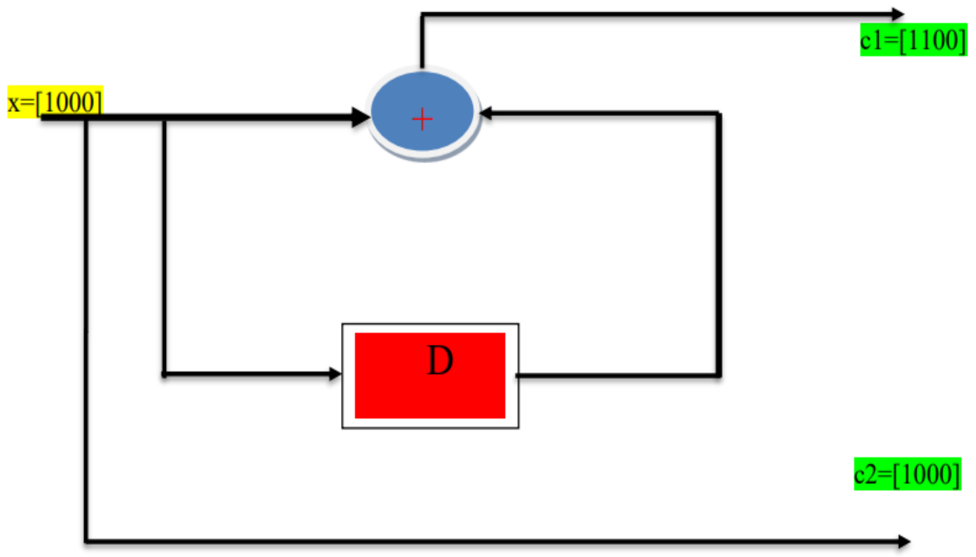


Figure 4.11: Non-recursive r=1/2 and K=2 convolutional encoder with input and output sequences.

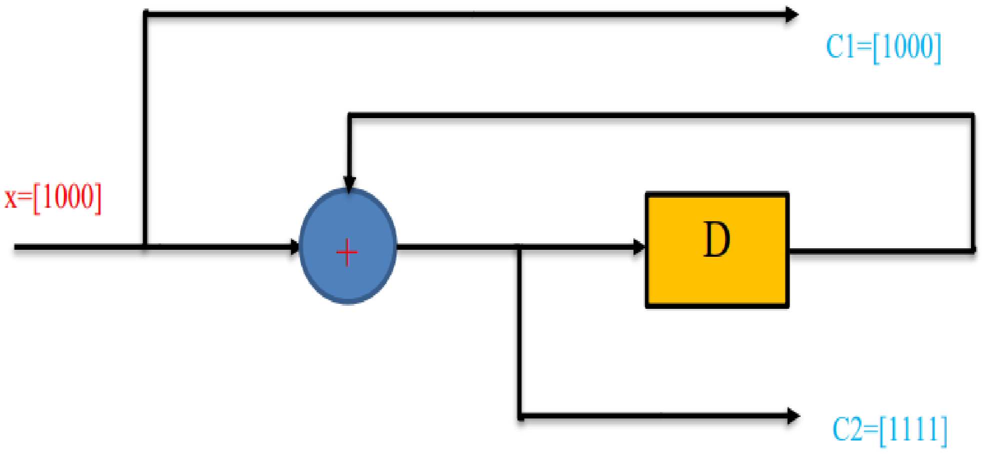


Figure 4.12: Recursive r=1/2 and K=2 convolutional encoder with input and output sequences.

From Figure 4.11 and Figure 4.12 given the same input sequence, the non-recursive encoder produces an output codeword with weight of 3 and the recursive encoder produces an output codeword with weight of 5. Thus, a recursive convolutional encoder tends to produce codewords with increased weight relative to a non-recursive encoder. This results in fewer codewords with lower weights and this leads to better error performance. For turbo codes, the main purpose of implementing RSC encoders as component encoders is to utilize the recursive nature of the encoders and not the fact that the encoders are systematic.

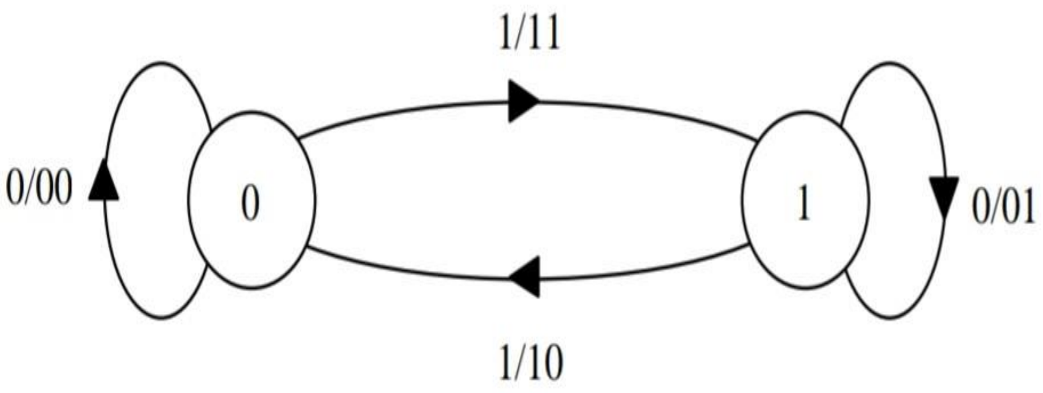


Figure 4.13: :State Diagram Of Recursive Systematic Convolutional Encoder

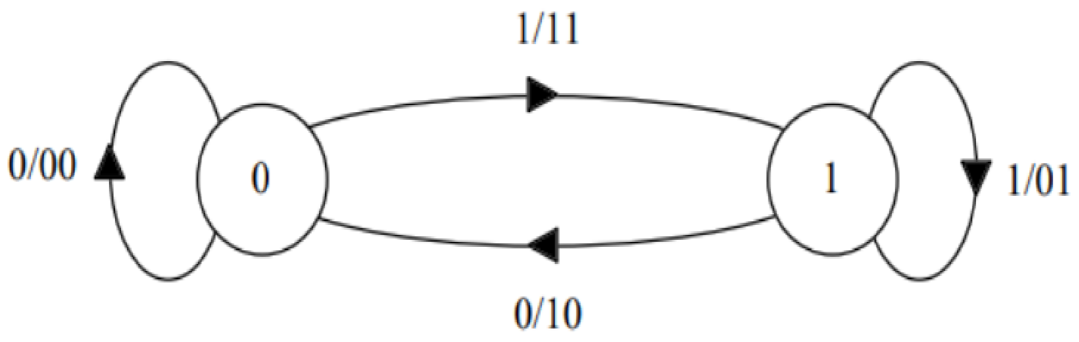


Figure 4.14: State Diagram Of Non-Recursive Systematic Convolutional Encoder

4.3.2 Recursive Systematic Convolutional Encoder

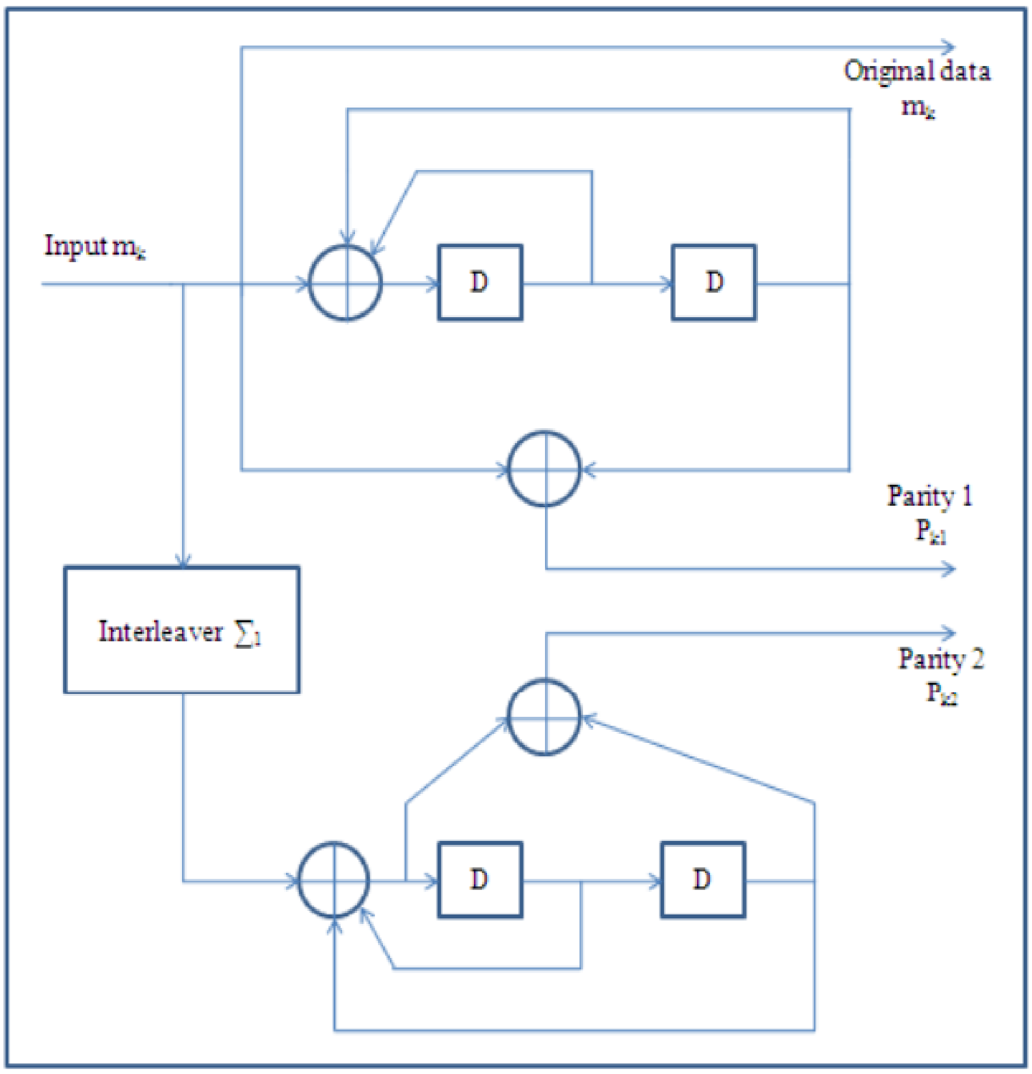


Figure 4.15: Example of rate R= 1/3 Recursive Systematic Convolutional Encoder

The Recursive Systematic Convolutional Encoder as constituent encoders in a parallel concatenation scheme. The information bit given as input m2 is same for interleaver also where reordering of information bits take place. The original data m2 is the systematic output.

The RSC1 produces Parity1 bit and RSC2 produces parity2 bit. Hence, both encoders are parallel concatenated, the same input is fed to both convolutional encoders. Recursive encoders are feedback encoders as there is feedback from output going to input. To have large minimum distance of codes, feedback encoders are used.

As larger the minimum distance of code , higher is the error correcting capability. The parallel concatenation is preferred as in serial concatenation, the output of first encoder is given as input to second encoder .

Convolutional codes are developed for real-time error correction. Convolutional codes generate one single code word from entire input bit stream. The encoded bit depends not only on current bit but additionally on antecedent bit information. More computational complexity arises as we increase the length of code exponentially.

Turbo codes produce high weight codes by using recursive convolutional encoders, useful to distinct the codes easily in decoding process. The difference between recursive and non-recursive convolutional encoders is that the recursive encoders have feedback connection whereas non- recursive encoders doesn’t have.

4.4 INTERLEAVER

4.4.1 Theory of Interleaver

Interleaving is a process, which is performed before transmission to disperse the data bits so that even if a part of the information is corrupted while passing through the channel, it can be recovered by rearranging the data after reception. There are different types of interleavers, which affects the performance of a system. In a digital communication system, interleaving is used to attain time diversity without any addition of redundant bits. Due to the rapid proliferation of digital speech coders which transform analog voices into efficient digital messages that are transmitted over wireless links, interleaving has become an extremely useful technique in all second and third generation wireless systems. Speech coders attempt to represent a wide range of voices in a uniform and efficient digital format. The encoded bits i.e. the source bits which consists maximum amount of information are more important than others and must be free from errors. Many speech coders produce several important bits in succession, and it is the function of the interleaver to spread those bits in time so that even if a deep fade or noise burst occurs, these important bits will not be corrupted. Spreading of these bits over time is necessary, because it becomes possible to make use of channel coding which protects the source data from corruption by the channel. Error correcting codes are designed to protect against channel errors, while interleavers scramble the time order of source bits before they are channel encoded.

4.4.2 Role of an Interleaver

The primary role of an interleaver is to disperse the sequences of bits in a bit-stream so as to minimize the effect of burst errors introduced in transmission. An interleaver is usually used in conjunction with some type of error correcting code. The error correcting codes can correct the lost information as long as the amount of lost bits in a single code word is within the limit defined by d_{min}. The limits sometimes get violated due to existence of errors in the form of bursts. The usual sources of bursty errors are listed below,

- Channel distortion and channel fading effects in wireless communications due to multipath propagation of the transmitted signal caused by reflection, diffraction, scattering, etc.
- Lightening and switching effects.
- Use of concatenated coding schemes, e.g. use of convolution code (Viterbi decoding) at the first stage usually generates a burst of errors when the decoding fails.

A basic interleaver is taken as a sequential device with single input and single output. A parallelism can be devised but the basic function remains the same. The interleaver takes a sequence of information bits with fixed width and generates another sequence of same width but in different order. It is then the role of a de-interleaver to retrieve back the original order. Thus the interleaver and deinterleaver strictly work in pair with each other.

An interleaver carries a specific permutation pattern π and the mapping of an input sequence $x \rightarrow x_0, x_1, x_2 \ldots x_k$ on to an output sequence $y \rightarrow y_0, y_1, y_2 \ldots y_k$ can be defined as

$$y_i = x_{\pi(i)} \qquad \ldots\ldots (1)$$

The reverse of this function (i.e. de-interleaving) is:

$$y_{\pi(i)} = x_i \qquad \ldots\ldots (2)$$

4.4.3 Interleaver Design

For turbo codes, an interleaver is used between the two component encoders. The interleaver is used to provide randomness to the input sequences. Also, it is used to increase the weights of the code words

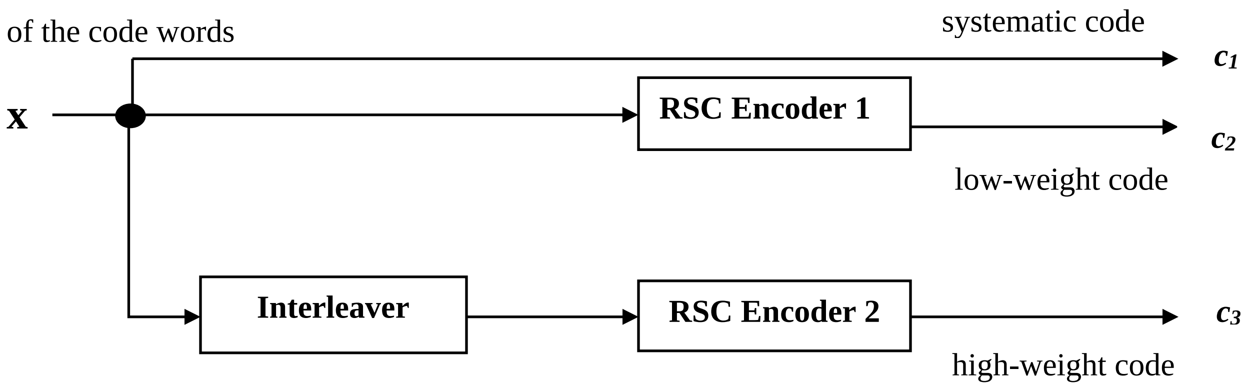


Figure 4.16: The interleaver increases the code weight for RSC Encoder 2 as compared to RSC Encoder 1

From Figure 4.16, the input sequence x produces a low-weight recursive convolutional code sequence c2 for RSC Encoder 1. To avoid having RSC Encoder 2 produce another low-weight recursive output sequence, the interleaver permutes the input sequence x to obtain a different sequence that hopefully produces a high-weight recursive convolutional code sequence c3. Thus, the turbo code's code weight is moderate, combined from Encoder 1's low-weight code and Encoder 2's high-weight code. Figure 4.17 shows an illustrative example

