EXPLORATION OF HIGH-LEVEL SYNTHESIS FOR 5G WIRELESS CHANNEL CODING ARCHITECTURE

A PROJECT REPORT

Submitted by

CB.EN.U4ECE17131 K SNEHITH REDDY
CB.EN.U4ECE17133 K JAYASAMPATH
CB.EN.U4ECE17144 N KIRAN KUMAR
CB.EN.U4ECE17168 K YESWANTH

Under the guidance of

DR B BALA TRIPURA SUNDARI

in partial fulfilment of the requirements for the award of the degree of

BACHELOR OF TECHNOLOGY

IN

ELECTRONICS AND COMMUNICATION ENGINEERING

AMRITA SCHOOL OF ENGINEERING AMRITA VISHWA VIDYAPEETHAM

COIMBATORE 641112

April 2021

AMRITA VISHWA VIDYAPEETHAM

AMRITA SCHOOL OF ENGINEERING, COIMBATORE, 641112

BONAFIDE CERTIFICATE

This is to certify that the project report entitled **"EXPLORATION OF HIGH LEVEL SYNTHESIS FOR 5G WIRELESS CHANNEL CODING ARCHITECTURE"**

Submitted by

CB.EN.U4ECE17131	**K SNEHITH REDDY**
CB.EN.U4ECE17133	**K JAYA SAMPATH**
CB.EN.U4ECE17144	**N KIRAN KUMAR**
CB.EN.U4ECE17168	**K YESWANTH**

in partial fulfilment of the requirements for the award of the **Degree of Bachelor of Technology** in **ELECTRONIC AND COMMUNICATION ENGINEERING** is a Bonafide record of the work carried out under my guidance and supervision at Amrita School of Engineering, Coimbatore.

Project Advisor	Project Coordinator
Name- Dr Bala Tripura Sundari B	Name-
Designation- Associate Professor, ECE	Designation-

Chairperson ECE
Dr M. Jayakumar

The project was evaluated by us on:

Internal Examiner External Examiner

DEDICATION

"We would like to dedicate our project to our family, teachers and friends."

ACKNOWLEDGEMENT

We would like to take this opportunity to express our gratitude to all the people who have helped us in our project.

Our sincere and hearty thanks and appreciations go firstly to our supervisor Ms. DR. B. Bala Tripura Sundhari, whose guidance and suggestions have given us much insight into the project work. Her insightful observation and effective feedback inspired us during the project. Secondly, we express our gratitude to our class advisors Ms Karthiga Balamurugan and Mr Manoj Kumar Panda for their encouragement to take this challenging project. It has been a joy and great privilege to study under their guidance and supervision. Furthermore, it is our honour to benefit from their personalities and diligence, which will treasure our whole life.

We would like to thank Mr N Mohan Kumar (Asst. Professor), Ms Karthiga Balamurugan (Asst. Professor) Department of Electronics and Communication Engineering for their valuable suggestions during project reviews and being supportive and encouraging towards the completion of the project.

Our thanks to all the staff of our college and to our friends, who really boosted our confidence to complete the project successfully and make it a fruitful one.

TABLE OF CONTENTS

ABSTRACT

In today's fast changing world, the demand for Mobile Internet is increasing day by day. The fourth generation (4G) systems are now in use world-wide. The present day's 4G LTE has some challenges left such as higher data rates and spectral efficiency due to the tremendous increase in the number of mobile internet users. This led us to a situation of replacing Turbo codes of 4G systems with a channel code that promises higher throughputs. Ever since, the 3GPP had accepted the LDPC codes as a channel coding scheme for 5G wireless communications, a lot of research is going on to optimise the decoder. In 5G, LDPC codes and polar codes are used for error correction for the data channel and control channel respectively. The prime objectives of fifth generation systems are higher data rate, higher spectral efficiency, higher throughput, higher bandwidth, and higher energy efficiency that too at lower latency. Channel coding plays a vital role in any wireless communication system. Our project presents a novel efficient high-throughput encoder and decoder for Low Density Parity Check codes for 5^{th} generation wireless Communications. This work proposes strategies to achieve high-throughput channel coding architecture for LDPC codes. The proposed design achieves peak throughputs in lower latencies, which meets the throughput and latency requirement for the 5G NR standard.

