EFFICIENT DESIGN OF ITERATIVE DECODING ALGORITHM AND ARCHITECTURE FOR CHANNEL CODING

A THESIS

SUBMITTED IN FULFILLMENT OF THE REQUIREMENT

FOR THE AWARD OF DEGREE OF

DOCTOR OF PHILOSOPHY

IN

ELECTRONICS AND COMMUNICATION ENGINEERING

by:

SHIVANI PASRICHA

(950806015)

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

THAPAR UNIVERSITY

2016

CERTIFICATE

I, **ShivaniPasricha** hereby declare that the thesis entitled, **"Efficient design of iterative decoding algorithm and architecture for channel coding"** submitted to Thapar University, Patiala, in partial fulfillment of the requirements for the award of the Degree of **Doctor of Philosophy in Electronics and Communication Engineering** is a record of original and independent research work done by me during 2009-2016 under the supervision and guidance of **Dr. Sanjay Kumar**, Professor and Head of Department, Department of Electronics and Communication Engineering, Thapar University, and it has not formed the basis for the award of any Degree/Diploma/Associateship/Fellowship or other similarly title to any candidate of any university.

Shivani Pasricha

SHIVANI PASRICHA

(Signature of Candidate) Date:

This is to certify that above statement made by the candidate is correct to the best of my knowledge.

Dr. Sanjay Kumar,

Guide, Professor and Head of Departmentof ECED,

Thapar University, Patiala.

Patiala.

ABSTRACT

The demand for information exchange is ever increasing for most of the communication models. The communication systems are designed and developed for transmitting the information from source to a remote destination. The major purpose for designing the communication system is the accurate and error-free transmission of information between source and destination. The symbols produced by the information source are given to the source encoder. The source encoder assigns code words to the symbols. For each distinct symbol, there is a unique codeword.Errors may be introduced while transmitting these binary sequences over the channel due to noise and interference.At the receiver end, some sort of decoder is used to perform the reverse operation of source encoder. The channel decoder provides decoded output which is converted into a sequence of symbols.

Errors may be introduced while transmitting the binary sequences over the channel due to noise and interference. Channel coding is done to avoid these types of errors. Some redundant binary bits are added to the input sequence by the channel encoder according to some pre-defined logic. The channel decoder at the receiver end reconstructs error-free accurate bit sequence, thus reducing the effects of channel noise and distortion. The error-control strategies are used by the channel encoder and decoder at the receiver to reduce the errors in the received signals. For example, forward error correction (FEC) or automatic repeat request (ARQ) techniques are used to correct errors and increase the reliability of the received signals. This thesis discusses the error control aspects of a communication system, in particular, the error-correcting Turbo codes and LDPC code.

Turbo codes, introduced in 1993, are designed by combining two or more relatively simple component codes to achieve a large coding gain. The information in turbo codes is encoded twice but in a different order. Recently, focus has turned on iterative decoding of turbo codes using “soft-in/soft-out” decoding scheme. The decoding of a complex and long code is

broken up in steps and the decoding steps are interlinked so that the information is not lost while decoding. This thesis presents realization of simplified VLSI architectures for turbo decoders using QPP and S-random Interleaver and also evaluates different VLSI architectures for 3GPP LTE/LTE-advanced turbo decoders for trade-offs in terms of throughput and area requirement.

The most significant concern for Turbo codes is to efficiently implement a turbo decoder. A simplified and efficient implementation of a turbo decoder using field programmable gate array (FPGA) technology has been presented in this thesis. An integer-based Turbo decoder has been designed based on the standard 2"s complement number system by analyzing issues of dynamic range and truncation effect. The integer-based Turbo decoder has been efficiently implemented by modifying the algorithm, applying integer arithmetic, and managing the compact hardware. The Max-Log-MAP decoding algorithm is used to modify the branch metric by weighing a-priori value for significantly improving the BER. The Turbo decoder generates 7-bit soft-decisions by using 8-level integer inputs, determines metrics on integers, and avoids complex floating point or fixed-point arithmetic. The memory address is manipulated to eliminate delay associated with interleaving and de-interleaving and to provide improvement in the throughput. Additionally, the Turbo decoder is implemented in a single-decoder structure by efficiently using the memory and logic cells. Verilog hardware description language has been used to describe the Turbo decoder. XCV300E FPGA chip has been used to implement the Turbo decoder. The design used approximately 28 RAM blocks out of 48 total RAM blocks and 3447 cells out of 6912 logic cells in the device. No additional external components were required in the design. Further, the power consumption of the designed device during normal operation was approximately 695 mW. FPGA implementation of turbo decoder with 8 iterations can operate at a frequency of and give throughput of more than 1 Mbps.

Different VLSI architectures have also been analyzed for 3GPP LTE/LTE-advanced turbo decoders for trade-offs in terms of throughput and area requirement. The performance of SWSISO MAP (maximum a posteriori)turbo decoder, standard SISO MAP turbo decoder, and PW SISO MAP turbo decoder have been analyzed on the basis of data flow graphs. Two different variants of quadratic permutation polynomial (QPP) interleaver have been proposed for simplifying complexity of "mod" operator implementation and providing best trade-off between power dissipation, delay, and area. Also, a turbo decoder with one variant of QPP interleaver has been implemented and analyzed. A novel technique for optimizing the area has been proposed to reduce the number of interleavers required for implementing a parallel window turbo decoder using multi-port memory. The circuit-level retiming and pipelining techniques have been used to improve the throughput without increasing the area complexity. The proposed decoders have been simulated using ModelSim 10.1 student edition and synthesized using Synopsys Design Compiler tool version D-2010.03-SP1. The synthesis results show an increased throughput of advanced decoders in comparison with the standard decoder, but at the same time, there is an increase in power and area requiremente. Considerable amount of area can be saved by applying an improved interleaving technique to PW decoders saves. Simulation results corresponding to throughput and latency for different architectures of turbo decoders have been shown in the work.

Interleaving is normally used in digital communication and storage systems to improve the performance of forward error correcting codes. For turbo codes, an interleaver is an essential constituent and its appropriate design is decisive for superior performance. Quadratic permutation polynomial (QPP) interleaver is a contention free interleaver that is appropriate for parallel turbo decoder realization. A new interleaver design, an alternative of QPP interleaver of turbo codes, has been suggested which permutes a series of bits with the identical statistical properties as a conformist QPP interleaver and performs superior than the

predictable QPP. New proposed architecture has been simulated and synthesized using Xilinx and HDL Designer tools. Very large scale amalgamation of architecture for the proposed interleaver has been investigated in terms of area, delay and power dissipation. Thermal power indulgence and device operation has been computed for the novel plan using QuartusII (32-bit) tool. Also, a contrast among the proposed alternative of QPP interleaver and the conformist QPP interleaver has been presented.

Low-Density parity-check (LDPC) codes were proposed by Robert Gallager in 1963 as error correction codes. These codes allow communicating over the noisy channels near the Shannon capacity limits. LDPC codes were overlooked for thirty years due to their high computational complexity for hardware technology at that time. Then LDPC codes were rediscovered by MacKay and Neal in 1990. LDPC codes are represented using the matrices or with a connected set of graph called as tanner graph. Tanner graph is used for mapping a parity check matrix with the nodes. In the tanner graph, the number of columns in the matrix is equal to number of variable nodes and the number of rows is equal to number of check nodes.

Message-passing decoding iterative algorithms are used at the decoder end of LDPC codes. The messages are exchanged between the variable nodes and the check nodes in discrete time steps of the message-passing decoding iterative algorithms. This process is continued till a specified number of iterations are processed. Several message-passing decoding algorithms have been proposed, among which the algorithm which provides the best performance is belief propagation (BP) algorithm.

In this thesis LDPC decoder has been realized using partially parallel decoder architecture and simplified soft-decision log-belief propagation algorithm.The realized architecture possesses high speed and greatly reduces the routing and check node complexity of the decoder. This work presents introduction to LDPC codes and its various decoding algorithms

followed by realization of LDPC decoder by using simplified message passing algorithm and partially parallel decoder architecture. Simplified message passing algorithm has been proposed for trade-off between low decoding complexity and decoder performance. It greatly reduces the routing and check node complexity of the decoder. Partially parallel decoder architecture possesses high speed and reduced complexity. The improved design of the decoder possesses a maximum symbol throughput of 92.95 Mbps and a maximum of 18 decoding iterations. The article presents implementation of 9216 bits, rate-1/2, (3, 6) LDPC decoder on Xilinx XC3D3400A device from Spartan-3A DSP family. The thesis has been organized in six chapters.

Chapter 1 presents the introduction, motivation, formulation of problem, objectives of the thesis, and organization of the thesis.

Chapter 2 deals with the preface to turbo codes and iterative decoding algorithms employed at decoder end of telecommunication system. The chapter also talks about FPGA implementation of the basic and proficient turbo decoder.

Chapter 3 evaluates various VLSI designs for 3GPP LTE/LTE advanced turbo decoders for comparison in terms of output and area constraints.

Chapter 4 discussesa new proposed interleaver blueprint which is an alternative of QPP interleaver, which permutes a series of bits with identical statistical allocation as a conformist QPP interleaver and executes better than conformist interleaver for turbo codes.

Chapter 5 presents preamble to LDPC codes and its different decoding algorithms followed by the understanding of LDPC decoder with the help of basic message passing algorithm technique and to some extent parallel decoder architecture design. Basic message passing algorithm significantly diminishes routing and checks node complication of decoder and also provides comparison among the low decoding complexity and performance of decoder.

Finally, **Chapter 6** concludes the research work. Further, a brief description about the future work/scope will be given as a motivational seed for further research work.

ACKNOWLEDGEMENT

I wish to express my sincere appreciation to those who have contributed to this thesis and supported me in one way or the other during this amazing journey.

First and foremost, I praise and thank our ALMIGTHY GOD whose blessings have bestowed in me the will power and confidence to carry out my Ph.D.

I would like to place my heart-felt and sincere thanks to my research guide, Dr. Sanjay Kumar, M.E., Ph.D., Professor and Head of Department, Electronics and Communication Engineering, Thapar University, Patiala. I feel it a pleasure to be indebted to my guide for his valuable support, advice and encouragement. I owe my gratitude and profound thanks to the doctoral committee members Dr. Rajesh Khanna, Professor, Electronics and Communication Engineering, Thapar University, Patiala and Dr. AmitKohli, Professor, Electronics and Communication Engineering, Thapar University, Patiala for their valuable suggestions and guidance.

I extend my thanks to my Director, Prof. Dr. RekhaAggarwal and Senior Director, Prof. B.P.Singh for giving me this opportunity to pursue Ph.D. I would like to greatly acknowledge with deep sense of gratitude theManagement of Amity School of Engineering and Technology Bijwasan, India andall the faculty members and office assistants in the Department of Electronics and Communication for their continuous motivation, moral and technical support during the research work.

Finally, it is great privilege to express my profound gratitude, deep love and affection to my family who has been with me always in all the difficulties that came across my life. Their love, patience, persistent encouragement, good understanding and prayers enabled me to complete the research work successfully.

ShivaniPasricha

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ACRONYMS

3GPP	3rd Generation Partnership Project
ARQ	Auto Repeat request
AWGN	Additive White Gaussian Noise
BCJR	Bahl, Cocke, Jelinek and Raviv
BPSK	Binary Phase Shift Keying
FEC	Forward Error Correction
FER	Frame Error Rate
FPGA	Field Programmable Gate Array
Kbps	Kilobits per second
LDPC	Low Density Parity Check
LLR	Log-Likelihood Ratio
LOVA	List Output Viterbi Algorithm
LTE	Long Term Evolution
MAP	Maximum A Posteriori
PCCC	Parallel concatenated convolutional code
PLVA	Parallel List Viterbi Algorithm
QPP	Quadratic Permutation Polynomial
RAM	Random Access Memory
ROM	Read Only Memory
SCCC	Serial concatenated convolutional code
SISO	Soft-In-Soft-Out
SISO-APP	Soft in Soft Out - a posteriori probability

SLVA	Serial List Viterbi Algorithm
SNR	Signal-to-Noise Ratio
VHDL	VHSIC Hardware Description Language
WCDMA	Wireless Code Division Multiple Access

Chapter 1
Introduction

1.1 Research Motivation and Contributions

There is an increasing need of information sharing in today‟s era of communication models. The basic function of any communication system is to transmit an input data from the transmitter to the receiver end. The set-up of a preliminary communication system is shown in figure 1.1.

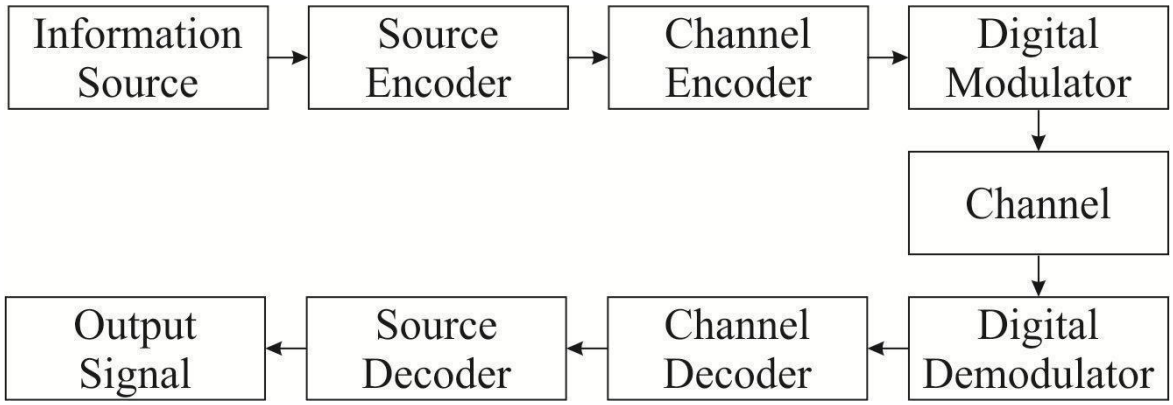


Figure 1.1 Basic Communication System

Input message transfer over the channel is carried out with highest possible data rate and accuracy. At transmitter end, the input source produces symbols which are given to the source encoder. The symbols are converted to code words by the source encoder. There is a distinct codeword for each distinct symbol. The source decoder does the inverse mapping of the source encoder code words, and thus, the binary channel decoder output is transformed back to symbol sequence. Noise in the channel may introduce errors in the binary sequences during communication over the channel. These errors can be avoided by using channel coding techniques. The encoder at the transmitter end for channel coding appends the output of the source encoder with redundant bits according to some algorithm. The decoder for channel coding block at the receiver side performs the inverse operation of the channel

encoder and retrieves the source encoder output. To retrieve the original bit sequence accurately, various error-controlling techniques are used in channel coding.

There is a dire need for authentic and reliable digital data transmission in the present era. Shannon in his paper [1] mathematically explained the communications channel in the presence of noise. He proposed the maximum value of theoretical capacity over a channel and also discussed the existence of channel codes that facilitated the maximum capacity to be achieved. Over a period of time, numerous channel codes which closely approach Shannon‟s limit have come into existence but each of them requires large block length to perform in an efficient manner. At higher values of E_b/N_o these codes provide higher values of coding gain but other parameters such as their cost, latency and complexity makes them impractical to closely approach the limit suggested by Shannon.

Turbo codes were introduced by P. Thitimajshima, Glavieux and C.Berrou [2] as a novel type of error-correcting codes. Initially, concatenated coding scheme [3] was used. The concatenated coding scheme was able to correct errors of comparatively longer codes and achieve comparatively better coding gain by making use of two or more relatively simple constituent codes, though, it had the disadvantage of having high complexity on the receiver side. The analysis techniques used by the turbo codes are different from the normal coding theory techniques and give excellent encoding and decoding performance [4, 5]. Turbo codes achieve this efficiency and closely approach the Shannon‟s limit by making use of two or more constituent codes which work on the input information and its interleaved version. Due to its remarkable properties, turbo codes find use in applications such as 3GPP used in IMT-2000 [6], WCDMA and CCSDS telemetry channel coding [7].

However, the complexity of the circuit and the power consumed by the turbo decoder implementations put a limitation on their use in the power-constrained systems. In the traditional approach, the hard decision value from the demodulator block is passed to the

error control decoder without conveying any information about the reliability of the hard decision passed. Improved results were obtained when the input to the demodulator was taken as quantized analog signal. This resulted in the inception of decoding algorithms with soft input. But this provided optimum results only if one code was used. In case of concatenated codes, new soft output algorithms were developed which provided a real number as an output. The real number provides information about the error probability. In other words it conveys the information about the reliability of hard decision of the decoder. This is quite significant in iterative decoding for the next stage decoding process. The two most widely used soft output decision algorithms are Viterbi algorithm and maximum a posteriori (MAP) algorithm. Viterbi algorithm [8] minimizes the probability of word or sequence error and MAP algorithm [9] minimizes the probability of bit error. Max-Log-MAP algorithm, that is a modified form of MAP, optimally balances performance and complexity [10].

This thesis analysis the Max-Log-MAP algorithm and updates the algorithm by weighing the a priori values with a scaling factor„s". Turbo codes" BER performance is evaluated for different values of scaling factor. Best BER performance is given for s=0.7 [11]. Also, different VLSI implementations of turbo decoder used in 3GPP LTE/LTE-advanced (i.e. decoder in standard form, sliding window technique decoder, parallel window technique turbo decoder, turbo decoder using parallel window technique and QPP interleaver, turbo decoder using parallel window and pipelining technique, turbo decoder using parallel window and retiming technique) have been analyzed for throughput and area requirements by using data flow graphs [13]. The analysis shows that the throughput starts increasing from standard window towards sliding window and parallel window, but the limitation is that simultaneously area requirement also increases proportionally. Circuit level pipelining and retiming techniques increase the throughput in a larger proportion as compared to area

increment. The parallel-window based architecture along with a novel interleaving technique [12], can give optimized area parameters.

Gallager proposed low-density parity-check codes [14]. At the time of their inception, LDPC codes were considered to be very complex so their use was quite limited. It was not until 30 years later that the work of Mackay [15, 16] brought them back to light. Mackay proved that LDPC codes can approach Shannon capacity limit closer than turbo codes [18] with increase in the block length [17]. As compared to the turbo codes, LDPC codes provide less complexity per iteration and error floor at much lower BER. Nowadays, LDPC codes are extensively used in applications such as IEEE 802.11 [21], IEEE 802.3 [19] IEEE 802.16 [20] and DBV-RS2 [22]. Other important characteristics of LDPC codes are that they are highly flexible in choice of code length and rate, they have natural stopping criterion and have good parallel decoding capability.

Based on the message alphabet type, the decoding algorithms for LDPC codes can be classified into hard- decision and soft-decision algorithms. If the alphabet takes binary values, then the algorithm is referred to as hard-decision and if the alphabet consists of infinite number of symbols, it is referred to as soft-decision. Soft-decision algorithms are more complex than hard-decision algorithms. The hard-decision algorithms are simple to implement as the decoding process makes use of binary values only, which further leads to a simplified binary structure and reduced wiring, thus making their hardware implementation simpler.

Belief propagation (BP) [15, 23, and 24] and Weighted Bit Flipping (WBF) algorithms [31] are widely used soft-decision algorithms. As compared to other algorithms, the BP algorithm provides an optimized error rate but its disadvantage is that it is the most complex algorithm. Efforts have been made to reduce the complexity of BP with least effect on its performance. Min-sum (MS) algorithm [28, 30, and 25] is a well-known approximation of BP. The

algorithms with comparatively less complexity have been thoroughly reviewed in [29]. The performance of WBF is not as good as that of BP but they are much simpler than BP and its approximations. In other words, they provide good performance/complexity tradeoff. The family consists of WBF algorithm, MWBF [26] and IMWBF [27]. In this thesis (3, 6) LDPC decoder has been developed with higher decoding output occupying lesser area by using SMPA and partially-parallel decoder architecture. The proposed SMPA algorithm significantly reduces the resource utilization as compared to the BP-algorithm and fully-parallel decoder architecture [32].

1.2 RESEARCH METHODOLOGY

Throughout this thesis, computer simulations have been used to obtain the results. The ADS software has been used to model simulation setup for turbo codes. VHDL language has been used for designing a novel interleaver. Some simulations were performed using ModelSim and synthesis has been done using Precision Synthesis. QuartusII tool (32-bit) has been used for obtaining total thermal P_D and device utilization. Graphs have been used to illustrate the simulation results. Tables have been used to show the interleaved sequence after using interleavers in the decoder structure. For the purpose of illustrating the facts, numerous figures have been included in the thesis.

1.3 THESIS OUTLINE

The thesis has been organized in six chapters.

Chapter 1 presents the introduction, motivation, formulation of problem, objectives of the thesis, and organization of the thesis.

Chapter 2 deals with the preface to turbo codes and iterative decoding algorithms employed at decoder end of telecommunication system. The chapter also talks about FPGA implementation of the basic and proficient turbo decoder.

Chapter 3 evaluates various VLSI designs for 3GPP LTE/LTE advanced turbo decoders for comparison in terms of output and area constraints.

Chapter 4 discusses a new proposed interleaver blueprint, an alternative of QPP interleaver, which permutes a series of bits with identical statistical allocation as a conformist QPP interleaver and executes better than conformist interleaver for turbo codes.

Chapter 5 presents preamble to LDPC codes and its different decoding algorithms followed by the understanding of LDPC decoder with the help of basic message passing algorithm technique and to some extent parallel decoder architecture design. Basic message passing algorithm significantly diminishes routing and checks node complication of decoder and also provides comparison among the low decoding complexity and performance of decoder.

Finally, **Chapter 6** concludes the research work. Further, a brief description about the future work/scope will be given as a motivational seed for further research work.

Chapter 2
Turbo Codes

2.1 Channel Coding

Since C. E. Shannon‟s pioneering work on mathematical theory for digital communications in 1948-49, the subject of error control coding has emerged as a powerful and practical means of achieving efficient and reliable communication of information in a cost effective manner. Suitability of numerous error control schemes in digital transmission systems, wire line and wireless, has been studied and reported in detail in the literature. Shannon‟s noisy channel coding theorem tells that adding controlled redundancy allows data exchange at low error rate

as long as , where and stand for the rate and capacity of the channel respectively. Channel coding does detection and correction of errors by making use of this controlled redundancy. Channel coding or Error Correction Coding technique depends on the system parameters and the channel type. The main function of channel codec is to maintain the reliability of message during transmission while reducing the required transmit power – coding gain and to have faithful reproduction of information at the receiving end. An error control code also facilitates the designing of a transmission system in the following ways:

a) Channel coding or error control code reduces the transmission power requirement of a digital transmission system. This feature makes it of great use in the field of wireless systems.
b) Maintaining the same level of performance with reduced size of a transmitting antenna or receiving antenna is feasible by use of channel, for example, Very Small Aperture Terminal.
c) The error control technique facilitates access to multiple users in the same radio frequency in a digital communication system, for example, cellular CDMA.

Efficient error control coding technique requires devising codes having good asymptotic error performance, optimum structural properties and efficient encoding and decoding strategies. For a channel code, the encoding process involves mapping of sequence from a k-valued set to a n-valued set with and are defined over a finite field. On the other hand, the decoding process involves the reverse mapping operation. The coding efficiency

of the code can be defined as ratio of I / O where and ere denotes the length of input sequence and represents the output sequence length. refers to elements in the finite field over which input message and output message and
has been defined. = = 2 for a binary code, which implies / . For a channel code, which implies that encoding adds some additional information. This additional information helps in reliable estimation of the original information sequence.

The process of error control or channel coding techniques can be classified into Forward Error correction and Automatic Repeat Request as shown in figure 2.1.

Forward Error Correction (FEC): This involves detection and correction of the erroneous symbols after decoding the received sequence.

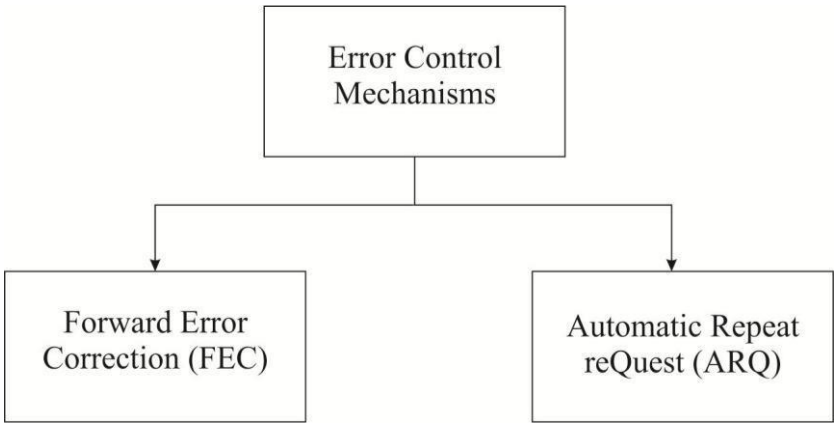


Figure 2.1 Error Control Mechanisms

Figure 2.2 shows structure-based classification of some FEC codes.

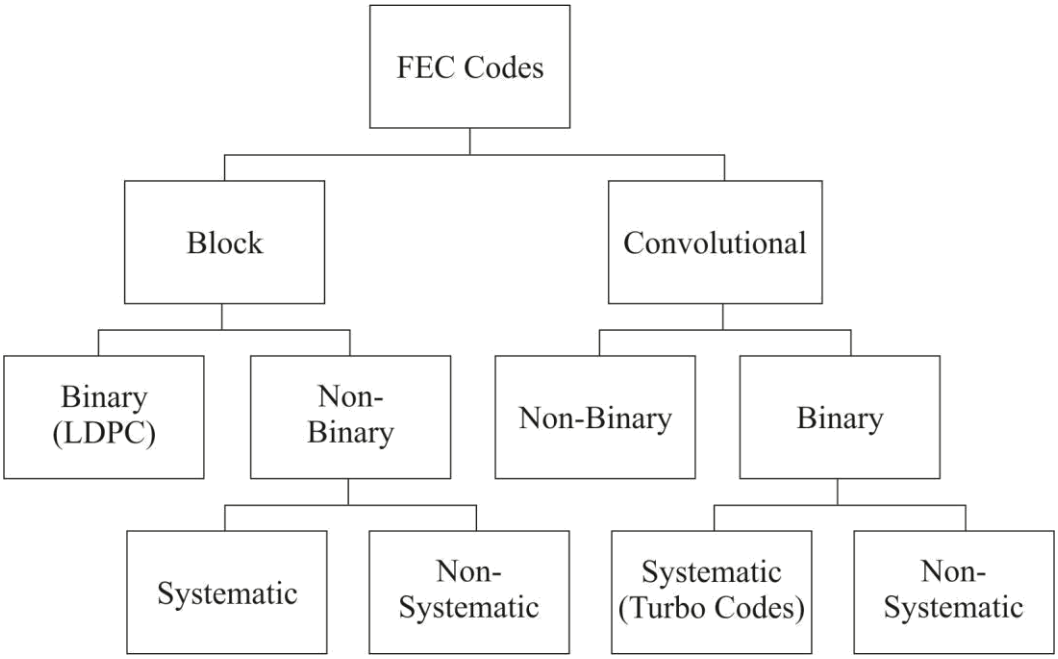


Figure 2.2 Classification of FEC codes

a) **Auto Repeat reQuest (ARQ)**: This is a simple method of error detection and retransmission. The channel decoder detects any error in the sequence received and requests for retransmission if any error is detected. The procedure goes on until an error-free sequence is received. The drawback of this scheme is extra memory required by the transmitter and also that it may involve considerable delay. Three important variants of ARQ scheme are: i) Stop and Wait, ii) Continuous and iii) Selective Repeat.

b) **Hybrid ARQ:** This technique is widely used in digital satellite communication systems. It incorporates the best features of both the schemes. The delay parameter and the bandwidth can be reduced significantly by combining a moderately powerful FEC with an ARQ scheme. Simultaneously the throughput of the transmission system increases.

This thesis discusses Turbo codes and LDPC codes in detail and proposes modifications at algorithmic and architectural level for improving the performance.

2.2 Turbo Codes

By combining couple of simple component codes, Forney proposed concatenated coding schemes as a method for obtaining big coding gains. The error-correction capability of the

resultant codes was that of much longer codes and the resultant codes had the structure that allowed simple to somewhat complex decoding. The combination of the convolutional code (inner side) and Reed-Solomon code (outer side) is the most largely used scheme. The modification of the concatenated scheme is the turbo code. Glavieux, Berrou and Thitimajshima introduced Turbo codes in a paper in 1993, wherein a bit-error probability (= 10^{-5}) was achieved at SNR of 0.7 dB by making use of the code having 1/2 code rate over AWGN channel and BPSK modulation. In conventional codes, the hard decision decoded bits are given in the final step. For proper working of the concatenated scheme, the algorithm used for decoding process should not only forward hard decisions among the decoders but should also affect an exchange of soft decisions to yield the most reliable result. In turbo codes, the soft decisions are exchanged between decoders and this process is repeatedly done to get optimized results.

2.2.1 Literature Review

Forney proposed concatenated coding techniques [3]. These techniques have the capability of correcting the errors of much longer codes with relatively easy to moderately complex decoding. Turbo codes can be explained as a modified concatenated structure with decoding algorithm undergoing number of iterations to produce reliable result. In 1993, Berrou, Glavieux and Thitimajshima introduced Turbo codes [2]. Berrou et al in [2] proposed that the BER of Turbo codes closely approximates Shannon‟s limit. The proposed encoder for turbo codes was built by the concatenation of a large non-uniform interleaver and two Recursive Systematic Convolutional (RSC) codes in a parallel manner. The decoder was implemented with P pipelined identical elementary decoders using modified Bahl et al algorithm [9] .In 1994, Patrick Robertson published a paper [33] in which he highlighted certain limitations of the BCJR algorithm. According to Patrick, proper description of the the interface between the decoders was not given. Further, he also highlighted that the main issue for the smaller blocks

i.e. trellis-termination was not addressed. To overcome the limitations, he proposed to formulate the associative iterative decoder in a comparatively simple manner by exchanging information between the decoders using log-likelihood ratios. He also proposed a method for reducing the iterations required for the same BER value. Another drawback of the codes proposed in [2] has been addressed in [33] that the BER curves show flattening at high SNRs. This paper proposes an interleaver design which can give marked improvement at high SNRs. In 1994, the theory of turbo codes was applied to linear block codes [34]. [35] Explains turbo decoding for multiple turbo codes and performance within 0.8dB of Shannon‟s limit for a rate

¼ code and at BER=10^{-5}. In [36], published in 1995, novel turbo encoder structures having constituent codes code rate 1/3, 2/3, 3/4 and 4/5 have been developed. The codes have been designed using primitive polynomials and random interleavers have been used for better asymptotic performance. The performance obtained from such codes lies within 1decibel of the Shannon‟s limit for BER of 10^{-6}. In [4], Benedetto and Montorsi proposed a technique for estimating the probability of error for a parallel concatenated coding scheme for any interleaver used. In [40], a module having soft input and giving soft-output used in decoding of PCCC [2, 33] and SCCC [37-40] has been analyzed in detail. Also, sliding window SISO algorithm is proposed which updates the probability distributions continuously without requiring trellis termination. Additive version of SISO algorithm is also proposed to overcome the complexity of multiplication operations. In 1997, Haganauer [41] generalized the usage of turbo principle from just decoding of parallel concatenated codes to serial decoding, source-controlled channel decoding, equalization, multiuser detection and coded modulation.

Turbo coding refers to concatenation of RSC encoders through pseudorandom interleaver [4]. Turbo codes give remarkable performance but also exhibit BER floor problem. The BER problem can be tackled by concatenating turbo codes with BCH or RS outer code. List

decoding can also be used to reduce BER and FER. PLVA and SLVA were introduced in [44-45]. In [46], list decoding is applied to turbo codes and performance comparison is made between convolutional codes and turbo codes using list decoding method. Finally, a decoding method based on serial LOVA is developed which achieves the desired result of reducing BER. In [47], asymmetric codes were explained which again targeted the BER and FER of turbo codes. The turbo codes of asymmetric nature consisted of one component code with good performance in "waterfall region" concatenated with another component code having good performance in the "error-floor" area of BER plot of Turbo codes. Asymmetric codes achieve relatively good BER performance in both the regions and better FER performance results than for either of the symmetric versions throughout the "waterfall" region. In [48], a novel type of turbo codes, referred to as turbo-SPC is proposed as an alternative to puncturing for changing code-rate. Turbo-SPC scheme significantly reduces decoding complexity, makes the cost reduction factor more conspicuous as the rate increases and also improves the error-rate floor problem.

SISO-APP (SISO- a posteriori probability algorithm) [40] algorithm is the generalized form of BCJR [9] algorithm. The advantage of SISO-APP over MAP is that it does not depend on whether the code is of systematic type or nonsystematic type, recursive or non-recursive. It is also independent of the concatenation scheme (serial, parallel, or hybrid). In [50], SISO-APP decoders have been synthesized and analyzed using a tile-graph technique [49]. The proposed technique discusses memory and latency parameters of SISO-APP architectures. Also, optimized SW and PW architectures were derived and a novel parallel window Data Flow Graph was discussed. [51] Presents a parallel VLSI architecture, comprising of multiple SISO elements, for low-latency turbo decoding. The proposed parallel architecture is then compared to the sequential architecture and it is found that the delay gets reduced by around 20 times and output is also reduced up to six-fold in the parallel architecture. SISO-MAP

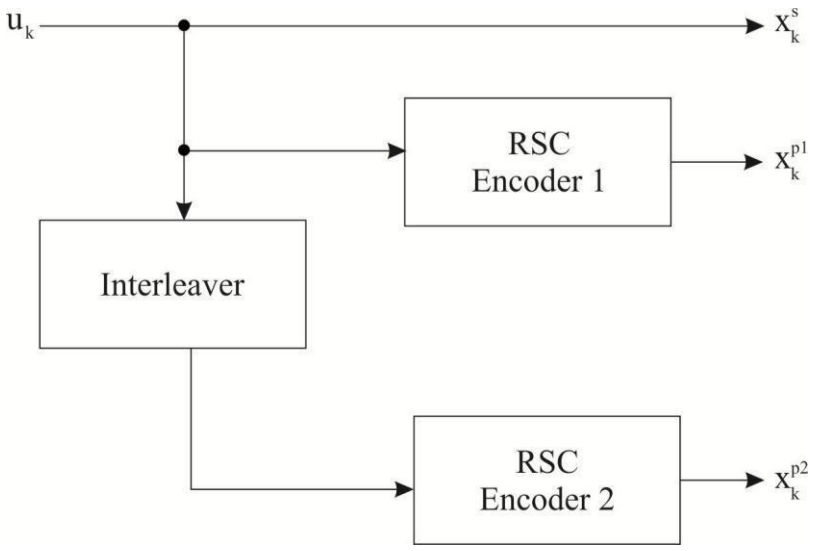
based turbo decoders were used in applications such as HSDPA or 3GPP-LTE, for decoding in an iterative manner in MIMO wireless systems (IEEE802). [52] Proposes radix-2 and radix-4 SISO-MAP decoder architectures. Implementation of the proposed architectures has been done in 180nm, 130nm and 90nm technology. Throughput, energy and silicon area-tradeoffs for the proposed architectures have been investigated in the paper. Various high speed recursion architectures were proposed in [53-55]. An advanced Radix-2 recursion architecture based on architectural and algorithmic level transformation has been proposed in [56] which possesses significantly lower hardware complexity and achieves comparable processing speed as Radix-4 architecture.

Decoders with low throughput (less than 10 Mb/s) have already been implemented, with numerous commercial applications in software or hardware. The paper [57] discusses different methods to increase turbo decoder‟s throughput such as parallel architectures and adequate stopping criteria. The paper also states that by directly wiring the SISO decoders, the concept of analog decoding will allow the concept of iterative decoding to be removed which may, in future, facilitate low energy consumption receiver deign. In flexible turbo decoding algorithm [58], mix of early-stop, parallel decoding, dual-path processing and radix-4 algorithms is proposed which reduces the power consumption and the latency. Parallelism techniques were classified into a three-level structure in [59]. The three levels are Turbo-decoder level Parallelism, BCJR-SISO decoder level parallelism and BCJR metric level parallelism. [60] Presents the architecture of turbo decoder utilizing both parallel-trellis stage level and parallel-SISO decoder level. Also proposed is an interleaver which allows contention-free access in the hybrid parallelism. Number of MAP decoders and memory banks have been used in [61] to achieve high throughput. To enable concurrent working of MAP decoders, QPP interleaver allowing contention-free access to memory modules is used.