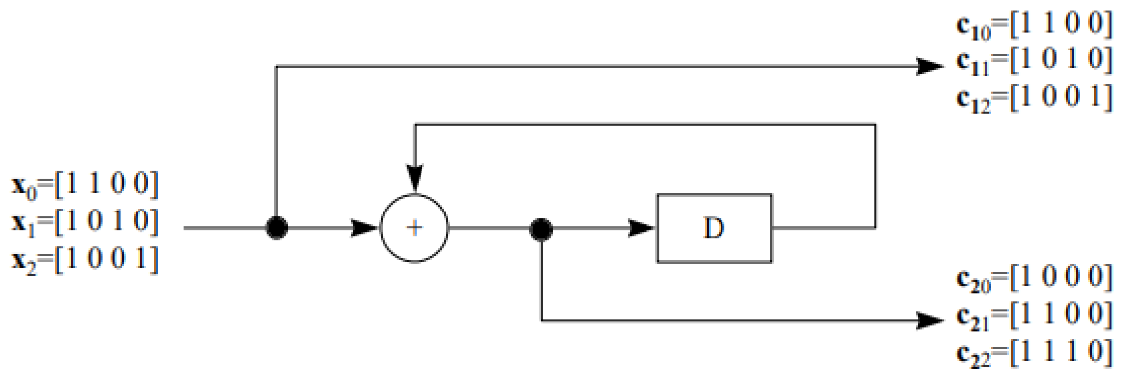


Figure 4.17: An illustrative example of an interleaver's capability

	Input Sequence **x_i**	Output Sequence **C_{1i}**	Output Sequence **C_{2i}**	Code word Weight **i**
i = 0	1 1 0 0	1 1 0 0	1 0 0 0	3
i = 1	1 0 1 0	1 0 1 0	1 1 0 0	4
i = 2	1 0 0 1	1 0 0 1	1 1 1 0	5

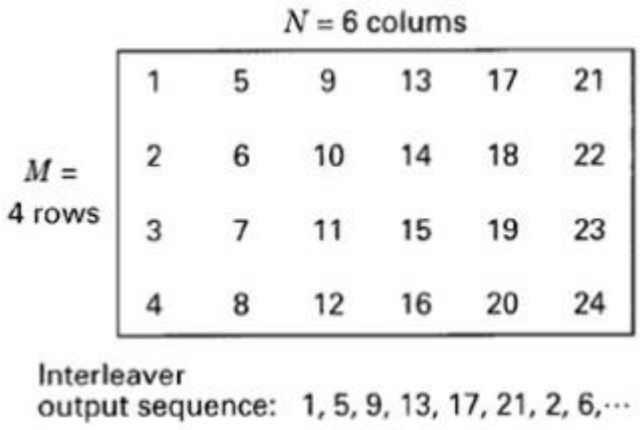
Figure 4.18: Input and Output Sequences for Encoder in Figure 4.17

The code word weight can be increased by utilizing an interleaver. The interleaver affects the performance of turbo codes because it directly affects the distance properties of the code. By avoiding low-weight code words, the BER of a turbo code can improve significantly.

4.4.4 Types of Interleaver

a. Block Interleaver

A block interleaver accepts coded symbols in blocks from encoder, shuffles the symbols and then feeds the rearranged symbols to the data modulator. The shuffling of block is accomplished by filling the columns of an M- row by N - column (M X N) array with encoded sequence. After the array is filled, these symbols are fed to the modulator one row at a time and transmitted over the channel. At the receiver, the deinterleaver performs inverse operation, the symbols are entered by rows and removed one column at a time.



N = 6 colums

M = 4 rows

1	5	9	13	17	21
2	6	10	14	18	22
3	7	11	15	19	23
4	8	12	16	20	24

Interleaver output sequence: 1, 5, 9, 13, 17, 21, 2, 6,···

Figure 4.19: M x N Block Interleaver

Example: As shown, 24 code symbols are place into the interleaver input. The output sequence to the transmitter consists of code symbols removed from array by rows.

Block interleaver is further categorized into various subtypes such as

- **Matrix Interleaver:** Matrix Interleaver is a type of block interleaver that performs interleaving by filling the input symbols row by row in a matrix of m rows and n columns and then output of the interleaver is read column by column. The column size n is called the depth and the row size m is the span. Such an interleaver is completely defined by m and n and is thus referred to as (m, n) matrix interleaver. At the deinterleaver, information is written column-wise and read out row wise.

- **Helical Interleaver**: A helical interleaver writes data into row–wise but reads data from its matrix diagonally instead of by column in such a way that consecutive interleaved data are never read from the same row or column.

- **Random Interleaver:** Random interleaver scrambles the data of different users with different pattern. Patterns of scrambling the data of users are generated arbitrarily. Because of the scrambling of data, burst error of the channel is randomized at the receiver side. The user specific Random Interleaver rearranges the elements of its input vector using a random permutation. The incoming data is rearranged using a series of generated permuter indices. A permuter is essentially a device that generates pseudo-random permutation of given memory addresses. The data is arranged according to the pseudo-random order of memory addresses. After randomization of the burst error which has rearranged the whole block of the data the latter can now be easily detected and corrected. Spreading is the important characteristic of random interleaver.

- **Odd-Even Interleaver:** An odd-even interleaver is a block interleaver in which the number of rows and columns must be odd numbers. The odd-even interleaver design is specifically for the r = ½ turbo code. A r = ½ turbo code is obtained by puncturing the two coded (nonsystematic) output sequences of a r = 1/3 turbo code. However, by puncturing these two coded output sequences, it is possible that an information (systematic) bit may not have any of its coded bits (both of the associated coded bits may be punctured out). Likewise, it is also possible for an information bit to have one or both of its coded bits. Thus, if an error occurs for an unprotected information bit (without any of its coded bits), the turbo code decoder may degrade on its performance.

The odd-even interleaver design overcomes this problem by allowing each information bit to have exactly one of its coded bits. As a result of this interleaver, the error correction capability of the code is uniformly distributed over all information bits.

b. **Convolutional Interleaver:** A convolutional interleaver is an interleaver which consists of a number of shift registers. The shift registers have a fixed delay each, which are positive integer multiples of a fixed integer. Every new data to the input of the interleaver is fed to the next shift register and the previous data in that register becomes part of the interleaver output. A convolutional interleaver has memory; that is, its operation depends not only on current symbols but also on previous symbols. In a typical convolutional interleaver, the delays are nonnegative multiples of a fixed integer (although a general multiplexed interleaver allows unrestricted delay values).

4.4.5 Pseudo-Random Interleaver

The random interleaver uses a fixed random permutation and maps the input sequence according to the permutation order. The length of the input sequence is assumed to be L. Figure 4.20 shows a random interleaver with L=4.

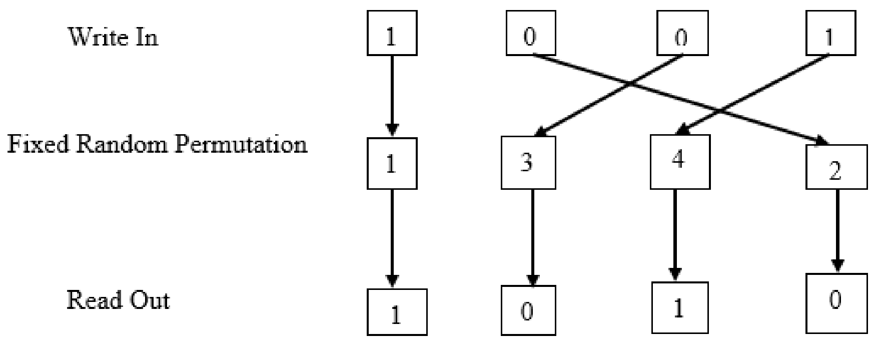


Figure 4.20: A random (pseudo-random) interleaver with L=4.

From Figure 4.20, the interleaver writes in [1 0 0 1] and reads out [1 0 1 0].

4.5 Data Assembler

The three 8-bit or 4-bit streams, such as input information; RSC encoded; and interleaved RSC encoded are appended by data Assembler to make a 24-bit or 12-bit encoded data respectively. The information bit stream, RSC1 encoded output and interleaver RSC2 output is applied to Data Assembler.

The process of Turbo Encoder is illustrated in below flowchart.

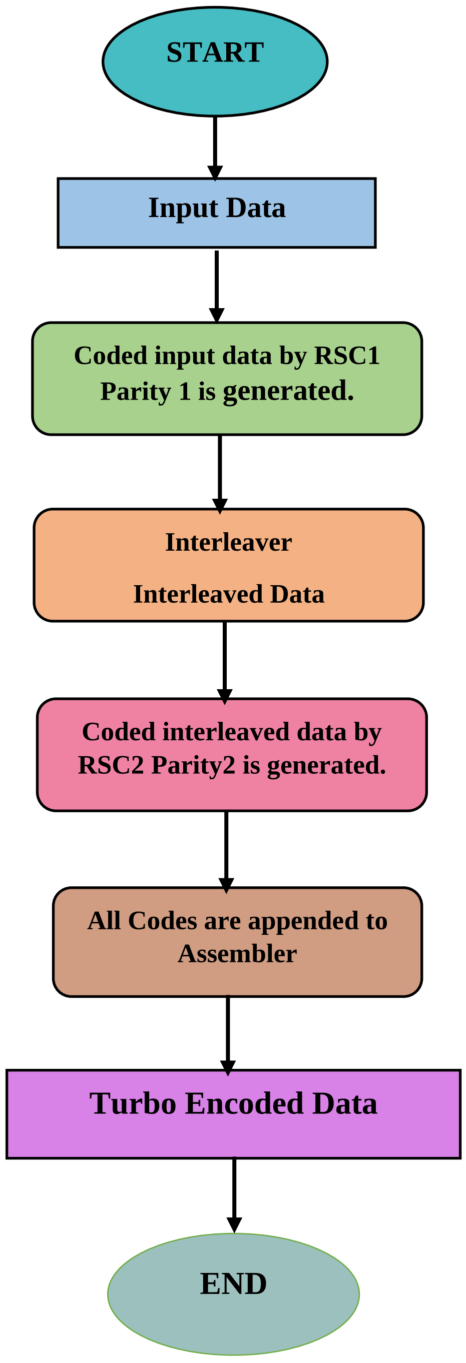


Figure 4.21: Flowchart of Process of Turbo Encoder.

4.6 Design of Turbo Decoder

A Turbo Decoder consists of two soft-input soft-output (SISO) decoders with two interleavers and one deinterleaver. The decoding process in turbo decoder is performed iteratively. The data is processed through two SISO decoders via the interleaver and the deinterleaver.

4.6.1 Introduction to Decoders

Decoder is a combinational circuit that has 'n' input lines and maximum of 2 power n output lines. One of these outputs will be active High based on the combination of inputs present, when the decoder is enabled. That means decoder detects a particular code. The output of decoder is nothing but minterms of 'n' input variables lines, when it is enabled.

Let us implement a higher order decoder using lower order decoder:

- **3 to 8 decoder:** a 3 to 8 decoder is implemented using 2 to 4 decoder. A 2 to 4 decoder has two inputs, A1 and A0 , and four outputs Y3 to Y0. Whereas, 3 to 8 decoder has three inputs A2, A1 & A0 and eight outputs, Y7 to Y0. Therefore, two 2 to 4 decoders are required for implementing one 3 to 8 decoder.

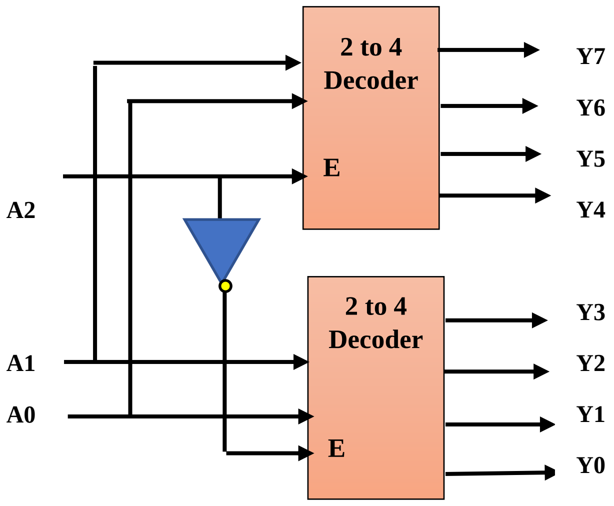


Figure 4.22: Block diagram of 3 to 8 decoder using 2 to 4 decoders

The parallel inputs A1 & A0 are applied to each 2 to 4 decoders. The complement of input A2 is connected to enable, E of lower 2 to 4 decoder in order to get outputs, Y3 to Y0. These are lower 4 min terms. The input, A2 is directly connected to Enable, E of upper 2 to 4 decoder in order to get outputs Y7 to Y4. These are higher 4 min terms.

4.6.2 Implementation of Turbo Decoder

To implement a Turbo Decoder , the first criteria is to choose a suitable decoding algorithm.

- **Decoding Algorithm**

The two main types of decoding algorithms available for decoding of turbo codes is as shown in figure. All algorithms are based upon the trellis-based estimation.

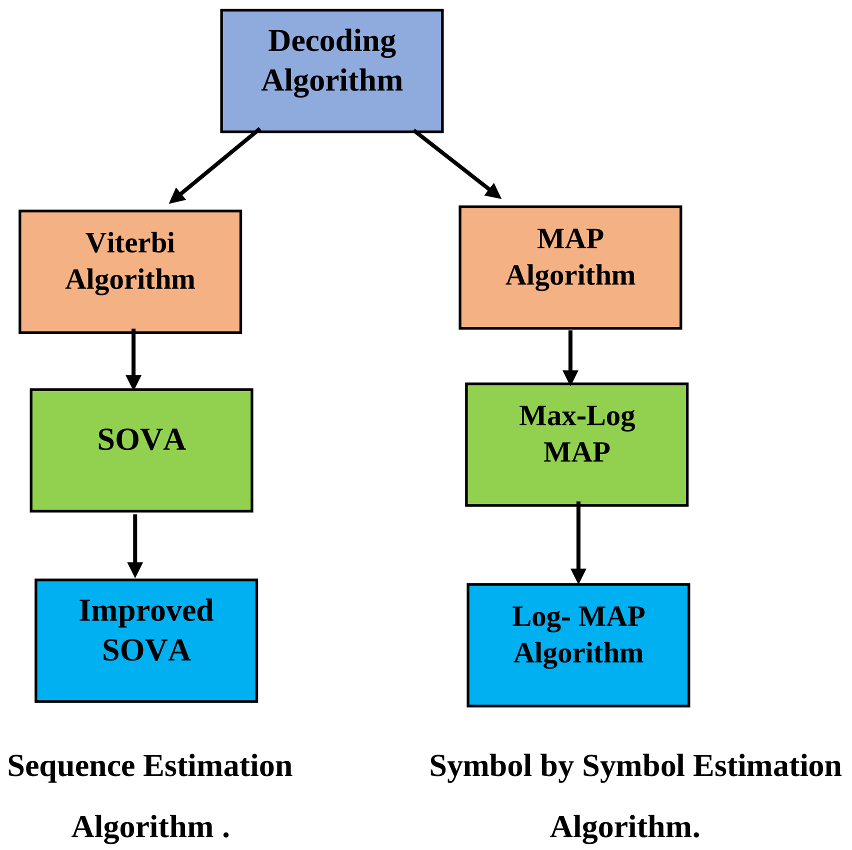


Figure 4.23: Turbo Decoding Algorithm

The trellis-based estimation algorithms are classified into two types. They are sequence estimation algorithms and symbol-by-symbol estimation algorithms. The Viterbi algorithm, SOVA (soft output Viterbi algorithm) and improved SOVA are classified as sequence estimation algorithms. Whereas, the MAP algorithm, Log-Map algorithm, and Max-Log-Map are classified as symbol-by-symbol estimation algorithms. In general, the symbol-by-symbol estimation algorithms are more complex than the sequence estimation algorithms. Therefore, Sequence estimation algorithms are preferred. The Viterbi algorithm is a hard-decision output decoding algorithm and soft-output is produced by SOVA algorithm.

A. Soft Output Viterbi Algorithm (SOVA)

The Soft-Output Viterbi Algorithm (SOVA) is a modified Viterbi Algorithm which can not only take in soft quantized samples but also provide soft decision outputs along with the hard decision. The SOVA can generate the hard decision in the same way as the Viterbi Algorithm. In addition, it can generate the reliability of the hard decision, which is named as the Soft-Output. The Soft-Output can be provided to its successor to improve the decoding performance of the concatenated decoder.