LIST OF SYMBOLS

SYMBOL	DESCRIPTION
H	Parity Check Matrix
Z	Expansion Factor or Lifting Size
K	Information Bits Length
N	Code word Length
C	Transmitted Code word
C’	Decoded Code word
r	Received Vector
n	Noise component
N_0	Noise Variance
L_i	i^{th} Log Likelihood Ratio component
E_s	Energy of the transmitted Signal
σ^2	Variance
E_N	Energy of the noise

LIST OF ABBREVIATIONS

ABBREVIATION	EXPANSION
5G NR	Fifth Generation New Radio
AWGN	Additive White Gaussian Noise
BER	Bit Error Rate
BG	Base Graph
BPSK	Binary Phase Shift Keying
FER	Frame Error Rate
FPGA	Field Programmable Gate Array
HLS	High-Level Synthesis
LLR	Log Likelihood Ratio
PCM	Parity Check Matrix
QAM	Quadrature Amplitude Modulation
QC-LDPC	Quasi Cyclic Low-Density Parity Codes
RTL	Register Transfer Level
SNR	Signal-to-Noise Ratio

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1
INTRODUCTION

INTRODUCTION

1.1 BACKGROUND

What is 5G? Well, we are all experiencing 4G now. 5G is the fifth generation of wireless network technology and so all the big carriers are working on building out their very own 5g networks right now all over the world. The main objective of 5g is again speed. Every new wireless communication's generation is notably faster and offers more capabilities than the previous generation networks. Remember the old and really slow 2G and 3G networks, and the way 4G is far better than those. 5G NR takes us a step forward not only in speed and latency but also in the frequency bands it supports. Comparing the frequency band of 4G with 5G, with 4G we can only go up to 2.5GHz while 5G covers a wide spectrum from 1GHz to 95GHz. This unprecedented flexibility that 5G allows opens up to seemingly endless new applications and hugely surpasses the limitations with 4G that we have today.5G frequencies can potentially overlap with 4G, but they are not compatible with one another. Once 5G becomes the mainstream 4G and 3G will become obsolete. To take advantage of 5g we need 5G supporting devices with 5G radios and a 5G network. Carriers are building up their 5g network in certain cities, one location at a time. So, we can't just get a software update to upgrade from 4G to 5G.

5G NR supports enhanced mobile broadband (eMBB), ultra-reliable and low-latency communications (uRLLC) and massive machine-type communications (mMTC). The 5G NR is taking a completely different path from LTE in the area of channel coding architecture. For 5G NR Low-Density Parity Check (LDPC) coding is replacing the existing Turbo coding that was previously used for PDSCH (Physical Downlink Shared Channel) coding and Polar Codes are replacing the Tail Biting Convolutional Codes (TBCC) used previously for PDCCH (Physical Downlink Control Channel) coding. Polar Codes and LDPC Codes are getting considerably more perception in view of their innate benefits of excellent bit error rate execution, quick encoding, and decoding processes. For 5G NR, the modulation schemes supported are QPSK, 16QAM, 64QAM, and 256QAM for uplink and downlink, $\pi/2$-BPSK, QPSK, 16QAM, 64QAM and 256QAM for downlink.

1.2 MOTIVATION

There are immense challenges in the 5G standard channel coding technique implementation which highly motivated us to work on the design of a high throughput decoder for LDPC codes. The 5G promises a throughput of 2GBPS to 200 GBPS in less than 1ms. Among the algorithms available for channel coding, message passing iterative decoder promises better results. This motivated us to work on the message passing min-sum iterative decoder and to make the min-sum algorithm more efficient and optimised. In this report, we propose an efficient channel coding scheme for 5G wireless communications implemented using High-Level Synthesis.