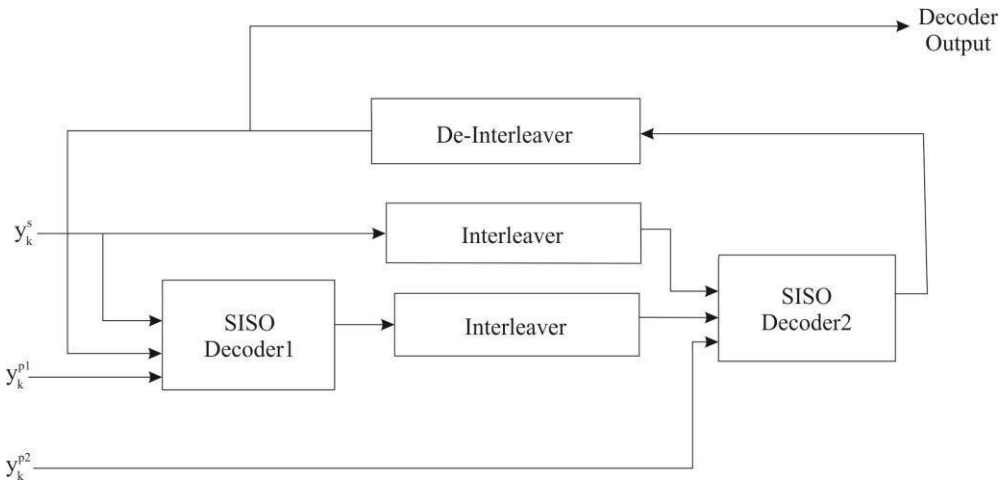
The proposed architecture can have any degree of parallelism and its structure can be adjusted according to the required throughput value.

2.2.2 Turbo Encoder

Figure 2.3 shows a turbo encoder consisting of two recursive systematic convolutional (RSC) encoders which are concatenated with an interleaver in between them. The encoder1 encodes the input data block. The same data block is also interleaved and encoder2 encodes it. The interleaver randomizes the burst error patterns, if any, so that the information can be retrieved correctly.



(a) Encoder



(b) Decoder

Figure 2.3 Turbo Encoder / Decoder

The main features of Turbo encoder are

- RSC encoders
- Parallel concatenation and Interleaving
- Iterative decoding

2.2.2.1 RSC encoder

The conventional convolutional encoder, which is not recursive or systematic can be converted to a RSC (Recursive Systematic Convolutional) encoder by feeding back one of its encoded outputs to the input side.

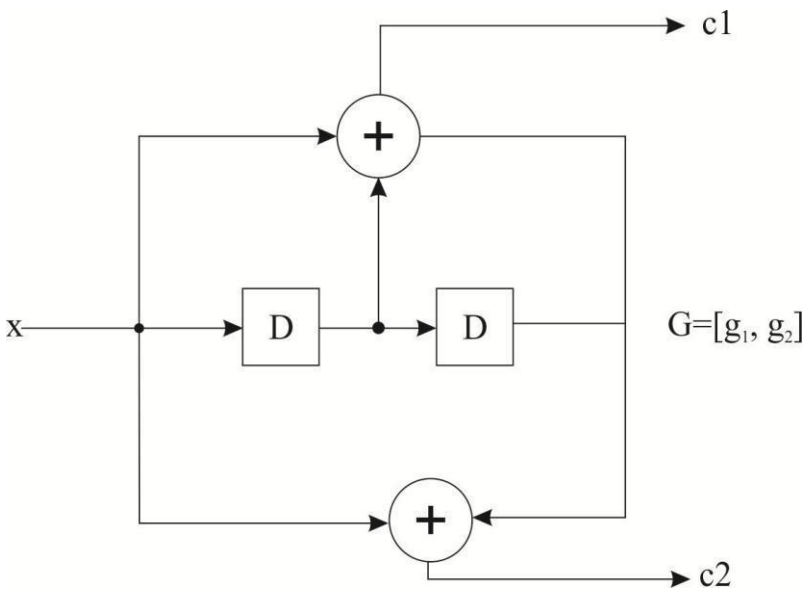


Figure 2.4(a) Non-Systematic Convolutional code with r=1/2 and K=3

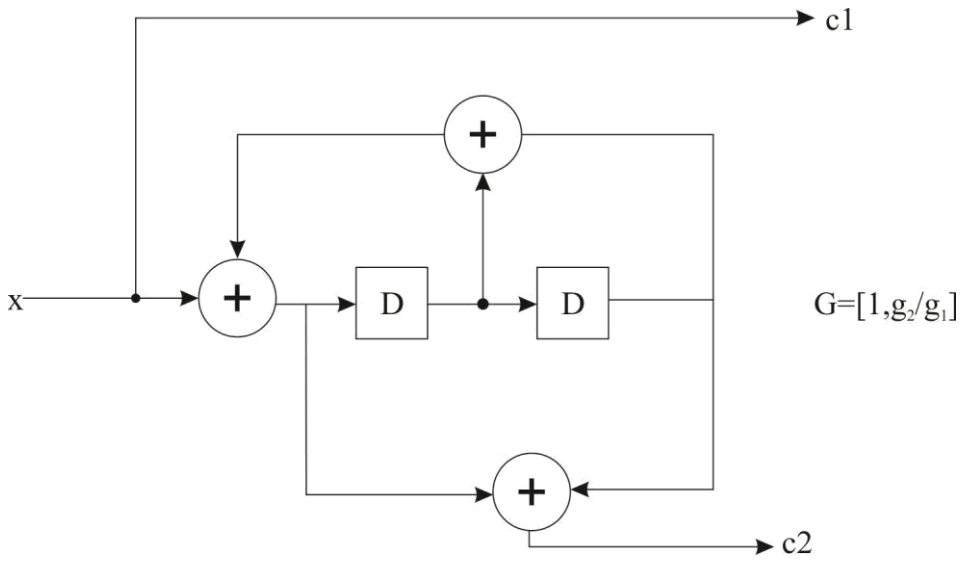


Figure 2.4(b) Recursuive Systematic encoder with r=1/2 and K=3

For each adder in the convolutional encoder, a generator polynomial is defined. [ ] and [ ] represent the generator polynomials of the above convolutional encoder shown in figure 2.4(a). The subscript 1 denotes the output terminal and the subscript 2 denotes the output terminal . A "1" in the generator polynomial represents a connection and a "0" in the polynomial represents no connection. The element in the row and column of the generator matrix (of the order k×n) of the convolutional encoder determines how element is determined by the input. The generator matrix for the above shown encoder (figure 2.4(a)) is given in the following equation

[ ] [ ] (2.1)

RSC encoder can be obtained from the conventional encoder by feeding back the first output to the input. Then the generator matrix gets modified as

(2.2)

Where 1 is the systematic output, is the feedback to the input of the RSC encoder and is the feed-forward output. Figure 2.4(b) shows the resulting RSC encoder. At low SNR, BER of RSC code is less than NSC code but at high SNR, BER of non-systematic code is less than

systematic convolution code for the same constraint length. Also, NSC codes do not lead themselves to parallel concatenation.

2.2.2.2 Concatenation of codes and interleaving

In the theory of coding, an inner code combines with an outer code to form concatenated codes which are a class of error correcting codes. The main purpose of concatenated codes is to have an error probability which decreases exponentially as the block length increases [62]. Concatenation is of two types: serial and parallel. In serial concatenation, the encoder1 output gets interleaved and this interleaved output is given as input to encoder 2. On the other hand, in parallel concatenation, the input data itself is interleaved before being given to the second encoder as input. To eliminate the chances of burst error occurrence and to randomize the code optimally, an interleaver is generally used in concatenated codes. for serial concatenation is given as [62]

(2.3)

Where is the total code rate.

for parallel concatenation is calculated as

(2.4)

That implies, (2.5)

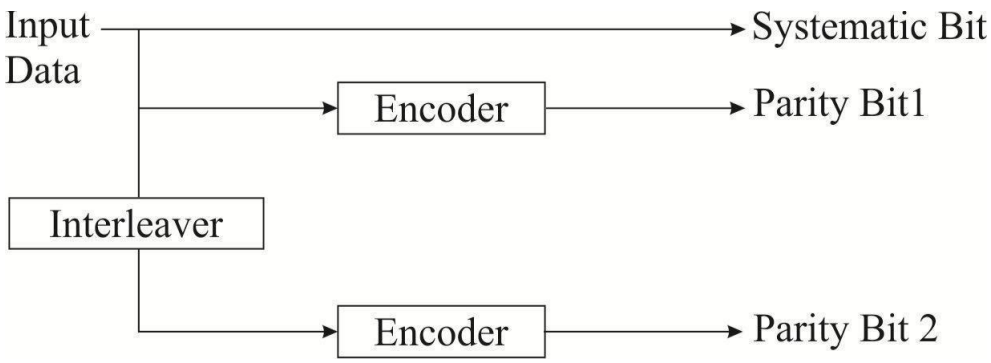


Figure 2.5(a) Parallel concatenation scheme

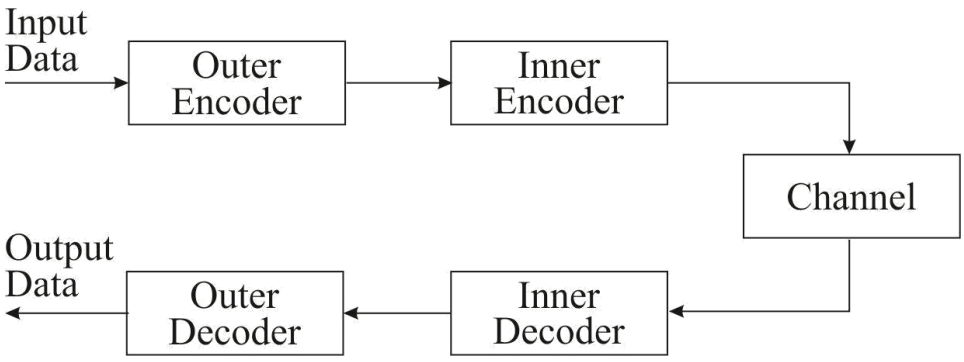


Figure 2.5(b) Serial concatenation scheme

The serial and parallel concatenation schemes are shown in Figure 2.5. Turbo codes use the parallel concatenated encoding scheme. However, the turbo code decoder is based on serial concatenation.

2.2.2.3 Iterative decoding

For iterative decoding, the three inputs to the turbo decoder are defined as $L(u_k)$ (the *a priori* value for bit u_k); r_k (received uncoded bits affected by noise); and c_k(received coded bits affected by noise). The expression for LLR of u_k after considering the received symbol r_k is given as

, (2.6)

that may be expressed as

+ . (2.7)

The expression for probability density function (r) for AWGN channel is

√ ( ) (2.8)

Where m is the mean and σ is the noise variance.

Conditional pdf can be expressed as

( ) (2.9)

( ) (2.10)

We can now compute

(2.11)

Log Likelihood Ratio can be re-computed by making use of the above expressions. Thus,

(2.12)

Say, represents the reliability estimate of the channel, and represents the *a priori* value for bit u_k.

Thus we get

(2.13)

After considering the whole observation sequence, the LLR of the information u_k may be given by the following expression

We can now rewrite it as

(2.14)

(2.15)

$L_e(u_k)$ represents extrinsic information. The *a priori* value is null for first half of iteration, thus

(2.16)

a priori information becomes the extrinsic information for second half of iteration, thus, the output may be expressed as

+ (2.17)

For the next iteration, becomes the *a priori* value of $L_a(u_k)$ and this process goes on. Initially, the extrinsic values are statistically independent and there is significant improvement from one iteration to another. But, the improvement after a few iterations becomes negligible as the initial information remains unchanged. Finally, the soft output decision after the last iteration can be obtained by taking the sum of the last two extrinsic

values and $L_c r_k$ and hard decision value can be computed by comparing the summation value with a threshold level equal to zero, that is, if the log likelihood ratio value is more than or equal to 0, then the decoded bit is considered as 1 otherwise it is considered as 0.

2.3 Decoding Of Turbo Codes

Turbo codes use Soft-In-Soft-Out decoding algorithms to retrieve the original information. In the conventional approach of decoding, hard value (0 or 1) is passed to the error control decoder. This approach has the drawback that the reliability of the information is not ascertained. On the other hand, SISO decoder outputs the estimate for each data bit. Estimate determination of same set of data bits is provided by the decoders but in different order. In the process of decoding, if all midway values are soft values, the decoders can achieve much from exchanging information. Performance gets enhanced by iterating the information exchange a number of times. After each iteration, estimates get re-evaluated by decoders by using information from other decoder and each bit is assigned the value of 0 or 1 in the final stage. In the design of turbo codes such decoders are essential. MAP and SOVA are the most commonly used decoding algorithms which are soft-decision based [63]. The performance of MAP algorithm is approximately .5 decibels better as compared to SOVA at low values of signal-to-noise ratio but it is more complex [64]. This is quite beneficial for turbo codes as any improvement in the BER of the first stage of iterative decoding increases overall performance significantly. On the other hand, SOVA is preferred due to its low latency values in number of real time applications.

The Turbo decoder consists of two SISO decoders and an interleaver/ de-interleaver pair to exchange information between two SISO decoders as shown figure 2.3.

2.3.1 SISO MAP Decoding Algorithm

Consider a convolutional encoder with code , which codes information bits into encoded bits. is called the constraint length of encoder. is called the code rate

of encoder and is defined as / . Number of storage elements in the encoder is given as which implies that at any time, encoder can be in any one state out of states. Next, for every pair of input variable and current state , there is a unique pair of output variable and next state( ).

The encoder passes from stage to . For input binary information { }, the turbo encoder provides output in two parts, and where represents the information sequence and is represented as { } and represents parity codes and is represented as{ } . An AWGN channel with noise spectrum density and BPSK modulation is assumed. The output is also obtained in two parts, and , where = { } and = { } and are actually noisy versions of the output. The objective of decoder is to find out original data out of noisy pattern Y= { }.

The MAP algorithm determines the data sequence by calculating each symbol separately from whole received sequence . The MAP decoder decides whether or depending on the sign of the following log-likelihood ratio (LLR)

$$LLR = \log \frac{P(x_k = +1 \mid y)}{P(x_k = -1 \mid y)} \tag{2.18}$$

Assume that for time , denotes the state of the encoder at time . Also, consider that Then LLR can be rewritten as

$$LLR = \log \frac{\sum_{k} \sum_{k-1} \gamma(y_k, T_{k-1}, T_k).\alpha_{k-1}(T_{k-1}).\beta_k(T_k)}{\sum_{k} \sum_{k-1} \gamma(y_k, T_{k-1}, T_k).\alpha_{k-1}(T_{k-1}).\beta_k(T_k)} \tag{2.19}$$

Where is the forward recursion metric, is the backward recursion metric and is the Branch metric. They can be expressed as

$$\alpha_k(T_k) = \sum_{S_{k-1}} \sum_{i=0} \gamma_i(y_k, T_{k-1}, T_k).\alpha_{k-1}(T_{k-1}) \tag{2.20}$$

$$\beta_k(T_k) = \sum_{S_{k+1}} \sum_{i=0} \gamma_i(y_{k+1}, T_k, T_{k+1}).\beta_{k+1}(T_{k+1}) \quad (2.21)$$

$$\gamma_i((y_{kk}^s, y_{k-1}^p), T_k, T_k) = q(x_k = i \mid T_{kk-1}, T_k).p(y_k^s \mid x_k = i).p(y_k^p \mid x_k = i, T_{kk-1}, T_k).P(T_{rk} \mid T_{k-1}) \quad (2.22)$$

Wherein, the parameter (/) can be either one or zero depending on whether is possible for the transition from state to or not. For AWGN channel, calculating | and (|) trivial. There is a fixed value for the last component | usually for all values of . By using logarithmic quantities, the equations can be converted to the additive form to avoid the complexity of multiplications.

Thus $\overline{\alpha}_k(T_k) = \log(\alpha_k(T_k)),\ \overline{\beta}_k(T_k) = \log(\beta_k(T_k)), \overline{\gamma}_k(T_k) = \log(\gamma_k(T_k))$ (2.23)

The forward recursion metric and the backward recursion metric can also be represented in the additive form, as shown below:

$$\overline{\alpha}_k(T_k) = \max^*_{(T_{k-1}, i)}(\overline{\gamma}_i(y_k, T_{k-1}, T_k) + \overline{\alpha}_{k-1}(T_{k-1})) \quad (2.24)$$

$$\overline{\beta}_k(T_k) = \max^*_{(T_{k+1}, i)}(\gamma_i(\overline{y}_{k+1}, T_k, T_{k+1}) + \beta_{k+1}(T_{k+1})) \quad (2.25)$$

Max* is defined as a maximization function with a correction term. The corrective term in max* can be implemented using a look-up table. However, the case in the iterative decoding is bit different. The „a priori" probability of information bits generated by the other MAP decoder must also be considered in turbo decoders.

2.4 Interleaver

The communication system consists of an information source (producing a message or sequence of messages), a transmitter (having the capability to operate on the message to convert it to a suitable signal for transmission), a channel (a medium like air or wire etc.), a receiver (practically inverse of transmitter operations) and the destination (a person or a machine). The noise source can be defined as burst noise (lightening, switch noise), white noise, or channel distortion. The basic communication system model is shown in figure 2.6.

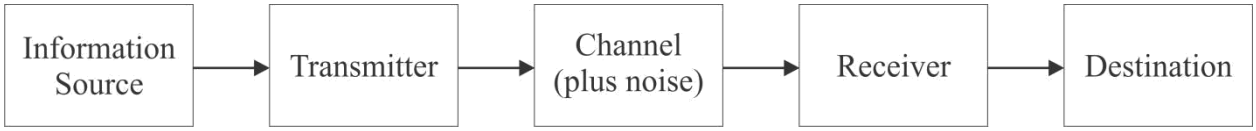


Figure 2.6 Basic Communication System Model

Parity bits are added with the information bits while sending the data over the channel. This new data sequence is a codeword. With the definition of codes, several properties associated with the code words are defined. The two important properties are hamming distance (or simply distance) and hamming weight (or simply weight). The hamming distance between two code words is the total number of positions, where the two code words are different. The hamming weight of a codeword is the total positions by which it is different with respect to the zero vectors. One of the properties, which define the quality of a code, is called the minimum distance which is the smallest distance between distinct code words, and it gives a measure of how good is the code in detecting and correcting errors. Up to errors can be detected and up to t errors can be corrected, with a code with minimum distance , provided the following relation gets satisfied:

(2.26)

Consider a code which can correct up to errors. Let‟s take a simplified data transmission example over a noisy channel as shown in Figure 2.7. We see that some data is lost during the transmission in such a way that two bits each from two of the transmitted code words have become corrupted. As the decoder can correct only one error in each codeword, both these code words end up as un-decodable.

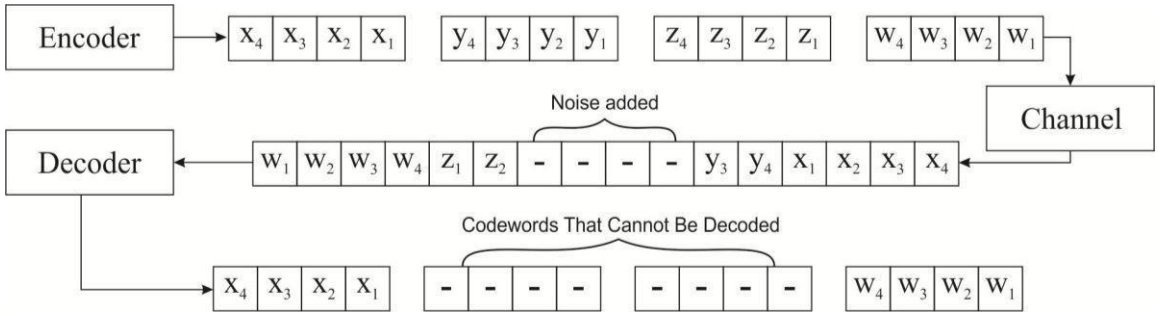


Figure 2.7 Transmissions and Reception of Data Over Noisy Channel

An alternate way to handle this situation is to introduce an interleaver between encoder and transmitter. Vice versa a de-interleaver (inverse of interleaver) has to be incorporated between receiver and decoder. The interleaver forms an important part of a channel coder and its most important objective is to randomize the bit sequence so that the burst error effect in transmission is minimized. The burst errors may occur due to distortion or switching effects in the channel or due to concatenated coding technique.

An interleaver, by definition, is single input - single output device with memory, taking a bit-stream of fixed width as input and generating permuted output bit-stream of the same width.

Thus the mapping of an input sequence on to an output sequence can be defined as where represents a

specific interleaving pattern. The interleaver operation can be parallelized to increase the throughput but the basic operation remains the same. The de-interleaver performs the reverse operation of retrieving back the original information.

2.4.1 Literature Review

Numerous architectures of interleaver have been investigated in the literature [69, 70-72, 73–83]. The bottleneck of memory access contention problem should be overcome to ensure high degree of parallelism [74, 51] required by 3GPP LTE. An idea to have a solution to the memory access contention problem came while designing the contention free interleavers discussed in [84–87]. QPP interleaver is suitable for 3GPP-LTE [11] as it allows parallel memory access which is contention-free. QPP Interleaver‟s algebraic features facilitate simplified hardware implementation and analytical designing of a parallel turbo decoder [75]. Also, as QPP derives its algebraic structure from the theorems of the permutation polynomials defined over integer rings, the parallelism degree for the decoder can be determined from the factors of the interleaver length [75]. A decoder optimized for 3GPP LTE application, that is, a parallel Turbo decoder architecture giving high throughput, can be

designed by using memory architecture with multi-bank feature and an interleaving address generator having low complexity [88]. In [89], QPP interleaver with a reconfigurable architecture has been proposed to provide contention-free memory access in the turbo decoder. This reconfigurable architecture eliminates the extra memory requirement for storing the interleaved addresses or the increase in the clock frequency as compared to the conformist approach. In [90], a novel non-linearity metric has been explained. Also, a novel interleaving metric has been proposed that depends on the spread factor and the non-linearity metric. [91] Compares the QPP and ARQ interleavers on basis of their design and performance. The coefficients of two QPPs must satisfy some sufficient conditions to generate identical permutations. The theorem explaining these sufficient conditions and the method for reduction of search time of QPP interleavers is given in [92]. The lengths used in the theorem are given by LTE standard up to 512. In [65], the appropriateness of algebraic interleavers providing contention-free memory-access for parallel decoding of turbo codes has been reinforced. Also, the simulation results in this paper illustrate better results for turbo codes using these interleavers. In [93], parallel interleaver architecture has been proposed generating on-the-fly interleaved addresses. Further, a parallel interleaving architecture has been proposed in [94] for accomplishing highly parallel multi-standard turbo decoding, which solves conflicting memory problem. In [95], the analysis of maximum-spread and largest-spread QPP interleavers has been done.

2.4.2 Types of Interleaver

Based on the input information type, the interleavers can be divided in two main types named as block interleaver and convolutional interleaver as shown in figure 2.8.

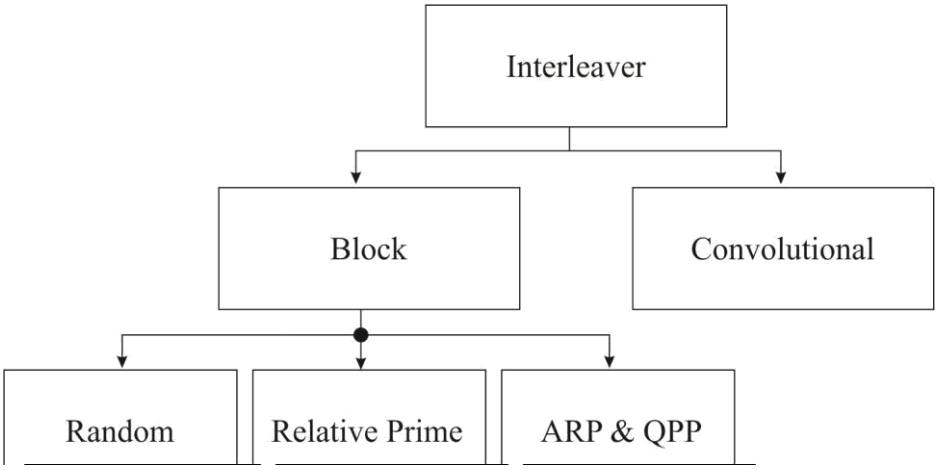


Figure 2.8 Classification of Interleavers

2.4.2.1 Block Interleaver

The input information in a block interleaver is in block form. The permutations are applied to the complete block of data. For example, in row-column matrix interleaver, information is retrieved column wise and entered row wise. Block interleavers can further be classified into random interleaver, S-Random interleaver, deterministic interleaver, ARP interleaver and QPP interleaver, depending on the manner of the address generation in the interleavers.

2.4.2.2 Convolutional Interleaver

Continuous stream of data is the input to the convolutional interleaver. Thus, it involves less latency as it need not wait for the whole block to start operation. It consists of number of branches made up of first-in-first-out (FIFO) shift registers. A commutator switch connects the input of the interleaver to the output of encoder. The input commutator switch shifts the code symbols from encoder in a sequential manner to different FIFO registers in all the branches in a cyclic way.

2.4.3 QPP Interleaver

3GPP LTE has adopted Quadratic permutation polynomial (QPP) interleaver for turbo coding. It can be used for parallel memory access due to its contention free property. It differs from previous 3G (third generation) interleavers in that it has defined algebraic structure. The structure of QPP is based on algebraic constructions via permutation polynomials over

integer rings. This structure was adopted as it is known that permutation polynomials generate contention-free interleavers [65, 66]. In contention-free interleavers, every factor of the interleaver length becomes a possible parallelism degree. The QPP interleaver can be expressed with a simple algebraic formula. For an information block length , the interleaving output position is specified by the quadratic expression [67]:

(2.27)

Where and are integer parameters and have dependency on the block size

. For each value of block size , a different set of parameters and is defined in LTE. Moreover, parameter is always an odd number whereas is always an even number. Each value of block size is an even number and is divisible by 4 and 8.Moreover, the block size *N* is always divisible by 16, 32, and 64 when

respectively.

The algebraic properties for the QPP interleaver are listed as under:

Property 1: has the same even/odd parity as *i*:

Property 2:

Property 3:

f(i)/4, *f(i+1)*/4, *f(i+2)*/4, and *f(i+3)*/4 has unique remainders:

{

{

2.4.3.1 QPP Contention-Free Property

Generally, for a contention-free Turbo interleaver/de-interleaver, the following constraint should be satisfied [65, 68]:

$$\lfloor \frac{\quad}{\quad} \rfloor \quad \lfloor \frac{\quad}{\quad} \rfloor \qquad (2.28)$$
$$(\quad -)$$

The memory index positions that are concurrently accessed by the *Q* MAP decoder cores are shown in equation (2.28). For a contention-free interleaver, these memory index positions should be unique during each read and write operation (figure 2.9).

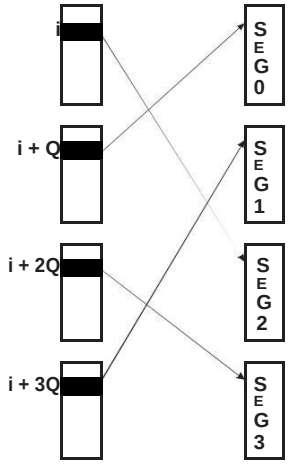


Figure 2.9 Contention-free Interleaving

2.4.3.2 Hardware Implementation of QPP Interleaver

Based on the algebraic analysis in [64], As the QPP interleaver is contention free always, thus its hardware implementation is comparatively simple.

Table 2.1 *f1, f2* and *N* (shown as K_i) values as defined in LTE standard [125]

i	K_i	f_1	f_2	i	K_i	f_1	f_2	i	K_i	f_1	f_2	i	K_i	f_1	f_2
1	40	3	10	48	416	25	52	95	1120	67	140	142	3200	111	240
2	48	7	12	49	424	51	106	96	1152	35	72	143	3264	443	204
3	56	19	42	50	432	47	72	97	1184	19	74	144	3328	51	104
4	64	7	16	51	440	91	110	98	1216	39	76	145	3392	51	212
5	72	7	18	52	448	29	168	99	1248	19	78	146	3456	451	192
6	80	11	20	53	456	29	114	100	1280	199	240	147	3520	257	220
7	88	5	22	54	464	247	58	101	1312	21	82	148	3584	57	336
8	96	11	24	55	472	29	118	102	1344	211	252	149	3648	313	228
9	104	7	26	56	480	89	180	103	1376	21	86	150	3712	271	232
10	112	41	84	57	488	91	122	104	1408	43	88	151	3776	179	236
11	120	103	90	58	496	157	62	105	1440	149	60	152	3840	331	120
12	128	15	32	59	504	55	84	106	1472	45	92	153	3904	363	244
13	136	9	34	60	512	31	64	107	1504	49	846	154	3968	375	248
14	144	17	108	61	528	17	66	108	1536	71	48	155	4032	127	168
15	152	9	38	62	544	35	68	109	1568	13	28	156	4096	31	64
16	160	21	120	63	560	227	420	110	1600	17	80	157	4160	33	130
17	168	101	84	64	576	65	96	111	1632	25	102	158	4224	43	264
18	176	21	44	65	592	19	74	112	1664	183	104	159	4288	33	134
19	184	57	46	66	608	37	76	113	1696	55	954	160	4352	477	408
20	192	23	48	67	624	41	234	114	1728	127	96	161	4416	35	138
21	200	13	50	68	640	39	80	115	1760	27	110	162	4480	233	280
22	208	27	52	69	656	185	82	116	1792	29	112	163	4544	357	142
23	216	11	36	70	672	43	252	117	1824	29	114	164	4608	337	480
24	224	27	56	71	688	21	86	118	1856	57	116	165	4672	37	146
25	232	85	58	72	704	155	44	119	1888	45	354	166	4736	71	444
26	240	29	60	73	720	79	120	120	1920	31	120	167	4800	71	120
27	248	33	62	74	736	139	92	121	1952	59	610	168	4864	37	152
28	256	15	32	75	752	23	94	122	1984	185	124	169	4928	39	462
29	264	17	198	76	768	217	48	123	2016	113	420	170	4992	127	234
30	272	33	68	77	784	25	98	124	2048	31	64	171	5056	39	158
31	280	103	210	78	800	17	80	125	2112	17	66	172	5120	39	80
32	288	19	36	79	816	127	102	126	2176	171	136	173	5184	31	96
33	296	19	74	80	832	25	52	127	2240	209	420	174	5248	113	902
34	304	37	76	81	848	239	106	128	2304	253	216	175	5312	41	166
35	312	19	78	82	864	17	48	129	2368	367	444	176	5376	251	336
36	320	21	120	83	880	137	110	130	2432	265	456	177	5440	43	170
37	328	21	82	84	896	215	112	131	2496	181	468	178	5504	21	86
38	336	115	84	85	912	29	114	132	2560	39	80	179	5568	43	174
39	344	193	86	86	928	15	58	133	2624	27	164	180	5632	45	176
40	352	21	44	87	944	147	118	134	2688	127	504	181	5696	45	178
41	360	133	90	88	960	29	60	135	2752	143	172	182	5760	161	120
42	368	81	46	89	976	59	122	136	2816	43	88	183	5824	89	182
43	376	45	94	90	992	65	124	137	2880	29	300	184	5888	323	184
44	384	23	48	91	1008	55	84	138	2944	45	92	185	5952	47	186
45	392	243	98	92	1024	31	64	139	3008	157	188	186	6016	23	94
46	400	151	40	93	1056	17	66	140	3072	47	96	187	6080	47	190
47	408	155	102	94	1088	171	204	141	3136	13	28	188	6144	263	480

The interleaving addresses in QPP can be calculated in a recursive manner (figure 2.10). Say, the interleaver starts at , is pre-computed as ( )

is incremented by y in the following cycles, thus (2.29)

follows: is computed recursively as

= (2.30)

Where is defined as

(2.31)

The initial value can be pre-computed as

(2.32)

The value of can also be computed in a recursive manner:

(2.33)

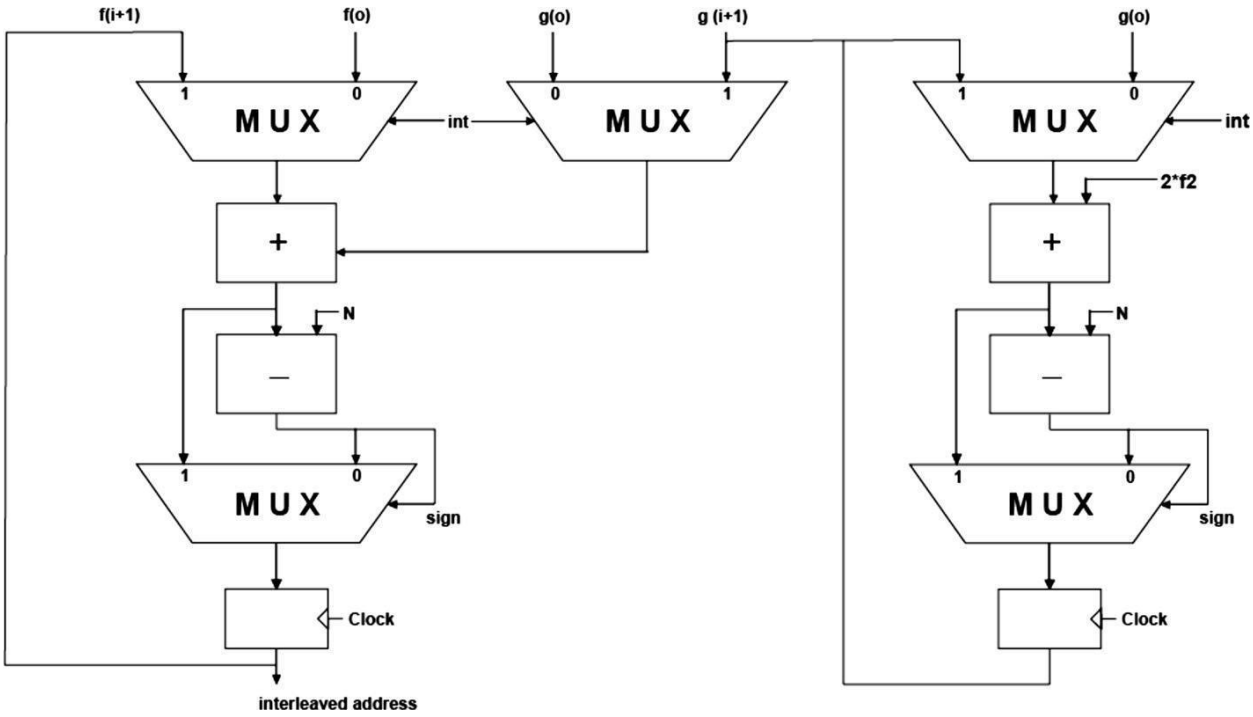


Figure 2.10 QPP Interleaver design

2.4.3.3 Parallel QPP Interleaver

The design of QPP interleaver can be done by look-up table approach but it results in large circuit area. Another approach can be implemented by online calculation which produces one interleaved address at each clock pulses. In this approach extra decoding latency is involved as decoder has to wait till the complete generation of corresponding interleaving addresses. Hence it causes extra decoding latency. To overcome this problem, parallel recursive architecture is used which reduces the required clock pulses and, in turn, reduces decoding latency. Also, QPP interleaver in parallel allows lower power consumption compared to its conformist approach. For parallel memory access, Y memory blocks, each of size P, are

required (figure 2.11). The algebraic analysis of the parallel design for ( ) can be discussed by considering two cases, and :

CASE 1: ,

(2.34)

Recursively, ( ) ,where

CASE 2: For $1 \leq k < Y$:

( ) (2.35)

(( ))

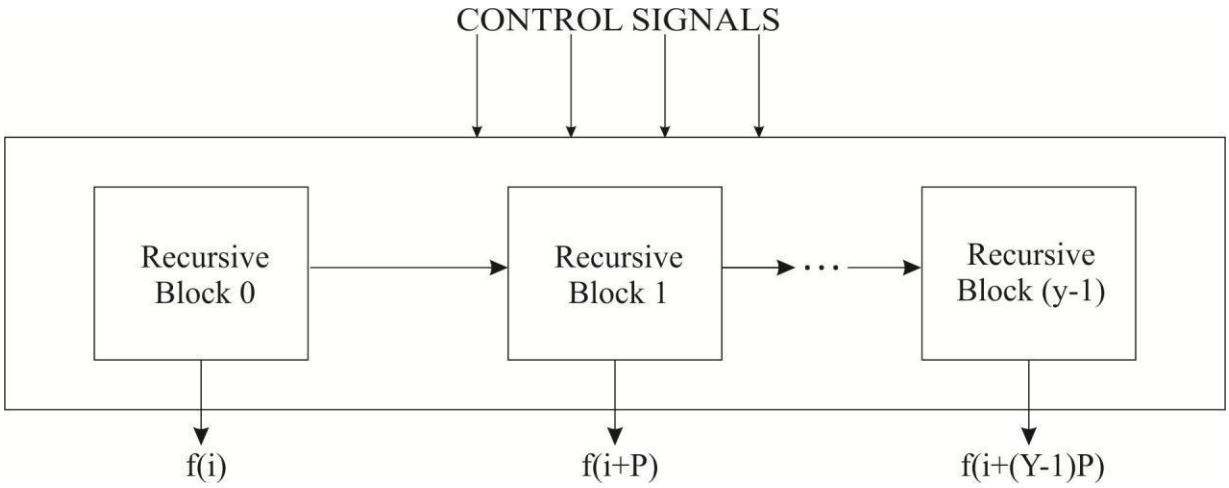


Figure 2.11 Architecture of QPP interleaver for parallel realization

2.5 Proposed Design

The key issue of applying Turbo codes is to find an efficient implementation of turbo decoder. This chapter addresses the implementation of a simplified and efficient turbo decoder in field programmable gate array (FPGA) technology. A simplified and efficient implementation of a Turbo decoder with minor performance loss has been proposed. An integer Turbo decoder based on the standard 2"s complement number system after considering the issues of dynamic range, truncation effect and other algorithm related subjects has been introduced. The efficient implementation comes from algorithm

modification, integer arithmetic and compact hardware management. Based on the Max-Log-MAP [119,120] decoding algorithm, the branch metric is modified by weighing a priori value, resulting in a significant BER improvement. The Turbo decoder takes in 8-level integer inputs, generates 7-bit soft-decisions and calculates all metrics on integers, avoiding complex floating point or fixed-point arithmetic. By manipulating memory address, delay associated with interleaving and de-interleaving is eliminated, resulting in much higher throughput. Also, by taking advantage of identical decoder function, Turbo decoder is implemented in a single-decoder structure, making efficient use of memory and logic cells.