At time k t , the ACSU in the Viterbi decoder selects a path with larger path metric between the two paths merging at the same states(It should be noticed that SOVA requires soft input. So the ACSU in the decoder should select the path with larger path metric). Without loss of generality, we assume path 1 is selected the survivor (that is, the path metric of path is larger). The probability of selecting the wrong path (path 2) as the survivor is

$$P_{s,k} = \frac{e^{-\Gamma_{s,k}(2)}}{e^{-\Gamma_{s,k}(1)} + e^{-\Gamma_{s,k}(2)}} = \frac{1}{1+e^{|\Delta_{s,k}|}} \leqslant \frac{1}{2}\ , \quad \Delta_{s,k} = |\Gamma_{s,k}(2) - \Gamma_{s,k}(1)| \geqslant 0$$

Where , (1) Γs k and , (2) Γs k is the path metric of path 1 and path 2 respectively . The absolute value computation in Equation (3-1) can make the formula independent of the actual path decision.

With probability Psk, , the Viterbi decoder made errors in all the e positions where the information bits of path 1 differs from path 2 (we can also say the decoder made a relevant decision at these positions); in other words if

$$u_j^{(1)} \neq u_j^{(2)}, \quad j = j_1, \dots, j_e$$

there are two paths merging at state 00 at time 9 t . Path 1 is selected as the survivor path and its path metric is 23. Path 2 is discarded and its path metric is 17.The discarded path can also be named as the competing path. The information bits of path 1 is

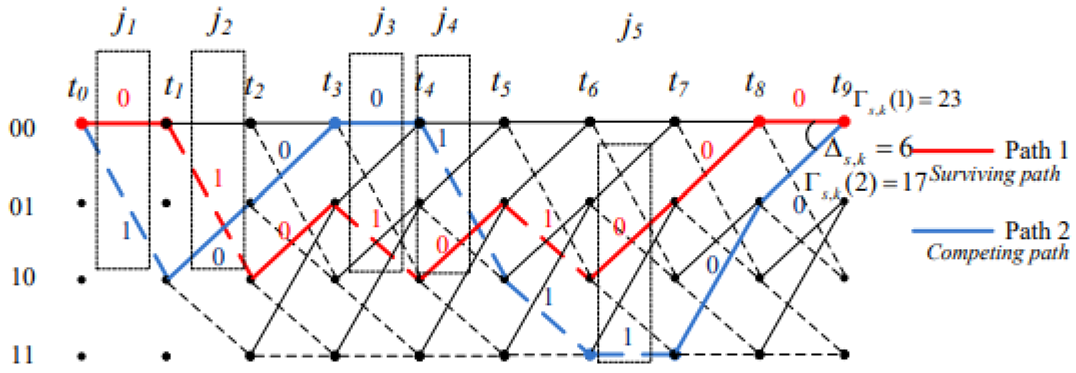
$$u_0^{(1)} \sim u_8^{(1)} = 0\ 1\ 0\ 1\ 0\ 1\ 0\ 0\ 0.$$

The information bits of path 2 is

$$u_0^{(2)} \sim u_8^{(2)} = 1\ 0\ 0\ 0\ 1\ 1\ 1\ 0\ 0.$$

So, there are 5 positions where the information bit of path 1 differs from path 2, and the Viterbi decoder made errors in those 5 positions with probability

$$P_{0,9} = \frac{e^{-17}}{e^{-23} + e^{-17}} = \frac{1}{1+e^{6}} \approx 0.0025$$



Assume the length before path 1 and path 2 merge is m δ . So there are e different information values, and m δ – e equal values. Assume the probabilities ^ Pj of previous erroneous decisions with path 1 have been stored. If path 1 is selected as the survivor path, these probabilities for the e differing decisions on this path according to the following equation

$$\hat{P}_j \leftarrow \hat{P}_j(1-P_{s,k}) + P_{s,k}(1-\hat{P}_j), \quad j = j_1, \ldots, j_e \qquad (3\text{-}2)$$

It is easily to demonstrate that ^ , 0 0.5 , < < Ps k Pj . Equation (3-2) requires that ^ Pj and Psk, are statistical independent. And it is approximately true for most of the practical codes. Both Psk, and ^ Pj are not suitable for hardware implementation. In practice, the log-likelihood ratio ^ Lj is used instead of ^ Pj , which is defined as

$$\hat{L}_j = \log\frac{1-\hat{P}_j}{\hat{P}_j} \qquad (3\text{-}3)$$

Where ^ Lj is named as the reliability of decision j u and it always has a positive value. The bigger ^ Lj is, the more reliable the decision j u is.

If the decision j u was relevant, its reliability ^ Lj can be updated using the formal value of ^ Lj and the path metric difference of path 1 and path 2, ie, Δsk, . The updating formula is now

$$\hat{L}_j \leftarrow f(\hat{L}_j, \Delta_{s,k}) = \min(\hat{L}_j, \Delta_{s,k}) \qquad (3\text{-}4)$$

Equation (3-4) is much easier to implement in hardware and does not deteriorate the performance of the algorithm . An example showing how to update ^ Lj . In this figure, path 1 and path 2 merge at state 00 at time 5 t , of which path 1 is selected as the survivor path. The path metric difference of the two paths is Δ0,5 . The information bits of path 1 is

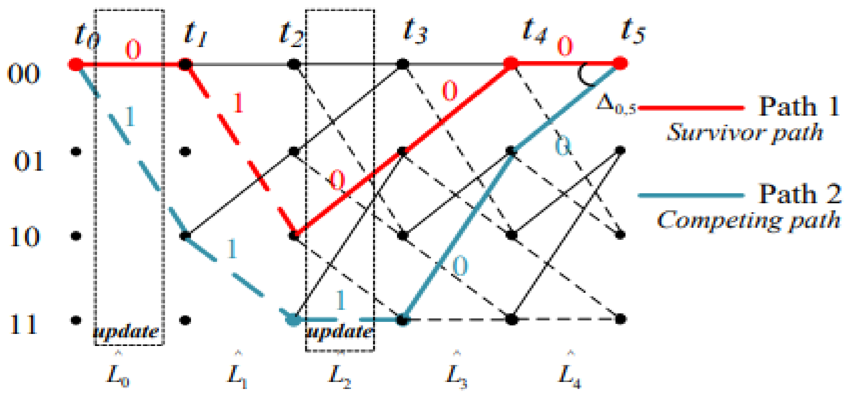
$$u_0^{(1)} \sim u_4^{(1)} = 0\ 1\ 0\ 0\ 0.$$

The information bits of path 1 is

$$u_0^{(2)} \sim u_4^{(2)} = 1\ 1\ 1\ 0\ 0.$$

So we can see that information bits 0 u and 2 u are relevant and their reliability ^ L0 and ^ L2 need to be updated. Before being updated, all the reliability values ^ ^ L L 0 4 ~ have been initialized to a very large value at time 0 t and some of them have been updated at time t t 0 4 ~ . At time 5 t , only ^ L0 and ^ L2 needed to be updated and the update equation is

$$\hat{L}_0 \leftarrow \min(\hat{L}_0, \Delta_{0,5}) \ \ and \ \ \hat{L}_2 \leftarrow \min(\hat{L}_2, \Delta_{0,5}) \qquad (3\text{-}5)$$



SOVA is an algorithm derived from Viterbi algorithm family. Viterbi is an efficient hard decision algorithm for implementing trellis decoding. Since it only produces hard decisions, but the component decoders of turbo decoders should output soft values, Viterbi algorithm cannot be applied to SISO unit directly. SOVA uses a modified path metric which takes into account the a-priori probabilities of the input symbols, and produces a soft output indicating the reliability of the decision.

In Viterbi algorithm, only the survivor path metric is stored and the value of the loser is discarded. For this reason, information about the reliability of the selected path becomes unknown. Different from Viterbi, SOVA saves not only the survivor path metric, but also the path metric difference at each place where 2 paths merge. These path metric differences are later used to produce soft output. Larger the difference, stronger is the confidence value of the decision made for that transition. When reconstructing the most likely path, the hard decisions are used, as in Viterbi decoding, starting at the end of trellis (the solid line). For producing soft information, we use the differences between the most likely path and the most likely path when choosing the other path in first step (dashed line). By calculating the differences between the two paths until they converge, a measurement for the reliability of the decision is obtained.

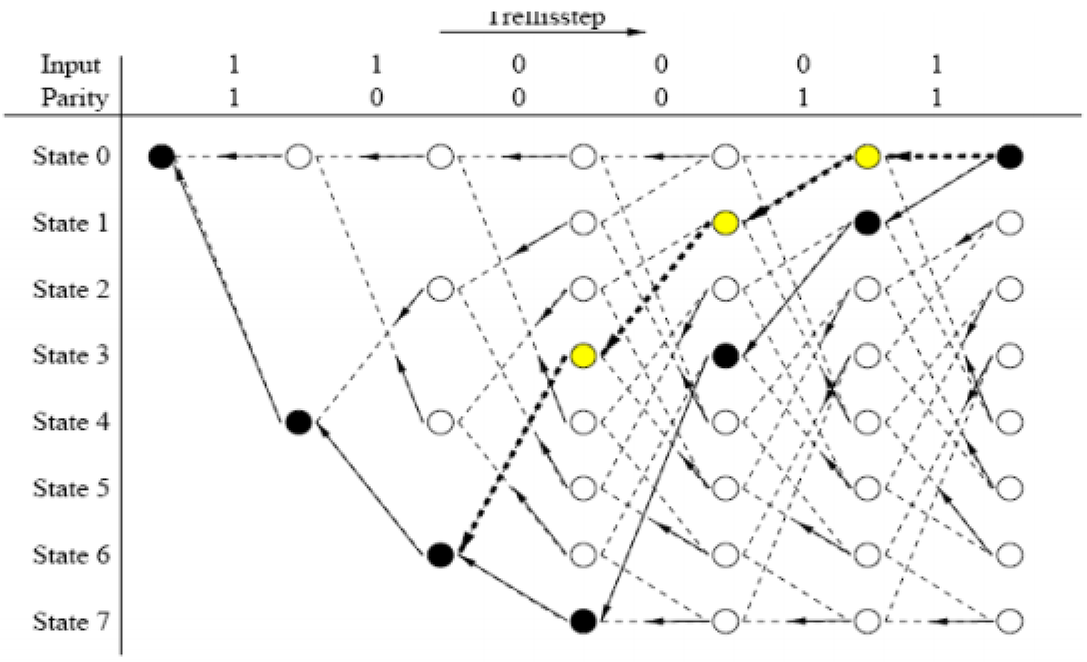
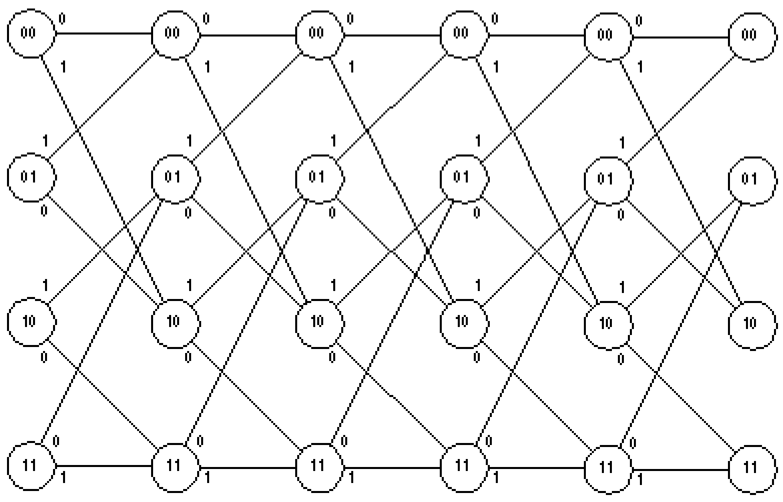


Figure 4.24: The decoding procedure of SOVA

Steps to be Followed during SOVA Algorithm:

Step 1: Form the Trellis

Inside the decoder, Soft-Output Viterbi Algorithm (SOVA) is used to determine the result with maximum likelihood. The process SOVA is similar to that Viterbi algorithm. A trellis is formed first.



The trellis is to shows how the codes are outputted by encoder. The bits in each node represents the states of the encoder. Each line represents a transition on receipt of codes from encoder The encoder output (decoder input) for all combination of states and input of the encoders are summarized as follows.

State 1	State 2	Input	Output	Next State 1	Next State 2	Decoder Input
0	0	0	0	0	0	00
0	0	1	1	1	0	11
0	1	0	0	1	0	00
0	1	1	1	0	0	11
1	0	0	1	1	1	01
1	0	1	0	0	1	10
1	1	0	1	0	1	01
1	1	1	0	1	1	10

Step 2: Determine the Accumulated Maximum Likelihood for Each State

From the above table, each state has its own input bit pattern. When bit streams are inputted to decoder, they can be compared to the input (x_0 x_1) to see whether they are matched. The result is represented by likelihood of input:

L_i = -1 if no bits are matched

0 if 1 bit are matched

1 if all 2 bits are matched

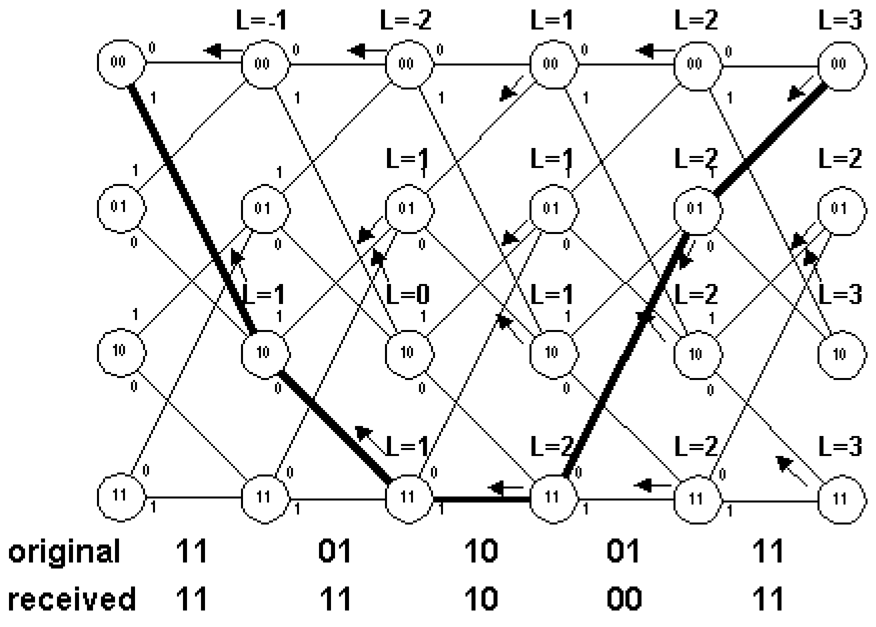
The overall likelihood of a transition is sum of likelihood of input and a-prior likelihood information (L_p).

$L = L_i + L_p$

L_p = 0.5 x a-prior information L_{an} if x_0 are 1

-0.5 x a-prior information L_{an} if x_0 are 0

The algorithm is as follows: Start from state 00, the overall likelihood of each transition are evaluated. The overall likelihood of each node is obtained by the maximum accumulated likelihood. With this algorithm, the following will be obtained.



Step 3: Find the Surviving Path

The surviving path is thus derived by tracing back from last stage (stage 5) to the first one (stage 0) and is the results of hard-output Viterbi Algorithm.

[1 0 1 0 1]

Step 4: Determine the Soft Output

To find the soft-output, non-surviving paths are considered. We consider the non-surviving path the one that is product by making different decision in one of the stage in tracing back. For example, when tracing back from (last stage) stage 5, one of the non-surviving path is found by tracing back to state 00 instead of state 01 from stage 5 to stage 4. Following the arrows, bit 1 is also 1. We found that bit 1 will have the following results when decision is changed in different stages:

Stage	1	2	3	4	5
Bit 1	X	X	0	1	1
Delta	-	-	3	1	2

where X means cannot trace back to state 00 in stage 0

Here we define a function delta to describe the tendency to have non-surviving path. It is the difference in overall likelihood when a different decision in a particular the stage.

The soft-output of bit 1 is evaluated by the following formula:

bit_value x *min*(*delta* which make bit 1 change to value other than *bit_value*)

bit_value = 1 for bit output = 1

-1 for bit output = 0

In bit one, the decision only changes when a state changes in stage 3. The minimum delta is 3. Thus the soft-output of the bit one is 3.

Repeat it for bit two (originally bit two = 0),

Stage	2	3	4	5
Bit 1	X	1	1	1
Delta	-	3	1	2

where X means cannot trace back to state 00 in stage 0

From the above results, in bit two, the decision changes when a state changes in stage 3, 4 and 5. The minimum delta is 1. Thus, the soft-output of the bit two is -1.