1.3 OBJECTIVE

The objective of our project is to design an efficient High-Level Synthesis framework to utilize in 5G wireless communications. Among the algorithms, the min-sum approximation algorithm promises better results. This work is to design a high performance Channel coding scheme using LDPC encoding and decoding implemented using High-Level Synthesis. The proposed work exploits the advantages offered by HLS for developing channel coding architecture for 5G wireless using LDPC codes.

1.4 ORGANISATION OF WORK

Organisation of this work is as follows, Chapter 2 deals with literature survey of channel coding schemes which illustrated several existing algorithms in the present day. It also has exploration of high-level synthesis which illustrates the transformation techniques of HLS. Chapter 3 deals with the approach of LDPC encoding and decoding. It also illustrates the proposed decoding algorithm along with the techniques for high throughput. Chapter 4 deals with the implementation details of the proposed design. It also specifies the exploitation of HLS for the proposed architecture which promises high throughputs. Chapter 5 summarizes proposed LDPC decoder design results as well as the performance of the proposed LDPC decoder design compared with Xilinx LDPC and base paper. It also has obtained module architecture and a few BER and FER plots obtained in different scenarios.

CHAPTER 2
LITERATURE SURVEY

LITERATURE SURVEY

2.1 INTRODUCTION TO HLS

High-Level Synthesis (HLS) also known as algorithm synthesis or C synthesis is an efficient automated hardware design process generated from the user's algorithm [2]. HLS reduces the effort of HDL design capture. HLS debugger allows flexibility in the final hardware implementation in order to meet the user's design constraints. It is a transformation of a behavioural C/C++ model into a Verilog RTL. In addition to this behavioural as an input, the tools also take things like constraints and directives [2]. The output in Verilog RTL will include a state machine and a data path and that data path will be composed of various different functional units, what we call building blocks for high-level synthesis and these include things like multipliers, adders, dividers, floating-point units, etc.

HLS will create a hardware description RTL from an untitled algorithm in C/C++, the big difference here is not the transformation from C/C++ to Verilog, it's the fact that there are clock cycles inserted into the design to make it fit more correctly into the downstream flow.

It will build a state machine, it will take care of all memories, registers and it will automatically break the entire design up into the discrete clock cycles that are required by the logic synthesis tools, downstream all of this taken care of by the high-level synthesis tool.

A basic high-level synthesis flow includes the following steps,

The behavioural model is input. It is read, the algorithm is processed and a number of algorithmic optimizations are implemented on the design [5]. These tend to be pretty language-focused and are very similar to a lot of compiler-type optimizations.

We then build a control and data flow graph, analyse that and perform optimizations, deal with the libraries that we have. This includes things like extracting timing information from various functional units etc.

One of the main tasks of a high-level synthesis tool is embodied in the Scheduler [7]. This is the algorithm inside the tool that we'll take a look at the control and data flow graph and split it up into clock cycles. This is a very important and high-value step and it takes into account a lot of the timing and latency constraints that we put into the tool at the beginning.

The last major step is binding. In binding, we basically tie all of the operations that are in the control and data flow graph to actual functional units that are going to appear in the Verilog RTL. In that process, we implement a very important step in high-level synthesis called sharing. What that means is that we determine exactly what functional units can be shared with other functional units and exactly the same thing for registers. So that we can minimize the number of resources that are present in the final design.

2.1.1 EXPLORATION OF HLS

High Level Synthesis can synthesize a hardware implementation from a high-level description. The techniques of HLS help in abstracting design to a level higher than RTL [2]. HLS has some constraints that are needed to be taken care of. Those constraints are latency data path delay, throughput and area utilization. Lower latency leads to better execution speed. Less resource utilization leads to reduced area which in turn leads to reduced power usage [1]. A design is better if it has lower latency and less resource utilization. HLS techniques are used to abstract the optimal design space exploration.