In this chapter, an integer Turbo decoder based on the standard 2‟s complement number system after considering the issues of dynamic range, truncation effect and other algorithm related subjects, has been proposed. The turbo decoder considered in this chapter has the following specifications:

- Code rate /
- Generator polynomial:
- Puncturing pattern: even/odd parity
- Block size: bits
- Interleaver: S-random interleaver with
- Trellis termination: none
- Considered modulation: BPSK

LLR is represented as:

In the iterative decoding, LLR is divided into three terms:

(2.36)

The last term is called “extrinsic information” and only this term should be fed back to the input of the other decoder as a-priori information. Therefore, must be subtracted from before it is fed back to the other decoder. and are

used to terminate the iteration of the Turbo decoding. Therefore, can be expressed as:

—— (2.37)

As mentioned before, the term is the information exchanged between the constituent decoders. In our implementation, we apply the Max-Log-MAP algorithm with a modification of the branch metrics. We weigh a-priori values with a scaling factor s, resulting in the following:

—— (2.38)

The effect of scaling factor on the BER performance was studied and at least 1000 errors were collected.

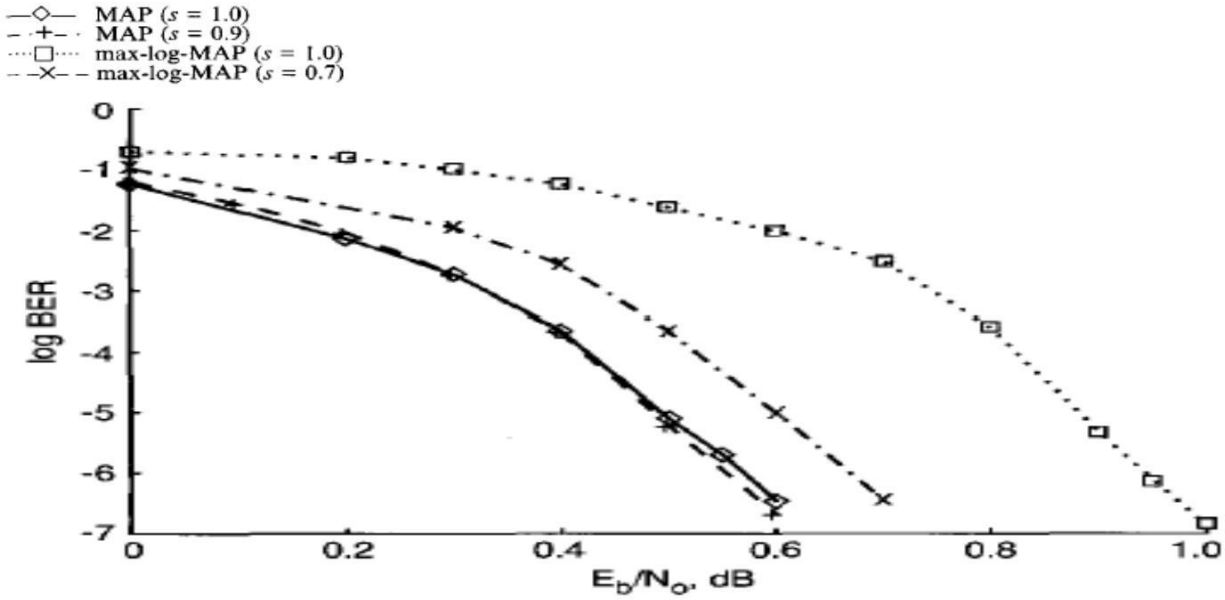


Figure 2.12 Performance of Turbo code with different scaling factors and block length 5114 bits

Figure 2.12 shows the BER performance of the best evaluated scaling factor compared to the standard Max-Log-MAP decoding algorithm for block length 5114 with AWGN and gives the best BER performance. It was found that a properly selected scaling factor can improve the performance of Max-Log-MAP by 0.3 dB, giving a near optimal (Log-MAP)

BER performance. The BER improvement is due to the fact that the scaling factors can effectively mitigate error propagation through iterations. The scaling factor„ ‟ allows the variation in the information exchanged between the decoders. In qualitative terms, a large value of „ ‟ makes the previous decoder outcome dominate the current decoding results, whereas a small value of „ ‟ makes one decoder less dependent on the other decoder‟s result.

2.5.1 Turbo decoder architecture

The implementation architecture for a Turbo decoder can take a serial or a parallel approach. In the serial approach, a pair of decoders is used repetitively and the data input is processed at the higher speed (denoted as bps in Figure 2.13) than the speed of incoming received bits (denoted as K bps). On the other hand, the parallel form of Turbo decoder requires multiple pair of MAP decoders and huge amount of memory for interleavers, de-interleavers and received data buffers. Unless the very high-speed decoder is needed, the serial approach is more practical.

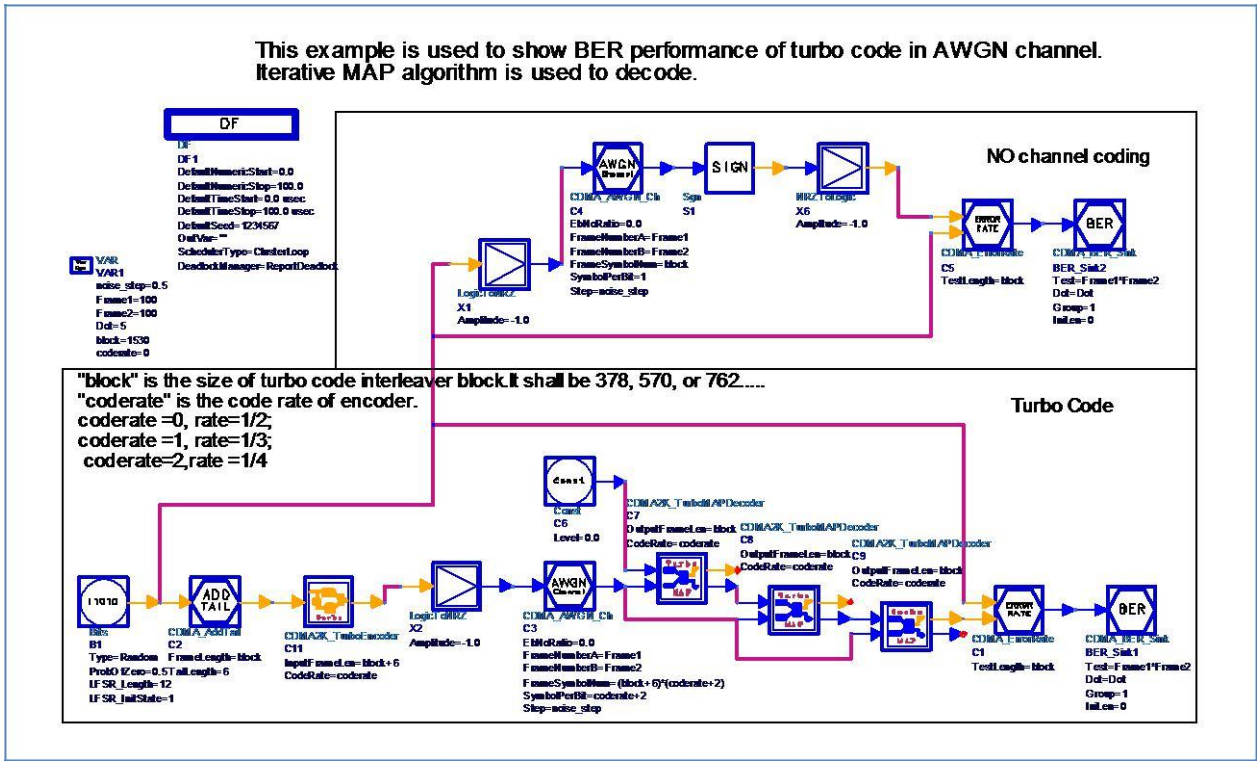


Figure 2.13 Simulation set-up for turbo codes

Now that the implementation has been reduced to a series of decoder operations, the obvious remaining drawback of the Map algorithm is the excessive memory required. The entire history of the state metrics must be stored towards the end of the trellis, at which point the backward algorithm begins and decisions can be output starting with the last branch, without the need to store any but the last set of state metrics computed backwards. This storage requirement is obviously very large. For an 8-state code, assuming 7-bit state metrics, 56 bits of storage per branch are required, which implies that for a 1000-bit block a total of 56,000 bits are required. The implementation of MAP decoder is based on a technique proposed in [4] which reduce the memory requirement for an 8-state code to just a few thousand bits, independent of the block length [122]. The technique can best be described by referring to the timing details of Figure 2.14, which indicates the bit processing times for one forward processor and two backward processors operating in synchronism with the received branch symbols that is, computing one set of state metrics during each received branch time (bit time for a binary trellis).

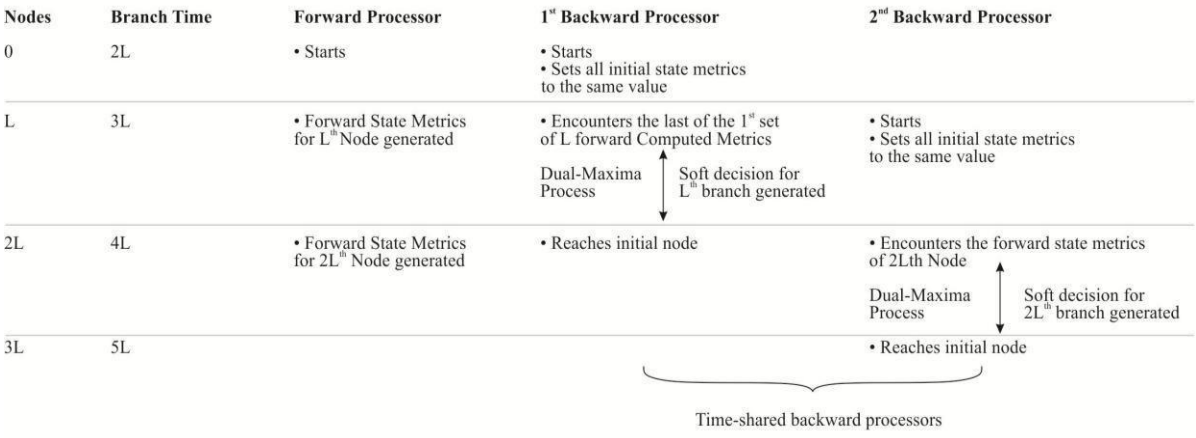


Figure 2.14 Timing details of forward and backward processors

The basic idea behind this approach is that Viterbi algorithm can start cold in any state at any time. After a few constraint lengths, the set of state metrics are as reliable as if the process had been started at the initial (or at the final) node. Let‟s say that this learning period for the trellis is branches, which are 16 in case of 8-state code. This is equally true for forward as

well as backward algorithm and assumes that subtracting at every node an equal amount from each, normalizes all state metrics. Let the received branch symbols be delayed by branch times. Then the forward algorithm will start at branch time . And also, it will compute all the state metrics for each node every branch time, storing these in memory. The first backward processor starts at the same time, but processes backward from the node, setting every initial state metric to the same value and not storing anything until branch time . At this point it has built up reliable state metrics and encounters the last of the first set of forward computed metrics. (In figure 2.16, the top line indicates the time index; the remaining lines are labeled according to the times at which the branches are processed. Also, unreliable metric branch computations are shown as dashed lines). At this point the branch soft decisions are output by performing the generalized dual-maxima process, and the backward processor proceeds until it reaches the initial node at time . Meanwhile, starting at time , the second backward processor begins processing with equal metrics at node, discarding all metrics until time , when it encounters the forward algorithm having computed the state metrics for the node. The generalized dual-maxima process is then turned on until time , at which point all soft decision outputs from the to the node will have been output. The two backward processors hop forward branches every time they have generated backward sets of state metrics, and they time-share the output processor since one generates the useful metrics, which are combined with those of the forward algorithm. For the backward algorithms, nothing needs to be stored except the metric set of the last node. The forward algorithm needs to store only 2 sets of state metrics, since after its first 2 computations (performed by time 4), its first set of metrics will be discarded, and the blank storage space can then be filled starting with the forward-computed

metrics for the node (at branch time 4 +1). Thus, the storage requirements for a 8-state code using 7-bit state metrics 112 bits in all, which for amounts to

approximately 1792 bits. The forward recursion (α-unit) and the backward recursion unit (β-unit) are identical except for the direction of recursion. The α-unit is shown in figure 2.15. It should be noted that the state metrics keep on increasing as the recursion goes on. Therefore, some normalization scheme needs to be adopted to avoid the explosion of the state metrics.

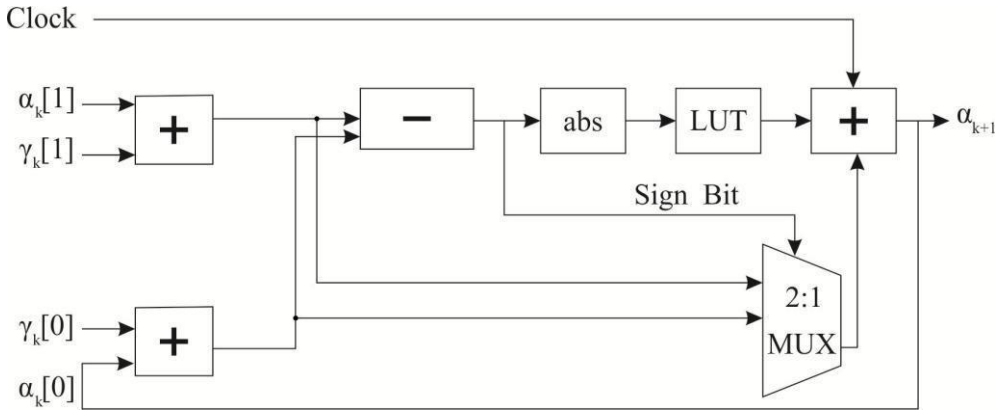


Figure 2.15 α-unit (forward recursion unit)

2.5.2 Numerical range

Most Turbo decoder implementations are based on fixed-point arithmetic [121]. Therefore, in the implementation of Turbo decoder, a significant effort must be focused on dynamic range, number density and normalization before choosing a number system. Since, our design is a simplified structure of Turbo decoder (of course, without significant loss in BER performance), we chose standard 2"s complement integer representation. For efficient implementation, it is required to estimate the numerical range of various metrics such that only a necessary number of bits would be used for each metric. In our implementation, assuming that the demodulator output produces 8-level (3-bit) output, the 3-bit value is converted into a 4-bit integer value ranging from –4 to +4 (without 0). With our modified Max-Log-MAP ( ), eight iterations and 3 dB SNR, simulations were performed to deduce the range of soft-decisions and the results indicate that they lie in the range of –48 to +48. In the same way, the extrinsic values range from –30 to +30 and branch metrics from – 13 to +13. As a result, 6 bits are assigned for extrinsic values and soft-decisions, 5 bits for branch metrics and 8 bits for internal metrics. Thus, the integer based implementation is truly

beneficial as compared to fixed-point implementations which require 8 bits for extrinsic values, 7 bits for demodulator outputs and at least 10 bits for internal metrics.

2.5.3 Node metric normalization

As the recursion process progresses, forward/backward node metrics accumulate. This can easily overflow and may underflow during the first few updates. The overflow/underflow problem can be solved by metric normalization.

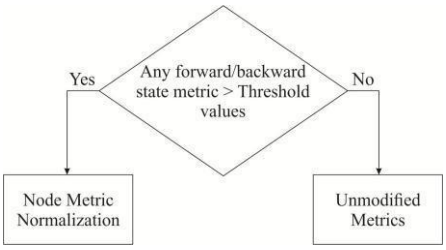


Figure 2.16 Forward/Backward Node Metric Normalization

A threshold value is used, which is compared with all the node metrics for each decoded bit in both forward and backward recursion. If the absolute value of any node metric is greater than , then all the node metrics are shifted towards the center as shown in figure 2.16. Note that the shifting is done for all node metrics so that the soft-decision values are not affected. The threshold value is chosen such that the update (the node metric plus a branch metric) does not cause overflow and the logic required for comparison is minimized. In our implementation, the threshold values are set to be –85 and 86. With normalization, node metrics can be represented using 8 bits.

2.5.4 Truncation effect

Truncation occurs in branch metric calculation [123] due to the division by 2 and the apriori scaling factor„s‟. Four truncation methods are considered round, fix, ceil and floor; each generate different truncation results as shown in Table 2.2.

Table 2.2 Truncated results by different truncation method

Z	-4	-3	-2	-1	0	1	2	3	4

Z/2	-2	-1.5	-1	-0.5	0	0.5	1	1.5	2
Round (Z/2)	-2	-2	-1	-1	0	1	1	2	2
Fix (Z/2)	-2	-1	-1	0	0	0	1	2	2
Floor (Z/2)	-2	-2	-1	-1	0	0	1	1	2
Ceil (Z/2)	-2	-1	-1	0	0	1	1	2	2

Although the round function is more systematic than other truncation functions, simulations in figures 2.17 and 2.18 shows that the floor and ceil functions perform better than the other two methods. In the proposed implementation, the floor function for truncation is used.

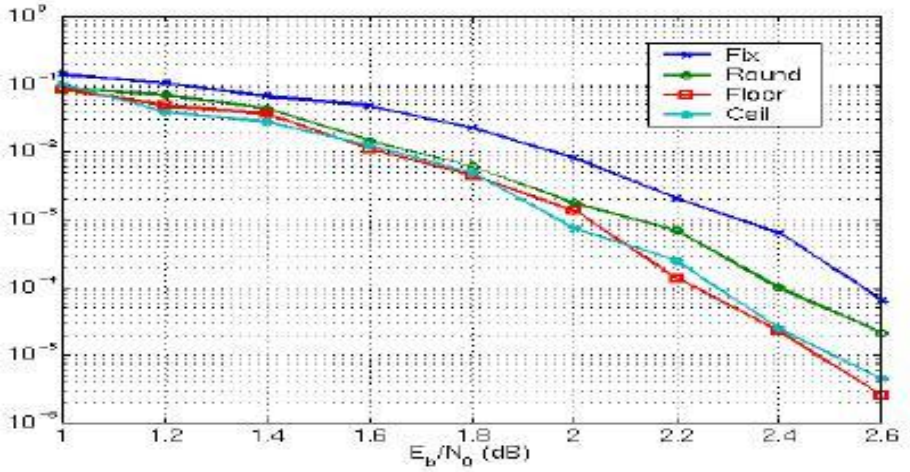


Figure 2.17 Truncation effect on BER

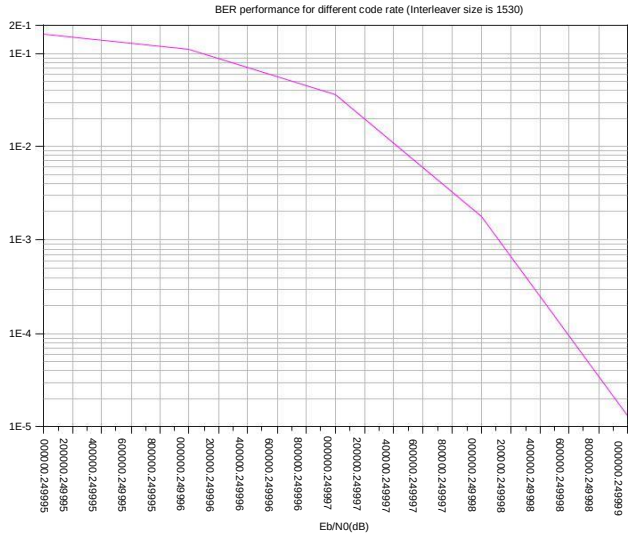


Figure 2.18 BER versus E_b/N_o

2.5.5 FPGA implementation

2.5.5.1 Interleaver/De-interleaver

Turbo decoder requires an interleaver and a deinterleaver to perform proper permutation of systematic bits and extrinsic values. In a conventional turbo decoder design, there will be a separate functional block to carry out the interleaving/deinterleaving function, which will require a fixed amount of time to permute/un-permute its contents [124]. However, in the proposed implementation, all the permutations are performed by address manipulation that requires no additional delay in the processing. Figure 2.19 shows the address generator that generates addresses for a decoder to read/write required information from a memory. Because the first decoder processes un-permuted information, systematic bits and extrinsic value are read sequentially from 0 to N-1. They are then processed and stored in the same address. For the second decoder, information is processed based on the permutation order; therefore, the sequential index is now used to retrieve a permutation index from ROM. This ROM based address that serves as the index is then used to retrieve the required information for processing. Once the information is processed, the result is again stored in the same index completing the interleaving/de-interleaving process. All the information is updated and stored properly by manipulating the memory address for decoders; therefore each decoder is ready to decode immediately after the other decoder is finished, resulting in no delay to process interleaving/de-interleaving. The overall decoding process takes far less clock cycles to finish compared to that with traditional interleaving/de-interleaving process.

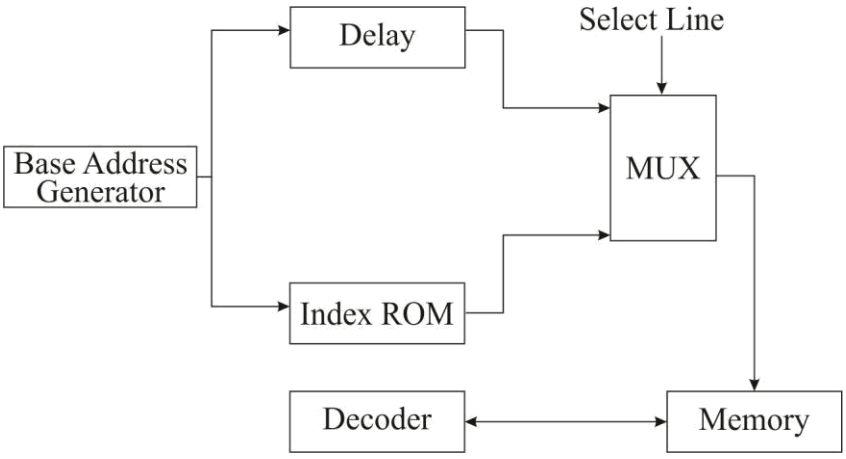


Figure 2.19 Address generator for interleaving and de-interleaving function

2.5.5.2 Single decoder design

In the proposed implementation, it is assumed that the same constituent code is used in the turbo encoder and the constituent decoders are identical. Therefore, the turbo decoder can be simplified by using a single decoder as shown in Figure 2.20. The switch is toggled according to the decoder number being operational while processing the data bits. When the first decoder is to be used, the switches are set to their lower position, which bypasses all interleaving and de-interleaving functions. When the first decoder has finished, the switches are then moved to the upper position to reconfigure the design as the second decoder that functions on permuted information. This process is repeated until the required number of iterations gets completed.

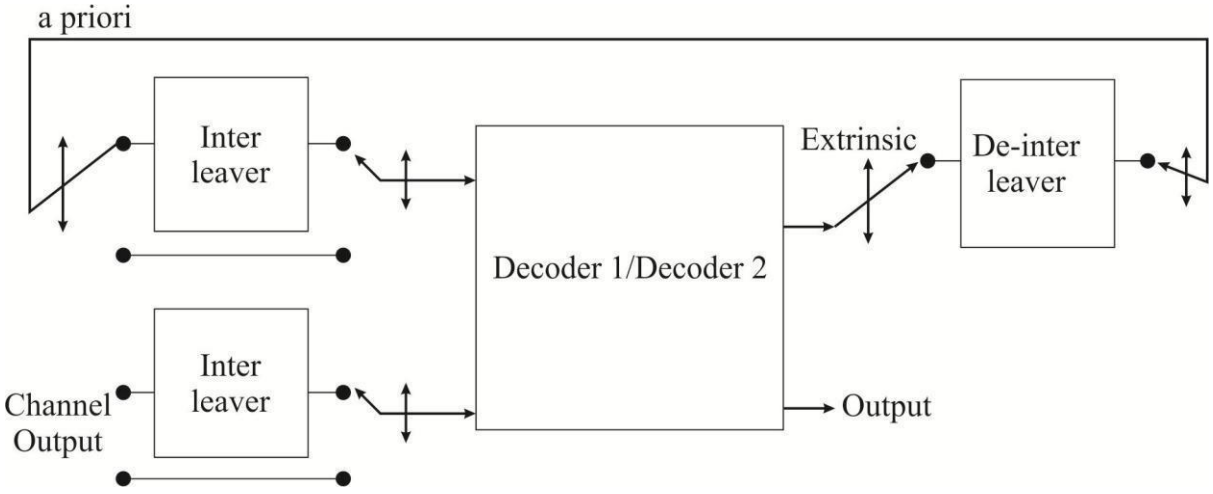


Figure 2.20 Single Decoder Design for Turbo Decoder

2.5.6 Results and Conclusion

Turbo decoder was described in Verilog hardware description language and implemented on Xilinx Virtex series XCV300E FPGA chip. The design utilized almost 3447 out of 6912 logic cells and approximately 28 RAM blocks out of 48 total RAM blocks in the device. In addition, the design required no external components and consumed approximately 695 mW during normal operation. The FPGA-based turbo decoder with 8 iterations can operate more than 1 Mbps throughput at a clock rate of 25 MHz. In this chapter, we have demonstrated a simplified and efficient implementation of a Turbo decoder with minor performance loss. The efficient implementation comes from algorithm modification, integer arithmetic and compact hardware management. Based on the Max-Log-MAP decoding algorithm, we modify the branch metric by weighing a priori value, resulting in a significant BER improvement. The Turbo decoder takes in 8-level integer inputs generates 7-bit soft-decisions and calculates all metrics on integers, avoiding complex floating point or fixed-point arithmetic. By manipulating memory address, delay associated with interleaving and de-interleaving is eliminated, resulting in much higher throughput. Also, by taking advantage of identical decoder function, we implemented our Turbo decoder in a single-decoder structure, making efficient use of memory and logic cells.

2.6 Conclusion

This chapter presents the theoretical perspective of channel coding and the turbo codes. A detailed description of the basic turbo encoder structure has been presented. Decoding algorithms for the turbo decoder have been discussed and a comparison has been drawn between MAP and SOVA algorithm. For implementation of Turbo decoder, Max-log-MAP algorithm has proven to be the most effective. Further, this chapter also discusses Interleavers in detail, in particular, QPP Interleaver due to its deterministic, algebraic structure and because it provides contention-free memory access. An exhaustive literature review of Turbo

codes and QPP Interleaver has also been done. Further, this chapter also talks about FPGA implementation of the basic and proficient turbo decoder.

Chapter 3
High-Performance VLSI Architectures for Turbo Decoders with QPP Interleaver

3.1 Introduction

Turbo encoder and decoder has been discussed in detail in chapter 2.The chapter also explains the need and types of interleavers, in particular, QPP interleaver. This chapter analyses different VLSI architectures for 3GPP LTE/LTE-advanced turbo decoders for trade-offs in terms of throughput and area requirement. Data flow graphs for standard SISO MAP (maximum a posteriori) turbo decoder, SW SISO MAP turbo decoder, PW SISO MAP turbo decoder have been presented, thus analyzing their performance. Two variants of quadratic permutation polynomial (QPP) interleaver have been proposed which tend to simplify the complexity of „mod" operator implementation and provide best compromise between area, delay and power dissipation. Implementation of decoder using one variant of QPP interleaver has also been discussed. A novel approach for area optimization has been proposed to reduce required number of interleavers for parallel window turbo decoder. Multi-port memory has also been used for parallel turbo decoder. To increase the throughput without any effective increase in area complexity, circuit-level pipelining and retiming have been used. Proposed architectures have been synthesized using Synopsys Design Compiler using 45-nm CMOS technology.

3.2 VLSI architectures of turbo decoders

Figure 3.1 shows the general block diagram of turbo decoder. The receiver converts the received analog data into digital form using analog to digital converter. The digital data is further de-multiplexed into three parts: systematic data, parity 1 and parity 2. The three parts are saved into three different memories. After that, a multiplexer chooses two out of the three data elements and processes one data element at a time. The control signals coming from the turbo control unit decide the selection of the data elements.

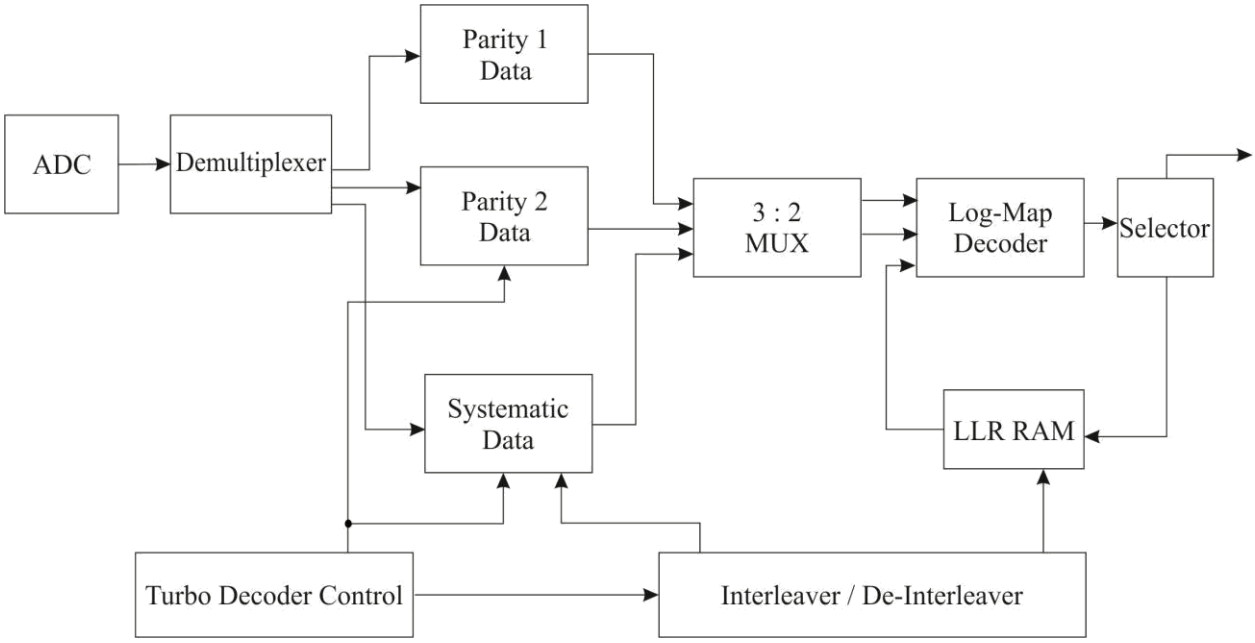


Figure 3.1 General block diagram of VLSI turbo decoder

The sequence of the data elements is processed by the Log-MAP decoder as a complex computational process, and the output a posteriori (APP) information is stored into LLR RAM. This process continues and after a few iterations, the final decision is made and transmitted as the final decoded bits.

3.2.1. Standard SISO MAP turbo decoder

The working of the standard turbo decoder is illustrated by the data flow graph (DFG), as shown in Figure 3.2. In the DFG, horizontal axis corresponds to sequence and vertical axis corresponds to the time taken. The working of the turbo decoder can be understood by the tiling diagram. The process starts with calculation of alpha metrics for the whole sequence. This is followed by calculation of beta metrics and LLRs in the subsequent clock cycles.

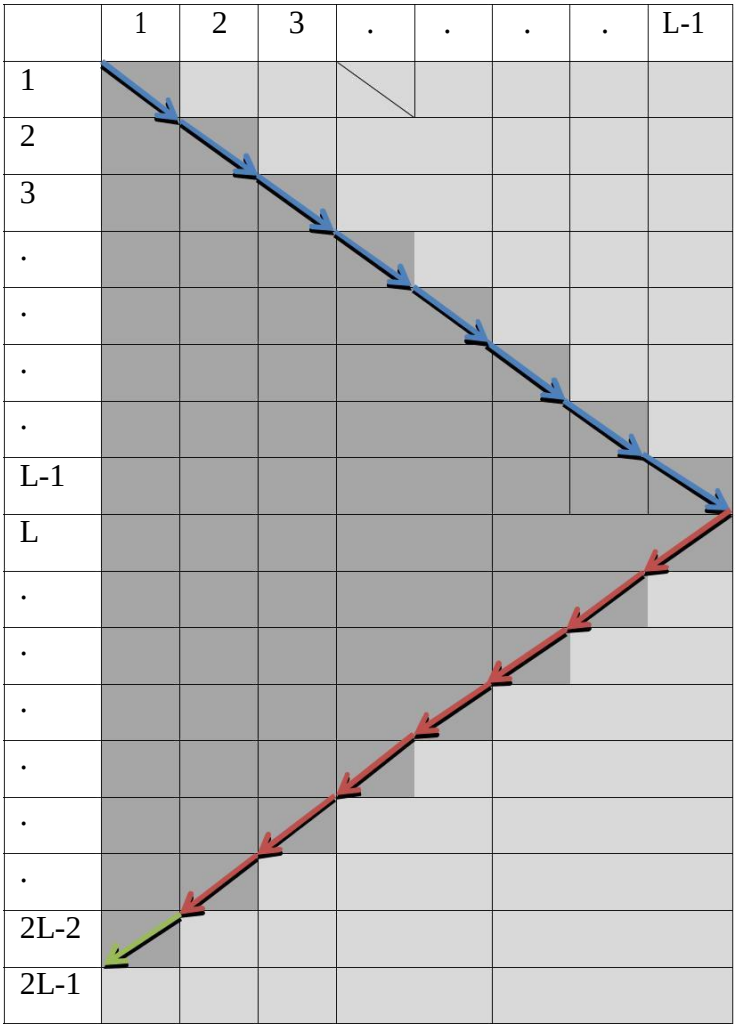


Figure 3.2 A simplified DFG representation of standard SISO MAP algorithm showing the resource utilization of α- and β-recursion flow with time

Representations:

- Metric Processing Unit (MPU); Input: , Output:
- Metric Processing Unit (MPU); Input: ,and Output :
- Metric Processing Unit (MPU); Input: , Output :
- Metric Processing Unit (MPU); Input: , and Output :

The implementation of standard turbo decoder is shown in Figure 3.3 [6]. As shown, input data is used to calculate branch metrics. These branch matrices are further used in the α-MPU (metric processing unit) for calculation of alpha metrics. The whole sequence is then stored into α-RAM. The Branch metrics are again calculated after this, and are used to calculate beta

metrics in β-MPU. The values of beta metrics, the branch metrics and alpha metrics are used to calculate the final log-likelihood ratio (LLR). The LLR values are stored into LLR RAM. These values can be used again in next iteration. Figure 3.4, 3.5 and 3.6 show implementation diagrams of different blocks of turbo decoders.