Using this algorithm, the soft-output will become

[3 -1 1 -1 2]

Feeding Data to Another Decoder

After the soft-output is evaluated by SOVA decoder, the data will be passed to the second decoder for further decoding. Before passing to data to the second decoder, two processes are performed to the decoder output:

the a-prior information (La_2) and the systematic data (x_0) are subtracted

d_1	3	-1	1	-1	2
La_2	0	0	0	0	0
x_0	1	1	1	-1	1
Le_1	2	-2	0	0	1

the result is multiplied by the scaling factor called channel reliability L_c. The reason of the factor is because the SOVA algorithm suffers a major distortion which is caused by over-optimistic soft outputs. The factor is used to compensate this distortion.

L_c = mean(Le_1) x 2 / var(Le_1)

Iterative Decoding

The data from first decoder L_cL_{e1}, together with the systematic data $a(x_k)$ and code information from of second encoder (y_2), are then fed into the second decoder for decoding, the decoding algorithm is the same as the first one.

After decoding, the output of second decoder is processed in the same way and fed back to the first decoder. The process continues. The number of iterations depends on the designer. Usually, the larger the iteration, the more accurate the data but the longer the time its takes for decoding.

Decision of Output

After iterations of decoding, the decoding results is the sign of the soft-output of the last decoder. Take the example, for the results of first decoder, the output become:

decoder output	3	-1	1	-1	2
result	1	0	1	0	1

which is the same as the input bit stream *u*. ie, the error can be recovered.

Soft output viterbi decoding algorithms is becoming a standard tool in communication receivers. It is necessary for turbo-decoding and turbo-like decoding. It is also used, with considerable performance improvement, in soft-input soft-output (SISO) concatenated decoders system performing such functions as equalization, demodulation, channel decoding, source decoding.

The well-known argument that soft decisions should be used as input values of a decoder is extended to "soft outputs". For binary variables "soft" values are defined as log-likelihood ratios. The "soft-output"-Viterbi algorithm (SOVA) is described in a compact way, as well as a general rule for "soft-in/soft-out" decoding of binary block codes.

The iterative soft output Viterbi algorithm (SOVA) is a sub-optimum algorithm when it is used to decode turbo codes. By normalizing its extrinsic information we get a performance improvement compared to the standard SOVA. In particular, if the extrinsic information is increased in the last decoding iteration, an additional coding gain improvement.

SOVA method using error filters to reduce the complexity of bit reliability determination further than that of the ordinary SOVA method. Error patterns corresponding to each of a handful of dominant i.e., most common error patterns are determined from experimental data. Error filters determine likelihoods of each postulated error pattern.

The Viterbi algorithm outputs hard decision bits for each state whereas the SOVA provides soft output information in form of log likelihood ratio (LLR) that includes a hard decision bit and reliability value, which allows for an iterative process in a turbo decoder.

The 4- bit input and the parity-1 are given as input to the Decoder-1 block. For the Decoder-2 block, one of the inputs is parity-2, which is interleaved parity. So, the other input for the Decoder-2 block should be interleaved input which will produce the interleaved version of actual input information. From this interleaved information, the actual information can be obtained using De-interleaver. The Turbo Decoder contains two identical Decoders and they use Viterbi algorithm as its decoding principle. Viterbi decoding is explained with the following example. Consider the actual Encoded message is corrupted. The received bits in red colour are corrupted bits. These errors are to be corrected by the decoder.

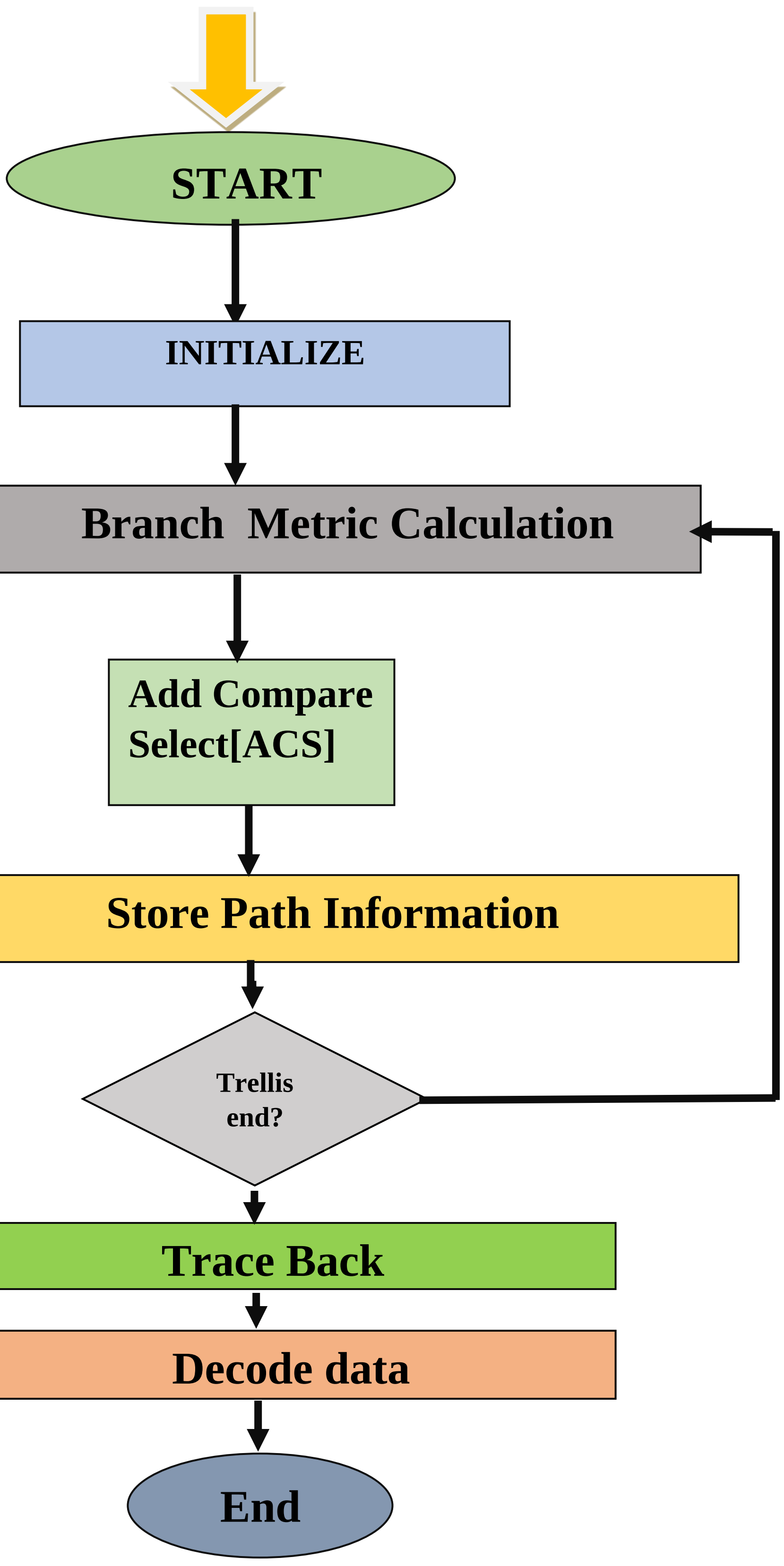


Figure 4.25: Flowchart of Soft Output Viterbi Algorithm

B. SISO Decoder Architecture

The SISO decoder architecture consists of forward and backward state metrics, LLR Computation, and memory (LIFO and FIFO) blocks. To control the flow of input symbol data LIFO and FIFO memory blocks are used. The FIFO 1 and 2 , and the LIFO 1 and 2 are used to buffer the input data symbols. The LIFO 3 and 4 are used to store forward state metric and LLR Values, respectively. The SISO decoder has two backward state metrics units,β1 and β2 , where β1 is used to provide the state metric values into β2, which generates the backward state metric to compute LLR values. The α and γ denote the forward state and branch metric values. The main components of SISO decoder are:

1. Branch and State Metric Unit and

2. LLR Computation Unit.

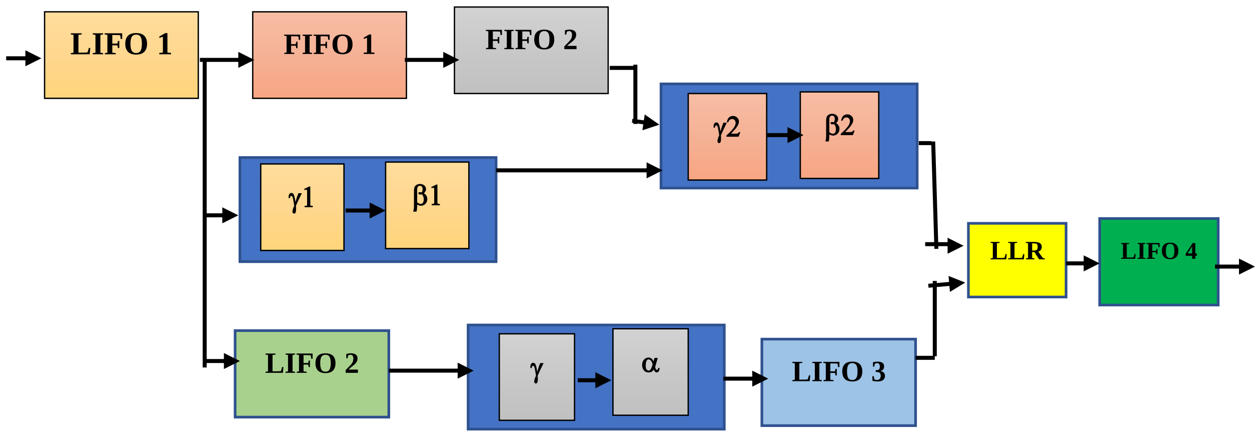


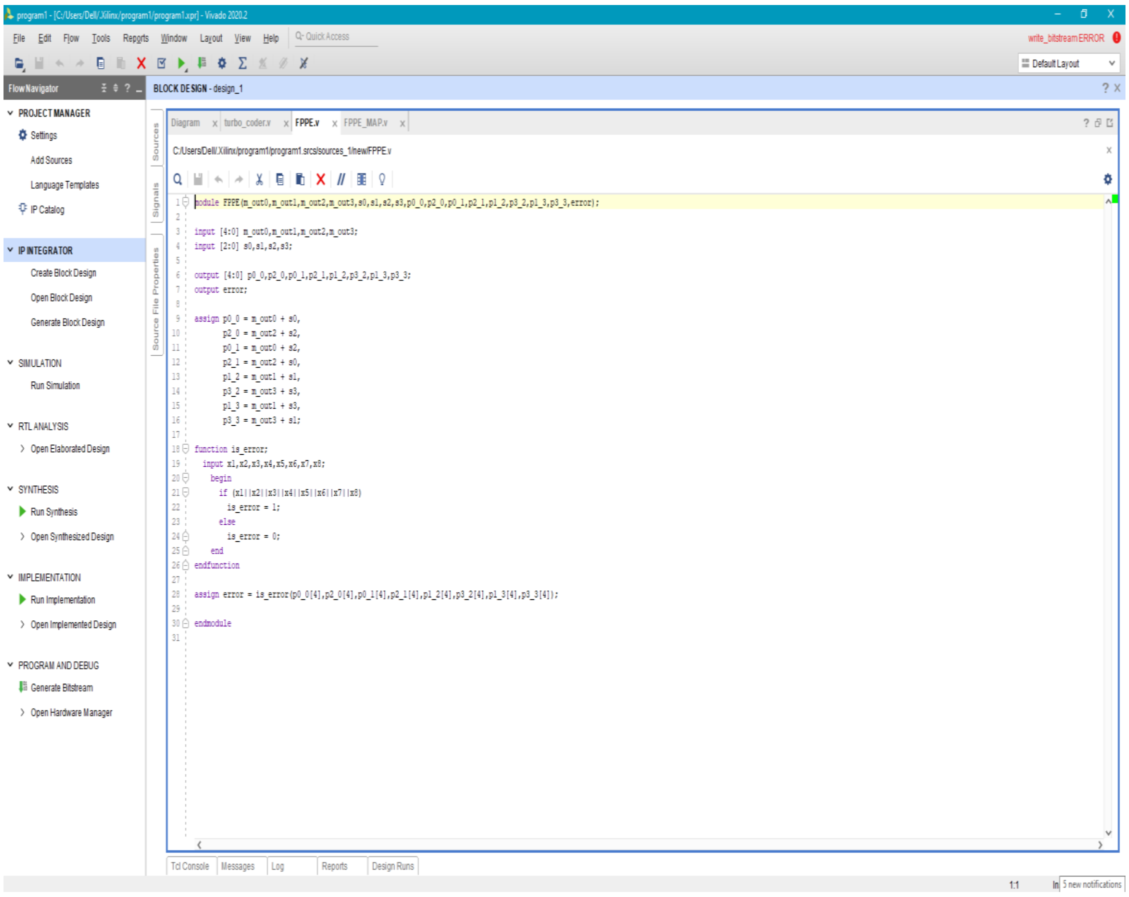
Figure 4.26: Block diagram of SISO Decoder

1. Branch and State Metric Unit: the branch and state metric units (BMU and SMU) are implemented using SOVA algorithm. The convectional BMU and SMU consists of branch metric value computation, add, compare, select, and normalization processes. To avoid overflow of state metric values, the SMU in turbo SISO decoder must include normalization processes.

2. LLR Computation Unit: LLR values are computed using forward and backward states, and branch metric values of all states. The LLR computation unit has a similar architecture to SMU. This architecture reduces the critical path delay and area usage.

4.7 Steps for Implementing Turbo Coder

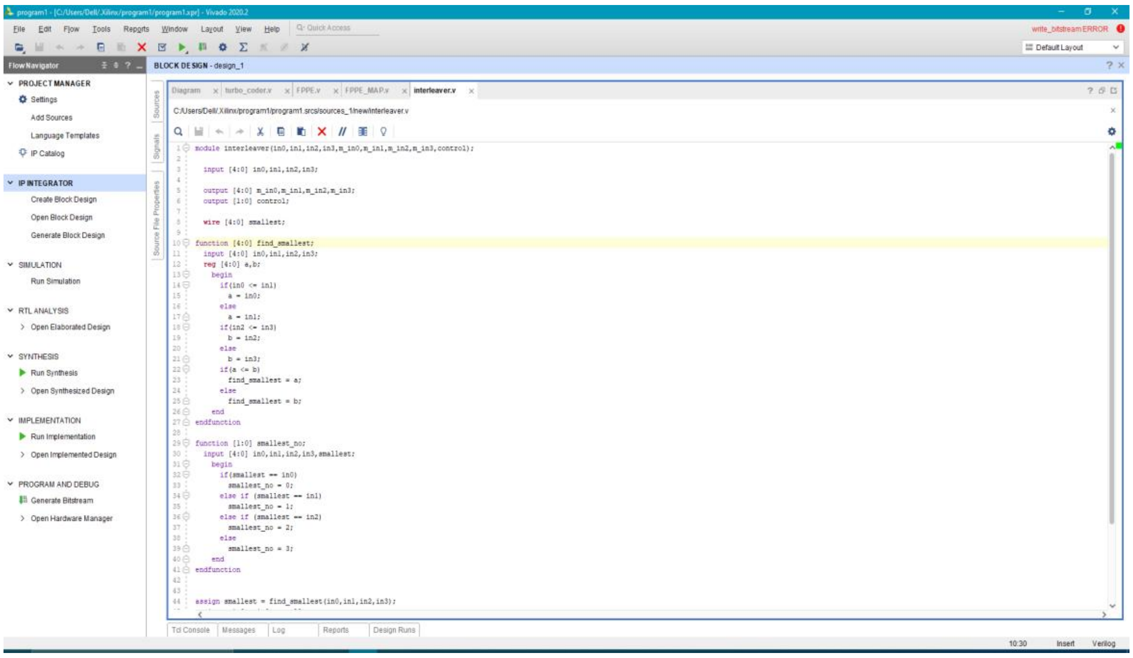
Step 1: Write the Verilog code for Recursive Convolutional Code in Xilinx Vivado 2020.2 version .