2.1.1.1 HLS TRANSFORMATION TECHNIQUES

High level synthesis transformations are very helpful in optimizing the user’s design. Loops in HLS provide more chances of parallelism. Some transformation techniques available in HLS are pipelining, unfolding, retiming. High complexity caused by control hazards can be resolved using unfolding loop transformation, pipelining and loop unrolling [2]. Few techniques are specified below.

2.1.1.2 PIPELINING

Pipelining is used to split a process into multiple stages and executes the stages in an overlapping fashion [10]. It enables concurrency. Multiple inputs can be processed at the same time. Pipelining increases the throughput of the design. Pipelining is essential for any design which promises lower latency. Pipelining increases the efficiency of resource utilization. Pipeline structure can execute two operations in a single clock cycle as shown in Fig 1.

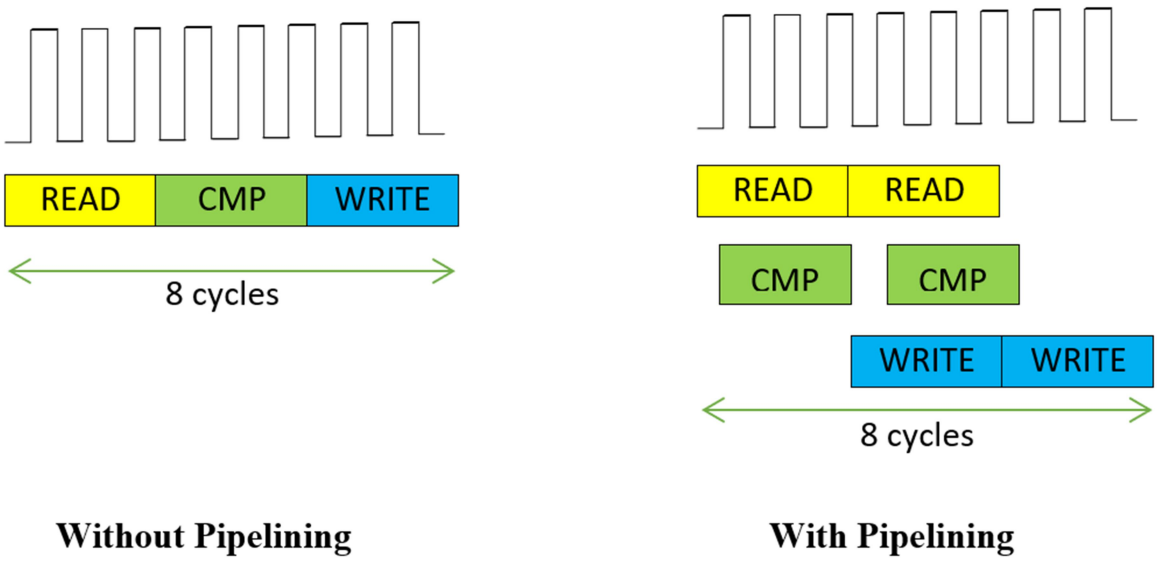

Fig 1 – Operations using Pipelining Illustration

Dependencies among iterations should be avoided in pipelining. Loops pipelining helps to execute loops in a faster manner. In the proposed architecture pipelining is used in row and column operations of LDPC decoder.

2.1.1.2 LOOP UNROLLING

Loop Unrolling reduces the number of loop iterations. In unrolling multiple iterations occur in a single iteration as shown in Fig 2. Loop control overhead is reduced using loop unrolling. Every unrolled loop will be processed in parallel. Complexity caused by control hazards can be resolved using loop unrolling. In the proposed architecture loop unrolling is used in the decoder.