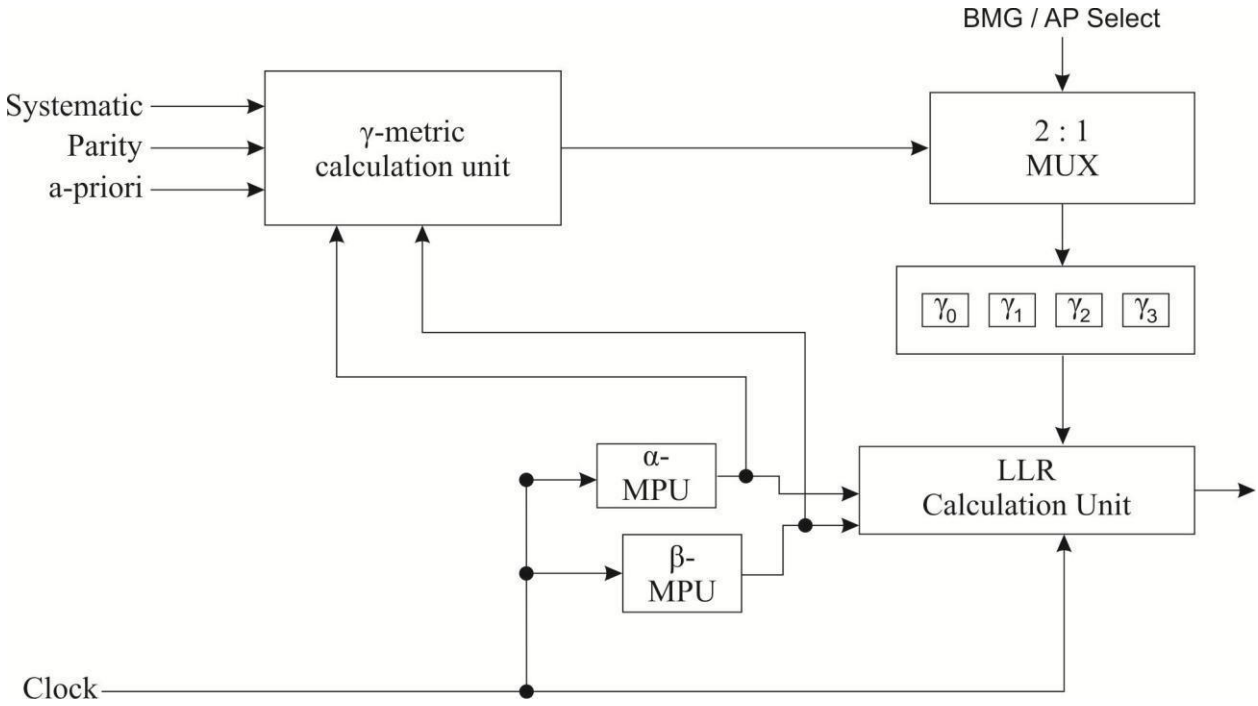


Figure 3.3 Standard SISO MAP Turbo Decoder

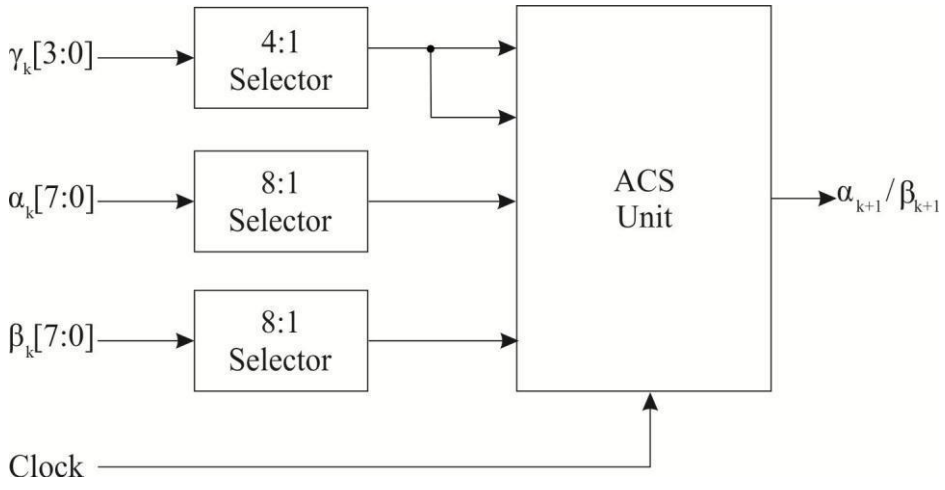


Figure 3.4 Implementation of α/β MPU

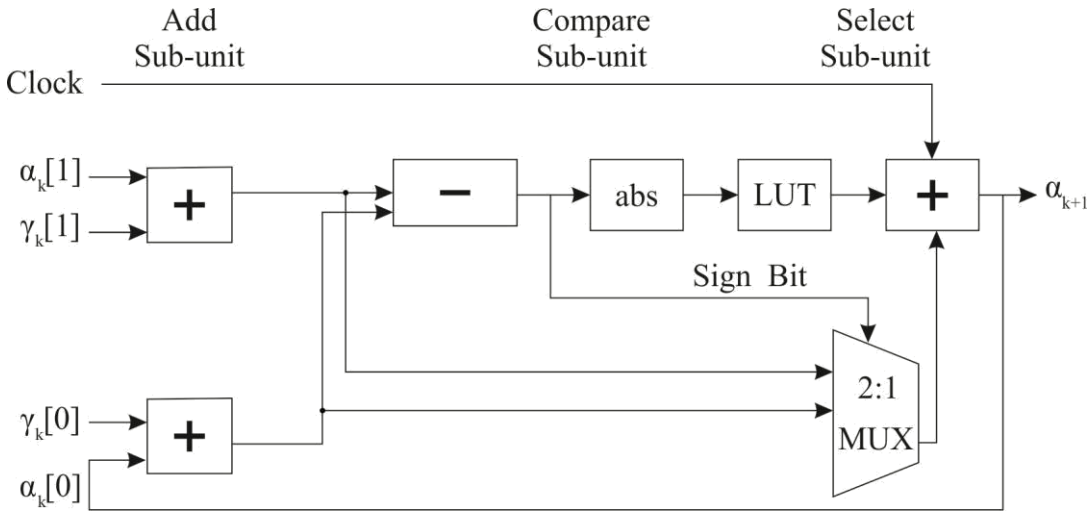


Figure 3.5 Implementation of Standard ACS unit

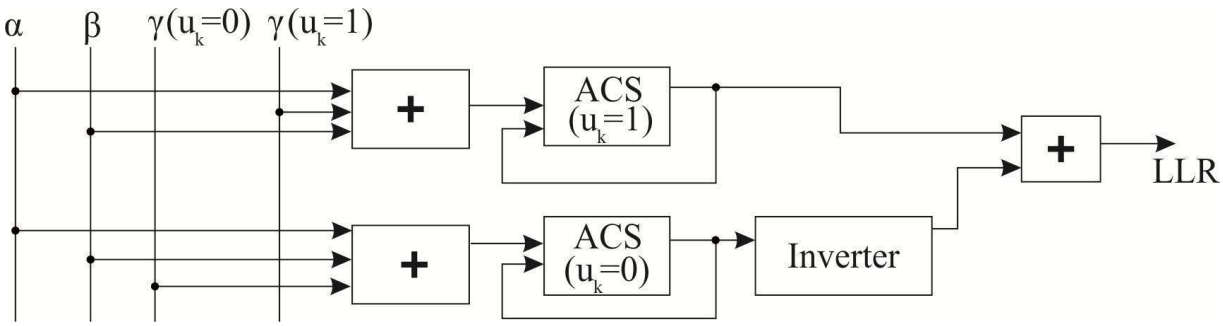


Figure 3.6 LLR unit implementation

3.2.2. Sliding window SISO MAP decoder

The major disadvantage of standard SISO MAP decoder is high latency and huge memory requirement; thus, truncated and continuous modes of communication are preferred over the terminated frame mode of communication. The DFG for SW turbo decoder is shown in Figure 3.7.

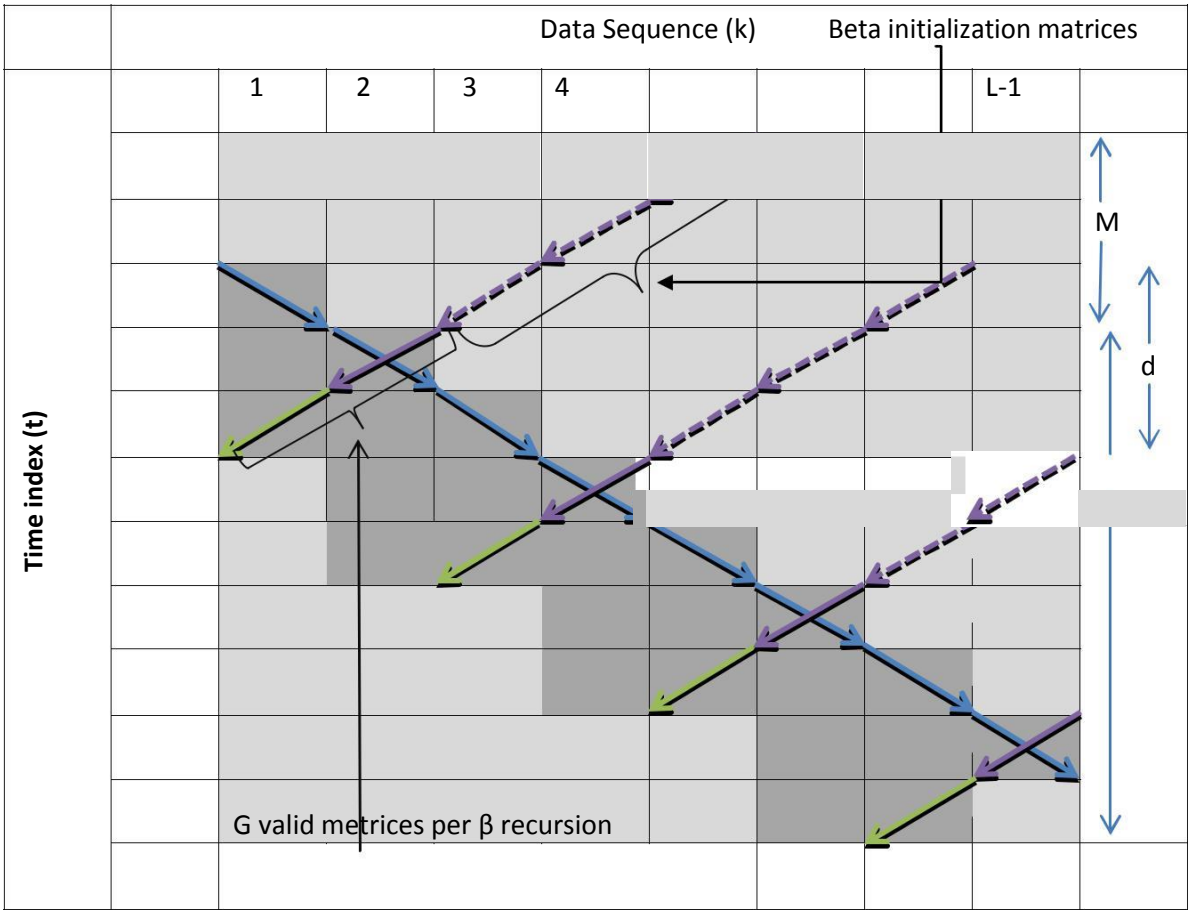


Figure 3.7 DFG for SW turbo decoder

The decoding process in sliding window (SW) SISO MAP decoder is initialized in the same manner as in the standard turbo decoder. It starts with the calculation of alpha metrics, followed by the calculation of beta metrics. After completion of initialization phase, LLR metrics are computed. The recursion pattern of SW SISO architecture can be configured by three parameters: (1) the metric warm-up depth, , (2) difference between the starting time of the - recursion flow and the ending time of the -recursion flow, and (3) the number of valid metric computations performed by the -recursion flow in relation to the total number of metric computations, . The parameters (width and height) of any feasible recursion pattern are given by:

(3.1)

(3.2)

To analyze the performance of SW architecture two parameters are required: 1) total decoding delay, and 2) metric storage lifetime,

Where (/) | | | / (3.3)

(/ { Σ }) (3.4)

Where number of code symbols to be decoded is represented by . As the total decoding delay and metric storage lifetimes get jointly minimized (according to a theorem stated in [59]), then by taking or , and taking optimized dimensions of the recursion pattern, minimized delay and storage lifetime be obtained.

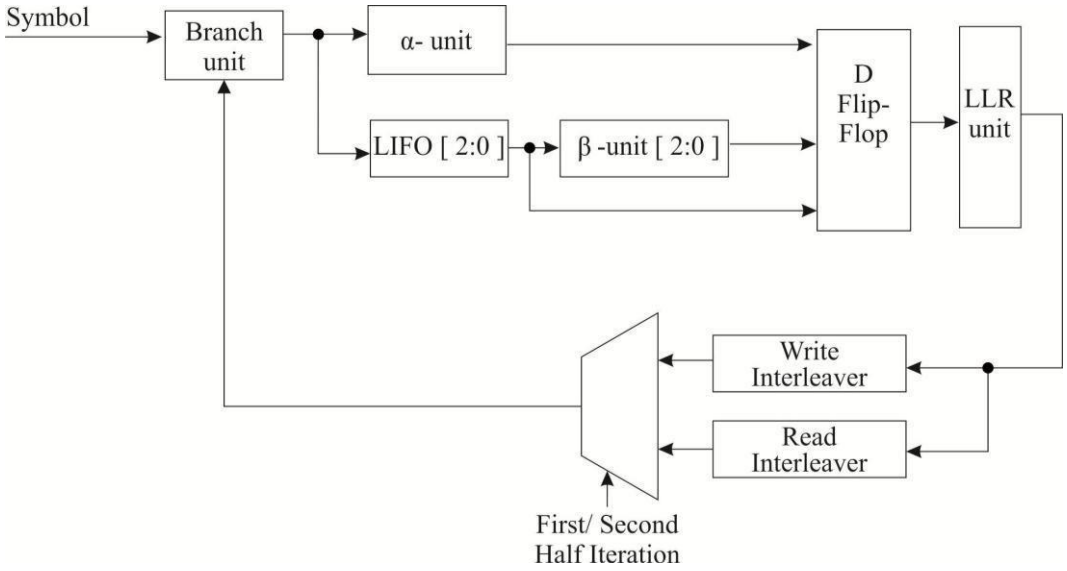


Figure 3.8 Implementation of SW SISO MAP decoder

Figure 3.8 shows implementation of SW SISO MAP decoder [131] and a comparison between SW SISO MAP decoder and the standard SISO MAP decoder is made on the basis of hardware requirement. Figure 3.8 shows that for SW SISO MAP decoder, hardware requirement is higher than standard SISO MAP decoder as SW SISO MAP decoder uses time multiplexing to reduce the latency. Another difference is in case of beta metrics computation, that is, instead of only one beta MPU, three beta MPUs have been used in SW SISO MAP decoder. First two MPUs work in coordination with each other. Third one is used for calculating the beta metrics for the last window in which termination metrics are available.

Section 3.2.4 discusses the hardware implementation of QPP interleaver [61], which is the standard interleaver used according to the 3GPP specifications. As the mod function involved in the QPP interleaver operation is highly complex to implement, so this section discusses variants of QPP interleaver, which provide a best compromise between area, delay and power dissipation.

3.2.3. Parallel window SISO MAP decoder

The main limitation of SW turbo decoder is that the decoding delay is greater than the time or clocks required for alpha metrics computation, which limits the maximum throughput of the decoder. The delay can be reduced with continuous frame mode of communication, which initiates with random probabilities of both alpha and beta metrics, and after obtaining a certain number of metrics, accurate probabilities are obtained. The actual decoding process starts after that. Parallel Window (PW) SISO algorithm involves usage of a number of SISOs in parallel. Each SISO processes one of the SWs. The main advantage of the PW turbo decoder is that it increases throughput proportionally Thus, there is a trade-off between the number of SISOs for error-correction performance and thereby having increased hardware complexity [50]. The main advantage of the PW turbo decoder is that it increases throughput proportionally. Figure 3.9 shows the basic DFG representing the working of a PW decoder. The hardware implementation is shown in Figure 3.10. As shown in Figures 3.9 and 3.10, the four-window hardware implementation involves the usage of four parallel BMG, beta, alpha and LLR units [132]. These units execute the data sequence in parallel in four ports; thus, four-port RAM and four-port LLR RAM are used.

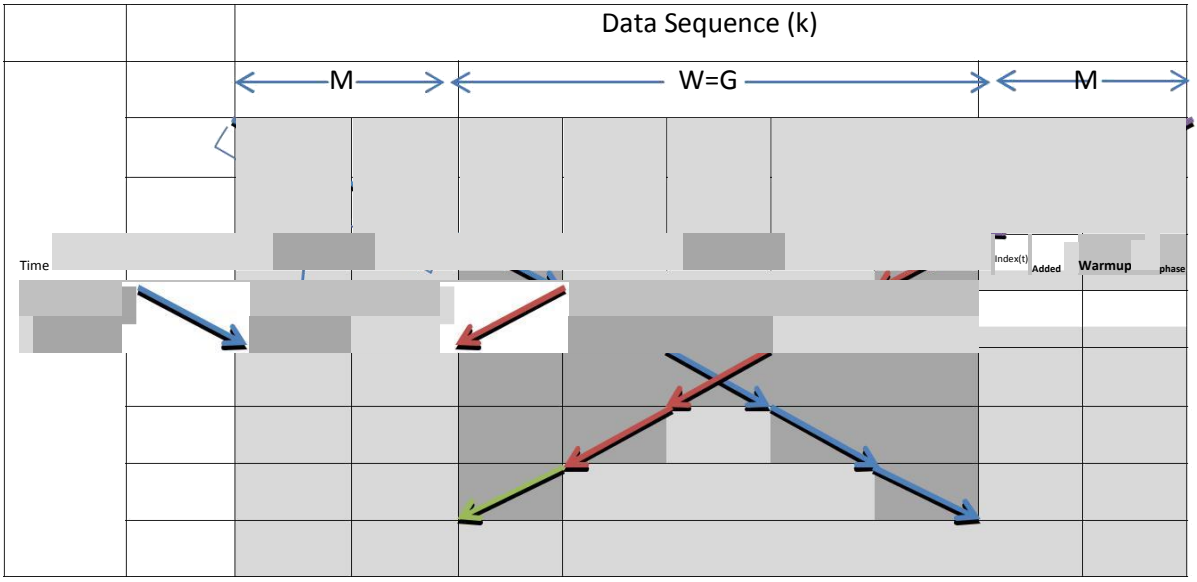


Figure 3.9 Basic DFG representing the working of a PW SISO Decoder

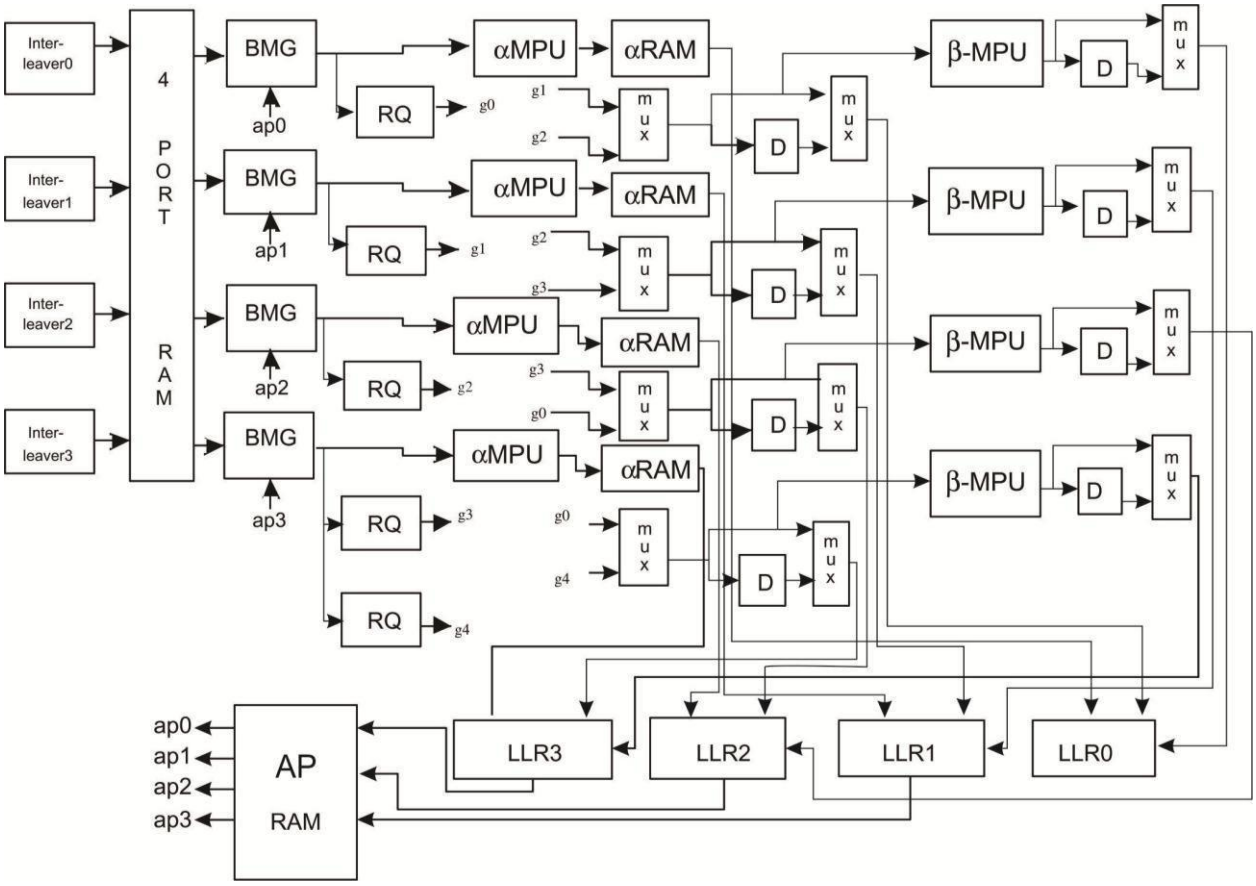


Figure 3.10 Hardware Implementation of Parallel Window Turbo Decoder

[132] 3.2.4 Hardware Implementation of QPP Interleaver

The QPP interleaver component of turbo decoder generates interleaved addresses corresponding to the sequential addresses in a recursive pattern.

If starting address to calculate interleaving address is assumed as , then the corresponding interleaved address, , is pre-computed as

(3.5)

For the next cycle, will be incremented by value , then, can be calculated recursively as:

( ) (3.6)

= ( ) (3.7)

where y (3.8)

y can also be calculated recursively as:

(3.9)

The initial value of y can be computed as:

(3.10)

QPP interleaver has numerous advantages over other types of interleavers: 1) it possesses a pure algebraic structure, 2) both interleaving and de-interleaving operations use same algorithm, and 3) gives contention-free memory access which facilitates parallelized decoding with high throughput, less latency, and optimum hardware usage [51].

The hardware implementation of mod operation is highly complex. Thus, two variants of QPP interleaver, variant1 and variant2, have been discussed in the sections below which account for low complexity and best compromise between area, delay and power dissipation.

3.2.4.1 Variant1 of QPP Interleaver:

The "mod" can be computed by using subtraction as the maximum value of or can be equal to . For these computations, three pre-calculated values of , and (are required.

The hardware implementation of interleaved address generator is shown in figure 3.11.

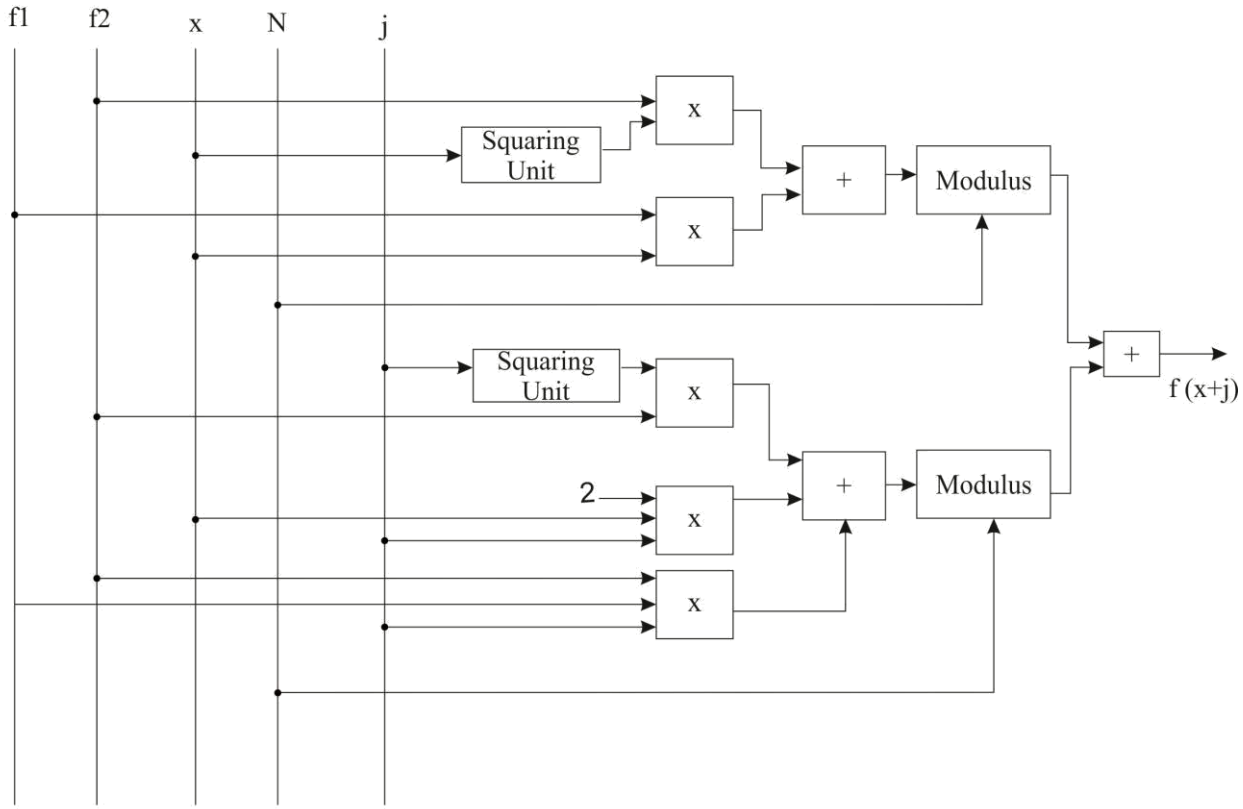


Figure 3.11 Implementation of Interleaved address generator

3.2.4.2 Variant2 of QPP Interleaver

The proposed design presents a combination of Random and QPP Interleaver. It combines the basic concept of QPP equation along with the use of two random functions for generating interleaved addresses. Instead of using the "mod" operator, the random functions make use of logical operators (in rand1 function, for values of which are powers of 2) and multiplication and division operators (in rand2 function, for values of which are non-

powers of 2) to get the interleaved addresses. The proposed design discusses a variant of QPP Interleaver which:

- Overcomes implementation difficulty of QPP algebraic equation; and
- Provides a best compromise between area, delay and power dissipation.

3.2.4.3 A Novel Area optimization technique for Parallel Window SISO Turbo Decoder

Figure 3.10 shows that numbers of interleavers are required for parallel window decoder. This may involve a lot of area. But if the difference between values of interleaved addresses

is carefully noticed, it is found that, at a particular time interval, there is a fix difference between all four interleaved values. Thus, the addresses can be simultaneously generated by using only one Interleaver and some adders and multiplexers, which can save some amount of area.

Table 3.1 Sequential versus Interleaved Addresses for PW turbo decoder using 4 windows

Sequential	Interleaved	Sequential	Interleaved	Sequential	Interleaved	Sequential	Interleaved
0	0	20	60	40	40	60	20
1	31	21	11	41	71	61	51
2	22	22	2	42	62	62	42
3	53	23	33	43	13	63	73
4	44	24	24	44	4	64	64
5	75	25	55	45	35	65	15
6	66	26	46	46	26	66	6
7	17	27	77	47	57	67	37
8	8	28	68	48	48	68	28
9	39	29	19	49	79	69	59
10	30	30	10	50	70	70	50
11	61	31	41	51	21	71	1
12	52	32	32	52	12	72	72
13	3	33	63	53	43	73	23
14	74	34	54	54	34	74	14
15	25	35	5	55	65	75	45
16	16	36	76	56	56	76	36
17	47	37	27	57	7	77	67
18	38	38	18	58	78	78	58
19	69	39	49	59	29	79	9

3.2.5 Circuit Level Improvements

Up gradations in standard window architecture at system level has been explained in the above sections. This section explains the up gradation of the different blocks of turbo decoder, so as to reduce critical path, to improve the speed and also to increase the maximum operating frequency.

3.2.5.1 Improved LLR (Log Likelihood Ratio) Unit:

System performance can be improved up to three times by using pipelining after every cascade structure. This will convert the block from one stage to four stage pipelined block, thereby increasing system performance up to three times.

3.2.5.2 Retimed ACS (Add Compare Select) Unit:

Pipelining technique cannot be applied to this unit. Thus, ACS unit using retiming technique [56] can be used to improve the speed. The proposed method uses two adders instead of three adders, and one LUT, thus reducing the delay. Figure 3.12 shows the implementation.

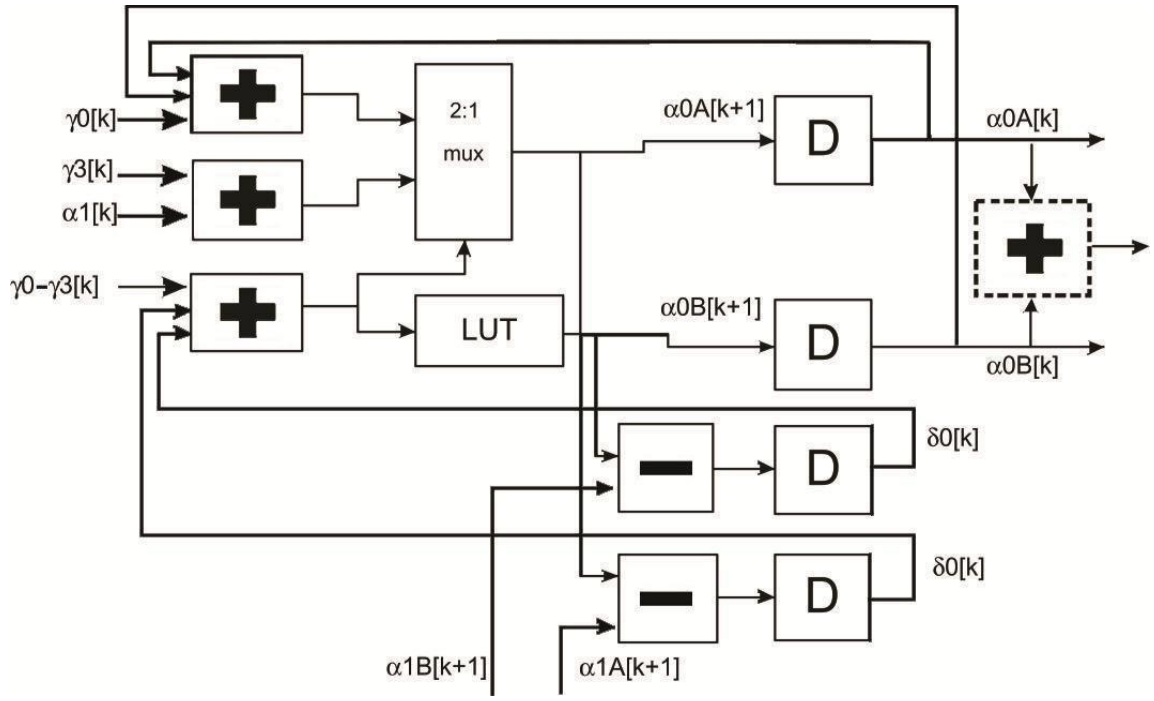


Figure 3.12 Hardware Implementation of Retimed ACS Unit [56]

3.3. Synthesis and simulation

3.3.1. Results for the proposed variant 2 QPP interleaver design

Simulation results prove that the conventional and the proposed design are functionally similar. The proposed design can be simulated for other values of N as well, as given by LTE standard. Three tools have been used to obtain the synthesis results, Xilinx, Quartus II (32-bit) and HDL Designer. The results show that the Xilinx tool does not support synthesis of "mod" operator for values of which are non-powers of 2.Tables 3.2–3.4 give comparison of various parameters of the proposed design and the existing design. The results show that the proposed design is the best compromise between area, delay and power dissipation.

● Device utilization for the device 5VLX30FF676 obtained by using Precision Synthesis software of HDL Designer tool for the conventional and the proposed designs is shown in the Table 3.2

Table 3.2 Device utilization report for N=40, f1=3 and f2=10 obtained by using precision Synthesis software

	Conventional Design	**Proposed Design**
Global Buffers	3.13%	3.13%
LUTs	4.98%	2.67%
CLB Slices	5%	2.69%
Dffs or Latches	1.22%	1.25%
IOs	63%	75%
Block RAMs	0.00%	0.00%
DSP48Es	0.00%	0.00%

- Total Thermal Power Dissipation and device utilization for the device EP4CGX30CF23C6 of Cyclone4 GX Family, obtained by using QuartusII (32-bit) Tool , for the conventional and the proposed design is shown in the table below:

Table 3.3 Thermal power dissipation and device utilization report for N=40, obtained by using QuartusII (32-bit) tool

	ConventionalDesign	**Proposed Design**
Total Thermal Power Dissipation	159.44mW	157.67mW
Total Logic Elements	9828/29,446	382/29,446
Total I/O pins	251	252

- Total Equivalent Gate Count and Total Delay for xc2s50e-ft256-7 device of Spartan2e family, obtained by using Xilinx Tool, for the conventional and the proposed design is shown in the table below:

Table 3.4 Total Equivalent gate count and total delay report for N=40, obtained by using Xilinx tool

	Conventional Design	**Proposed Design**
Total Equivalent Gate Count	No Result*	5062
Total Delay	No Result*	6.140ns

*The code did not get synthesized due to the “mod” operator.