C:/Users/Dell/.Xilinx/program1/program1.srcs/sources_1/new/FPPE.v

```
module FPPE(m_out0,m_out1,m_out2,m_out3,s0,s1,s2,s3,p0_0,p2_0,p0_1,p2_1,p1_2,p3_2,p1_3,p3_3,error);

input [4:0] m_out0,m_out1,m_out2,m_out3;
input [2:0] s0,s1,s2,s3;

output [4:0] p0_0,p2_0,p0_1,p2_1,p1_2,p3_2,p1_3,p3_3;
output error;

assign p0_0 = m_out0 + s0,
       p2_0 = m_out2 + s2,
       p0_1 = m_out0 + s2,
       p2_1 = m_out2 + s0,
       p1_2 = m_out1 + s1,
       p3_2 = m_out3 + s3,
       p1_3 = m_out1 + s3,
       p3_3 = m_out3 + s1;

function is_error;
  input x1,x2,x3,x4,x5,x6,x7,x8;
    begin
      if (x1||x2||x3||x4||x5||x6||x7||x8)
        is_error = 1;
      else
        is_error = 0;
    end
endfunction

assign error = is_error(p0_0[4],p2_0[4],p0_1[4],p2_1[4],p1_2[4],p3_2[4],p1_3[4],p3_3[4]);

endmodule
```

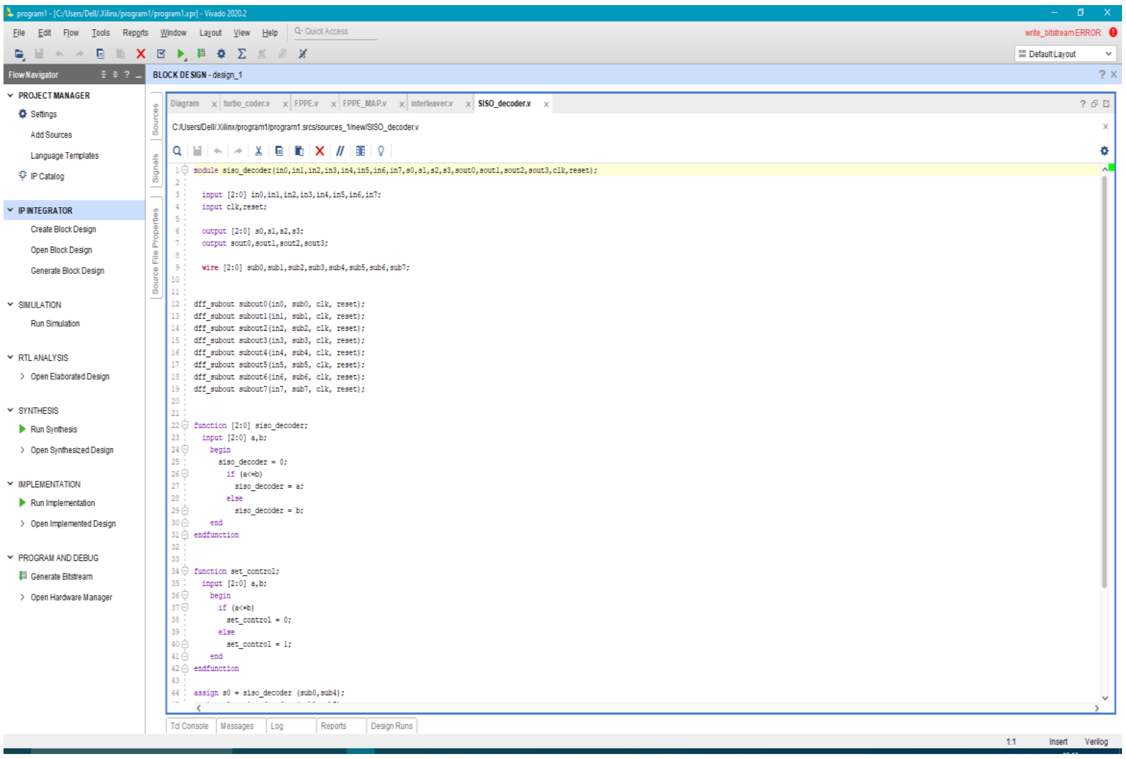
Step 2: Write the Verilog code for Interleaver.



C:/Users/Dell/.Xilinx/program1/program1.srcs/sources_1/new/interleaver.v

```
module interleaver(in0,in1,in2,in3,m_in0,m_in1,m_in2,m_in3,control);

  input [4:0] in0,in1,in2,in3;

  output [4:0] m_in0,m_in1,m_in2,m_in3;
  output [1:0] control;

  wire [4:0] smallest;

function [4:0] find_smallest;
  input [4:0] in0,in1,in2,in3;
  reg [4:0] a,b;
    begin
      if(in0 <= in1)
        a = in0;
      else
        a = in1;
      if(in2 <= in3)
        b = in2;
      else
        b = in3;
      if(a <= b)
        find_smallest = a;
      else
        find_smallest = b;
    end
endfunction

function [1:0] smallest_no;
  input [4:0] in0,in1,in2,in3,smallest;
    begin
      if(smallest == in0)
        smallest_no = 0;
      else if (smallest == in1)
        smallest_no = 1;
      else if (smallest == in2)
        smallest_no = 2;
      else
        smallest_no = 3;
    end
endfunction


assign smallest = find_smallest(in0,in1,in2,in3);
```

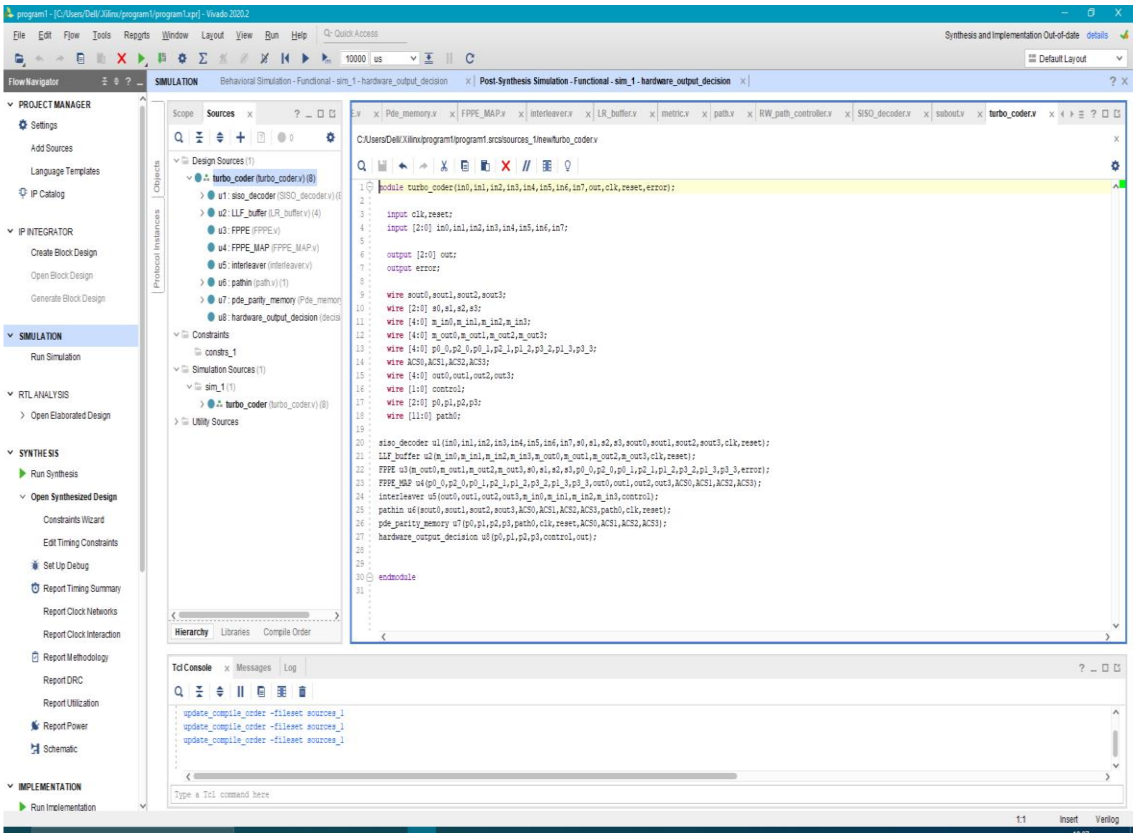
Step 3: Write the Verilog code for SISO Decoder.



C:/Users/Dell/.Xilinx/program1/program1.srcs/sources_1/new/SISO_decoder.v

```
module siso_decoder(in0,in1,in2,in3,in4,in5,in6,in7,s0,s1,s2,s3,sout0,sout1,sout2,sout3,clk,reset);

  input [2:0] in0,in1,in2,in3,in4,in5,in6,in7;
  input clk,reset;

  output [2:0] s0,s1,s2,s3;
  output sout0,sout1,sout2,sout3;

  wire [2:0] sub0,sub1,sub2,sub3,sub4,sub5,sub6,sub7;


dff_subout subout0(in0, sub0, clk, reset);
dff_subout subout1(in1, sub1, clk, reset);
dff_subout subout2(in2, sub2, clk, reset);
dff_subout subout3(in3, sub3, clk, reset);
dff_subout subout4(in4, sub4, clk, reset);
dff_subout subout5(in5, sub5, clk, reset);
dff_subout subout6(in6, sub6, clk, reset);
dff_subout subout7(in7, sub7, clk, reset);


function [2:0] siso_decoder;
  input [2:0] a,b;
    begin
      siso_decoder = 0;
        if (a<=b)
          siso_decoder = a;
        else
          siso_decoder = b;
    end
endfunction


function set_control;
  input [2:0] a,b;
    begin
      if (a<=b)
        set_control = 0;
      else
        set_control = 1;
    end
endfunction

assign s0 = siso_decoder (sub0,sub4);
```

Step 4: Write the Verilog Code for Turbo Coder and call all the subprograms to it.



C:/Users/Dell/.Xilinx/program1/program1.srcs/sources_1/new/turbo_coder.v

```
module turbo_coder(in0,in1,in2,in3,in4,in5,in6,in7,out,clk,reset,error);

  input clk,reset;
  input [2:0] in0,in1,in2,in3,in4,in5,in6,in7;

  output [2:0] out;
  output error;

  wire sout0,sout1,sout2,sout3;
  wire [2:0] s0,s1,s2,s3;
  wire [4:0] m_in0,m_in1,m_in2,m_in3;
  wire [4:0] m_out0,m_out1,m_out2,m_out3;
  wire [4:0] p0_0,p2_0,p0_1,p2_1,p1_2,p3_2,p1_3,p3_3;
  wire ACS0,ACS1,ACS2,ACS3;
  wire [4:0] out0,out1,out2,out3;
  wire [1:0] control;
  wire [2:0] p0,p1,p2,p3;
  wire [11:0] path0;

siso_decoder u1(in0,in1,in2,in3,in4,in5,in6,in7,s0,s1,s2,s3,sout0,sout1,sout2,sout3,clk,reset);
LLF_buffer u2(m_in0,m_in1,m_in2,m_in3,m_out0,m_out1,m_out2,m_out3,clk,reset);
FPPE u3(m_out0,m_out1,m_out2,m_out3,s0,s1,s2,s3,p0_0,p2_0,p0_1,p2_1,p1_2,p3_2,p1_3,p3_3,error);
FPPE_MAP u4(p0_0,p2_0,p0_1,p2_1,p1_2,p3_2,p1_3,p3_3,out0,out1,out2,out3,ACS0,ACS1,ACS2,ACS3);
interleaver u5(out0,out1,out2,out3,m_in0,m_in1,m_in2,m_in3,control);
pathin u6(sout0,sout1,sout2,sout3,ACS0,ACS1,ACS2,ACS3,path0,clk,reset);
pde_parity_memory u7(p0,p1,p2,p3,path0,clk,reset,ACS0,ACS1,ACS2,ACS3);
hardware_output_decision u8(p0,p1,p2,p3,control,out);



endmodule
```

```
update_compile_order -fileset sources_1
update_compile_order -fileset sources_1
update_compile_order -fileset sources_1
```

Step 5: Save all the Programs and generate the Waveforms for Turbo Coder by Running the Simulation. Observe the waveforms.

Step 6: Observe the Schematic of Turbo Coder.

Step 7: Run the Synthesis of Turbo Coder.

Step 8: Run the Implementation of Turbo Coder.

Step 9: Observe the output for each block of Turbo Coder and finally the Turbo Coder is implemented.

Chapter 5

ADVANTAGES AND APPLICATIONS

5.1 Advantages of Turbo Codes

1. Remarkable Power efficiency in AWGN (Additive White Gaussian Noise) and flat fading channels for moderately low Bit Error Ratio (BER).

2. Design tradeoffs suitable for delivery of multimedia services.

5.2 Applications of Turbo Codes

1. Wireless multimedia

- Data: use large frame sizes
  - Low BER, but long latency.
- Voice: use small frame sizes
  - Short latency, but higher BER.

2. Combined equalization and error correction decoding.

3. Combined multiuser detection and error correction decoding.

Chapter 6

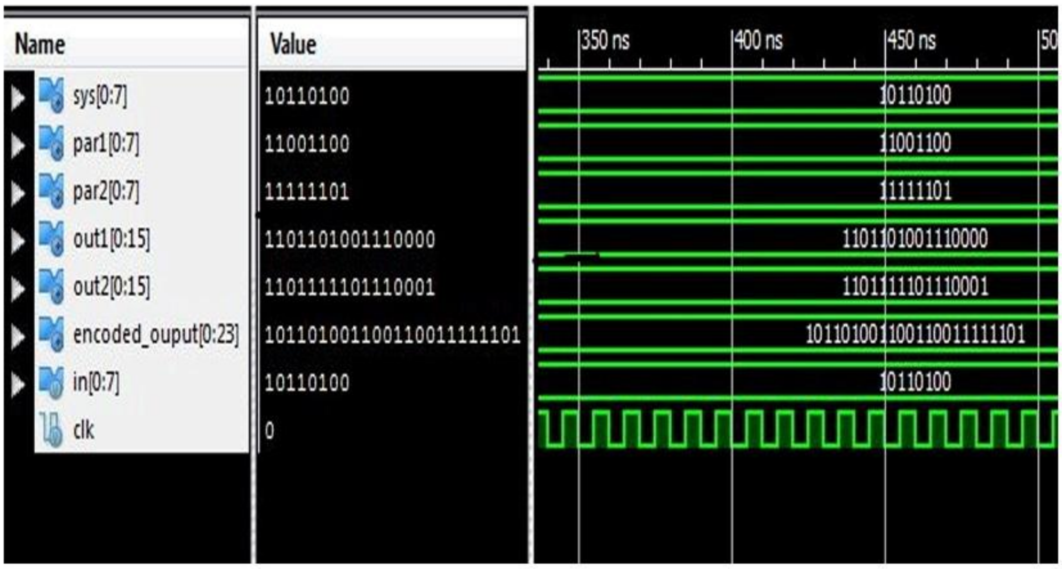
RESULT

The Turbo encoder and decoder simulations are done using Verilog HDL in Xilinx Vivado 2020.2. Verilog Design Suite is a Xilinx based software suite. It may be used for HDL design synthesis and analysis. Recursive Convolutional encoders, Turbo encoder and decoder simulation outputs are performed using Xilinx Vivado.

A) Turbo Encoder

1. Simulation result of Turbo encoder

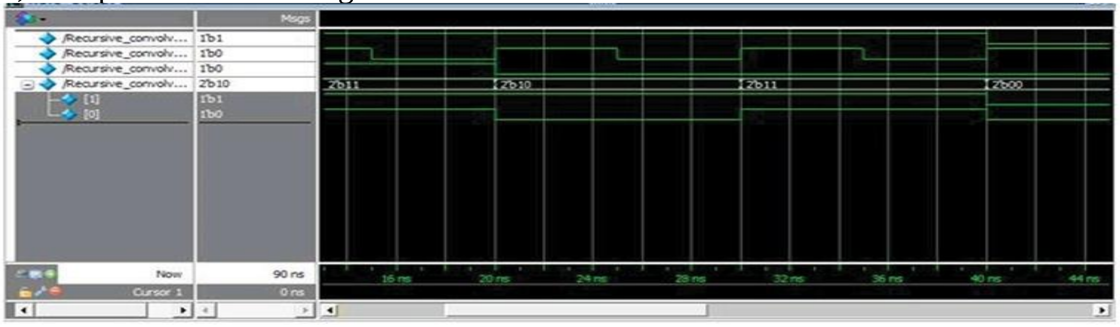
The input given to turbo encoder is '10110100' and output obtained is '10110100 11001100 11111101'. The first 8 output bits are input bits and remaining bits represents parity.



Name	Value
sys[0:7]	10110100
par1[0:7]	11001100
par2[0:7]	11111101
out1[0:15]	1101101001110000
out2[0:15]	1101111101110001
encoded_ouput[0:23]	101101001100110011111101
in[0:7]	10110100
clk	0

Figure 6.1: Simulation of turbo encoder

2. Output of RSC encoder using Xilinx Vivado. ½ rate RSC encoder. That is, for 1 input bit, 2 output bits have been generated.