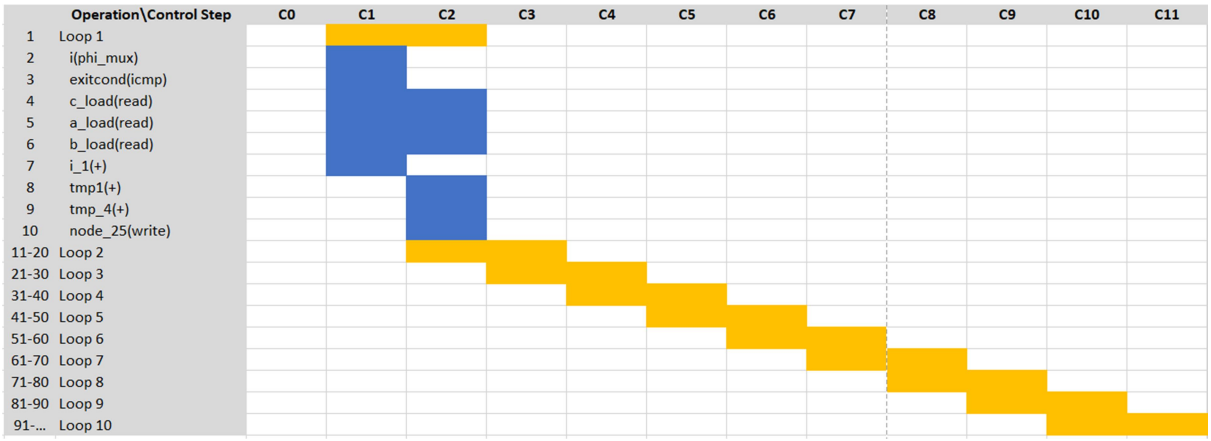

Fig 2 – Loop Unrolling Illustration

2.1.1.3 ARRAY PARTITIONING

When there is a bottleneck in bandwidth Array Partitioning is used. Array partitioning means that a large array is divided into small subarrays in the lane of internal FPGA memory block which is called Block Random Access memory (BRAM). This increases the memory bandwidth. When different resources access a data at the same time, data access conflict occurs. Array Partitioning reduces this conflict.

2.2 INTRODUCTION TO CHANNEL CODING

Channel coding is considered as a crucial component of today's Wireless Communications [4]. The Third Generation Partnership Project (3GPP) has recently finalised two physical layer Channel Coding Schemes such as Low-Density Parity Check (LDPC) and Polar codes. These codes replace the Turbo and Convolution codes used for 4G long term evolution (LTE) [5]. These codes promise improved throughputs, latency and reliability compared to 4G systems.

Channel coding is a vital and complex component of any wireless mobile communication systems. Channel coding is nothing but correcting the transmission errors that are caused by poor signal strength, noise, and interference of the channel. This is achieved by employing a channel encoder within the transmitter (be it the base station or the handset) to process each information block. The information block has K information bits and the rest are parity bits. Encoder converts it into a longer encoded block comprising N bits greater than K encoded bits, which are transmitted.

In the receiver, the extra (N − K) encoded bits provide the channel decoder with information that permits it to detect and proper transmission errors within the first K information bits.

2.3 LDPC ENCODING AND DECODING

2.3.1 STRUCTURE OF LDPC CODES

LDPC codes were originally devised by Gallager and re-discovered by Mackay et al. In this section, the LDPC codes are discussed and we also specify the main characteristics of 5G LDPC codes. These codes are Quasi-Cyclic LDPC codes which belong to a family of protograph codes [3]. A QC-LDPC code can be characterised by null space of an array of sparse circulants of same size [5]. LDPC codes can be depicted using two ways: one using PCM i.e. a matrix kind representation and the other one using graphical representation. The graphical representation is also known as Tanner Graph.

2.3.1.1 MATRIX REPRESENTATION

First way of depicting the LDPC codes is using a parity check matrix. The PCM of a QC-LDPC code is defined by its base graph and an expansion factor. 5G NR offers two base graphs of different sizes. Base Graph (BG1) of size 46*68 for longer code word lengths. Base Graph (BG2) of size 42*52 for shorter code word lengths. Entries of the base graph are replaced by a circularly right shifted identity matrix and a zero matrix of size Z*Z. where Z is the expansion factor. For the sake of simplicity, we describe part of BG2 as an example to show matrix representation as shown in Fig 3. The PCM of the below BG2 is obtained by replacing entries with a matrix of size Z*Z.