3.3.2. Results for the proposed decoder design

Six different architectures of Radix-2 turbo decoders (with block length 80 bits) havebeen implemented:

(1) Standard turbo decoder

(2) SW turbo decoder

(3) PW turbo decoder

(4) PW turbo decoder using improved interleaving technique

(5) PW-based turbo decoder with pipelining

(6) Retimed PW-based turbo decoder with pipelining

PW-based turbo decoder: Four-window parallel decoders have been used in this architecture.

PW turbo decoder using improved interleaving technique: This architecture uses a single QPP interleaver and adders instead of using multiple interleavers.

PW-based turbo decoder with pipelining: This architecture uses four-stage pipelining in LLR unit to reduce critical time.

Retimed PW-based turbo decoder with pipelining: This architecture uses a retimed design of ACS unit to further reduce the delay.

ModelSim 10.1 student edition has been used for simulation, and synthesis has been done using Synopsys Design Compiler tool version D-2010.03-SP1. Results for all the architectures on 180-nm technology tabulated in Table 3.5 show increased throughput for advanced decoders as compared to the standard decoder but at the same time, also show increase in the area and power requirement. Improved interleaving technique when applied to PW decoders saves considerable amount of area. Simulation results corresponding to latency and throughput for different architectures of turbo decoders have been shown in Figures 3.13-3.19.

Table 3.5 Synthesis result for different architectures of turbo decoders

	Standard Turbo Decoder	Sliding Window based approach	Parallel window based approach	Parallel Window based approach using improved QPP	Parallel Window based approach with pipelining	Parallel Window based approach with retiming
Clock frequency	63.6 MHz	63.6MHz	63.6MHz	63.6MHz	144.9 MHz	188.6 MHz
Throughput (1 iteration)	7.5 Mbps	8.5 Mbps	19.5 Mbps	19.5 Mbps	40 Mbps	52.8 Mbps
Latency (us)	2.7	2.0	1.4	1.4	0.64	0.498
Cell Area ($*10^6$)	957166	1455825	2501889	2499499	2662879	3117934
Dynamic Power (mw)	5.4831	20.8597	50.9684	50.0929	220.7872	347.7969
Static Power (uw)	122.67	180.48	278.32	276.78	298.59	352.37

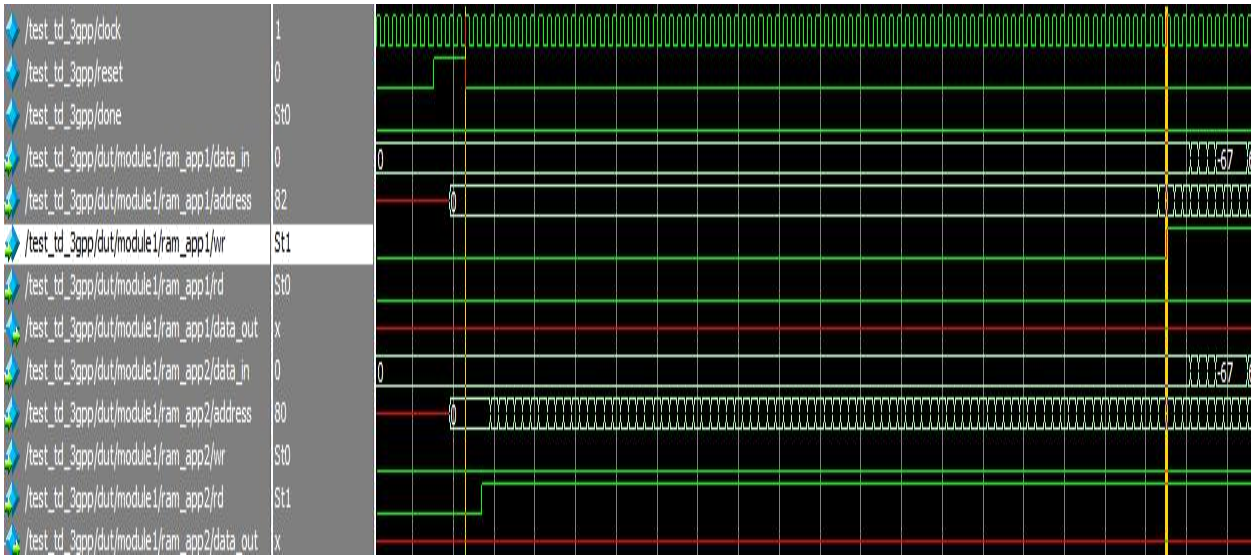


Figure 3.13 Simulation Diagram Corresponding to latency for Standard SISO MAP Turbo Decoder

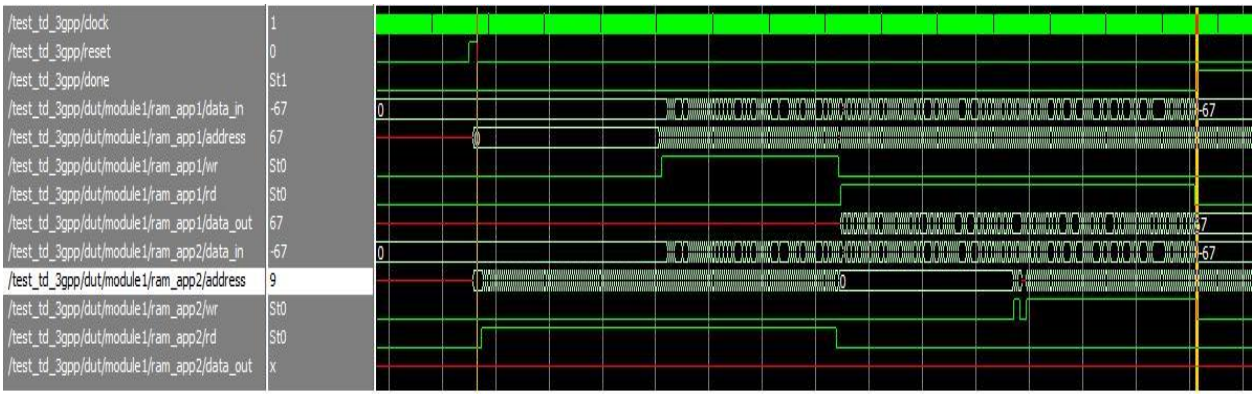


Figure 3.14 Simulation Diagram corresponding to throughput for Standard SISO MAP Decoder

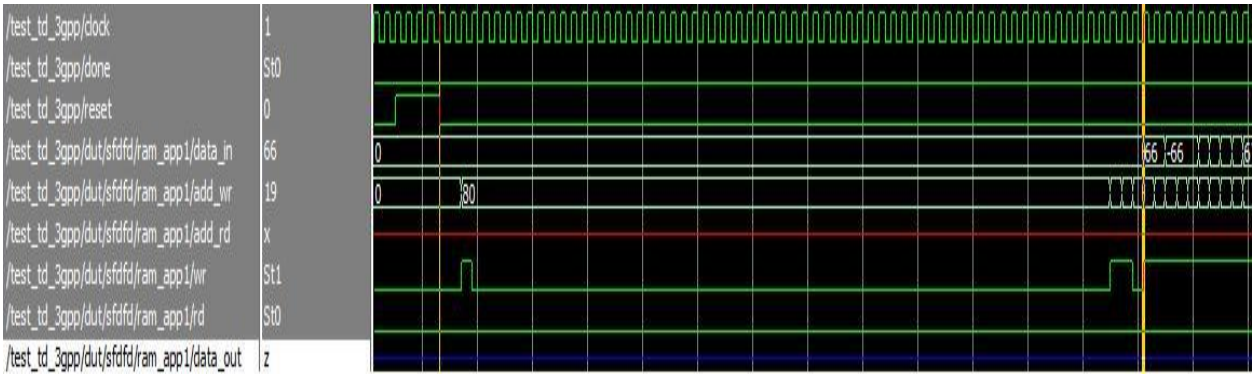


Figure 3.15 Simulation Diagram Corresponding to latency for SW SISO MAP Turbo Decoder

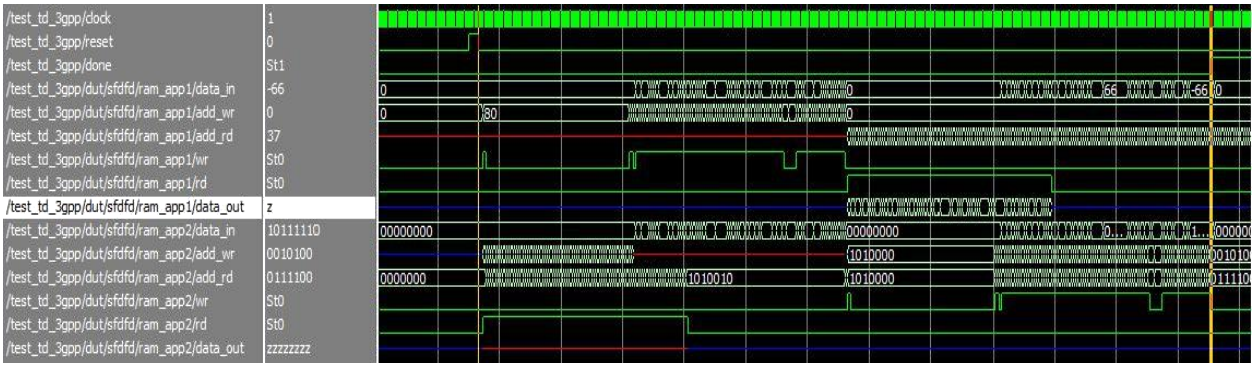


Figure 3.16 Simulation Diagram Corresponding to throughput for SW SISO MAP Turbo Decoder

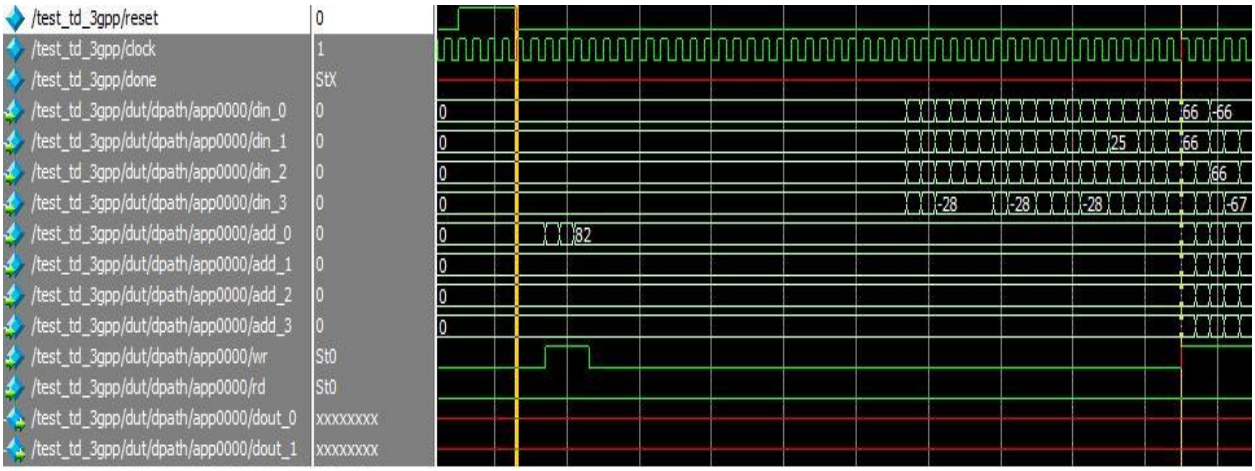


Figure 3.17 Simulation Diagram Corresponding to latency for PW and PW using improved QPP interleaver SISO MAP Turbo Decoder

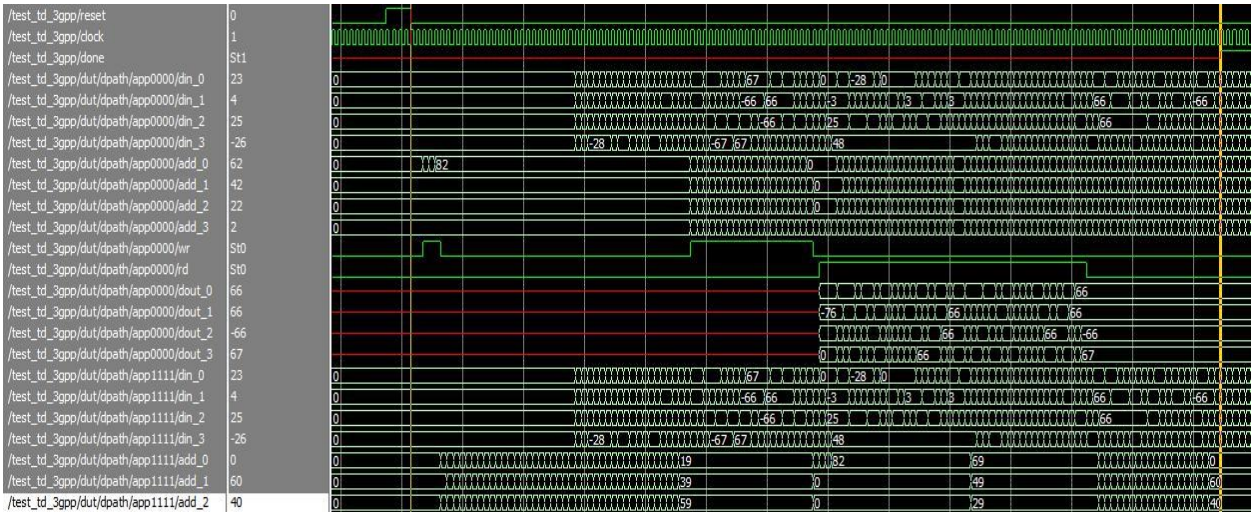


Figure 3.18 Simulation Diagram Corresponding to throughput for PW and PW using improved QPP interleaver SISO MAP Turbo Decoder

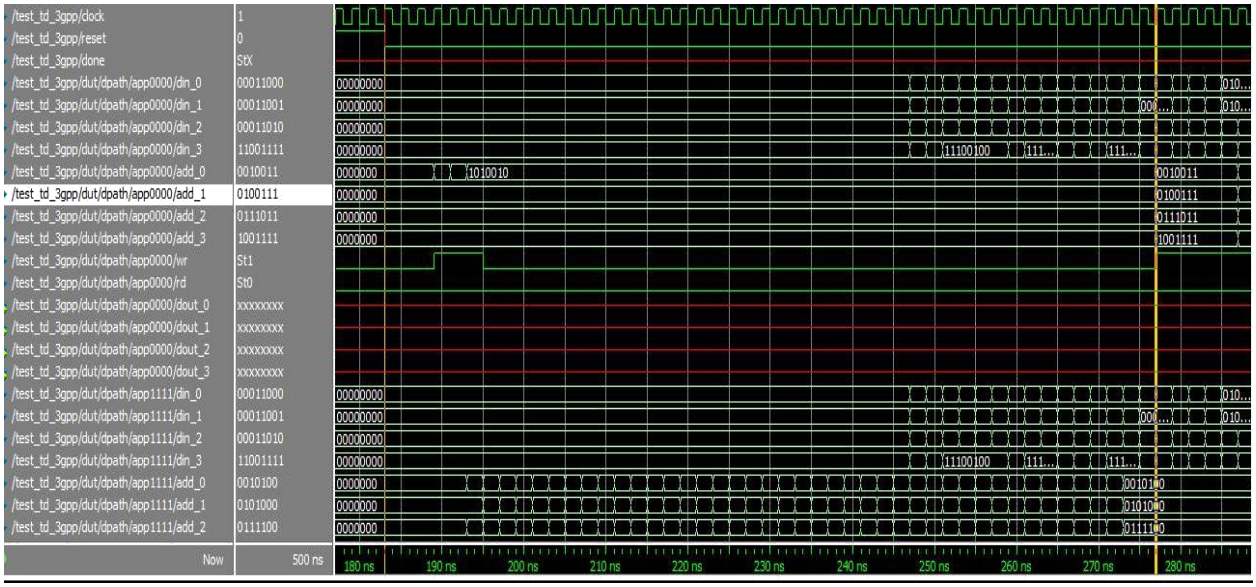


Figure 3.19 Simulation Diagram Corresponding to latency for PW with Pipelining and PW with retiming SISO MAP Turbo Decoder

3.4 Conclusion

Six different architectures for turbo decoders have been designed and analyzed for their trade-offs in terms of latency, throughput and area requirement. The throughput increases from standard window towards SW and PW, but at the same time area requirement also increases proportionally. By applying circuit-level pipelining and retiming, throughput increases in larger proportion when compared to area increment. A novel interleaving technique gives area-optimized results. The proposed technique can be applied to turbo decoder for any parallelism degree.

Chapter 4
Novel Interleaver Design for Turbo Codes

4.1 Introduction

Long term evolution (LTE) was developed by the third generation partnership project (3GPP) for the Wideband Code Division Multiple Access (WCDMA) based air interface [125]. The channel coding scheme for LTE is Turbo coding [69], as discussed in chapter 2. Turbo decoder is one of the major blocks in a LTE wireless receiver [126–11]. VLSI sequential architectures of Turbo decoders suffer from high decoding latency due to the iterative decoding process, the interleaving/de- interleaving process between iterations and the forward and backward recursion in the MAP (maximum a posteriori) algorithm [70–72]. The interleaving process (done by interleavers) writes the soft values generated by the MAP decoder into random/ pseudo-random positions after permutation. Latency can be lowered by reducing the number of required decoding iterations, but that may adversely affect the coding gain. Number of SISO decoder units can be operated in parallel, each processing one of the sliding windows to implement parallel decoding schemes. An essential requirement to implement such parallel decoding architecture is parallel interleaving of outputs of multiple concurrent SISO units, which may improve error correction performance, latency, and throughput of the entire decoder [51].Parallel turbo decoder implementation can be achieved by using a contention-free interleaver, such as, Quadratic permutation polynomial (QPP) interleaver. In the literature, many interleaver architectures have been extensively investigated [73-92, 93-95]. This chapter proposes a novel interleaver design, a variant of QPP interleaver, for turbo codes, which permutes a sequence of bits with the same statistical distribution as a conventional QPP interleaver and performs as well as or better than the conventional QPP.

4.2 Proposed Design of Novel Interleaver

4.2.1 Theoretical Perspective

The „mod‟ operation in QPP algebraic equation can be difficult to implement in hardware if the operands are not known in advance. The proposed design addresses the implementation difficulty of QPP algebraic equation and also provides a best compromise between area delay and power dissipation. Thus a variant of QPP Interleaver has been proposed which would be easy to implement as compared to the previous designs and would provide a best compromise between area, delay and power dissipation.

Implementation ease

Most of the VHDL synthesis tools do not support the „„„mod‟‟‟ operator but multiplication and division by powers of 2 is supported by many synthesis tools. Logical operators have the advantage that they can be mapped directly to logic gates and also that their synthesis is quite easy. The proposed design uses logical operators (in „„„rand1‟‟‟ function, for values of N which are powers of 2) and multiplication and division operators (in „„„rand2‟‟‟ function, for values of N which are non-powers of 2) to get the interleaved addresses.

Efficient Trade-Off between Area, Delay and Power Dissipation

The implementation of arithmetic operators like division and multiplication, and logical operators involves less delay, less resource usage and less power dissipation as compared to the „„„mod‟‟‟ operator implementation. To prove the above mentioned claims, the proposed design (using multiplication, division and logical operators) has been simulated and synthesized for two values of N, 48 (nonpowers of 2) and 64 (powers of 2) in this chapter. The proposed design may be tested for other values of f1, f2 and N as well, as given by LTE standard. This chapter also presents a comparison between various parameters of the proposed variant of QPP and the conventional QPP Interleaver.

4.2.2 Hardware Implementation

The proposed design has been implemented in VHDL by making use of two procedures „„rand1“““ and „„rand2“““ (figure4.1). These two procedures further make use of modules like „„powers“““, „„nonpowers“““, „„I2V““““ and „„V2I“““. „„Powers“““ module (figure4.2) computes mod value for values of N which are powers of 2, simply by using arithmetic and logical operator. „„rand1“““ module (figure4.3) makes use of „„powers“““ module along with other interleaver input parameters and finds out the interleaved address values. Similarly, „„nonpowers“““ module (figure4.4) computes mod for values of N which are not the powers of 2, by using arithmetic and relational operators. „„rand2“““ module then makes use of „„nonpowers“““ module along with other interleaver input parameters to find out the interleaved address values. „„I2V““““ (figure4.5) and „„V2I““““ (figure4.6) modules provide integer to vector and vector to integer conversions respectively. The hardware implementation of each of these modules has been shown below in a sequential manner, starting with the main design module. The hardware implementation of „„rand2“““ module is similar to that of „„rand1“““, except that „„rand2“““ module has been implemented by making use of „„nonpowers“““ module instead of „„powers“““ module.

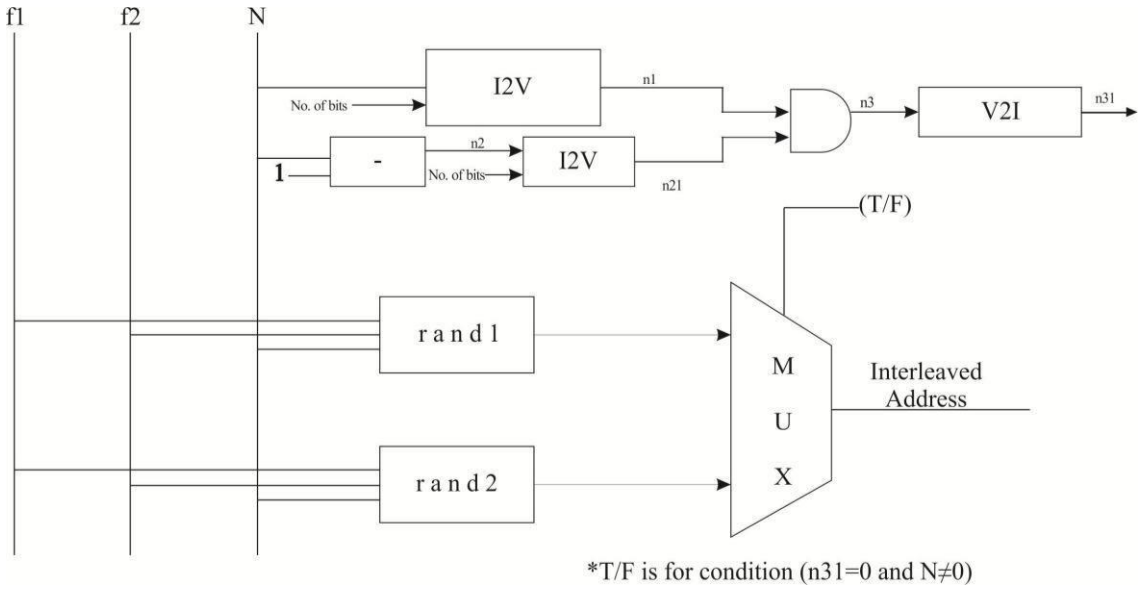


Figure4.1 Hardware implementation of the main module of the proposed design

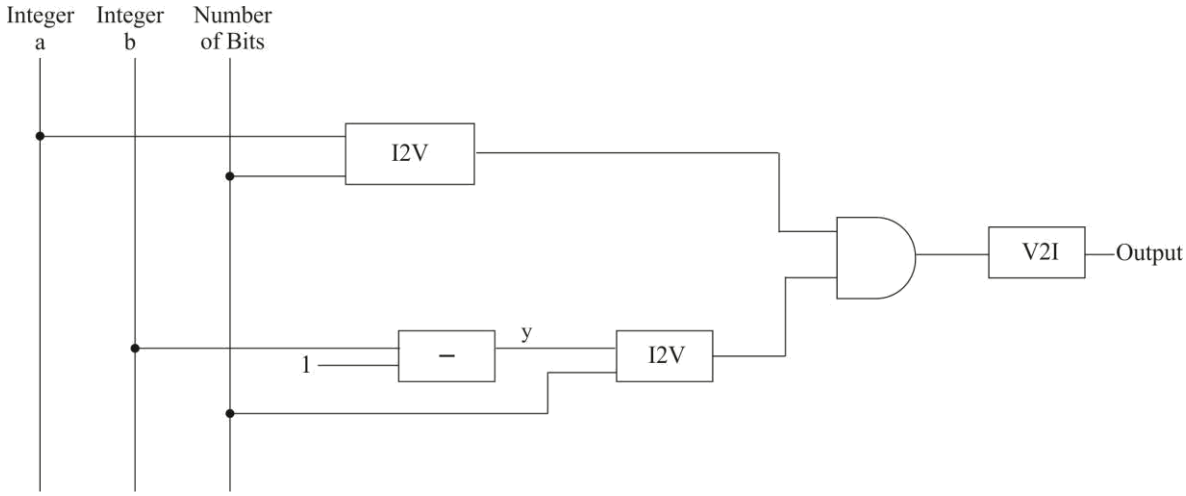


Figure4.2 Hardware implementation of ‘‘powers’’

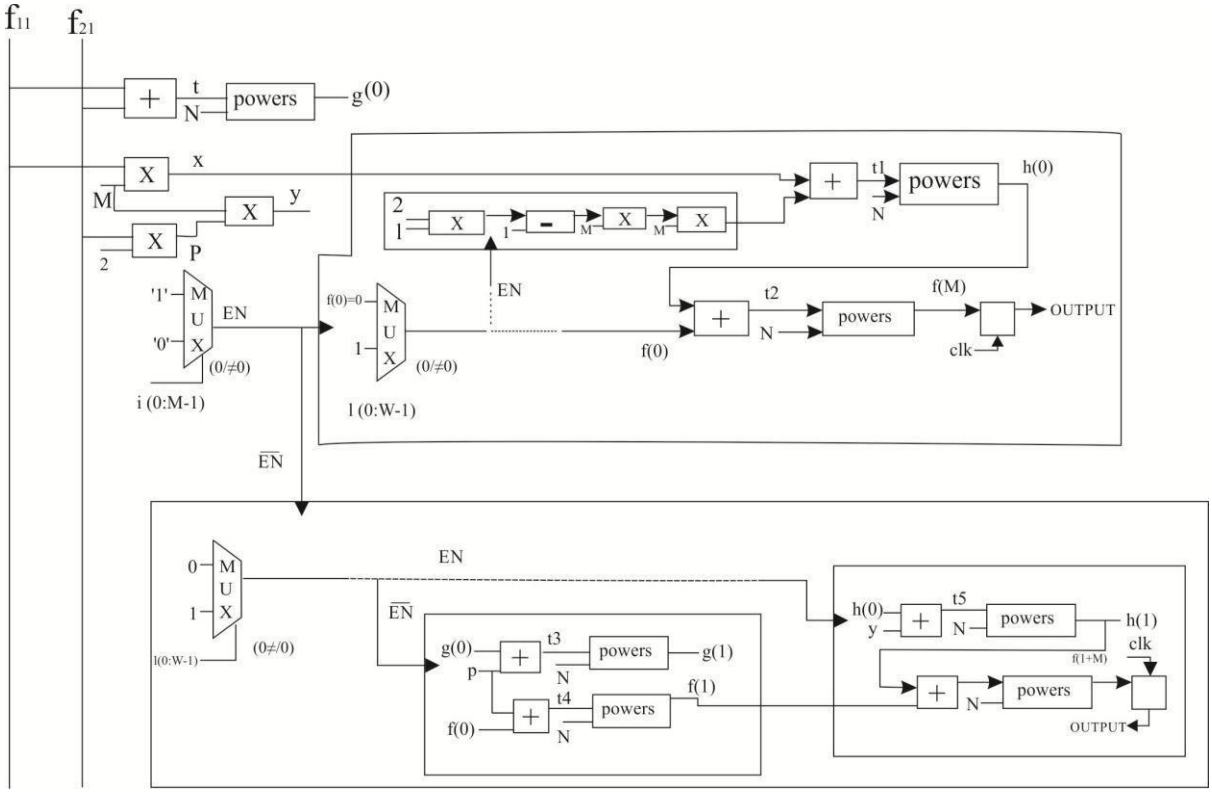


Figure4.3 Hardware implementation of ‘‘rand1’’

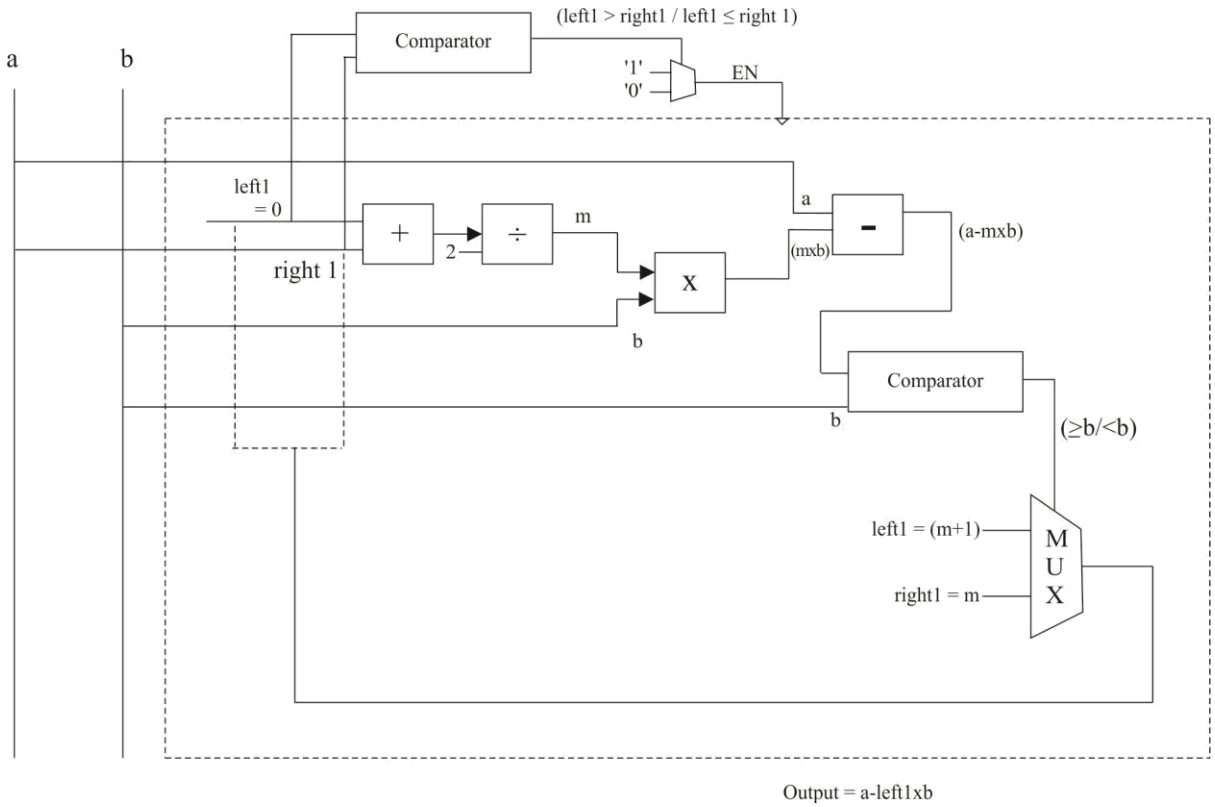


Figure4.4 Hardware implementation of ‘‘nonpowers’’

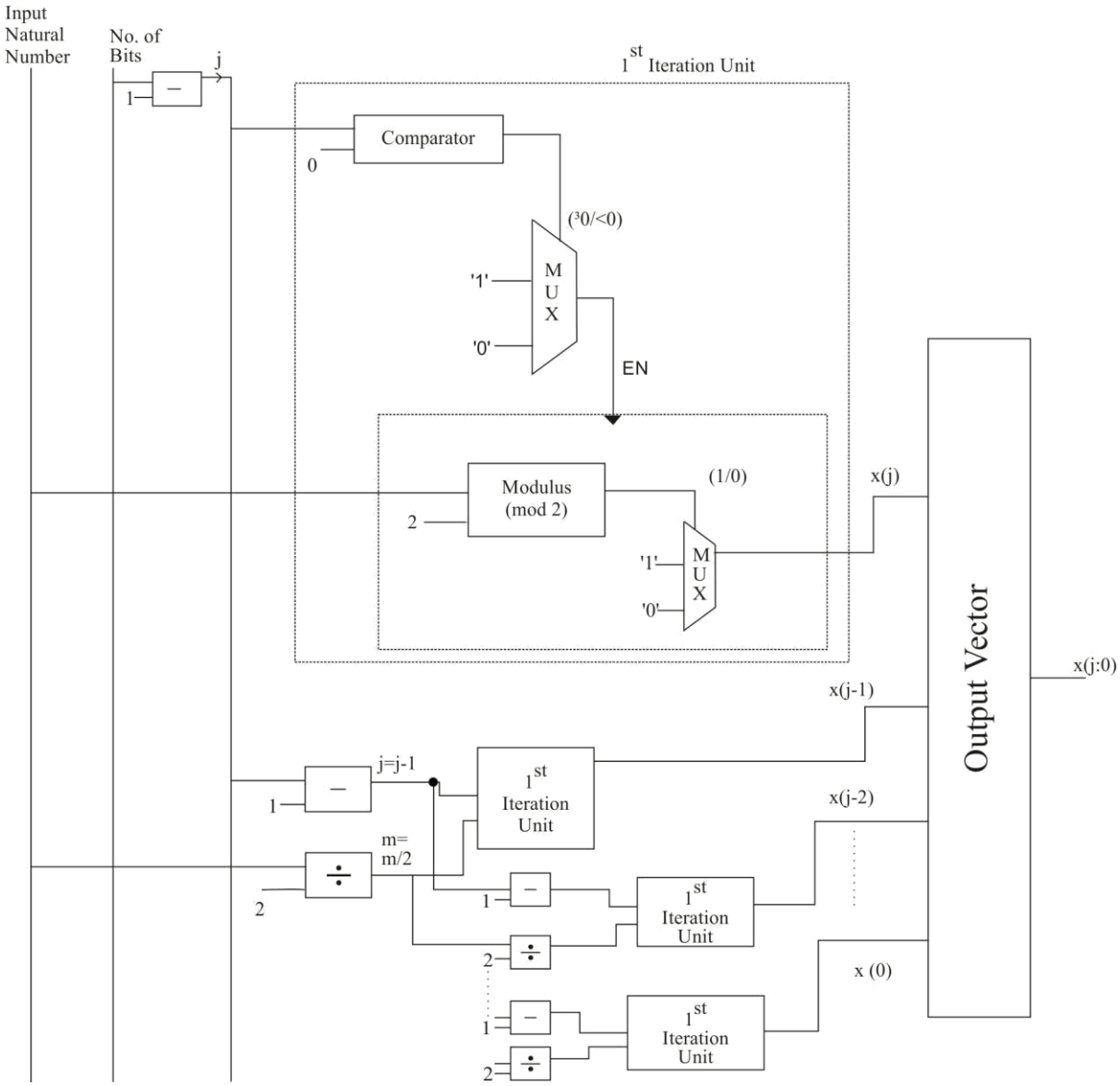


Figure4.5 Hardware implementation of ‘‘I2V’’

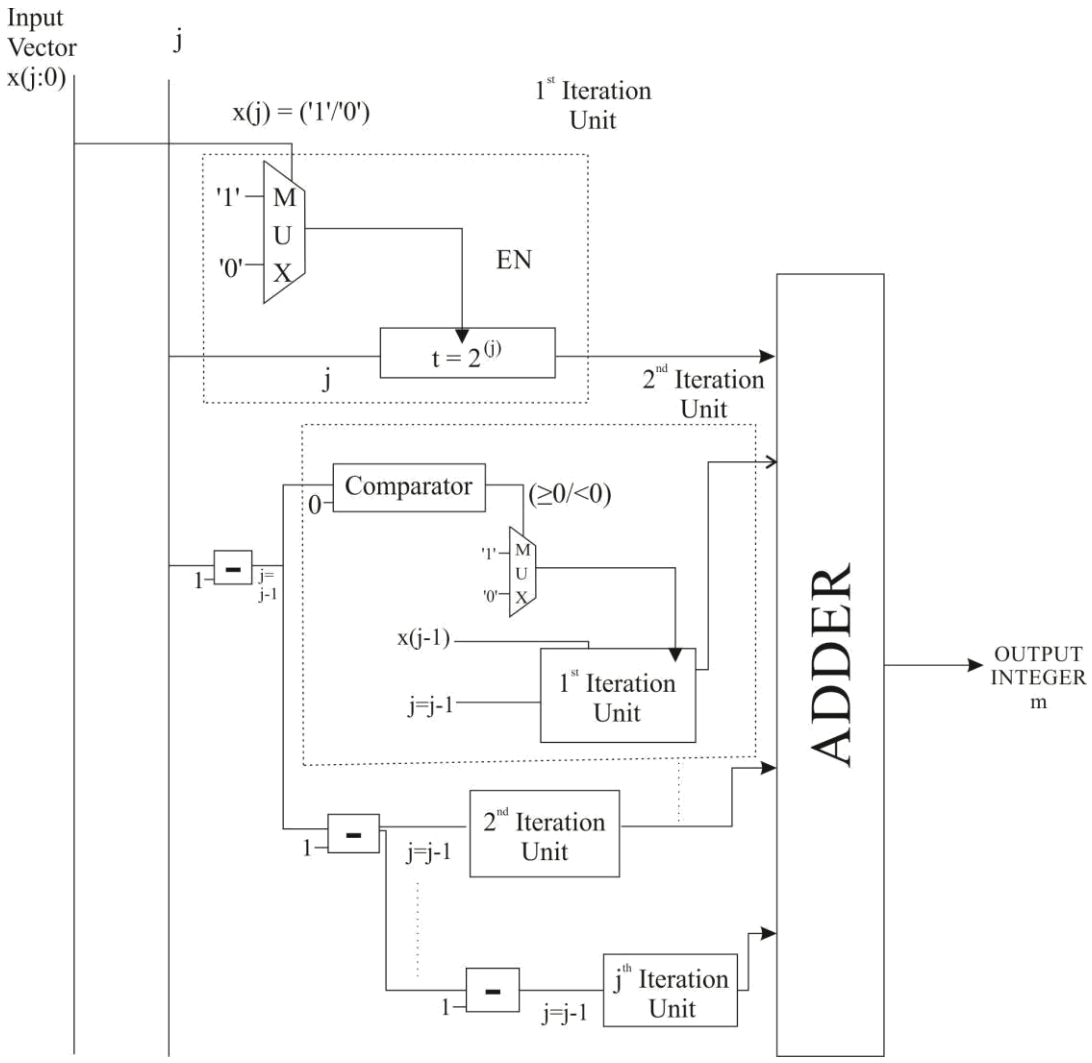


Figure4.6 Hardware implementation of ‘‘V2I’’

4.3 Simulation and Synthesis Results

4.3.1 Simulation Results

To prove the functional similarity between the proposed design and the conventional design, simulation results have been shown in tabular form in the tables 4.1 and 4.2 for both the designs, by considering two values of , that is, 48 and 64. The proposed design generates the interleaved set of addresses similar to the existing interleaver. Simulation has been done through HDL Designer tool of Mentor Graphics.