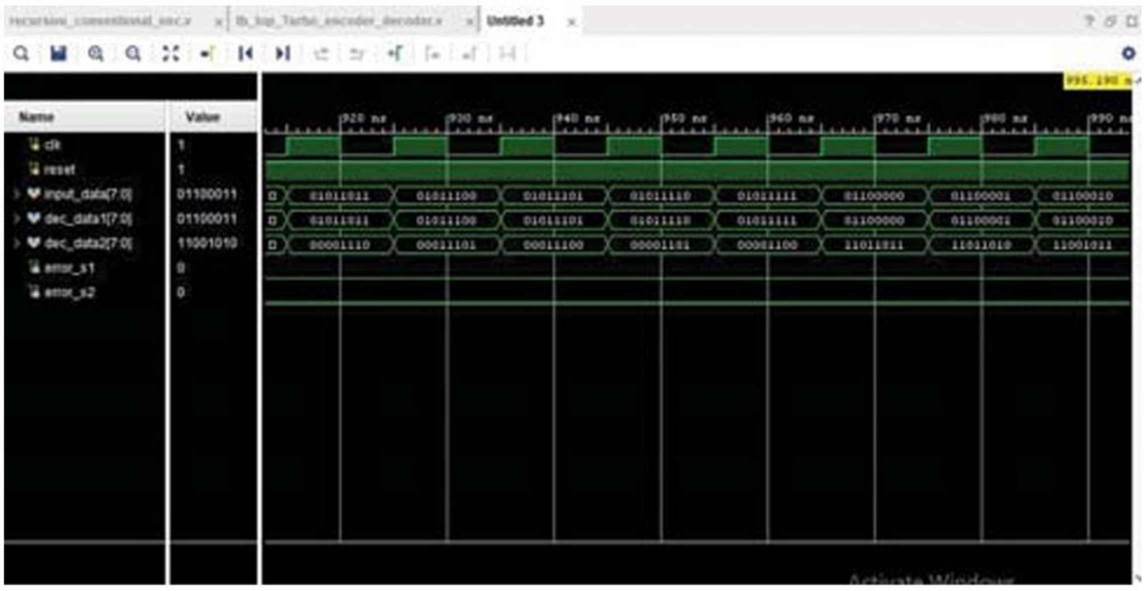
	Msgs
/Recursive_convolv...	1'b1
/Recursive_convolv...	1'b0
/Recursive_convolv...	1'b0
/Recursive_convolv...	2'b10
[1]	1'b1
[0]	1'b0
Now	90 ns
Cursor 1	0 ns

Figure 6.2: Simulation of RSC Encoder

B) Turbo Decoder

1. Simulation output of Turbo Decoder using Vivado.

The two SISO component decoders generate dec_data1 and dec_data2 which is of 8 bits each. Both SISO decoders generate zero errors.



```
Name              Value
clk               1
reset             1
input_data[7:0]   01100011   0  01011011  01011100  01011101  01011110  01011111  01100000  01100001  01100010
dec_data1[7:0]    01100011   0  01011011  01011100  01011101  01011110  01011111  01100000  01100001  01100010
dec_data2[7:0]    11001010   0  00001110  00011101  00011100  00001101  00001100  11011011  11011010  11001011
error_s1          0
error_s2          0
```

Figure 6.3: Simulation Result of Turbo decoder

C) Schematic Diagram of Turbo Coder

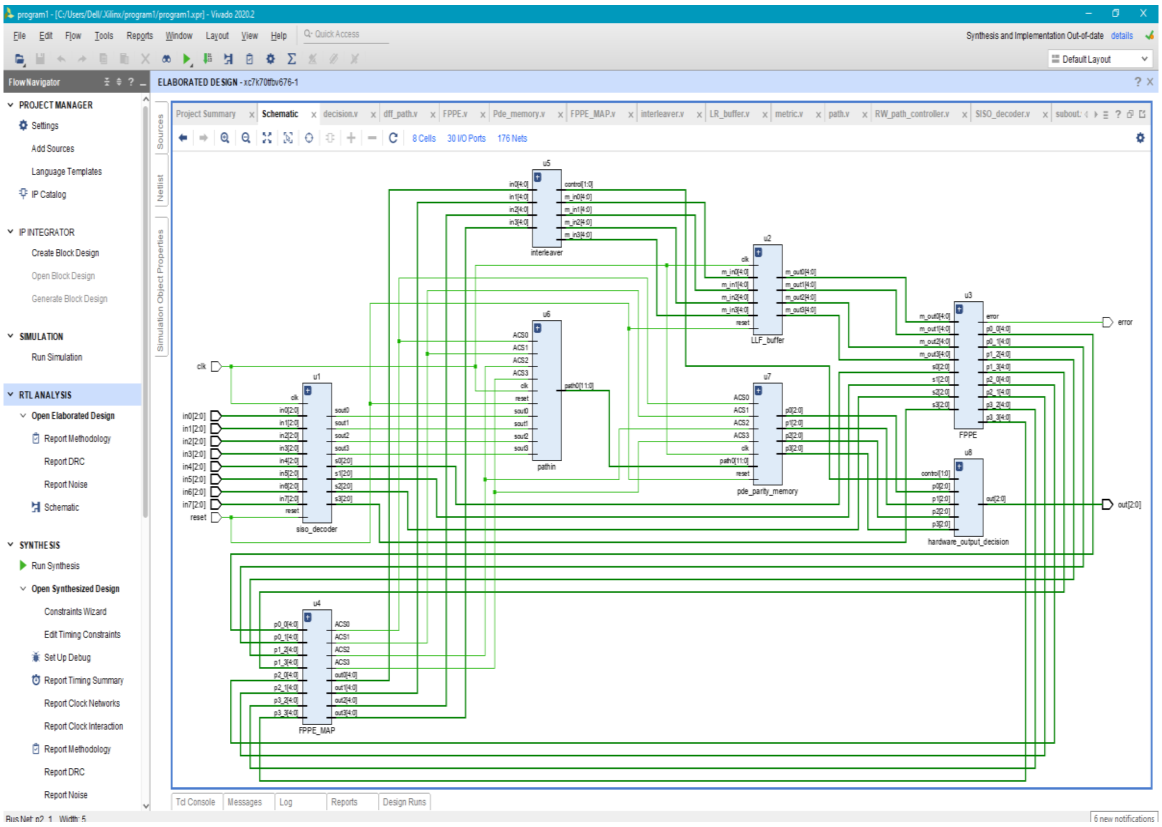


Figure 6.4: Schematic of Turbo Coder

D) Schematic of Turbo Coder after Implementation.

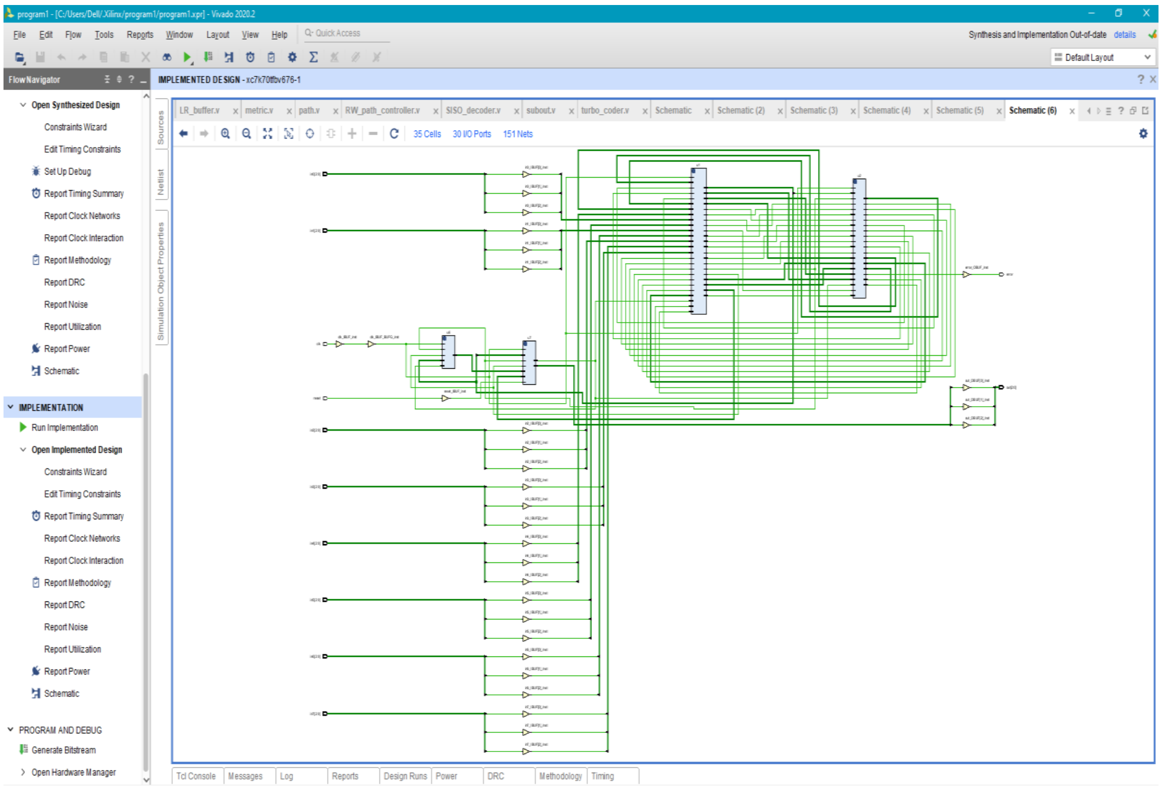


Figure 6.5: Schematic of Turbo Coder after Implementation

Chapter 7

CONCLUSIONS

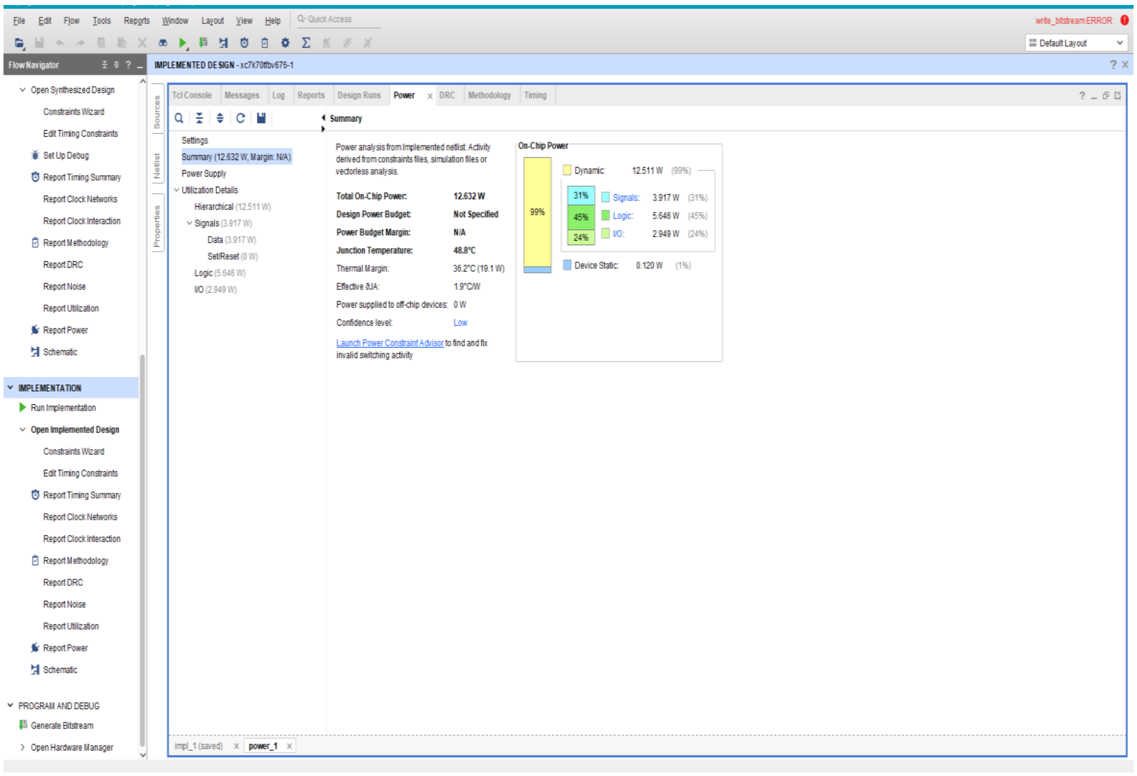
The Turbo encoder and Decoders are designed and simulated in Verilog HDL using Xilinx Vivado 2020.2 version for 8-bit input. Input information bits are given to Turbo encoder that encodes data and then transmitted to Turbo Decoder through channel to obtain 24-bit output data. At decoder, the received data may contain errors, which are decoded using SOVA algorithm to obtain original information. Here, the decoder successfully corrects the error and retrieves the original message. Synthesis is done in Xilinx Vivado 2020.2 and results are tabulated.

7.1 Future Enhancement

In this project, we implemented a turbo encoder and decoder scheme using the SOVA algorithm. The advantage of turbo codes over existing coding schemes is that it attains a very low BER at low signal-to-noise ratios. This makes it suitable for wireless applications where low transmission power is desired. However, the performance of turbo codes on Rayleigh and Ricean fading channels remain an active subject of research. The portability of this code to Advanced Design System (ADS) would be very beneficial for code re-usability and also the synthesis capability of ADS would result in faster product development.

APPENDICES

1. Power Report



IMPLEMENTED DESIGN - xc7k70tfbv676-1

Summary

Settings
Summary (12.632 W, Margin: N/A)
Power Supply
Utilization Details
Hierarchical (12.511 W)
Signals (3.917 W)
Data (3.917 W)
Set/Reset (0 W)
Logic (5.646 W)
I/O (2.949 W)

Power analysis from Implemented netlist. Activity derived from constraints files, simulation files or vectorless analysis.

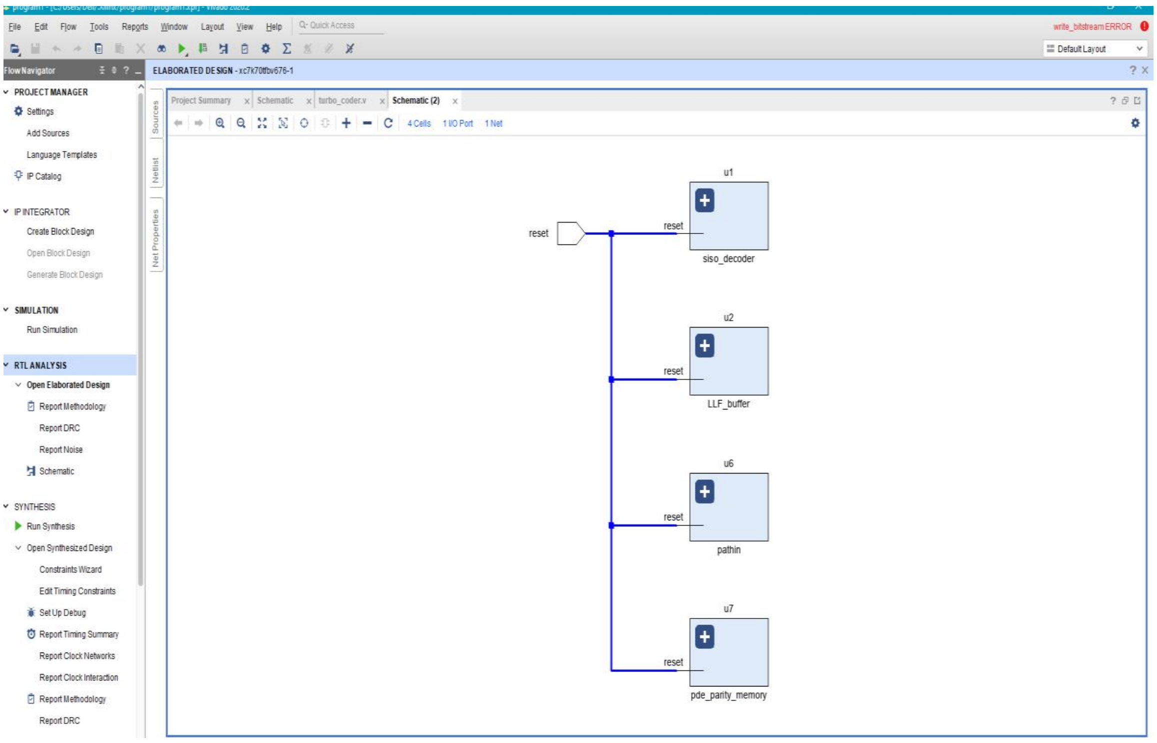
Total On-Chip Power:	12.632 W
Design Power Budget:	Not Specified
Power Budget Margin:	N/A
Junction Temperature:	48.8°C
Thermal Margin:	36.2°C (19.1 W)
Effective ϑJA:	1.9°C/W
Power supplied to off-chip devices:	0 W
Confidence level:	Low