24	14	23	37	−1	−1	47	−1	−1	8	1	0	−1	−1	−1	−1	−1	−1	−1	−1
5	−1	−1	12	19	12	19	8	29	31	−1	0	0	−1	−1	−1	−1	−1	−1	−1
8	35	−1	46	47	−1	−1	−1	43	−1	0	−1	0	0	−1	−1	−1	−1	−1	−1
−1	41	6	−1	36	28	28	14	12	37	1	−1	−1	0	−1	−1	−1	−1	−1	−1
8	16	−1	−1	−1	−1	−1	−1	−1	−1	−1	5	−1	−1	0	−1	−1	−1	−1	−1
41	42	−1	−1	−1	26	−1	27	−1	−1	−1	1	−1	−1	−1	0	−1	−1	−1	−1
27	−1	−1	−1	−1	7	−1	31	−1	30	−1	17	−1	−1	−1	−1	0	−1	−1	−1
−1	7	−1	−1	−1	13	−1	9	−1	−1	−1	6	−1	37	−1	−1	−1	0	−1	−1
3	43	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	8	−1	−1	−1	−1	−1	0	−1
−1	2	−1	−1	−1	−1	−1	−1	30	−1	40	35	−1	−1	−1	−1	−1	−1	−1	0

Fig 3: Example of BG2 base matrix – 10 rows and 20 columns

2.3.1.2 TANNER GRAPH REPRESENTATION

The second way of depicting LDPC codes is using Tanner Graphs. It is based on Bi-partite graph design. These graphs are made from PCM. A parity check matrix is a sparse matrix, where it has more 0's than 1's. It has two kinds of nodes. One class of nodes is the "Bit nodes" which corresponds to n bits of the code word of size n or simply the number of columns in PCM. The other class of nodes is the "Check nodes' ' which corresponds to the number of parity check equations or simply the number of rows in PCM. Let us consider a parity check matrix derived from a (6,3) code. The tanner graph representation is as shown in the Fig 4.

$$\text{PCM (H)} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 1 & 0 \end{bmatrix}$$

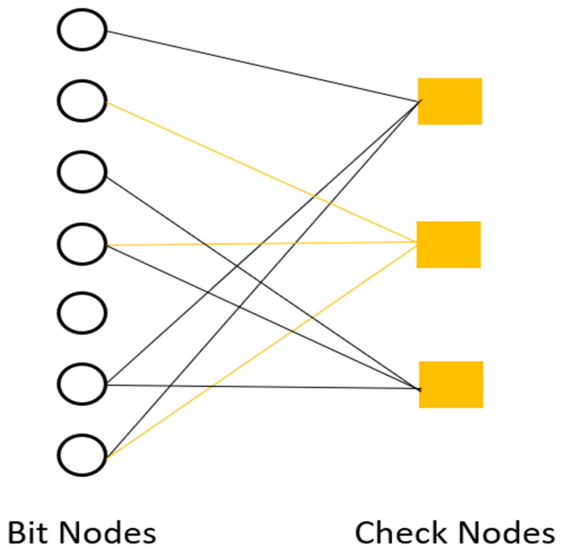

Fig 4 – Tanner Graph Illustration

2.3.1.3 5G NR LDPC CHARACTERISTICS

As specified above, QC-LDPC codes have been accepted as the channel coding scheme for 5G wireless communications. General structure of the LDPC base graph is depicted in Fig-. The rows are divided into two parts: core check rows and extension check rows. The columns are divided into three categories: information columns, core parity columns and extension parity columns. As shown in the Fig 5, the general structure of a 5G base graph is divided into five different sub categories namely A, D, E, O and I. The submatrix A corresponds to information bits. D corresponds to the first set of parity bits and is a square matrix with double diagonal structure. The importance of double diagonal structure is specified in the next chapter. The first column of double diagonal structure is three. Submatrix O is a zero matrix. Submatrix E corresponds to SPC (single parity check) rows. Submatrix 'I' is an identity matrix which corresponds to the second set of parity bits.