Table 4.1 Sequential vs. Interleaved Addresses for

Sequential	Interleaved	Sequential	Interleaved	Sequential	Interleaved	Sequential	Interleaved	Sequential	Interleaved
0	0	15	57	30	18	45	11	60	36
1	23	16	48	31	41	46	2	61	59
2	14	17	7	32	32	47	25	62	50
3	37	18	62	33	55	48	16	63	9
4	28	19	21	34	46	49	39		
5	51	20	12	35	5	50	30		
6	42	21	35	36	60	51	53		
7	1	22	26	37	19	52	44		
8	56	23	49	38	10	53	3		
9	15	24	40	39	33	54	58		
10	6	25	63	40	24	55	17		
11	29	26	54	41	47	56	8		
12	20	27	13	42	38	57	31		
13	43	28	4	43	61	58	22		
14	34	29	27	44	52	59	45		

Table 4.2 Sequential vs. Interleaved Addresses for

Sequential	Interleaved	Sequential	Interleaved	Sequential	Interleaved	Sequential	Interleaved	Sequential	Interleaved
0	0	10	22	20	44	30	18	40	40
1	19	11	41	21	15	31	37	41	11
2	14	12	36	22	10	32	32	42	6
3	33	13	7	23	29	33	3	43	25

4	28	14	2	24	24	34	46	44	20
5	47	15	21	25	43	35	17	45	39
6	42	16	16	26	38	36	12	46	34
7	13	17	35	27	9	37	31	47	5
8	8	18	30	28	4	38	26		
9	27	19	1	29	23	39	45		

4.3.2 Synthesis Results

The conventional and the proposed designs have been synthesized by using Precision Synthesis software of HDL Designer Tool. The synthesis result provides device utilization for the device 5VLX30FF676 of Virtex-V family. Total Thermal Power Dissipation and device utilization for the device EP4CGX110DF31C7 of Cyclone4 GX Family have been obtained by using Quartus II (32-bit) Tool. Total Equivalent Gate Count and Total Delay for xc2s50e-ft256-7 device of Spartan2e family have been obtained by using Xilinx Tool. The results have been obtained for the conventional as well as the proposed design and a comparison has been drawn.

Table 4.3 Device utilization for the device 5VLX30FF676 of Virtex-V family for

	Conventional Design	**Proposed Design**
Global Buffers	3.13%	3.13%
LUTs	1.55%	0.77%
CLB Slices	1.56%	1.02%
Dffs or Latches	1.07%	1.02%
IOs	99%	99%
Block RAMs	0.00%	0.00%
DSP48Es	0.00%	0.00%

Table 4.4 Total Thermal Power Dissipation and device utilization for the device EP4CGX110DF31C7 of Cyclone4 GX Family for

	Conventional Design	Proposed Design
Total Thermal Power Dissipation	178.31mW	178.90mW
Total Logic Elements	400/109424	397/109424
Total I/O pins	395/508	396/508

Table 4.5 Total Equivalent Gate Count and Total Delay for xc2s50e-ft256-7 device of Spartan2e family for

	Conventional Design	Proposed Design
Total Equivalent Gate Count	5083	5062
Total Delay	6.788ns	6.140ns

Table 4.6 Device utilization for the device 5VLX30FF676 of Virtex-V family for

	Conventional Design	Proposed Design
Global Buffers	3.13%	3.13%
LUTs	4.15%	2.67%
CLB Slices	4.17%	2.69%
Dffs or Latches	1.43%	1.25%
IOs	75%	75%
Block RAMs	0.00%	0.00%
DSP48Es	0.00%	0.00%

Table 4.7 Total Thermal Power Dissipation and device utilization for the device EP4CGX110DF27C7 of Cyclone4 GX family for

	Conventional Design	Proposed Design
Total Thermal Power Dissipation	167.81mW	166.27mW
Total Logic Elements	11565/109424	378/109424
Total I/O pins	299/426	300/426

Table 4.8 Total Equivalent Gate Count and Total Delay for xc2s50e-ft256-7 device ofSpartan2e family for

	Conventional Design	Proposed Design
Total Equivalent Gate Count	No Result*	5062
Total Delay	No Result*	6.140ns

*The code did not get synthesized due to the “mod” operator.

4.4 Conclusion

The proposed design has been designed and analyzed for total delay, total thermal power dissipation, total equivalent gate count and total device utilization.

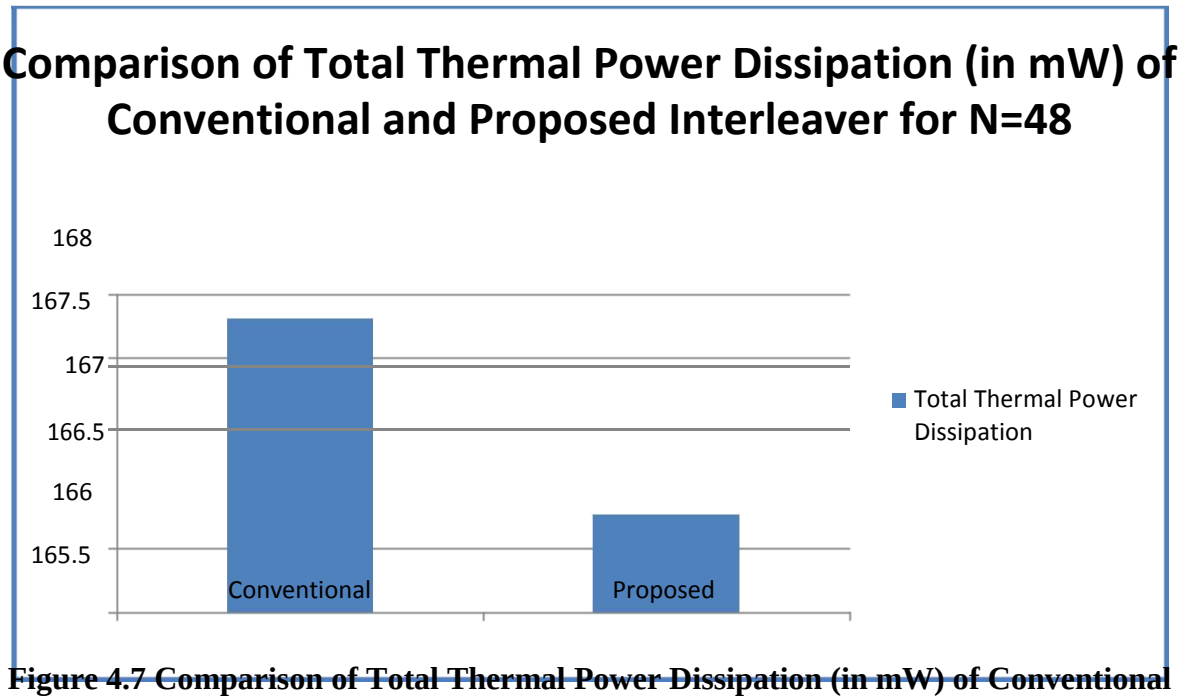


Figure 4.7 Comparison of Total Thermal Power Dissipation (in mW) of Conventional and Proposed Interleaver for

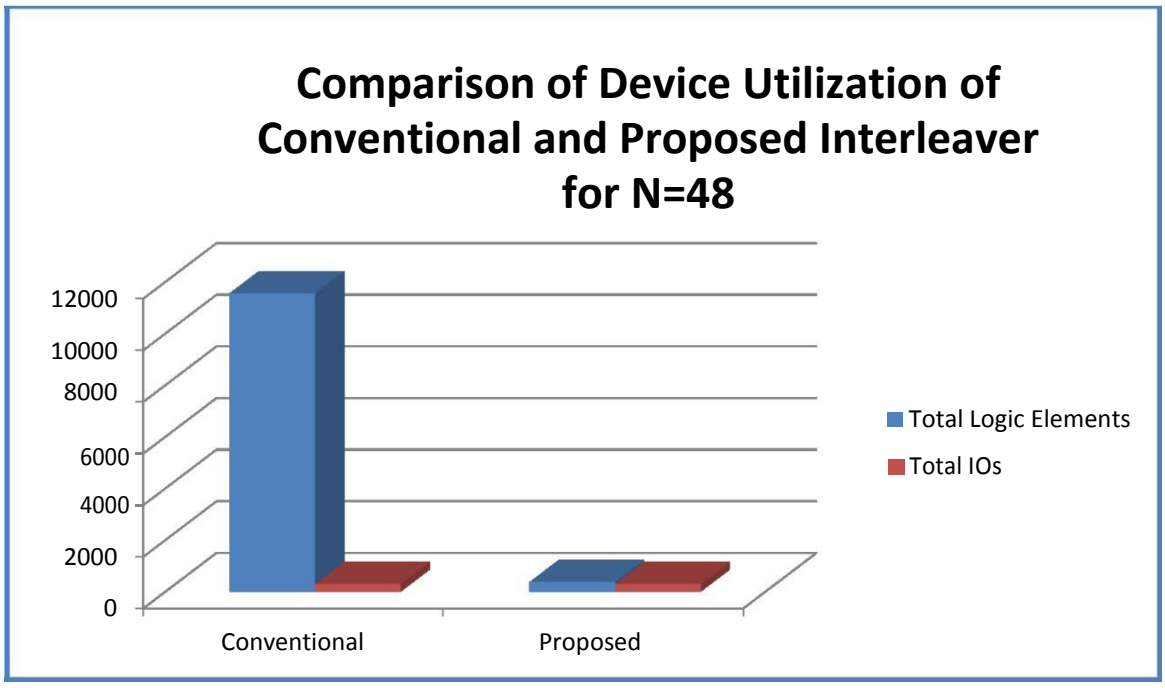


Figure 4.8 Comparison of Device Utilization of Conventional and Proposed Interleaver for

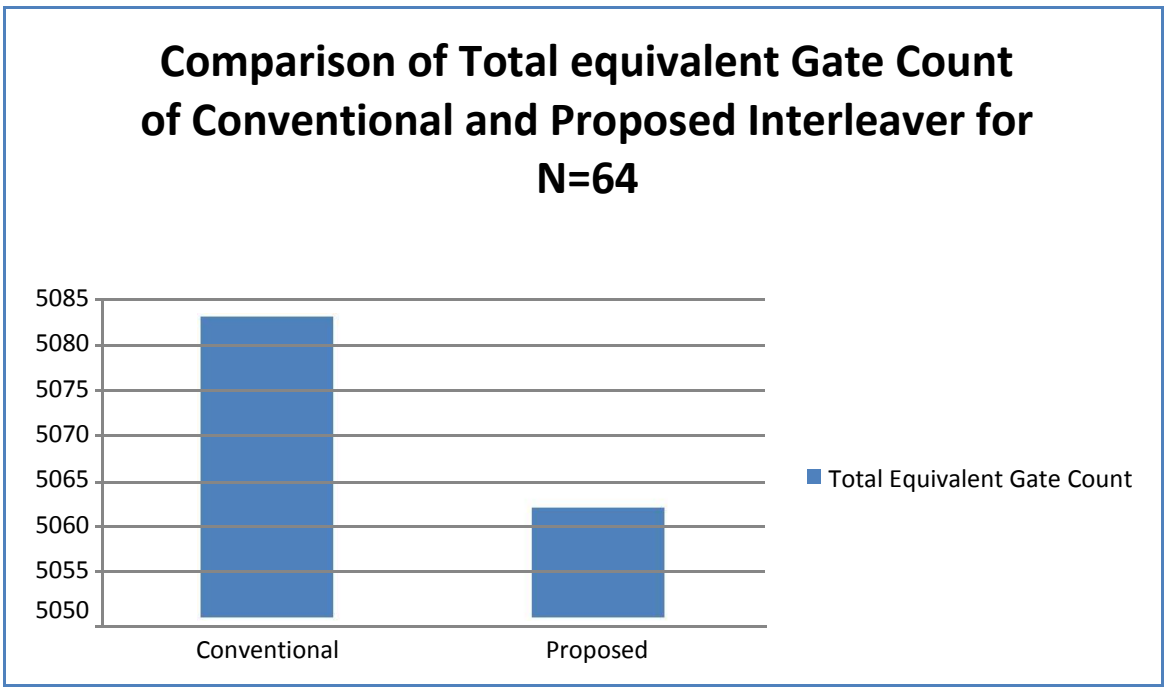


Figure 4.9 Comparison of Total Equivalent Gate Count of Conventional and Proposed Interleaver for

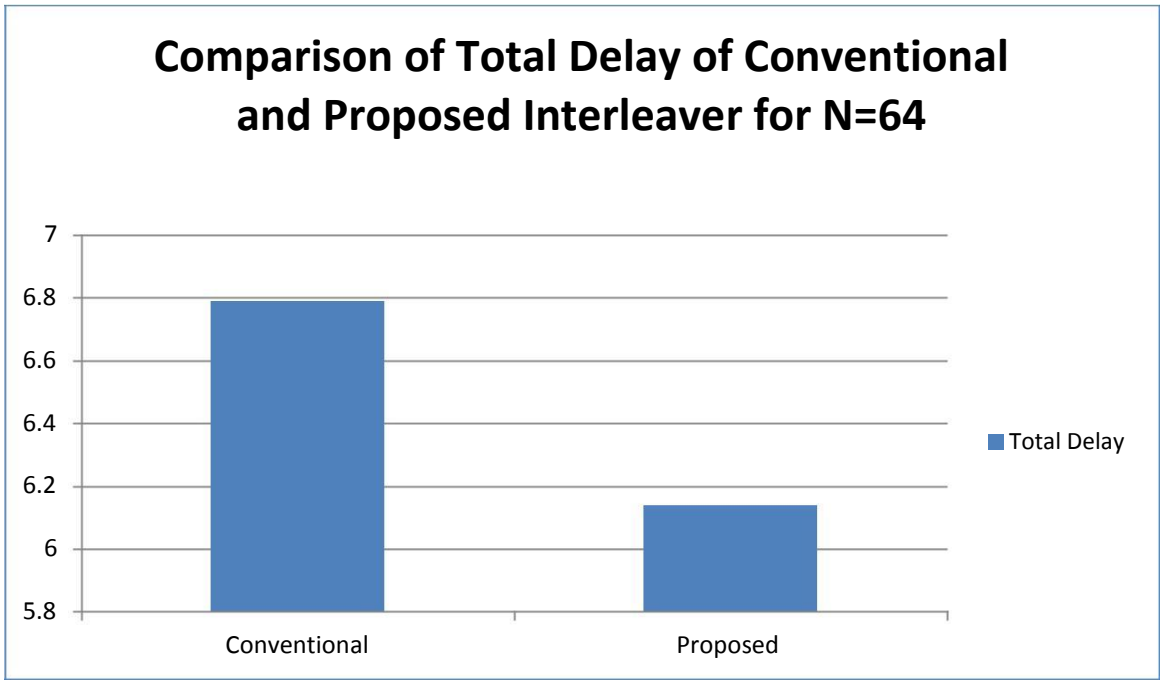


Figure 4.10 Comparison of Total Delay of Conventional and Proposed Interleaver for

As shown from tables 4.3-4.8 and figures 4.7-4.10, the proposed design provides the optimal results. The parameter values of the proposed design may be equal to those of existing design

for some parameters but overall, the proposed design provides the best compromise between area, delay and power dissipation. Moreover, it also overcomes the implementation difficulty of “mod” operator.

Chapter 5
Low Density Parity Check Codes

5.1 Introduction

Low-density parity-check (LDPC) codes are error correcting codes that facilitate reliable communication over the noisy channels and approach the Shannon capacity limit. Robert Gallager introduced LDPC codes in 1963 [14]. Because of their high computational complexity LDPC codes were ignored for 30 years. The LDPC codes were then rediscovered by MacKay and Neal in 1990. Mackay and Neal re-introduced the Low-density parity-check codes[15] and proved that optimally designed LDPC code had capability to perform near Shannon limit. The LDPC codes can be represented by a matrix with number of elements in every column having value 1 and number of elements in every row having value 1. Rest of the values are 0.The code rate of the matrix is / .

Tanner depicted LDPC codes by a graph [23]. A Tanner Graph can be defined as a bipartite structure in which there are two sets of nodes, variables nodes and check nodes. The Tanner Graph can be uniquely derived from H (i.e. parity-check) matrix. The graph has connections joining check nodes and variable nodes only if the element at the intersection of row and column is equal to 1.

$$\mathbf{H} = \begin{array}{c} \begin{matrix} v_1 & v_2 & v_3 & v_4 & v_5 & v_6 & v_7 \end{matrix} \\ \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 \end{bmatrix} \end{array} \begin{matrix} c_1 \\ c_2 \\ c_3 \\ c_4 \end{matrix}$$

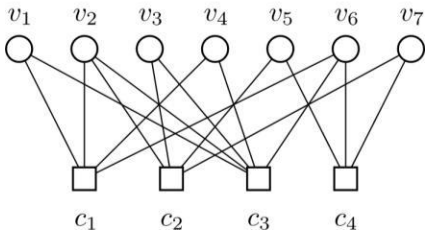


Figure 5.1 Representation of LDPC code by its Parity-Check matrix and Tanner Graph

As far as the design efficiency of the code is concerned, more connecting edges between the two sets imply greater efficiency of the code. However, the use of iterative algorithms for decoding is facilitated by the sparseness of H [17]. The short cycles significantly diminish the efficiency of iterative decoding algorithms, as the decoder goes on working locally, thus inhibiting the proper exchange of decoding information [96, 97]. However, short cycles are bound to occur, particularly for finite-sized block LDPC codes, [98, 99].

Consider a bipartite graph having message nodes and check nodes. LDPC can also be represented by a matrix of dimensions in which any particular matrix element entry is 1 iff there exists a connection betweencheck node and message node. Then the code may be represented by a vector set with the condition . represents the parity check matrix. On the other hand, any matrix (binary) can be shown by a Tanner graph and codeword can be obtained from the empty space of . It is to be noted that any binary and linear code cannot be represented by a sparse and a bilateral graph. But, if there is any possibility of showing a binary linear code with sparse, bilateral graph, then that code is a low-density parity-check code. The efficiency of LDPC codes depends on sparsity of graph structure.

The most important issue in the formulation of Low Density Parity Check code is the formation of its sparse parity-check matrix having the required characteristics. The literature discusses numerous formation methods of LDPC codes. Generally, there are two main techniques for formation of parity-check matrices of LDPC codes, that is, computer-based methods and algebraic techniques. The algebraic approach is based on finite mathematics, [31, 100 and 101], or combinatorial techniques, [102-105]. The algebraic methods find usage in industry due to their simplified encoding techniques. On the other hand, techniques like Gallager [106], MacKay [17] and density evolution (DE), [107, 108], which are computer-

based give excellent performance (i.e. near-capacity performance) for very large block lengths and are very flexible in code-designing.

5.2 Literature Survey of LDPC Codes

Numerous algorithms for LDPC codes have been introduced having different complexity and performance parameters. A reduction in the storage requirement has been proposed in [109] by adopting a novel merged-schedule message-passing algorithm.[109] Also discusses a mechanism to update the implementation mechanism in BCJR algorithm of the forward and backward recursions using "max-quartet" computations that compute without lookup-tables and give very less performance loss. But compromising exchange between complexity and performance of the decoder remains an issue [110]. [29] Presents large number of log-likelihood-ratio based Belief Propagation decoding algorithms. One of the soft-decision decoding algorithms, Iterative log Belief-propagation (BP) algorithm, exhibits high decoding capability and complexity. On the other hand, Bit-Flip algorithm, a type of hard decision decoding, shows reduced performance and decoding complexity. Thus to achieve a compromising exchange between intricacies of decoder structure and capabilities, simplified message passing algorithm (SMPA) was introduced [111]. The LDPC codes can be decoded by exchanging log-likelihood ratio (LLR) messages between symbol and parity-check nodes and using optimized implementations of the sum-product algorithm i.e. SPA [112].The complication aspect and throughput of decoder design also depend on the decoder architecture. Fully parallel architecture has the advantage of giving high through put but its disadvantage is that it has very high decoder complexity. This design complication increases as the code length increases. The serial decoder architecture has the advantage of having fewer complications but has reduced performance as disadvantage. The output decreases as the length of the code increases. This issue led to development of partially parallel decoder [113] to provide compromising interchange between output and design complications. [114]

Discusses a serial Galois Field (64)-Low Density Parity Check decoder based on a less complex type of the Extended Min-Sum algorithm. The architecture and throughput have been discussed in detail and the implementation results when tested for different values of code rates and lengths show that the partially parallel decoder architecture performed at less than 0.7 dB from the Belief Propagation algorithm.[116] Presents a novel decoding algorithm that aids in the design of LDPC decoder and also fulfills requisites of internal information exchange.[109] Proposes an optimum, reconfigurable full-decoder architecture along with an efficient memory and interconnection network. It is clear from the simulation results that the proposed architecture achieves an output of 19.2 Gbps for data frame 2304 bits long. Also, the proposed design exhibits reductions of the order of 89.13%, 69.83% and 60.5% in power consumption, silicon area and interconnect length respectively.

5.3Decoding Algorithms of LDPC codes

Message-passing algorithms are the decoding algorithms for LDPC codes. They are so called since their procedure involves transfer of information data over the connections of Tanner graph. Another name for message-passing algorithms is the iterative algorithms as the exchange of data between the two sets of nodes goes on repetitively till desired output is obtained. The various decoding algorithms can be categorized on the basis of message-type passed or type of computations done at the nodes.

In iterative decoding algorithms, the check and variable nodes exchange the information at discrete intervals of time. An initial received value is associated with each variable node

where . Each received value is a variable (random) belonging to the set of output alphabet , defined at the output of the channel. The message sent by a variable node

(belonging to some message alphabet) is processed by the check node along with the messages received from its neighboring adjacent nodes. After that, a suitable message is sent back by the check node to its every adjacent variable node from alphabet . Then, new

messages are produced by each variable node from its own received value and the messages obtained from the neighboring check nodes. These new messages are sent to its neighboring check nodes. This process continues until a result is obtained after a particular number of iterations have occurred. Dummy messages from the check nodes form initial iteration. This is followed by communication of received values from variable to check nodes. In determination of the next message, the point to be noted is that message sent on a connecting edge does not depend on the message just received on that particular connecting edge. Thus, the only information that is communicated in each step from a node to its adjacent node is the extrinsic information. A particular codeword can be estimated from a message which can take continuous or discrete values. The estimated value of the codeword is determined from the message sign whereas reliability parameter is determined from the absolute value [24].

Iterative algorithms can be classified into hard-decision and soft-decision algorithms. The message alphabet may take bit value, e.g., { } or any value such as the set of real numbers. If takes value from the set{ }, thenit is a hard-decision algorithm; for any other value of , it would be referred to as soft-decision algorithm. Any increase in the alphabet size of messages would increase the complications in computations of iterative algorithms.

5.3.1 Hard-decision Algorithms

Implementation of hard decision algorithms is very simple as the computations for decoding require only binary values. In these algorithms, only binary values are used throughout the decoding process. Gallager's algorithm [106] is an example of hard-decision algorithm, and is a part of Majority Based algorithms set [115] in which the alphabet is of range{ }. Another example of hard-decision algorithm is the Bit Flipping algorithm (BF) [106]. In Bit Flipping algorithm, a flipping function is defined which counts total unsatisfied syndrome bits in which every variable node takes part. There is a binary buffer associated with each

variable node which stores the hard decision; this buffer will flip its value if the flipping function output is greater than a particular threshold value. Decoding process goes on till each and every check node equation is satisfied or till the maximum number of iterations has taken place.

5.3.2 Soft-decision Algorithms

In contrast to hard-decision algorithms, soft-decision algorithms have a continuous alphabet. Belief propagation (BP) [15, 24] and Weighted Bit Flipping algorithms (WBF) algorithms are examples of soft-decision algorithms.

The Belief propagation algorithm is highly complicated but it is the most efficient decoding algorithm from the error rate point of view. Min-Sum (MS) algorithm [24, 25, 28 and 117] is a less complicated version of Belief propagation. [112] gives an overview of less complex algorithms. The WBF algorithms show a good tradeoff between throughput and design complications as compared to BP. The WBF algorithms include Modified Weighted Bit Flipping (MWBF) [26] and Improved Modified Weighted Bit Flipping (IMWBF) [27].In contrast to BF algorithm, these algorithms flip only one bit per iteration.

Iterative algorithms can be implemented using probability, likelihood ratio (LR) or log-likelihood ratio (LLR). The most widely and practically used domain is LLR domain as it is less complicated and more robust. A review of BP algorithm in LLR domain is given below:

Say, represents the variable nodes set and represents the check nodes set. Let denotes the message sent from node a to node b.

Initialization:

Let represents received values from AWGN channel. Then variable node messages are given by

— (5.1)

Check node operation:

(∏ ( ) (——)) (5.2)

Variable node operation:

Σ (5.3)

Where —

Hard Decision:

Σ (5.4)

Decoding is considered as complete if condition is fulfilled where for 0 otherwise In case the specified condition fails, then go to check node operation step. Decoding failure is declared if a codeword cannot be found within some specified maximum iterations.

5.3.3 Log-Belief Propagation decoding algorithm

Before description of algorithm, say, we have parity check matrix for LDPC codes. It has number of variable nodes and number of check nodes. Here, represents the entry at the metacentre of row and columnof the matrix. Say, we have set of variable nodes which determine check node as { }and set of check nodes which determine variable node as { }. / Refers to set of bits excluding and / denote set of bits excluding .

If we directly apply Belief Propagation decoding algorithm then complexity is too high due to use of various multiplier units. So to reduce the complexity, Log-BP decoding algorithm is introduced. It reduces the complex multiplication operations to simple addition operations [118].

- Let [ ] be the conditional probability that be the conditional probability that is 0. is 1 and

- Here, and represent the input bit and the number of variable nodes respectively.

Initialization: This step involves computation of the message from each variable node to check node ( ), with likelihood ratio (/)

( ) (5.5)

Where, is +1 when

is -1 when

Check – node operation: This step involves computation of check-to-variable messages ( ).

( ) Π ( ) (5.6)

Where, is ∑ | |

Variable – node operation: This involves computation of the message from variable node to check node. Thus, for each variable node

Where, ( ) ∑ ( ) (5.7)

| |

- Log-likelihood ratio (LLR) is updated to ∑ (5.8)
- Output decision is 0 if and 1if .

In [29], several less complex algorithms have been proposed which can make the implementation of the conventional belief-propagation algorithm comparatively easy. However, the relatively less complex algorithms have comparatively low throughput.

Min-sum algorithm, a simple decoding algorithm has been discussed below. The algorithmic complexity can be reduced by using the hyperbolic functions tanh (·) and $\tanh^{-1}$ (·) and suitable approximations. Given two random variables A and B, the following expressions can be defined:

(5.9)

Above equation can be expressed as:

( )/ (5.10)

The following expressions are obtained after applying the Jacobian algorithm [111, 114]

( ) | || |

b)

( ) | || |

The standard Belief-Propagation algorithm is more complex in contrast to min-sum algorithm but the decoding output of min-sum algorithm is better than the standard BP decoder.

5.4 Proposed Algorithm

This chapter presents realization of LDPC decoder by using simplified message passing decoding algorithm and partially parallel decoder architecture. The different computation steps of the proposed algorithm are listed below:

1. Compute likelihood ratio of each variable node;
2. Compute intrinsic message i.e. where, denotes most significant bit of i.e. sign bit.
3. Compute check-to-variable message for each check node b i.e.

/

4. Update the likelihood ratio value. The updated value of likelihood ratio is given by $ULR = LR + \sum$

Where {–

5. The *ULR* value is used to compute finally the variable-to-check message.
6. Finally, LLR can be computed as $LLR = LR + \sum$ .Decision is made on the basis of sign of LLR. If LLR comes out to be positive, bit is assumed to be zero whereas it is taken to be one if LLR comes out to be negative or zero. The proposed algorithm provides trade-off between low decoding complexity and decoder performance.

5.4.1 Construction of parity-check matrix

The (3, 6) parity check matrix can be constructed by using two regular and fixed matrices and one random matrix.[135, 133, 113 and 114]. Say and are two regular and fixed matrices(figures 5.2 and 5.3), each of same order, say . Matrix is composed of number of identity matrices, each of order whereas matrix is composed of number of "shifted" identity matrices , again of the order . The "shifted" identity matrix may be obtained by shifting the identity matrix towards right by a particular number of positions, which can be calculated using "mod" operator.

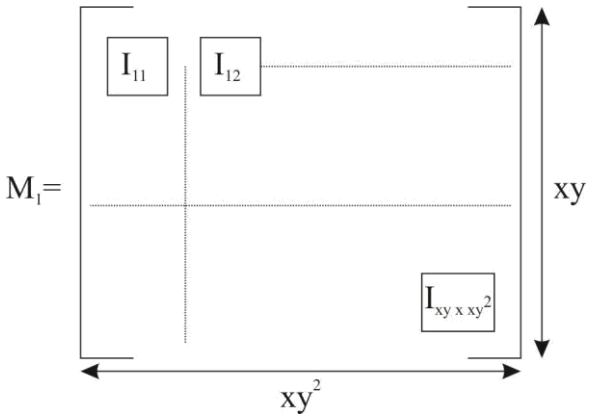


Figure 5.2 Regular and Fixed matrix

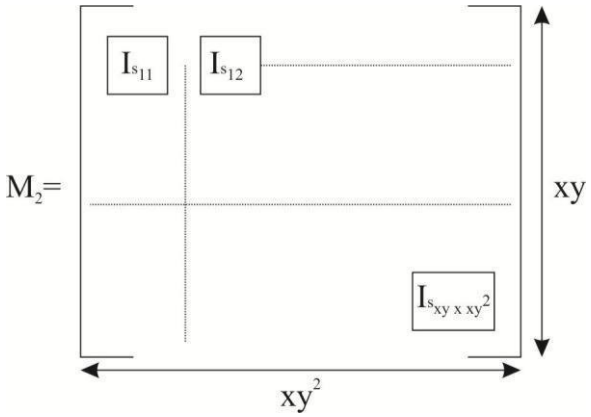


Figure 5.3 Regular and Fixed matrix

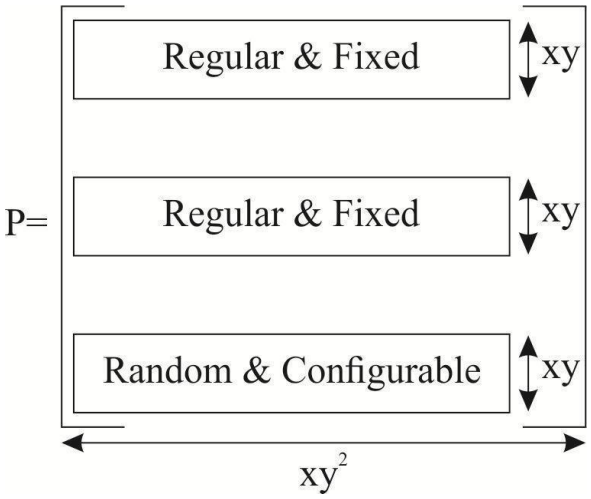


Figure 5.4 Parity-Check matrix

Say, matrix , is a random matrix. Its dimensions are of the same order as that of regular and fixed matrices i.e. Matrix is random in the sense that the placement of

ones in the matrix and the change in error positions in subsequent iterations is totally random Thus *P* can be considered as is [ ] (figure 5.4).

5.4.2 Partial-parallel decoder architecture

This article presents the design of partial-parallel decoder architecture having high speed and low complexity (Figure 5.5). The architecture has been designed for (3, 6) parity-check matrix [113, 114, 134].