Launch Power Constraint Advisor to find and fix invalid switching activity

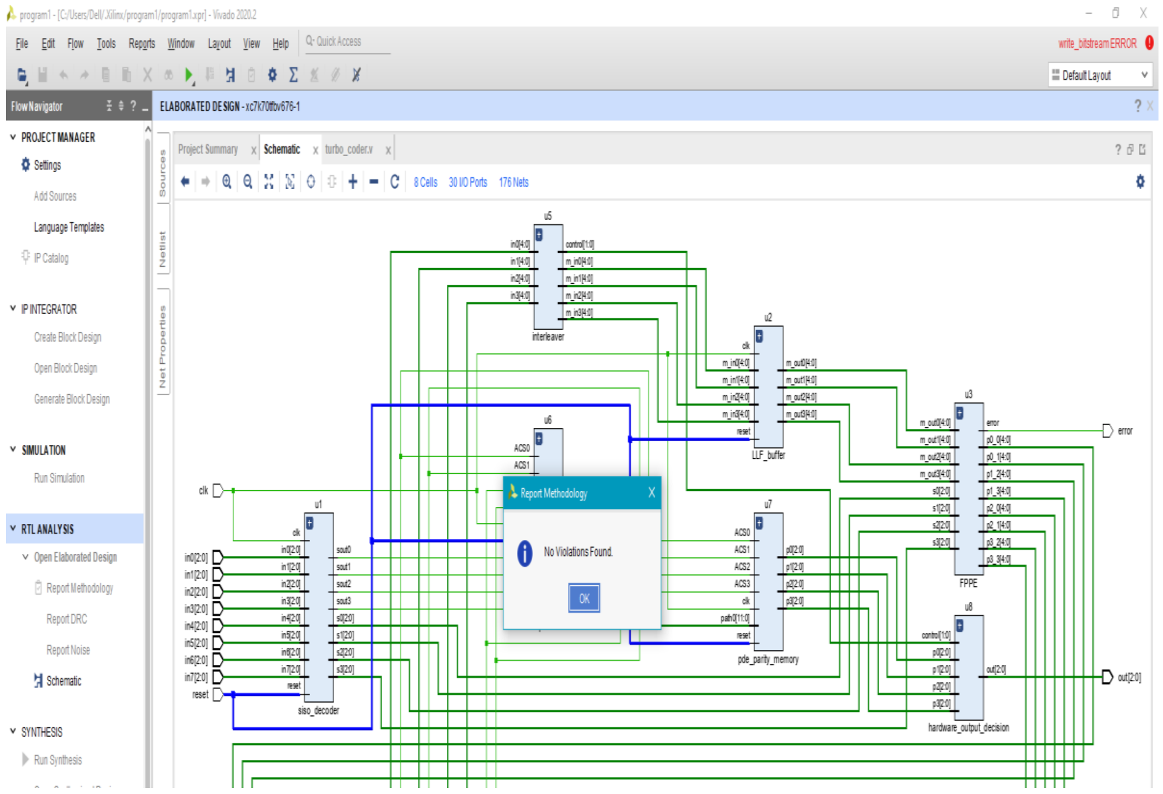
On-Chip Power

Dynamic:	12.511 W	(99%)
Signals:	3.917 W	(31%)
Logic:	5.646 W	(45%)
I/O:	2.949 W	(24%)
Device Static:	0.120 W	(1%)

2. Elaborated Design : Schematic

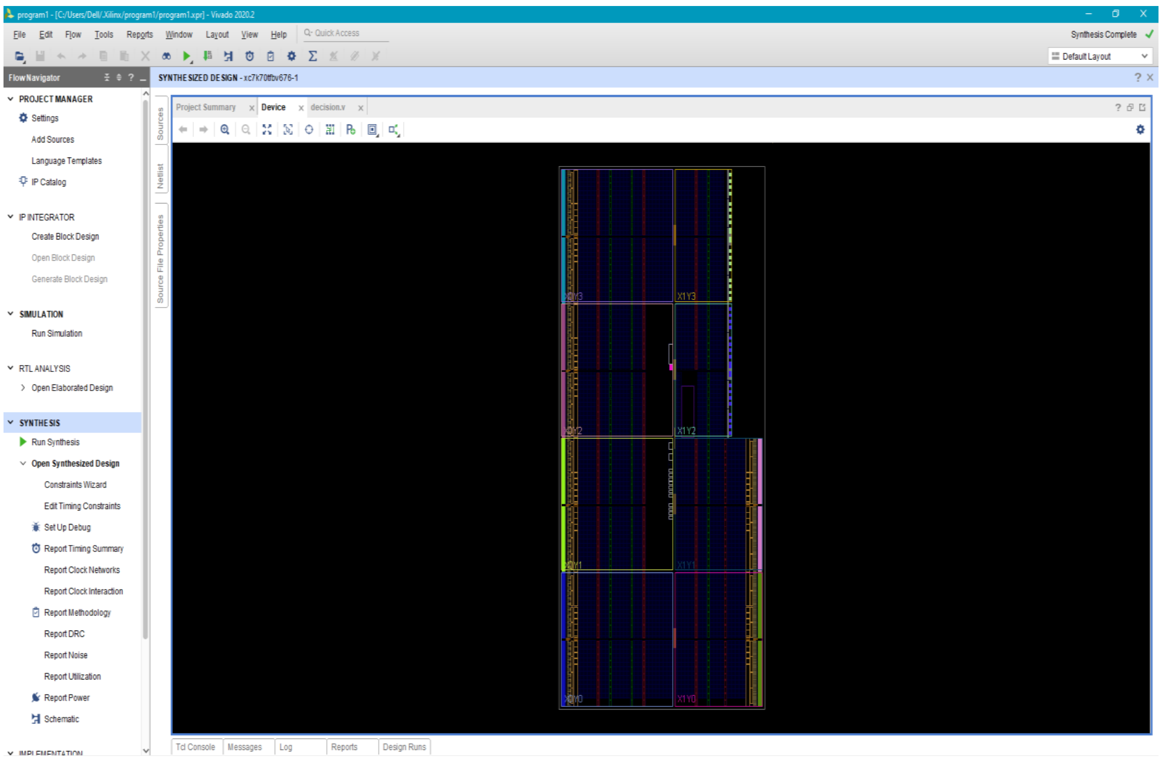


3. Methodology Report

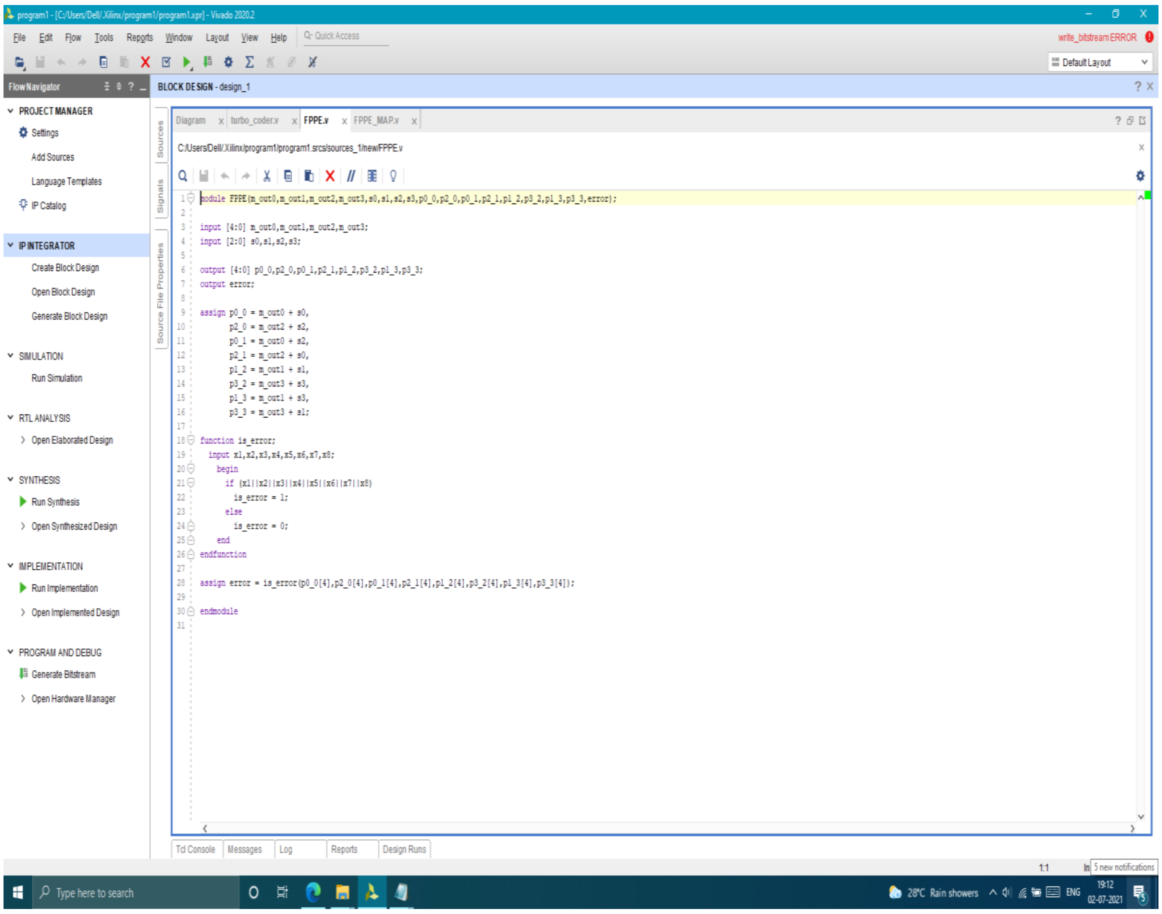

program1 - [C:/Users/Dell/Xilinx/program1/program1.xpr] - Vivado 2020.2
ELABORATED DESIGN - xc7k70tfbv676-1
Report Methodology
OK

No Violations Found.

4. Synthesis Design


program1 - [C:/Users/Dell/Xilinx/program1/program1.xpr] - Vivado 2020.2
SYNTHESIZED DESIGN - xc7k70tfbv676-1
Synthesis Complete

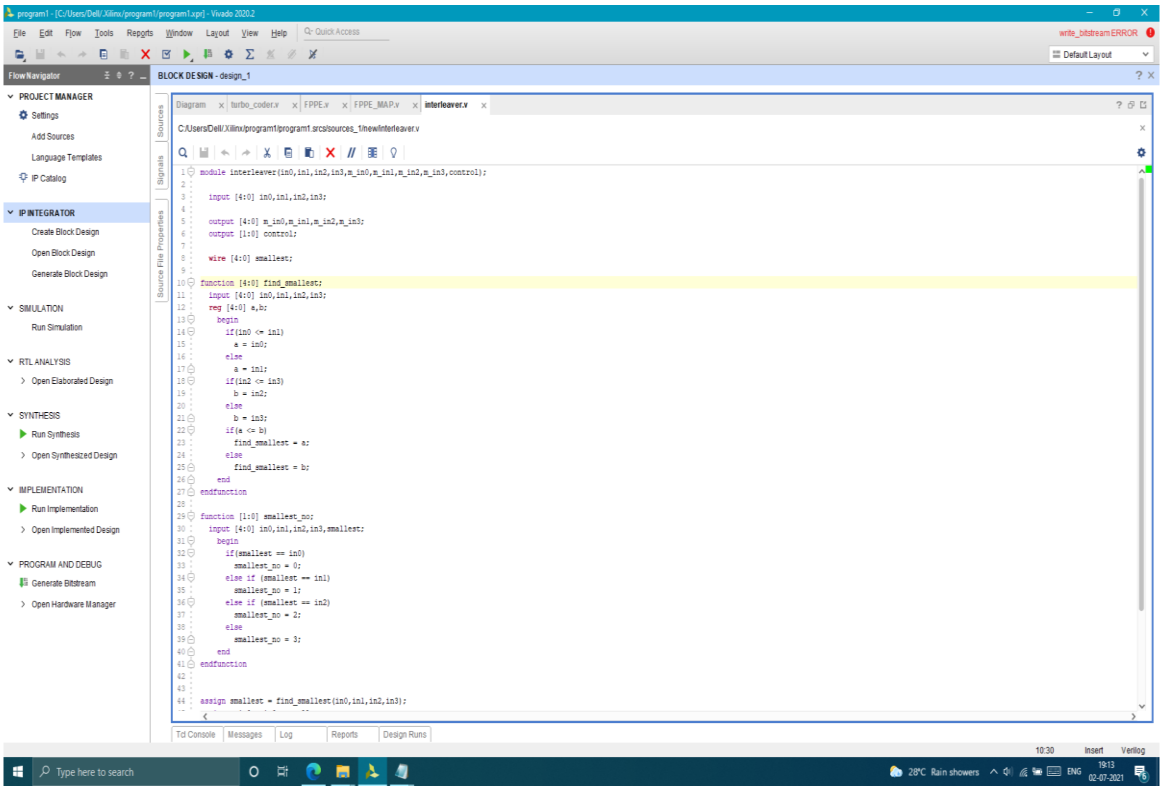
5. Verilog Code for Recursive Convolutional Encoders



C:/Users/Dell/.Xilinx/program1/program1.srcs/sources_1/new/FPPE.v

```
module FPPE(m_out0,m_out1,m_out2,m_out3,s0,s1,s2,s3,p0_0,p2_0,p0_1,p2_1,p1_2,p3_2,p1_3,p3_3,error);

input [4:0] m_out0,m_out1,m_out2,m_out3;
input [2:0] s0,s1,s2,s3;

output [4:0] p0_0,p2_0,p0_1,p2_1,p1_2,p3_2,p1_3,p3_3;
output error;

assign p0_0 = m_out0 + s0,
       p2_0 = m_out2 + s2,
       p0_1 = m_out0 + s2,
       p2_1 = m_out2 + s0,
       p1_2 = m_out1 + s1,
       p3_2 = m_out3 + s3,
       p1_3 = m_out1 + s3,
       p3_3 = m_out3 + s1;

function is_error;
  input x1,x2,x3,x4,x5,x6,x7,x8;
    begin
      if (x1||x2||x3||x4||x5||x6||x7||x8)
        is_error = 1;
      else
        is_error = 0;
    end
endfunction

assign error = is_error(p0_0[4],p2_0[4],p0_1[4],p2_1[4],p1_2[4],p3_2[4],p1_3[4],p3_3[4]);

endmodule
```