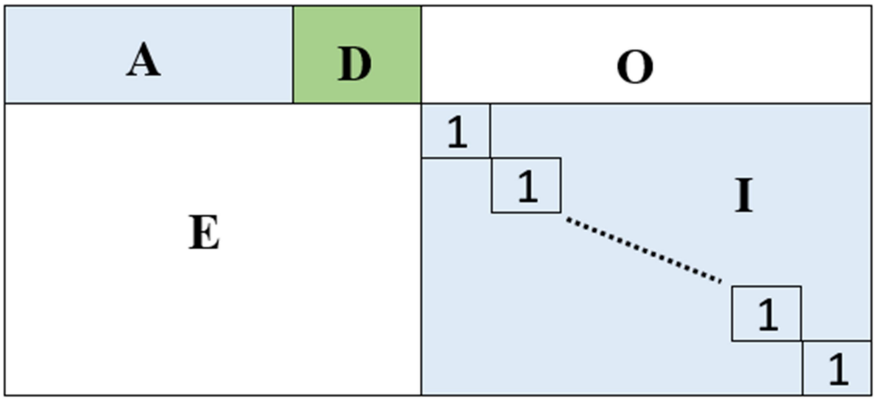

Fig 5 – General structure of 5G Base Graph

The 3GPP considered two different base graphs. Both BG1 and BG2 are rate-compatible. They both have similar structures. Let K be the information bit length in the code word. BG1 supports larger block length i.e. 500 K 8448. It supports high rate transmissions i.e. 1/3. While BG2 is most recommended for the smaller block lengths ($40 \leq K \leq 2560$). It supports lower code rates ($1/5 \leq R \leq 2/3$), where R is the code rate of the transmission.

2.3.1.4 EXPONENT MATRIX AND LIFTING SIZE

Expansion factor Z is also known as lifting sizes. The number of shift coefficient designs is 8 for BG1 and BG2. In 5G standard all lifting sizes are divided into eight sets based on parameter t, where 't' is defined as $t * 2^i$. The different sets of shift coefficients and its lifting sizes are listed in Table 1.

EXPONENT MATRIX	LIFTING SIZE SET
P_1 (a = 2)	{ $a \times 2^j$ | j = 0,1,2,3,4,5,6,7 }
P_2 (a = 3)	{ $a \times 2^j$ | j = 0,1,2,3,4,5,6,7 }
P_3 (a = 5)	{ $a \times 2^j$ | j = 0,1,2,3,4,5,6 }
P_4 (a = 7)	{ $a \times 2^j$ | j = 0,1,2,3,4,5 }
P_5 (a = 9)	{ $a \times 2^j$ | j = 0,1,2,3,4,5 }
P_6 (a = 11)	{ $a \times 2^j$ | j = 0,1,2,3,4,5 }
P_7 (a = 13)	{ $a \times 2^j$ | j = 0,1,2,3,4 }
P_8 (a = 15)	{ $a \times 2^j$ | j = 0,1,2,3,4 }

Table 1 – Relation between exponent matrix and Lifting Size

2.3.1.5 LIFTING SIZES

Lifting sizes or expansion factor Z supported by standard 5G NR QC-LDPC codes is listed in Table.

Z		j							
		0	**1**	**2**	**3**	**4**	**5**	**6**	**7**
a	**2**	2	4	8	16	32	64	128	256
	3	3	6	12	24	48	96	192	384
	5	5	10	20	40	80	160	320	
	7	7	14	28	56	112	224		
	9	9	18	36	72	144	288		
	11	11	22	44	88	176	352		
	13	13	26	52	104	208			
	15	15	30	60	120	240			

Table 2 – Lifting Sizes supported by 5G NR

2.3.2 LDPC ENCODING

The LDPC code is a kind of linear block code based on parity check matrix H of size m*n, where m represents the number of check nodes, n represents the number of bit nodes present in the tanner graph of the matrix. The parity check matrix

is a sparse matrix which helps in reducing the complexity for computations. There are a few existing methods to construct LDPC codes, such as Progressive Edge-Growth (PEG-LDPC), QC-LDPC and so on.