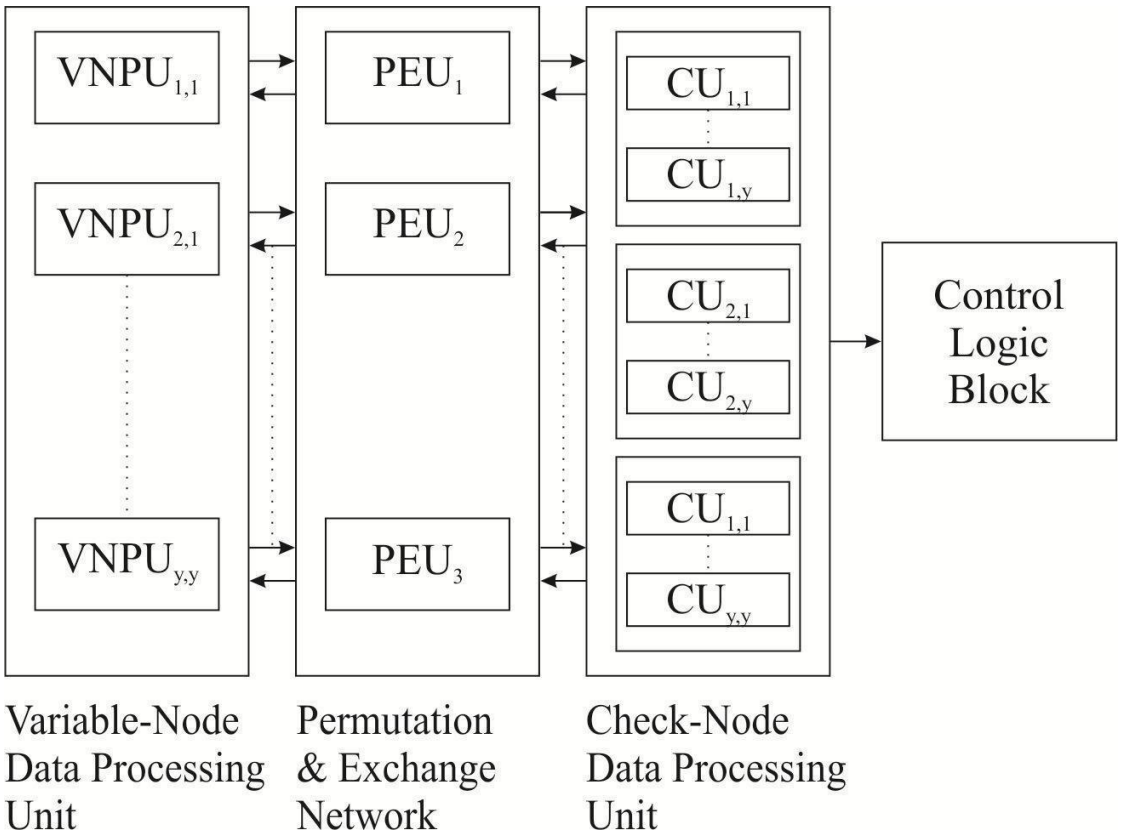


Figure 5.5 Principal partial-parallel decoder architecture

The architecture presents implementation of:

(1) Variable node data processing unit (VNDPU): Each VNDPU has 36 variable node processing units (VNPUs). Each VNPU further consists of RAM memory blocks: three extrinsic RAM (E-RAM) blocks each of size (256 × 1), one Intrinsic RAM (I-RAM) block of size (256 × 8) and one decision RAM (D-RAM) block of size (256 × 1), in addition to a computation unit for performing variable node computations (Figure 5.6). Each E-RAM block has memory locations. Any memory location out of these available locations can

store the extrinsic information exchanged between a particular variable node and its neighbouring check node. This significantly reduces the size of RAM as given by [110]

(2) Check node data processing unit (CNDPU): Each check node data processing unit (CNDPU) has 3 check units (CUs) for performing check node computations. Each CU further consists of 6 check nodes.

(3) Permutation and exchange network (PEN): The PEN consists of 3 permutation and exchange units which are used for permutation and exchange of messages between CNDPU and VNDPU. Exchange of information between variable node and check node is done through bidirectional permutation and exchange network. During variable-to-check message, permutation is done and during check-to-variable message, reverse permutation is done.

(4) Control block: All the CNUs communicate the parity check results to a control block. The control block determines if all the parity check equations specified by the parity-check matrix have been fulfilled.

In the each decoding iteration, the proposed decoder works in two modes: VNP (variable node processing) and CNP (check node processing) mode. The decoder processes each iteration in cycles of processing time, with each processing mode occupying half of the processing cycles.

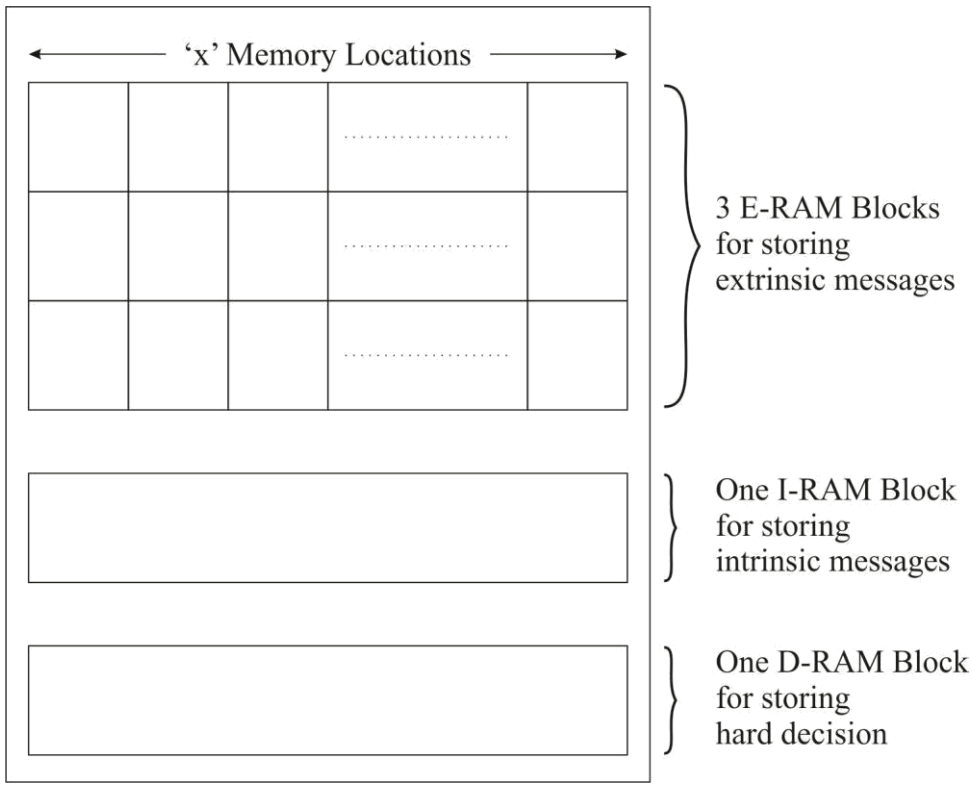


Figure 5.6 Memory blocks in VNDPU

5.4.3 Variable node data processing unit (VNDPU)

VNDPU consists of 36 VNPUs in the proposed architecture. Each VNPU performs three operations:

(1) Accessing 1-bit check-to-variable messages from the E-RAM block;

(2) Computation of 1-bit hard decision, 1-bit variable-to-check messages and updated value of LLR from the stored 5-bit intrinsic message and the check-to-variable message;

(3) Storing the computed values in their respective memory blocks i.e. hard decision in the D-RAM block, updated LLR in the I-RAM block and the computed variable-to-check messages in the E-RAM block.

Thus, as input VNPU has three 1-bit check-to-variable messages, 5-bit intrinsic message during initialization and 8-bit intrinsic message during decoding iteration as input. After processing and the required computations, VNPU outputs three 1-bit variable-to-check messages, one 1-bit hard decision value and one 8-bit intrinsic message. The variable-to-

check messages are stored in the E-RAM, hard-decision value is stored in the D-RAM and intrinsic message is stored in the I-RAM block.

5.4.3.1 Architecture of VNU

The architecture shown in the Figure 5.7realizes the variable node unit of SMPA algorithm. All the input and output messages are in 2"s complement format. In this architecture, there are six 8-bit binary adders and MSB of intrinsic output i.e. updated LLR is used as hard-decision bit. The architecture also associates one address decoder with each E-RAM block. The address decoders (AD1, AD2 and AD3) provide the read address for variable-to-check messages and write address for check-to-variable messages. The three address decoders along with the three permutation and exchange units (AD1 with PEU1; AD2 with PEU2; AD3 with PEU3) jointly specify the complete connectivity between the variable nodes and the check nodes. All the three address decoders are implemented as mod-8 binary counters, counting from 0 to , and are initialized to value 0 while the decoder is performing variable node computations. While in check node computation mode, the three decoders are initialized to different values. AD1 is initialized to value 0; AD2 is initialized to value for a particular ; AD3 is randomly initialized as it is used for realization of random matrix M3. AD3 can be randomly initialized to a value for a given

value of but for two different values of , the value of is also different. Another constraint that needs to be checked for is that for a single value of n but two different values of m, the difference between two p values is given by .

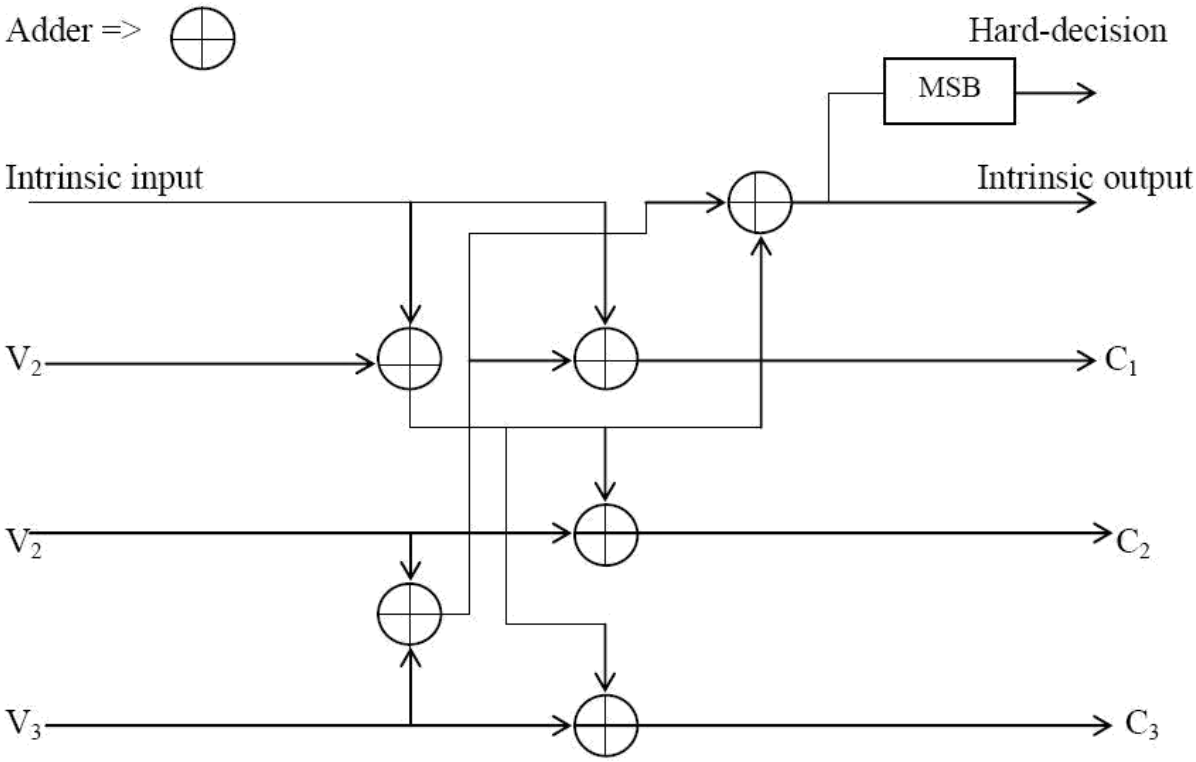


Figure 5.7 Architecture of VNU

5.4.4 Permutation and exchange network

Permutation and Exchange network provides connectivity between the variable nodes and the check nodes as specified by 3 sub-matrices and . PEN in the proposed architecture consists of 3 permutation and exchange units (PEUs) which provide for permutation, reverse permutation and exchange of data between variable nodes and check nodes.

(1) PEU1 connects to , thus connecting six VNPU elements with the CU having the same x-index.

(2) PEU2 connects to , thus connecting six VNPU elements with the CU having the same y-index.

(3) PEU3 connects to , This connection is required to be of random type so that randomness of matrix M3 is maintained. PEU3 permutes (or reverse permutes) data in two stages as shown in Figure 5.8.

5.4.5 Check node data processing unit (CNDPU)

The CNDPU performs certain operations to convert variable-to-check extrinsic message to check-to-variable message as listed below:

(1) Three single-bit variable-to-check messages are accessed from the E-RAM blocks.

(2) The accessed data is passed through a permutation and exchange unit of permutation and exchange network.

(3) Computations are performed on the permuted data and variable-to-check extrinsic messages are generated from the check-to-variable messages. In addition, parity check is also performed on the hard decision bits and the results are communicated to the control block that checks if all the parity check conditions specified by the parity-check matrix P have been satisfied or not. If the conditions are satisfied, decoding process can be stopped.

(4) The computed data from CNDPU is again passed through permutation and exchange unit of PEN for reverse permutation process (Figure 5.9).

(5) The resultant message after reverse permutation is written back to the ith memory location of E-RAM, from where the message was initially fetched.

5.4.5.1 Architecture of CU

Each CU performs operation of one check node i.e. computation of check-to-variable messages. Figure 5.10shows the architecture of CU. As shown in the figure, the complexity of CU is less than

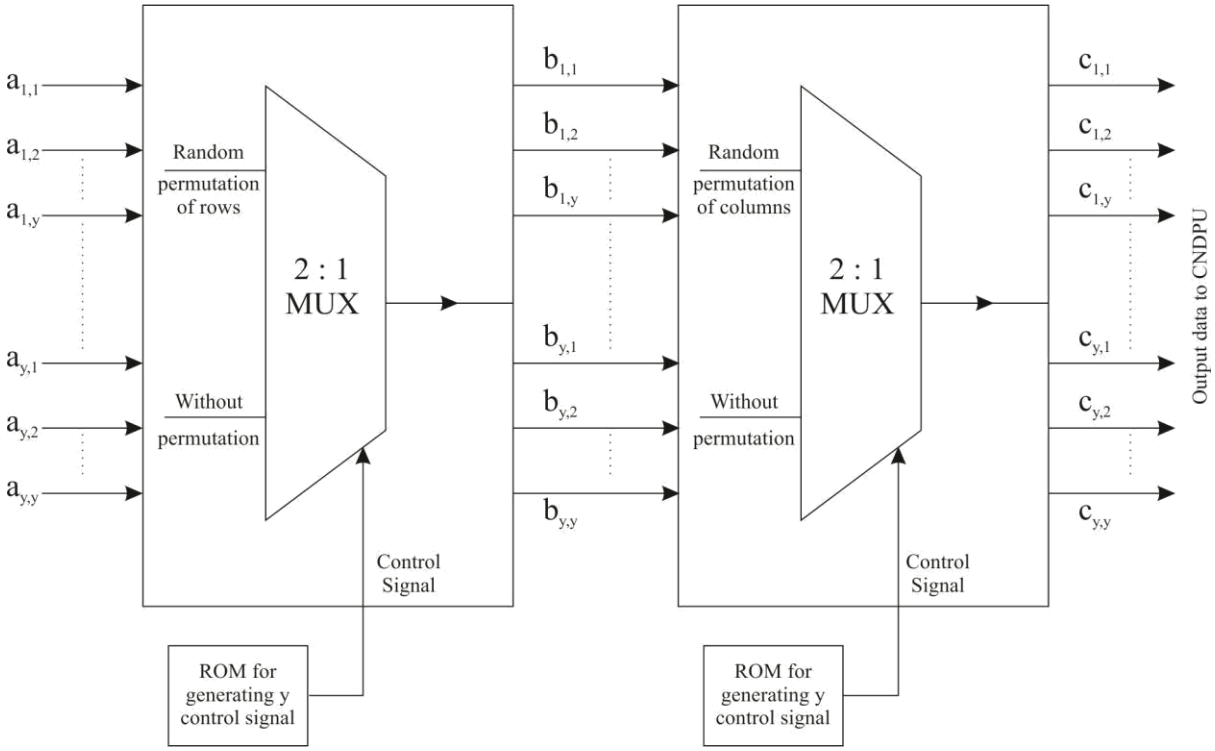


Figure 5.8 Block diagram of PEU3

the architectural complexity of Log-BP algorithm. It consists of only twelve XOR gates. Input to CU is k i.e.6-bit variable-to-check message coming from six VNPUs and the output is also of 6-bits sending to various VNPUs. This architecture realizes the "check node computation" of the SMPA algorithm completely [113,114]. The complete processing of data through VNDPU, PEN and CNDPU is shown in Figure 5.11.

5.4.6 Increasing throughput of the proposed decoder

The throughput of the proposed architecture can be increased by using pipelining process. In the proposed scheme, pipelining has been used at two levels: at the sub-unit level and at the decoder level. At the sub-unit level, pipelining scheme is used to pipeline the operations on CNDPU and VNDPU. At the decoder level, pipelining of processing of consecutive frames is done so as to increase the overall throughput.

5.4.7 Pipelining in VNDPU

Full-duplex connections are used for realizing the connections between the VNPUs and the different memory banks. Thus pipelining technique can be used for increasing the throughput.

The address for writing the check-to-variable messages is same as the address from where variable-to-check messages were initially accessed. Thus, assuming that VNPU takes three clock cycles to perform three operations, after a delay of three clock cycles the write address can be obtained.

5.4.8 Pipelining in CNDPU

As the architecture of CNDPU is not very complex, thus apart from accessing and writing the data on to the memory location, the permutation, computation and reverse permutation steps can be performed in a single clock cycle. Thus overall three clock cycles are used for CNDPU operations. Also, full duplex connections are used for interfacing various CUs of CNDPU with PEN.

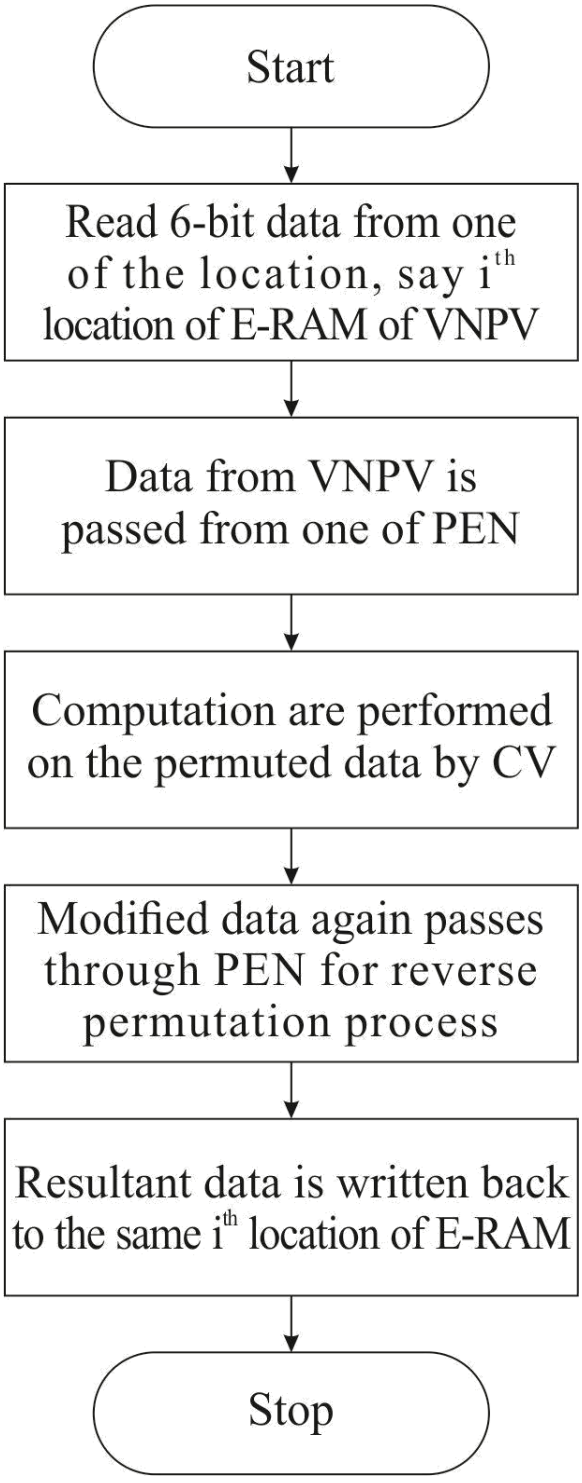


Figure 5.9 Flow chart showing CNDPU computations

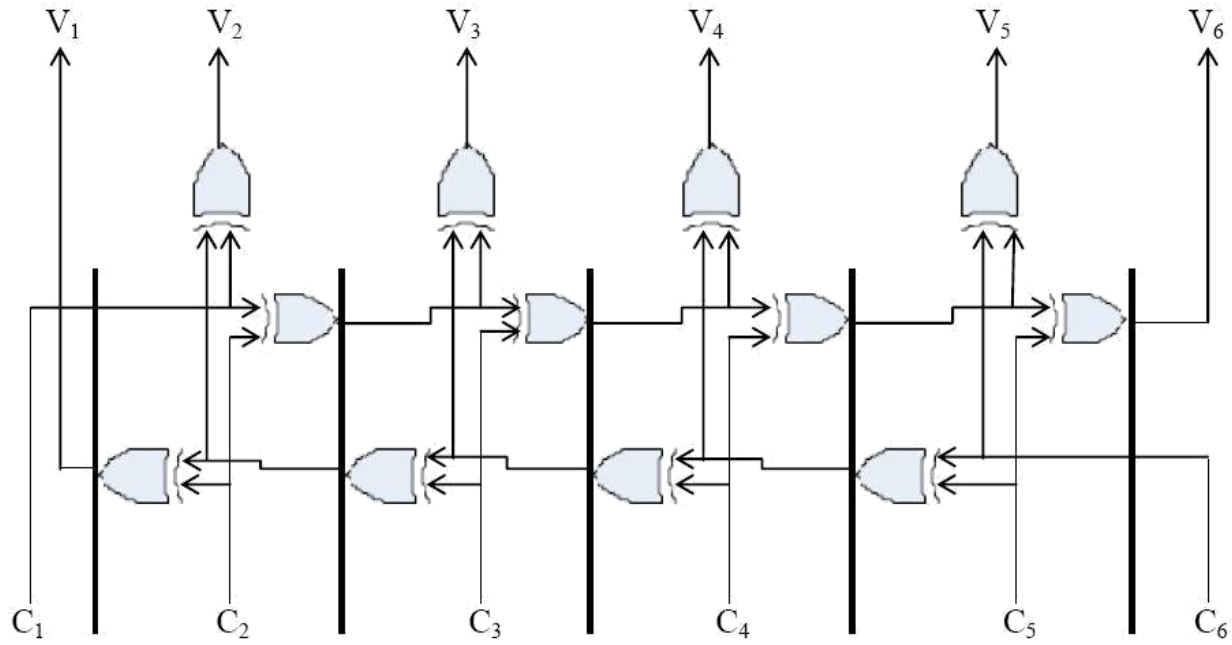


Figure 5.10 Architecture of CU

5.4.9 Pipelining at decoder level

In the proposed architecture, the processing of consecutive frames is pipelined to achieve higher throughput. While the decoder is reading out one frame from the decoder, the next frame is iteratively decoded and the third frame is loaded into the decoder.

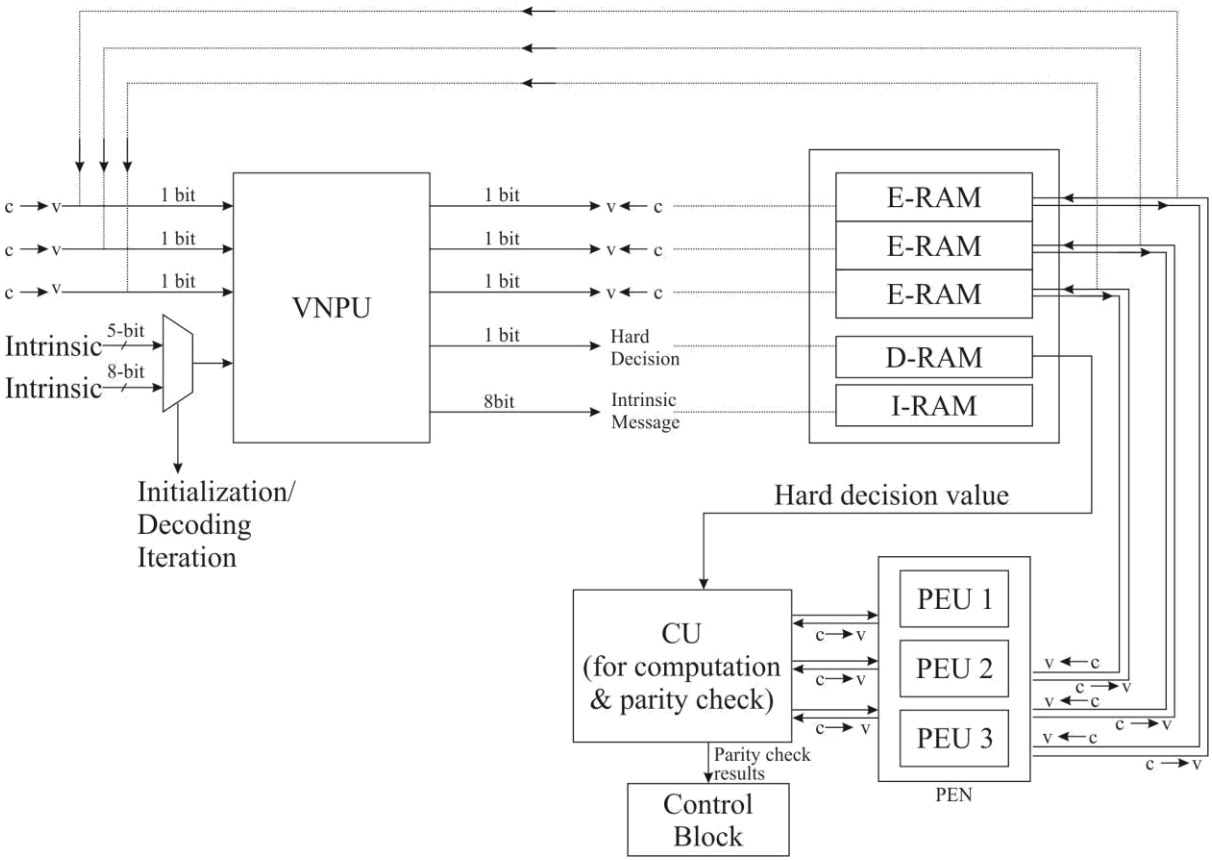


Figure 5.11 Processing data path through VNDPU, PEN and CNDPU

Thus, the decoder works on three code frames simultaneously. To realize this pipelining, two I-RAMs and two D-RAMs are used. The first I-RAM is RAM because of 5-bit quantization of LLR i.e. initial value of intrinsic message and the second I-RAM is of size because the value of LLR is updated to 8-bit quantization for further processing of iterative decoding. Input frame is loaded into the first I-RAM and the second I-RAM is used for storing the intrinsic message during iterative decoding. Output frame is read out from the first D-RAM of decoder and the second D-RAM is used for storing the hard-decision during iterative decoding.

5.4.10 Simulation

The decoder gets triggered at the positive edge of the clock. The global clock signal „clk" is used for synchronous operation. The LLR information is fed through the signal „din" using „load" control signal. „I1" signal gives the memory address of I-RAM in which input LLR

information is loaded. Signal „c‟ shows the VNU in which LLR information is loaded. The complete process of loading intrinsic information takes around 9474 clock cycles (258 for initialization and 9216 for input). After decoding the input, 1-bit output is obtained as decoded data using a high „data out ready‟ control signal. The decoding and retrieval of output from the decoder is synchronous with the clock.

Figure 5.12shows the block diagram of LDPC decoder along with the control signals. In this architecture, the number of input and output pins used is less compared with architecture proposed by [135]. Output is obtained by using only 1 pin rather than 36 pins. Also, this architecture does not need 14 pins for loading the address in each „Int‟ memory of 36 variable nodes (Figure 5.13).

Figure 5.14 shows the simulation result at the end of 18 iterations. This „int‟ signal shows the number of iterations at end of clock pulse 19092.

Figure 5.15 shows the starting of decoded output at the port „dout‟. This port is synchronous with „data out ready‟ port. Output is available in the next positive edge of clock cycle when the „data out ready‟ signal is „high‟.

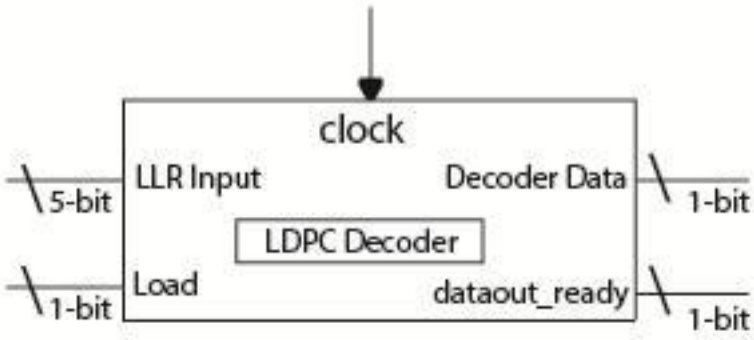


Figure 5.12Control signals of LDPC decoder

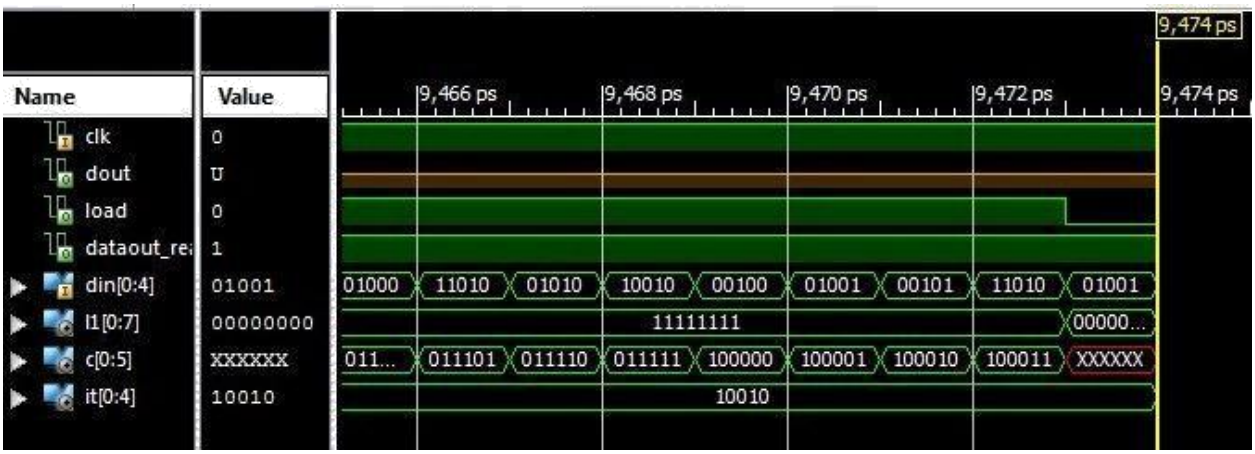


Figure 5.13 Decoder has loaded all the LLR information

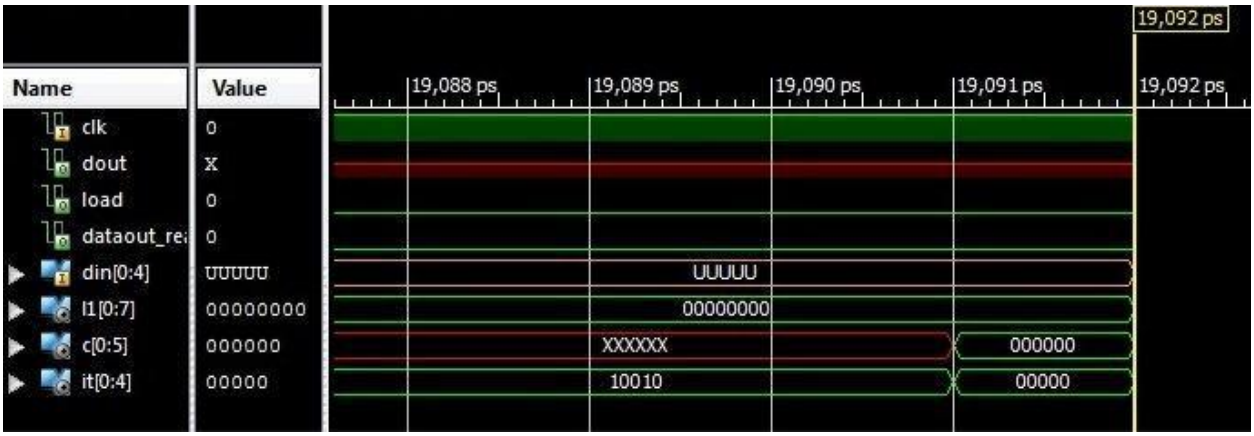


Figure 5.14 Decoder status after all iterations

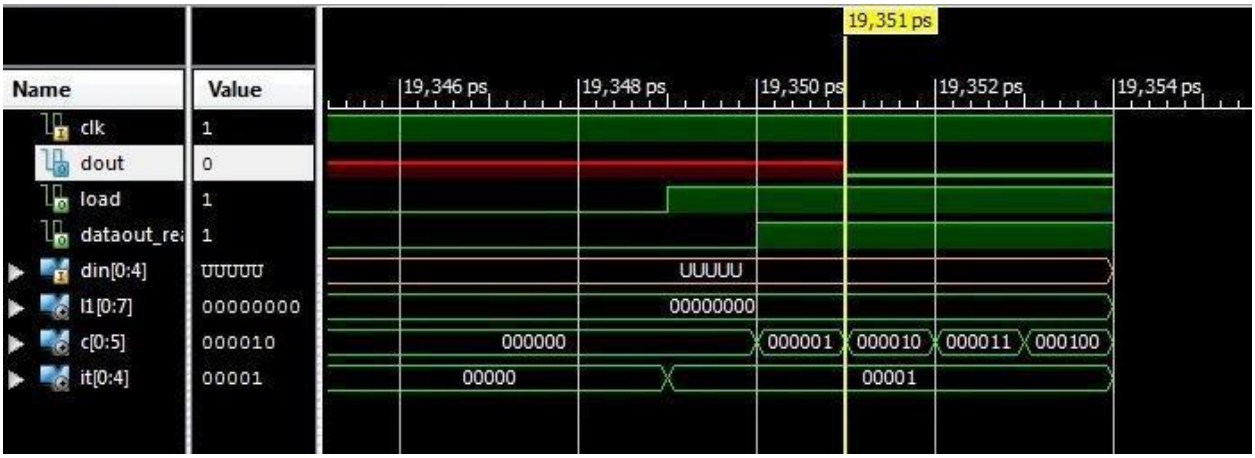


Figure 5.15 Start of decoded information at output

5.4.11 FPGA implementation

Partially parallel decoder architecture has been implemented for (3, 6)-regular LDPC code and on Xilinx XC3D3400A device from Spartan-3A DSP family. The implementation involves the following configurations:

● First and second intrinsic RAMs i.e. „Int" RAM is configured as dual port block RAM. The size of first RAM is i.e. 1.25 K-bit RAM. The size of second RAM is i.e. 2 K-bit RAM. First and second decision RAMs i.e. „Dec" RAM is configured as dual port distributed RAM. The size of first and second decision RAMs is 256 × 1 (i.e. 256 bits RAM).

● Each variable node contains 3 extrinsic RAMs i.e. E1, E2 and E3. These RAMs are configured as distributed RAM. The size of these RAMs are (i.e. 256 bits RAM).

VHDL has been used for data entry and resource utilization has been obtained through the synthesis process. Xilinx Development System tool suite has been used to place and to route the synthesized implementation for the target XC3D3400A device with the speed option-5. Table 5.1shows the statistic of hardware resource utilization. Maximum clock frequency suggested by Xilinx Development System tool suite is 96.28 MHz. If we have „a" decoding iterations for each code frame, then number of clock pulses utilizes for decoding the one frame of input will be (2a (L + 2) + (L + 2)). Here, 2 is added due to number of pipelining stages and extra (L + 2) is due to initialization process. Maximum symbol throughput will be (96.28) · k2. L/(2a (L + 2) + (L + 2)) = 96.28 (9216/9546) = 92.95 Mbps if maximum decoding iteration is 18. Figure 5.16 shows place and route decoder implementation using Xilinx FPGA editor tool.