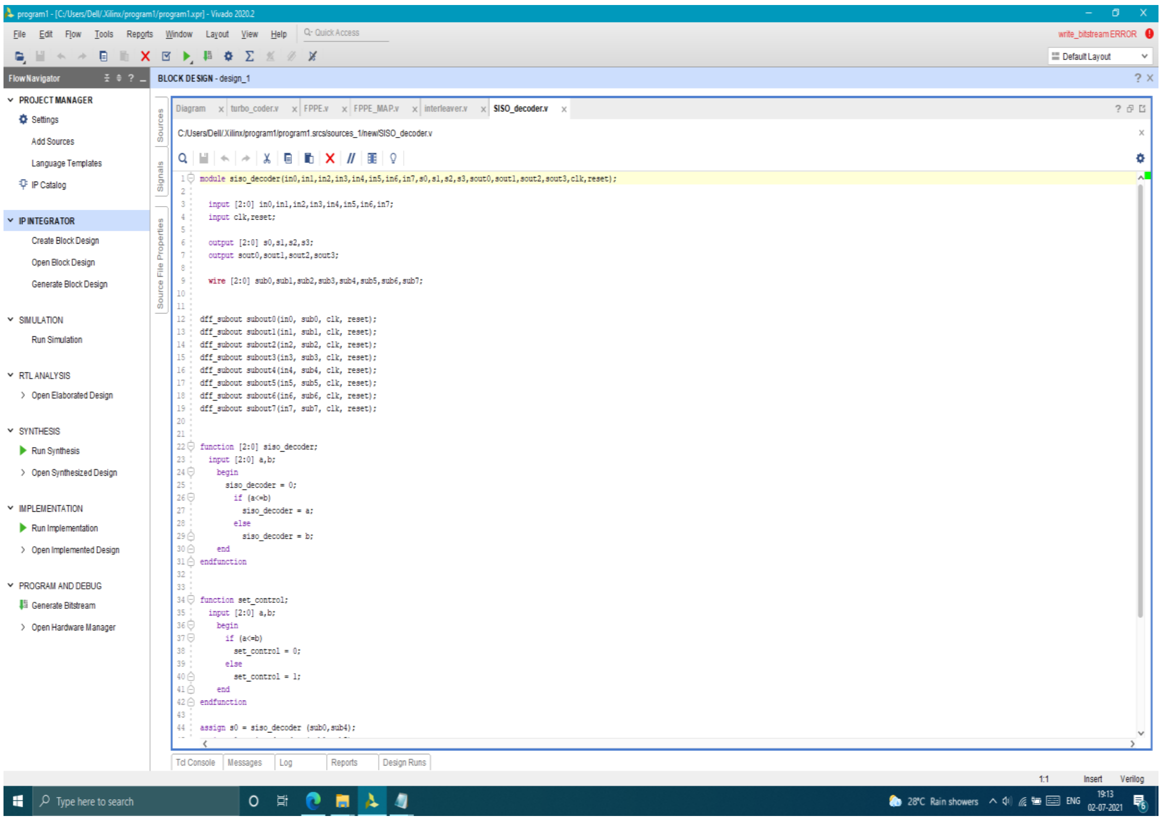
6. Verilog Code for Interleaver



C:/Users/Dell/.Xilinx/program1/program1.srcs/sources_1/new/interleaver.v

```
module interleaver(in0,in1,in2,in3,m_in0,m_in1,m_in2,m_in3,control);

  input [4:0] in0,in1,in2,in3;

  output [4:0] m_in0,m_in1,m_in2,m_in3;
  output [1:0] control;

  wire [4:0] smallest;

function [4:0] find_smallest;
  input [4:0] in0,in1,in2,in3;
  reg [4:0] a,b;
    begin
      if(in0 <= in1)
        a = in0;
      else
        a = in1;
      if(in2 <= in3)
        b = in2;
      else
        b = in3;
      if(a <= b)
        find_smallest = a;
      else
        find_smallest = b;
    end
endfunction

function [1:0] smallest_no;
  input [4:0] in0,in1,in2,in3,smallest;
    begin
      if(smallest == in0)
        smallest_no = 0;
      else if (smallest == in1)
        smallest_no = 1;
      else if (smallest == in2)
        smallest_no = 2;
      else
        smallest_no = 3;
    end
endfunction


assign smallest = find_smallest(in0,in1,in2,in3);
```

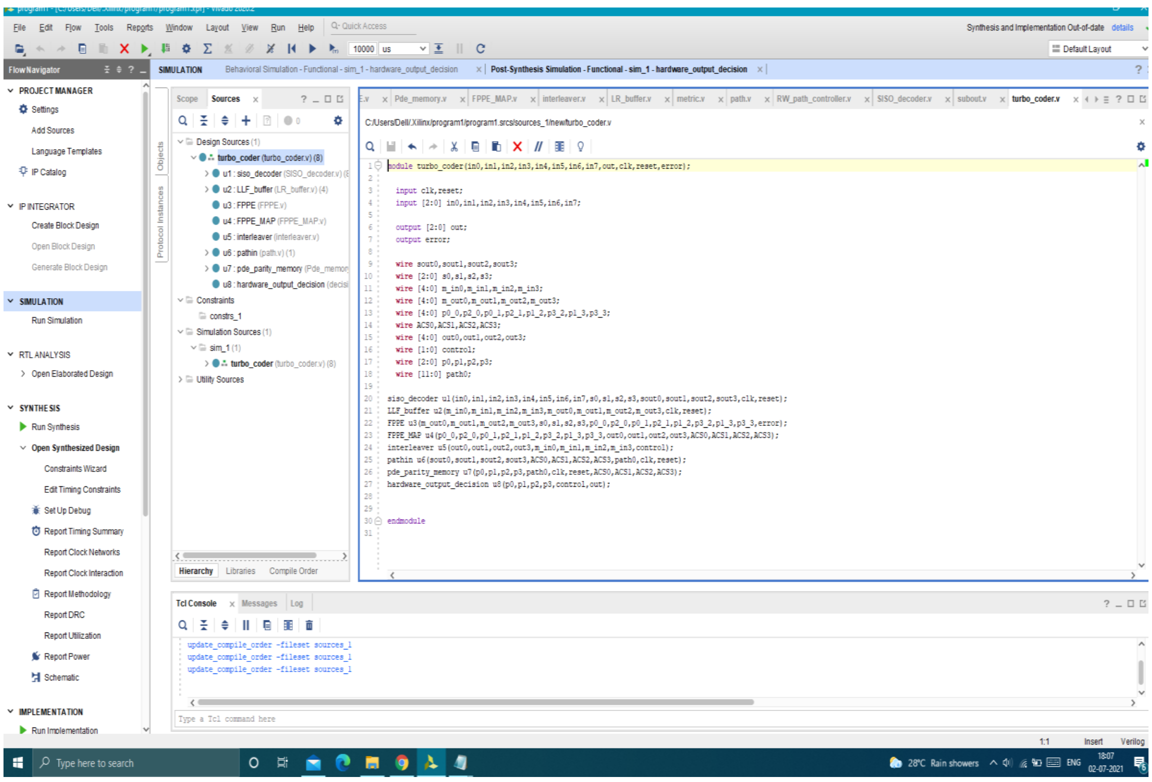
7. Verilog Code for SISO Decoder



C:/Users/Dell/.Xilinx/program1/program1.srcs/sources_1/new/SISO_decoder.v

```verilog
module siso_decoder(in0,in1,in2,in3,in4,in5,in6,in7,s0,s1,s2,s3,sout0,sout1,sout2,sout3,clk,reset);

  input [2:0] in0,in1,in2,in3,in4,in5,in6,in7;
  input clk,reset;

  output [2:0] s0,s1,s2,s3;
  output sout0,sout1,sout2,sout3;

  wire [2:0] sub0,sub1,sub2,sub3,sub4,sub5,sub6,sub7;


dff_subout subout0(in0, sub0, clk, reset);
dff_subout subout1(in1, sub1, clk, reset);
dff_subout subout2(in2, sub2, clk, reset);
dff_subout subout3(in3, sub3, clk, reset);
dff_subout subout4(in4, sub4, clk, reset);
dff_subout subout5(in5, sub5, clk, reset);
dff_subout subout6(in6, sub6, clk, reset);
dff_subout subout7(in7, sub7, clk, reset);


function [2:0] siso_decoder;
  input [2:0] a,b;
    begin
      siso_decoder = 0;
        if (a<=b)
          siso_decoder = a;
        else
          siso_decoder = b;
    end
endfunction


function set_control;
  input [2:0] a,b;
    begin
      if (a<=b)
        set_control = 0;
      else
        set_control = 1;
    end
endfunction

assign s0 = siso_decoder (sub0,sub4);
```

8. Verilog Code for Turbo Coder

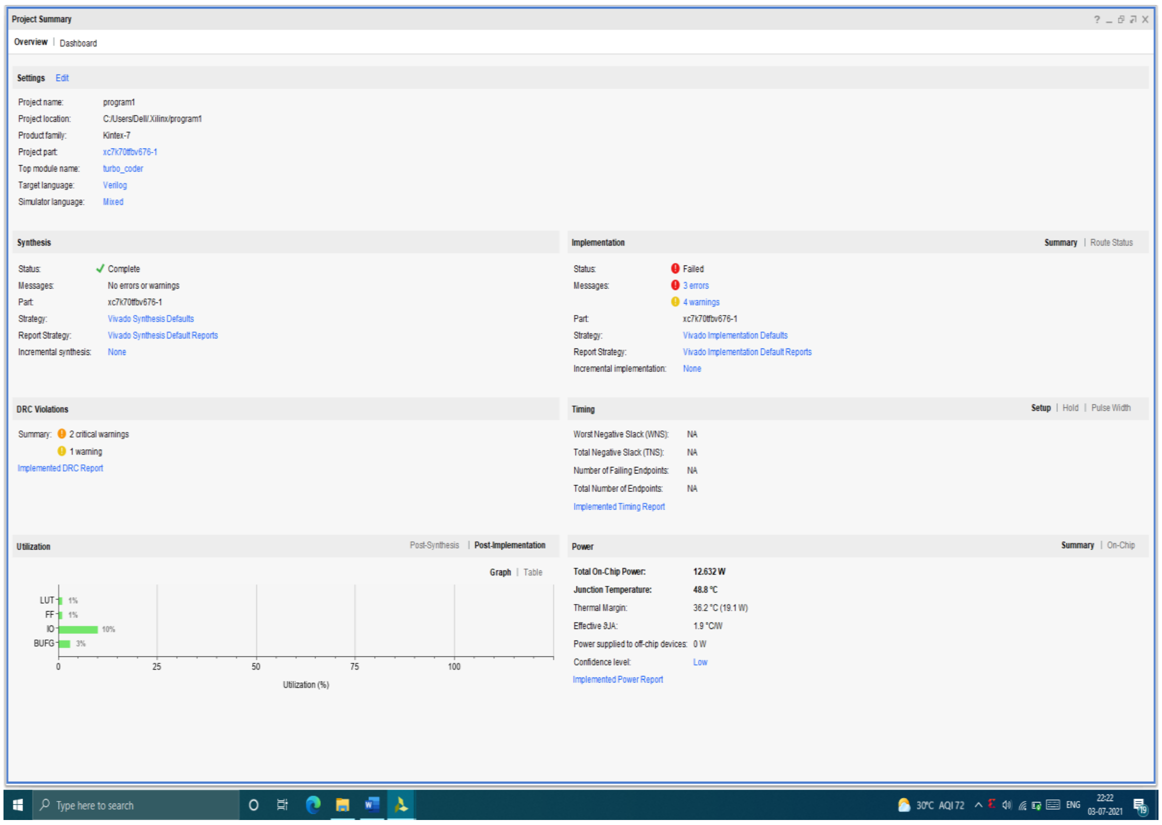


C:/Users/Dell/.Xilinx/program1/program1.srcs/sources_1/new/turbo_coder.v

```verilog
module turbo_coder(in0,in1,in2,in3,in4,in5,in6,in7,out,clk,reset,error);

  input clk,reset;
  input [2:0] in0,in1,in2,in3,in4,in5,in6,in7;

  output [2:0] out;
  output error;

  wire sout0,sout1,sout2,sout3;
  wire [2:0] s0,s1,s2,s3;
  wire [4:0] m_in0,m_in1,m_in2,m_in3;
  wire [4:0] m_out0,m_out1,m_out2,m_out3;
  wire [4:0] p0_0,p2_0,p0_1,p2_1,p1_2,p3_2,p1_3,p3_3;
  wire ACS0,ACS1,ACS2,ACS3;
  wire [4:0] out0,out1,out2,out3;
  wire [1:0] control;
  wire [2:0] p0,p1,p2,p3;
  wire [11:0] path0;

siso_decoder u1(in0,in1,in2,in3,in4,in5,in6,in7,s0,s1,s2,s3,sout0,sout1,sout2,sout3,clk,reset);
LLF_buffer u2(m_in0,m_in1,m_in2,m_in3,m_out0,m_out1,m_out2,m_out3,clk,reset);
FPPE u3(m_out0,m_out1,m_out2,m_out3,s0,s1,s2,s3,p0_0,p2_0,p0_1,p2_1,p1_2,p3_2,p1_3,p3_3,error);
FPPE_MAP u4(p0_0,p2_0,p0_1,p2_1,p1_2,p3_2,p1_3,p3_3,out0,out1,out2,out3,ACS0,ACS1,ACS2,ACS3);
interleaver u5(out0,out1,out2,out3,m_in0,m_in1,m_in2,m_in3,control);
pathin u6(sout0,sout1,sout2,sout3,ACS0,ACS1,ACS2,ACS3,path0,clk,reset);
pde_parity_memory u7(p0,p1,p2,p3,path0,clk,reset,ACS0,ACS1,ACS2,ACS3);
hardware_output_decision u8(p0,p1,p2,p3,control,out);


endmodule
```

```
update_compile_order -fileset sources_1
update_compile_order -fileset sources_1
update_compile_order -fileset sources_1
```

9. Project Summary



Project Summary

Overview | Dashboard

Settings Edit

Project name:	program1
Project location:	C:/Users/Dell/.Xilinx/program1
Product family:	Kintex-7
Project part:	xc7k70tfbv676-1
Top module name:	turbo_coder
Target language:	Verilog
Simulator language:	Mixed

Synthesis

Status:	✓ Complete
Messages:	No errors or warnings
Part:	xc7k70tfbv676-1
Strategy:	Vivado Synthesis Defaults
Report Strategy:	Vivado Synthesis Default Reports
Incremental synthesis:	None

Implementation Summary | Route Status

Status:	Failed
Messages:	3 errors
	4 warnings
Part:	xc7k70tfbv676-1
Strategy:	Vivado Implementation Defaults
Report Strategy:	Vivado Implementation Default Reports
Incremental implementation:	None

DRC Violations

Summary: 2 critical warnings
1 warning
Implemented DRC Report

Timing Setup | Hold | Pulse Width

Worst Negative Slack (WNS):	NA
Total Negative Slack (TNS):	NA
Number of Failing Endpoints:	NA
Total Number of Endpoints:	NA

Implemented Timing Report

Utilization Post-Synthesis | Post-Implementation

Graph | Table

Resource	Utilization (%)
LUT	1%
FF	1%
IO	10%
BUFG	3%

Power Summary | On-Chip

Total On-Chip Power:	**12.632 W**
Junction Temperature:	**48.8 °C**
Thermal Margin:	36.2 °C (19.1 W)
Effective ϑJA:	1.9 °C/W
Power supplied to off-chip devices:	0 W
Confidence level:	Low

Implemented Power Report

REFERENCES

[1] V. Kavinilavu1, S. Salivahanan, V. S. Kanchana Bhaaskaran2, Samiappa Sakthikumaran, B. Brindha and C. Vinoth ***“Implementation of Convolutional Encoder and Viterbi Decoder using Verilog HDL”***, IEEE 3rd International Conference on Electronics Computer Technology,2011

[2] Tepoju Vivek Vardhan, Bandi Neeraja, Boya Pradeep Kumar, Chandra Sekhar Paidimarry ***“Implementation of Turbo Codes Using Verilog HDL and Estimation of Its Error Correction Capability”***, IEEE Asia Pacific Conference on Postgraduate Research in Microelectronics and Electronics (PrimeAsia),2015

[3] Cagri Tanriover, Bahram Honary, Jun Xu, and Shu Lin**,** ***“Improving turbo code error performance by multifold coding”***, IEEE Communication letters, VOL. 6, NO. 5, MAY 2002

[4] Yaqin Shi, Ming Zhan, Jie Zeng ***“FPGA Implementation and Power Estimation of a Memory-Reduced LTE-Advanced Turbo Decoder”***, IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData), 2018.

[5] Arash Ardakani, Mahdi Shabany, ***“An Efficient Max-Log MAP Algorithm for VLSI Implementation of Turbo Decoders”***, IEEE International Symposium on Circuits and System (ISCAS), 2015

[6] Claude Berrou, Ramesh Pyndiah, Patrick Adde, Catherine Douillard and Raphaël Bidan, ***“Application of turbo codes”***, IEEE 2005

[7] Aswathy Narayanan; Senthil Murugan; Ramesh Bhakthavatchalu, ***“Low Latency Max Log MAP based Turbo Decoder”***, International Conference on Communication and Signal Processing (ICCSP),2019

[8] Wang Huahua, Liu Wenwen, ***“Analysis of turbo decoding algorithm in LTE systems”***, Project of the National Science and Technology Major Special: “LTE wireless comprehensive test instrument development 2012 IEEE

[9] Moeed Israr and Tad Kwasniewski, ***“Digital IC design of turbo codes”***, 9th International Database Engineering & Application Symposium (IDEAS’05) 2005 IEEE

[10] Bashar Tahir, Stefan Schwarz, and Markus Rupp ***“BER Comparison Between Convolutional, Turbo, LDPC, and Polar Codes”***, IEEE 24th International Conference on Telecommunication (ICT), 2017

[11] Aldrin Claytus Vaz, C Gurudas Nayak and Dayananda Nayak ***“Performance Comparison between Turbo and Polar Codes”***, 3rd International Conference on Electronics, communication and Aerospace Technology (ICECA), 2019