Table 5.1 FPGA resource utilization

Resources	In Arch[136]	In this Arch.
Slices	11,792	7,021
Slices reg.	10,105	1,558

4 – input LUT	15,933	12,872
IOBs	68	9
Block Rams	90	72

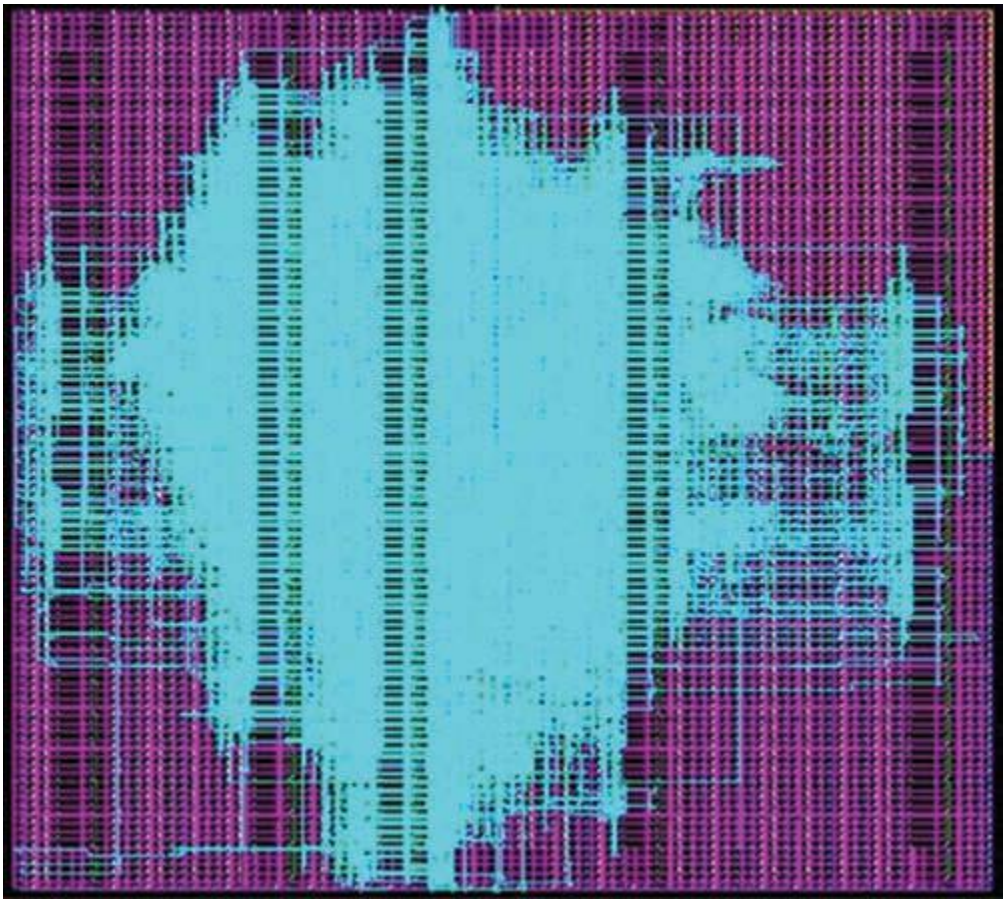


Figure 5.16 Place and route output

5.4.12Results

In this chapter, LDPC decoder has been implemented on the Xilinx XC3D3400A FPGA device developed with high decoding throughput and less area consumption. The improved decoder provides maximum decoding throughput of 92.95 Mbps in 18 iterations and maximum slice utilization is 7021. The proposed SMPA algorithm used in this chapter greatly reduces the resources utilization. The hardware resource utilization result obtained by the partially parallel decoder is very less compared with fully parallel decoder architecture.

5.5 Conclusion

This chapter starts with introduction to LDPC codes followed by exhaustive literature review of LDPC codes. The chapter clearly states the difference between the hard and soft decoding algorithms and dicusses Log-BP algorithm in detail. Further, this chapter also presents realization of LDPC decoder by using simplified message passing decoding algorithm and partially parallel decoder architecture.

Chapter 6
Conclusion and Future Scope

Channel coding is done to avoid the errors that may be introduced while transmitting binary sequences over the channel due to noise and interference. Turbo codes and LDPC codes are the two most prominent error-correcting techniques which provide near Shannon limit performance.

In the initial chapters, this thesis presents the theoretical perspective of channel coding and the turbo codes. A detailed description of the basic turbo encoder structure has been presented. Further, Interleavers have also been discussed in detail, in particular, QPP Interleaver due to its deterministic, algebraic structure and because it provides contention-free memory access. An exhaustive literature review of Turbo codes and QPP Interleaver has also been done. The most significant concern for Turbo codes is to efficiently implement a turbo decoder. Decoding algorithms for the turbo decoder have been discussed and a comparison has been drawn between MAP and SOVA algorithms. For implementation of Turbo decoder, Max-log-MAP algorithm has proven to be the most effective. Further, this thesis also talks about FPGA implementation of the basic and proficient turbo decoder. The efficient implementation comes from algorithm modification, integer arithmetic and compact hardware management. Based on the Max-Log-MAP decoding algorithm, the branch metric has been modified by weighing a-priori value, resulting in a significant BER improvement. The Turbo decoder takes in 8-level integer inputs generates 7-bit soft-decisions and calculates all metrics on integers, avoiding complex floating point or fixed-point arithmetic. By manipulating memory address, delay associated with interleaving and de-interleaving is eliminated, resulting in much higher throughput. Also, by taking advantage of identical decoder function, Turbo decoder has been implemented in a single-decoder structure, making efficient use of memory and logic cells. The design utilized almost 3447 out of 6912 logic

cells and approximately 28 RAM blocks out of 48 total RAM blocks in the device. In addition, the design required no external components and consumes approximately 695 mW during normal operation.

Six different architectures for turbo decoders have been designed and analyzed for their trade-offs in terms of latency, throughput and area requirement. The throughput increases from standard window towards SW and PW, but at the same time area requirement also increases proportionally. By applying circuit-level pipelining and retiming, throughput increases in larger proportion when compared to area increment. A novel interleaving technique gives area-optimized results. Simulation results corresponding to throughput and latency for different architectures of turbo decoders have been shown in the work.

For turbo codes, an interleaver is an essential constituent and its appropriate design is decisive for superior performance. A new interleaver design has been suggested, an alternative of QPP interleaver of turbo codes, which permutes a series of bits with the identical statistical sharing as a conformist QPP interleaver and gives superior performance than the predictable QPP. New proposed architecture has been simulated and synthesized using Xilinx and HDL Designer tools. Very large scale amalgamation of architecture for the proposed interleaver has been investigated in terms of area, delay and power dissipation. Thermal power indulgence and device operation has been computed for the novel plan using Quartus II (32-bit) tool. Also, a contrast among the proposed alternative of QPP interleaver and the conformist QPP interleaver has been presented

Next, this thesis presents introduction to LDPC codes followed by exhaustive literature review of LDPC codes. The difference between the hard and soft decoding algorithms has been highlighted and Log-BP algorithm has been discussed in detail. Further, this thesis also presents realization of LDPC decoder by using simplified message passing decoding algorithm and partially parallel decoder architecture. Further, LDPC decoder has been

developed with high decoding throughput and less area consumption and implementation has been done on the Xilinx XC3D3400A FPGA device. The improved decoder provides maximum decoding throughput of 92.95 Mbps in 18 iterations and maximum slice utilization is 7021. The proposed SMPA algorithm greatly reduces the resource utilization. The hardware resource utilization result obtained by the partially parallel decoder is very less as compared to fully parallel decoder architecture.

The future work can be extended in numerous ways by optimizing the decoder at any of the three levels: decoder level, algorithmic level and at metric level. Although the proposed design shows better performance in terms of latency, throughput and resource utilization and power dissipation, yet future work can be focused on FPGA implementation of the proposed technique with less hardware requirements.

REFERENCES

[1] Shannon, C. E. (1948). A mathematical theory of communication. *Bell System Technical Journal, 27,* 379–423. doi:10.1002/j.1538-7305.1948.tb01338.x

[2] Berrou, C., Glavieux, A., & Thitimajshima, P. (1993). Near Shannon limit error-correcting coding and decoding turbo codes. *IEEE International Conference on Communication, 2,* 1064–1070.

[3] Forney Jr, G. D. (1966). *Concatenated codes*. Cambridge, MA: MIT Press.

[4] Benedetto, S., & Montorsi, G. (1996). Unveiling turbo codes: Some results on parallel concatenated coding schemes. *IEEE Transactions Informatics Theory*, *42,* 409–428. doi:10.1109/18.485713

[5] Takeshita, O. Y., & Costello Jr, D. Y. (2000). New deterministic interleaver designs for turbo codes. *IEEE Transactions on Information Theory, 46,* 1988–2006. doi:10.1109/18.868474

[6] Japan Association of Radio Industries and Businesses (ARIB). (1998). Japan‟s Proposal for Candidate Radio Transmission Technology on IMT-2000: W-CDMA.

[7] Consultative Committee for Space Data Systems, Telemetry Channel Coding, May 1999.

[8] Viterbi, A. J. (1967).Error Bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Transactions on Information Theory, 13*(2*),* 260–269.

[9] Bahl, L., Cocke, J., Jelinek, F. & Raviv, J. (1974). Optimal decoding of linear codes for minimising symbol error rate. *IEEE Transactions on Information Theory, 20,* 284– 287.

[10] Mathana, J. M., Rangarajan, P., & Perinbam, J. R. P. (2013). Low complexity reconfigurable turbo decoder for wireless communication systems. *Arabian Journal for Science and Engineering, 38*, 1–14.

[11] Verma, S. & Kumar, S. (2011). An FPGA Realization of Simplified Turbo Decoder Architecture. *International Journal of Physical Sciences*, *6*(10), 2338–2347.

[12] Pasricha, S. & Sharma, S. (2016).Novel Interleaver Design for Turbo codes. *Wireless Personal Communications, Springer, 86*(2), 727-749.

[13] Verma, S. & Kumar, S. (2015). High-performance VLSI architectures for turbo decoders with QPP interleaver. *International Journal of Electronics*, *102*(4), 599-618.

[14] Gallager, R. (1963). *Low-Density Parity-Check Codes*. Cambridge, MA: MIT Press.

[15] MacKay, D. J. C. (1999). Good error-correcting codes based on very sparse matrices. *IEEE Transactions on Information Theory, 45*, 399 – 431.

[16] Davey, M. & MacKay, D. (1998). Low-density parity check codes over GF (q). *IEEE Communications Letters, 2,* 165 -167.

[17] MacKay, D. & Neal, R. (1997). Near Shannon limit performance of low density parity check codes. *Electronics Letters, 33*, 457-458.

[18] MacKay, D (1998). Gallager codes that are better than Turbo codes. *Proceedings 36th Allerton conference on Communication, Control, and Computing, s* (pp. 201-210), Monticello, IL.

[19] *IEEE P802.3an, "1OGBASE-T task force.* Retrieved from http: //www. ieee8O2.org/3/an.

[20] IEEE 802.16e, "Air interface for fixed and mobile broadband wireless access systems," IEEE P802.16e/D12 Draft, October 2005.

[21] IEEE 802.11n, "Wireless LAN medium access control and physical layer specifications," IEEE P802.11n/D3.07, March 2008.

[22] *T.T.S.I. digital video broadcasting (DVB) second generation framing structure for broadband satellite applications.* Retrieved from http://www.dvb.org.

[23] Tanner, R. (1981). A recursive approach to low complexity codes. *IEEE Transactions on Information Theory, 27*, 533-547.

[24] Richardson, T. & Urbanke, R. (2001). The capacity of low-density parity-check codes under message-passing decoding. *IEEE Transactions on Information Theory, 47*, 599-618.

[25] Xiao, H. & Banihashemi, A.H. (2004). Graph-based message-passing schedules for decoding LDPC codes. *IEEE Transactions on Communications, 52*, 2098-2105.

[26] Zhang, J. & Fossorier, M. (2004). A modified weighted bit-flipping decoding of low-density parity-check codes. *IEEE Communication Letters, 8*, 165-167.

[27] Jiang, M., Zhao, C., Shi, Z. and Chen, Y. (2005).An improvement on the modified weighted bit-flipping decoding algorithm for LDPC codes. *IEEE Communications Letters, 9*, 814- 816.

[28] Fossorier, M., Mihaljevic, M. and Imai, H. (1999). Reduced complexity iterative decoding of low-density parity check codes based on belief propagation. *IEEE Transactions on Communications, 47*, 673- 680.

[29] Chen, J., Dholakia, A., Eleftheriou, E., Fossorier, M. & Hu, X.Y. (2005). Reduced-complexity decoding of LDPC codes. *IEEE Transactions on Communications, 53*(8), 1288- 1299.

[30] Fossorier, M. (2001). Iterative reliability-based decoding of low-density parity check code. *IEEE Journal Select Areas Communicable, 19*, 908-917.

[31] Kou, Y., Lin,S. & Fossorier,M. (2001). Low-density parity-check codes based on finite geometries: a rediscovery and new results. *IEEE Tranactions on Information Theory, 47*, 2711- 2736.

[32] Verma S.,& Sharma, S. (2016.). FPGA implementation of low complexity LDPC iterative decoder. *International Journal of Electronics, 103*(7), 1112-1126. DOI: 10.1080/00207217.2015.1087052

[33] Robertson, P. (1994).Illuminating the structure of code and decoder of parallel concatenated recursive systematic (turbo) codes. *Proceedings Global Telecommunications Conference, 1994. GLOBECOM '94. Communications: the Global Bridge., IEEE, 3*, 1298-1303.

[34] Pyndiah, R., Glavieux, A., Picart, A. & Jacq, S. (1994). Near optimum decoding of product codes. *Global Telecommunications Conference, 1994. GLOBECOM '94. Communications: The Global Bridge., IEEE, 1*, 339-343.

[35] Divsalar, D. & Pollara, F. (1995). Multiple Turbo Codes for Deep-Space Communications, *The Telecommunications and Data Acquisition Progress Report 42-121*, 66-77.

[36] Divsalar, D. & Pollara, F. (1995).On the design of turbo codes. *TDA Progress report 42-123*, 99-121.

[37] Benedetto, S. & Montorsi, G. (1996). Iterative Decoding of Serially Concatenated Convolutional Codes. *Electronics Letters, 32* (13), 1186-1188.

[38] Benedetto, S., Divsalar, D., Montorsi, G. & Pollara, F. (1996). Serial Concatenation of Interleaved Codes: Performance Analysis, Design, and Iterative Decoding. *The Telecommunications and Data Acquisition Progress Report 42-126*, 1-26.

[39] Couleaud, J. Y. (1995). High Gain Coding Schemes for Space Communications. *ENSICA Final Year Report*, University of South Australia, the Levels, Australia.

[40] Benedetto, S., Divsalar, D., Montorsi, G. & Pollara, F. (1996). A Soft-Input Soft-Output Maximum A Posteriori (MAP) Module to Decode Parallel and Serial Concatenated Codes.s *TDA Progress Report 42-127*, 1-20.

[41] Hagenauer J. (1997). The Turbo Principle: Tutorial Introduction and state of the art. *Proceedings International Symposium on Turbo Codes and Related Topics*, (pp. 1-11), Brest, France.

[42] Costello, D. J. & Meyerhans, G. (1996). Concatenated turbo codes. *Proceedings International Symposium on Information Theory and Applications*, (pp.571–574), Victoria, B.C., Canada.

[43] Anderson, J. D. (1996). Turbo codes extended with outer BCH code. *Electronic Letters*, *32*, 2059–2060.

[44] Nill, C. & Sundberg, C. (1995). List and soft symbol output Viterbi algorithms: Extensions and comparisons. *IEEE Transactions on Communications, 43*, 277–287.

[45] Seshadri, N. & Sundberg, C. -E. W. (1994). List Viterbi decoding algorithms with applications. *IEEE Transactions on Communications, 42*, 313–323.

[46] Narayanan, Krishna R. & St¨uber, Gordon L. (1998). List Decoding of Turbo Codes. *IEEE Transactions On Communications, 46* (6), 754-762.

[47] Takeshita, O. Y., Collins, O. M., Massey, P. C. & Costello, D. J. (1999). A Note on Asymmetric Turbo-Codes. *IEEE Communications Letters, 3*(3), 69-71.

[48] Li Ping (2001). Turbo-SPC codes. *IEEE Transactions on Communications*, *49* (5), 754-759.

[49] Mansour, M. M. & Shanbhag, N. R. (2002). Design methodology for high speed iterative decoder architectures. *Acoustics, Speech, and Signal Processing (ICASSP), 2002 IEEE International Conference on*, (pp.3085-3088), Orlando, FL.

[50] Mansour, M. M., & Shanbhag, N. R. (2003). VLSI architectures for SISO-APP decoders. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 627–650. doi:10.1109/TVLSI.2003.816136

[51] Dobkin, R., Peleg, M. & Ginosar, R. (2005). Parallel Interleaver Design and VLSI Architecture for Low-Latency MAP Turbo Decoders. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 13* (4), 427-438.

[52] Studer, C., Fateh, S., Benkeser, C & Huang, Q. (2012). Implementation Trade-offs of Soft-Input Soft-Output MAP Decoders for Convolutional Codes. *IEEE Transactions on Circuits and Systems I: Regular Papers, 59* (11), 2774 – 2783.

[53] Boutillon, E., Gross, W. & Gross, P. (2003).VLSI architectures for the MAP algorithm. *IEEE Transactions on Communications, 51*(2), 175-185.

[54] Lee, S., Shanbhag, N. & Singer, A. (2005). A 285-MHz pipelined MAP decoder in 0.18um CMOS. *IEEE Journal of Solid State Circuits, 40*(8), 1718-1725.

[55] P. Urard et al.(2004). A Generic 350 Mb/s turbo codec based on a 16-state Turbo decoder, *Proceedings Solid-State Circuits Conference, 2004. Digest of Technical Papers. ISSCC. 2004 IEEE International*, 424-433.

[56] Wang, Z., & Huang, X. (2010, February). VLSI architectures for turbo decoders, *VLSI (1st ed.)*. Rijeka: In-Tech Publication.

[57] Boutillon, E., Douillard, C. & Montorsi, G. (2007). Iterative Decoding of Concatenated Convolutional codes:Implementation Issues. *IEEE Proceedings*, *95*(6), 1201-1227.

[58] Duk Gun Choi et al. (2007).An FPGA Implementation of High-Speed Flexible 27-Mbps 8-stateTurbo Decoder. *ETRI Journal*, *29* (3), 363-370.

[59] Muller**,** O. Baghdadi**,** A. & Jézéquel, M. (2010). Parallelism Efficiency in Convolutional Turbo Decoding. *EURASIP Journal on Advances in Signal Processing, 2010*, 11.

[60] Wong, C., Lai, M., Lin, C., Chang, H. & Lee, C. (2010). Turbo Decoder Using Contention-free Interleaver and Parallel Architecture. *IEEE Journal Of Solid-State Circuits*, *45*(2), 422-432.

[61] Sun, Y., & Cavallaro, J. R. (2011). Efficient hardware implementation of a highly-parallel 3GPP LTE/LTE-advance turbo decoder. *Integration, the VLSI Journal, 44*, 305–315. doi:10.1016/j.vlsi.2010.07.001

[62] Abughalieh, N., Steenhaut, K., Lemmens, B. & Nowé, A. (2011). Parallel Concatenation vs. Serial Concatenation Turbo Codes for Wireless Sensor Networks. *Proceedings Communications and Vehicular Technology in the Benelux (SCVT), 2011 18th IEEE Symposium on*, 1-6, Ghent. doi: 10.1109/SCVT.2011.6101314.

[63] B. Vucetic, B. & Yuan, J. (2000). *Turbo Codes - Principles and Applications*. Norwell, MA, USA :Kluwer Academic Publishers.

[64] Pietrobon, S. S. (1998). Implementation and Performance of a Turbo/MAP Decoder. *International Journal of Satellite Communications*, *16*, 23–46.

[65] Takeshita, O. Y. (2006). On maximum contention-free interleavers and permutation polynomials over integer rings. *IEEE Transactions on Informations Theory, 52*, 1249–1253.

[66] Sun, J., & Takeshita, O. Y. (2005). Interleavers for turbo codes using permutation polynomials over integer rings. *IEEE Transactions on Information Theory*, *51*, 101–119.

[67] Multiplexing and channel coding, 3GPP TS 36.212 version 8.4.0, September 2008.

[68] Nimbalker, A., Blankenship, K. T., Classon, B., Fuja, T. E., & Costello, D. J. (2008). Contention- free interleavers for high-throughput turbo decoding. *IEEE Transactions on Communications, 56*(8), 1258–1267.

[69] Jung, P., & Nasshan, N. (1994). Performance evaluation of turbo-codes for short frame transmission systems. *Electronic Letters, 30*(2), 111–113.

[70] Jung, P., & Nasshan, N. (1994). Dependence of the error performance of turbo-codes on the interleaver structure in short frame transmission systems. *Electronic Letters, 30*(4), 287–288.

[71] Perez, L. C., Seghers, J., & Costello, D. J, Jr. (1996). A distance spectrum interpretation of turbo codes. *IEEE Transaction on Information Theory, 42*, 1698–1709.

[72] Daneshgaran, F., & Mondin, M. (1997). On design of interleaver for turbo codes. In *Proceedings CISS'97* (pp. 509–514). Baltimore, MD.

[73] Khandani, A. K. (1999). Dynamic generation of good turbo-code interleavers. Waterloo, ON, Canada: Department of Electrical and Computer Engineering, University of Waterloo. Technical Report UW-E& CE99-02.

[74] Andrews, K. S., Heegard, C., & Kozen, D. (1998). Interleaver design methods for turbo codes. In *Proceedings IEEE international symposium on information theory (ISIT'98)* (p. 420). Cambridge, MA.

[75] Crozier, S. N.(2000). New high-spread high-distance interleavers for turbo codes. *In Proceedings 20th B. Symposium communications,* (pp. 3–7), Kingston, ON, Canada.

[76] Barbulescu, A. S., & Pietrobon, S. S. (1994). Interleaver design for turbo codes. *Electronics Letters, 30*(25), 2107–2108.

[77] Garello, R., Montorsi, G., Benedetto, S., & Cancellieri, G. (2001). Interleaver properties and their application to the trellis complexity analysis of turbo codes. *IEEE Transactions on Communications, 49*, 793–807.

[78] Hokfelt, J., Edfors, O., & Maseng, T. (2001). A turbo code interleaver design criterion based on the performance of iterative decoding. *IEEE Communications Letters, 5,* 52–54.

[79] Khandani, A. K. (1998). Group structure of turbo-codes. *Electronics Letters, 34*(2), 168–169.

[80] Takeshita, O. Y., & Costello, Jr., D. J. (1998). New classes of algebraic interleavers for turbo-codes. *In Proceedings IEEE international symposium information theory (ISIT'98)* (p. 419). Cambridge, MA.

[81] Hokfelt, J., Edfors, O., & Maseng, T. (1999). Interleaver design for turbo codes based on the performance of iterative decoding. *In 1999 IEEE international conference on communications, 1999. ICC '99* (Vol. 1, pp. 93–97). Vancouver, BC.

[82] Popovski, P., Kocarev, L., & Risteski, A. (2004). Design of flexible-length s-random interleaver for turbo codes. *IEEE Communications Letters*, *8*(7), 461–463.

[83] Kusume, K., & Bauch, G. (2006). Cyclically shifted multiple interleavers. In *Proceedings IEEE global telecommunications conference (GLOBECOM 2006).* San Francisco, California, USA.

[84] Nimbalker, A., Blankenship, T. K., Classon, B., Fuja, T. E., & Costello, Jr. D. J. (2004). Contention-free interleavers. In *Proceedings 2004 IEEE international symposium on information theory* (p. 54). Chicago, IL.

[85] Berrou, C., Keroue´dan, S., Saouter, Y., Douillard, C., & Je´ze´quel, M. (2004). Designing good permutations for turbo codes: Towards a single model. In *Proceedings international conference on communications* (Vol. 1, pp. 341–345). Paris, France.

[86] Giulietti, A., Van der Perre, L., & Strum, M. (2002). Parallel turbo coding interleavers: avoiding collisions in access to storage elements. *IEEE Electronics Letters, 38*(5), 232–234.

[87] Blankenship, T. K., Classon, B., & Desai, V. (2002). High-throughput turbo decoding techniques for 4G.In *Proceedings international conference 3G wireless and beyond* (pp. 137–142). San Francisco, CA.

[88] Nimbalker, A., Blankenship, K. T., Classon, B., Fuja, T. E., & Costello, D. J. (2008). Contention- free interleavers for high-throughput turbo decoding. *IEEE Transactions on Communications*, *56*(8), 1258–1267.

[89] Lee, S., Wang, C., & Wern-Ho, S. (2010). Architecture design of QPP interleaver for parallel turbo decoding. In *Vehicular technology conference (VTC 2010-Spring), 2010 IEEE* (Vol. 71, pp. 1–5).

[90] Takeshita, O. Y. (2007). Permutation polynomial interleavers: An algebraic-geometric perspective. *IEEE Transaction for Information Theory, 53*(6), 2116–2132.

[91] Nimbalker, A., Blankenship, Y., Classon, B., & Blankenship, T. K. (2008). ARP and QPP interleavers for LTE turbo coding. In *Proceedings Wireless communications and networking conference, 2008. WCNC 2008. IEEE* (pp. 1032–1037), Las Vegas, NV.

[92] Trifina, L., Tarniceriu, D., & Munteanu, V. (2011). Improved QPP interleavers for LTE standard. *10th International symposium on signals, circuits and systems (ISSCS), 2011* (pp. 1–4).

[93] Wang, G., Vosoughi, A., Shen, H. Cavallaro, J. R., & Guo, Y. (2013). Parallel interleaver architecture with new scheduling scheme for high throughput configurable turbo decoder. In *2013 IEEE international symposium on circuits and systems (ISCAS)* (pp. 1340–1343).

[94] Wang, G., et al. (2014). Parallel interleaver design for a high throughput HSPA+/LTE multi-standard turbo decoder. *IEEE Transactions on Circuits and Systems I: Regular Papers, 61*(5), 1376–1389.

[95] Chi, C.-L. (2013). Quadratic permutation polynomial interleavers for LTE turbo coding. *International Journal of Future Computer and Communication, 2*(4), 364–367.

[96] Yazdani, M. R., Hemati, S. & Banihashemi, A. H. (2004). Improving belief propagation on graphs with cycles. *IEEE Communications Letters, 8* (1), pp. 57- 59.

[97] Chen, R., Huang, H. & Xiao, G. (2007). Relation Between Parity-Check Matrixes and Cycles of Associated Tanner Graphs. *IEEE Communications Letters, 11* (8), 674-676.

[98] Tao, T., Jones, C.R., Villasenor, J. D. & Wesel, R. D. (2004). Selective avoidance of cycles in irregular LDPC code construction. IEEE Transaction on Communications, 52(8), 1242- 1247.

[99] Hu, Y., Eleftheriou, E. & Arnold, D. M. (2005). Regular and irregular progressive edgegrowth tanner graph. *IEEE Transactions on Information Theory,* 51(1), 386-398.

[100] Lucas, R., Fossorier, M. P. C., Kou, Y. & Lin, S. (2000). Iterative decoding of one-step majority logic deductible codes based on belief propagation. *IEEE Transactions on Communications, 48* (6), 931-937.

[101] Xu, J., Chen, L., Djurdjevic, I., Lin, S. & Abdel-Ghaffar, K. (2007).Construction of Regular and Irregular LDPC Codes: Geometry Decomposition and Masking. IEEE Transactions on Information Theory, 53(1), 121-134.

[102] MacKay, D. J. C. & Davey, M. C. (2000). Evaluation of Gallager Codes for Short Block Length and High Rate Applications. *Codes, Systems, and Graphical Models: Volume 123 of IMA Volumes in Mathematics and its Applications,* 113-130, SpringVerlag, New York, 2000.

[103] Vasic, B. (2001). Structured iteratively decodable codes based on Steiner systems and their application in magnetic recording. *Global Telecommunications Conference, 2001. GLOBECOM '01. IEEE, 5*, 2954-2960.

[104] Vasic, B. & Milenkovic, O. (2004). Combinatorial constructions of low-density parity check codes for iterative decoding. *IEEE Transactions on Information Theory, 50* (6), 1156-1176.

[105] Johnson, S. J. & Weller, S. R. (2001). Construction of low-density parity-check codes from Kirkman triple systems. *Global Telecommunications Conference, 2001. GLOBECOM '01. IEEE, 2*, 970-974.

[106] Gallager, R. G. (1962). Low-density parity check codes. *IRE Trans. Inf. Theory, 39* (1), 37-45.

[107] Richardson, T. J., Shokrollahi, M. A. & Urbanke, R. L. (2001). Design of capacity approaching irregular low-density parity-check codes. *IEEE Transactions on Information Theory, 47* (2), 619-637.

[108] Luby, M. G., Mitzenmacher, M. Shokrollahi, M. A. & Spielman, D. A. (2001). Improved low-density parity-check codes using irregular graphs. IEEE Transactions on Information Theory, 47 (2), 585-598.

[109] Mansour, M. M., & Shanbhag, N. R. (2003). High-throughput LDPC decoders. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 11*(6), 976–996. doi:10.1109/TVLSI.2003.817545

[110] Lingyan, S., Hongwei, S., Keirn, Z., & Kumar, B. V. K. V. (2006). Field programmable gate array (FPGA) for iterative code evaluation. *Transactions on Magnetics, 42*(2), 226–231. doi:10.1109/TMAG.2005.861744

[111] Chandrasetty, V. A., & Aziz, S. M. (2011). FPGA Implementation of a LDPC decoder using a reduced complexity message passing algorithm. *Journal of Networks, 6*(1), 36–45. doi:10.4304/jnw.6.1.36-45

[112] Xiao-Yu, H., Eleftheriou, E., Arnold, D. M., & Dholakia, A. A. D. A. (2001). Efficient implementations of the sum-product algorithm for decoding LDPC codes. *Global Telecommunications Conference, 2001. GLOBECOM '01, IEEE, 2*, 1036–1036.

[113] Zhang, T., & Parhi, K. K. (2001a, September). VLSI implementation-oriented (3, k)-regular low-density parity-check codes. In *IEEE Workshop on Signal Processing Systems (SiPS)* (pp. 25–36). Antwerp, Belgium. doi:10.1109/ SIPS.2001.957328

[114] Zhang, T., & Parhi, K. K. (2001b). Joint code and decoder design for implementation-oriented (3, k)-regular LDPC codes. In *Proceedings of IEEE Asilomar Conference* (pp. 1232–1236). Pacific Grove, CA. doi:10.1109/ACSSC.2001.987687

[115] Zarrinkhat, P. & Banihashemi, A.(2004).Threshold values and convergence properties of majority-based algorithms for decoding regular low-density parity-check codes. *IEEE Transactions on Communications, 52*, 2087-2097.

[116] Masera, G., Quaglio, F., & Vacca, F. (2007). Implementation of a flexible LDPC decoder. *IEEE Transactions on Circuits and Systems II: Express Briefs, 54*(6), 542–546. doi:10.1109/TCSII.2007.894409

[117] Madi, A. A., Mansouri, A., & Ahaitouf, A. (2012, March). Design, simulation and hardware implementation of low density parity check decoders using min-sum algorithm. *IJCSI International Journal of Computer Science Issues, 9*(3), 83–91.

[118] Zhang, T., & Parhi, K. K. (2003). An FPGA implementation of (3, 6)-regular low-density parity-check code decoder. *EURASIP Journal on Applied Signal Processing,*

Special Issue on Rapid Prototyping of DSP Systems, *2003*(6), 530–542. doi:10.1155/S1110865703212105

[119] Cheng, J., -F, & Ottosson, T. (2000). Linearly approximated log-MAP algorithms for turbo decoding. *Vehicular Technology Conference Proceedings, 2000. VTC 2000-Spring Tokyo. 2000 IEEE 51st*, 3, (pp. 2252-2256), Tokyo. pp. 2252-2256.

[120] Robertson, P., Villebrun, E. & Hoeher, P. (1995). A comparison of optimal and sub-optimal MAP decoding algorithms operating in the log domain. *Communications, 1995. ICC '95 Seattle, 'Gateway to Globalization', 1995 IEEE International Conference on*, 2, (pp. 1009-1013), Seattle, WA.

[121] Viterbi, A. J. (1998). An intuitive justification and a simplified implementation of the MAP decoder for convolutional codes. *IEEE journal on selected areas in communication, 16*(2), 260-264.

[122] G. Montorsi, G. & Benedetto, S. (2000).Design of fixed-point iterative decoders for concatenated codes with interleavers. In Proceedings *Global Telecommunications Conference, 2000. GLOBECOM '00. IEEE*, 2, (pp. 801-806), San Francisco, CA.

[123] H. Michel, H. & When, N. (2001). Turbo-decoder quantization for UMTS. *IEEE Communications Letters. 5* (2), 55-57.

[124] Park, G., Yoon, S., Kang, C. & Hong, D. (2000). An implementation method of a turbo-code decoder using a block-wise MAP algorithm. *Vehicular Technology Conference, 2000. IEEE-VTS Fall VTC 2000. 52nd, , 6,* (pp.2956-2961), Boston, MA..

[125] GPP TR 25.913 V7.3.0 (2006-03) Requirements for evolved UTRA (Release 7)

[126] Sharma, S. (2012). A novel weighted multilevel space-time trellis coding scheme. *Journal of Computers and Mathematics with Applications. 63*(1), 280–287.

[127] Jain, D., & Sharma, S. (2014). Adaptively grouped multilevel space-time trellis codes. *Wireless Personal Communications: An International Journal, 74(*2), 415–426.

[128] Jain, D., & Sharma, S. (2014). Adaptively grouped multilevel space–time trellis codes combined with beam-forming and component code selection. *Wireless Personal Communications, An International Journal, 77*(4), 2549–2563.

[129] Jain, D., & Sharma, S. (2014). A novel grouped multilevel dynamic space–time trellis coding scheme. *International Journal of Communication Systems*, *28*, 1168-1179 doi:10.1002/dac.2754.

[130] Sharma, S., Attri, S., & Chauhan, R. C. (2003). A Simplified and efficient implementation of FPGA based turbo decoder. In *IEEE international proceedings of performance, computing, and communications conference, 2003*, 207–213.

[131] Dawid, H., & Meyr, H. (1995). Real-time algorithms and VLSI architectures for soft output MAP convolutional decoding. *Proceedings of Personal, Indoor, and Mobile Radio Communications, PIMRC'95. Wireless: Merging onto the Information Superhighway, 1*, 193–197.

[132] Worm, A., Lamm, H., & Wehn, N. (2001). VLSI architectures for high-speed MAP decoders. *Proceedings of 14th VLSI Design*, 446–453.

[133] Li, Y., Elassal, M., & Bayoumi, M. (2004). Power efficient architecture for (3, 6) low density parity check code decoder. In *Circuits and Systems, 2004, ISCAS '04.Proceedings of the 2004 International Symposium, 4* (pp. IV–81-4) Los Angles, CA doi:10.1109/ISCAS.2004.1328945

[134] Parhi, K. K. (1999). VLSI digital signal processing systems: Design and implementation. New York, NY: John Wiley & Sons.

[135] Zhang, T., & Parhi, K. K. (2002, October 16–18). A 54 MBPS (3, 6)-regular FPGA LDPC decoder. In *Signal Processing Systems, 2002 (SIPS '02), IEEE Workshop on* (pp. 127–132). doi:10.1109/SIPS.2002.1049697

[136] Zhang, T., Wang, W., & Parhi, K. K. (2001, May). On finite precision implementation of low density parity-check codes decoder. *Proceedings of 2001 IEEE International Symposium on Circuits and Systems* (pp. 202–205). Sydney.doi:10.1109/ISCAS.2001.922207

LIST OF PUBLICATIONS

[1] Verma, S. & Sharma, S. (2016). FPGA implementation of low complexity LDPC iterative decoder. *International Journal of Electronics, 103*(7), 1112-1126. DOI: 10.1080/00207217.2015.1087052 (SCI Indexed; Impact factor: 0.459)

[2] Pasricha, S. & Sharma, S. (2016).Novel Interleaver Design for Turbo codes. *Wireless Personal Communications, Springer, 86*(2), 727-749.(SCI Indexed; Impact factor: 0.653)

[3] Verma, S. & Kumar, S. (2015). High-performance VLSI architectures for turbo decoders with QPP interleaver. *International Journal of Electronics*, *102*(4), 599-618. (SCI Indexed; Impact factor: 0.459)

[4] Verma, S. & Sharma, S. (2011). An FPGA Realization of Simplified Turbo Decoder Architecture. *International Journal of the Physical Sciences,6*(10), 2338–2347. (Formerly SCI Indexed; when published Impact Factor = 0.54)

[5] Pasricha, S. & Sharma, S. (2009). FPGA based design of Reed Solomon codes. *Indian Journal of Science and Technology, 2* (4) 48-52.