Among them we used, parity check matrix of the QC-LDPC expanded by Z circulant permutation matrix. We devised a new approach to encode LDPC codes using Double Diagonal Structure which is specified clearly in the next section. The parity check matrix has a special structure called Double diagonal structure. The structure is explained in detail in the next section. This new approach executes in lesser complexities than a regular LDPC encoding.

2.3.2 LDPC DECODING

There are various methods of newly discovered algorithms available for decoding of LDPC codes. Efficiency of the LDPC codes majorly depends on decoding. The less the complexity in decoding the more efficient is the approach. Primarily there are two kinds of decoding. They are Hard Decision Decoding and Soft Decision Decoding. Few Algorithms like sum-product algorithm, belief propagation algorithm, message passing algorithm works on Hard Decision Decoding. However, soft decision decoding is recommended for better results.

2.3.2.1 HARD DECISION DECODING

Consider a parity check matrix H as shown below. The process starts from the H matrix. Out of the check nodes for every bit C_n, has a store that is modified by C_n where C is the code word of length n. When there are many non-zero check nodes, the greater the probability of error.

$$H = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \end{bmatrix}$$

The error is recovered by flipping the faulty bit. One or more errors can be rectified using this scheme. If C_n consists of an error and different bits influencing its check nodes are still an error. The bits in the boxes are errors. At the decoding of C_n very faulty bits may or may not be recovered. Then bit

nodes and check nodes which are not instantly attached can affect the existing C_n. Thus the probability of error is maximum in this type of decoding.

2.3.2.2 SOFT DECISION DECODING

It is based on belief propagation and indeed a better decoding process than hard decision decoding. Soft decision decoding starts from true values of the channel. For example, in BPSK over AWGN channel decoding of one bit operations happens by using the received vector. That decision is made after the LLR calculations.

2.3.2.3 DECODING ALGORITHMS

2.3.2.3.1 SUM PRODUCT ALGORITHM

Sum product algorithm is a soft decision decoding algorithm which is based on belief propagation. The input to the bit node is the LLR. Bit node procedure can be described by the following equation

$$b_i = LLRn * \sum_{j \neq 1} Cj$$

Where b is the bit node and C_j is the j^{th} bit in the codeword C. n is the collection of bit nodes. Check node process can be indicated by the following equation,

$$C_k = 2\tanh^{-1}\left(\prod_{k \neq 1} \tanh\left(\frac{bi}{2}\right)\right)$$

Where k is the degree of check node. Here multiple non-linear behaviours are needed in the algorithm. Thus, completes in large computational complexity.

2.3.2.3.2 MIN-SUM ALGORITHM

This is the most recommended algorithm as it is having the lower complexity compared to the above algorithms. But optimising this algorithm is also a tough challenge. In the beginning of the process, the LLR values are applied to bit nodes which in turn reached check nodes as well. The row and column updating operations will happen through every iteration. It is also a belief propagation algorithm. At the end of every iteration the estimated received value gets updated.

Then the decision process starts and the bit which is transmitted is estimated. Our proposed technique is mentioned in the next sub section.

2.3.2.3.3 PROPOSED DESIGN

Various algorithms for LDPC encoding and decoding are specified in the above sections. We are going to use Min-Sum approximation as the decoding algorithm for the proposed design. We also made a few modifications for the existing algorithm in order to achieve higher throughputs. The details of the modifications are clearly specified in the next chapter. We exploited the advantages offered by HLS for achieving higher throughputs in lower latencies.

CHAPTER 3
APPROACH

APPROACH

The block diagram of the proposed design is as shown in Fig 6.

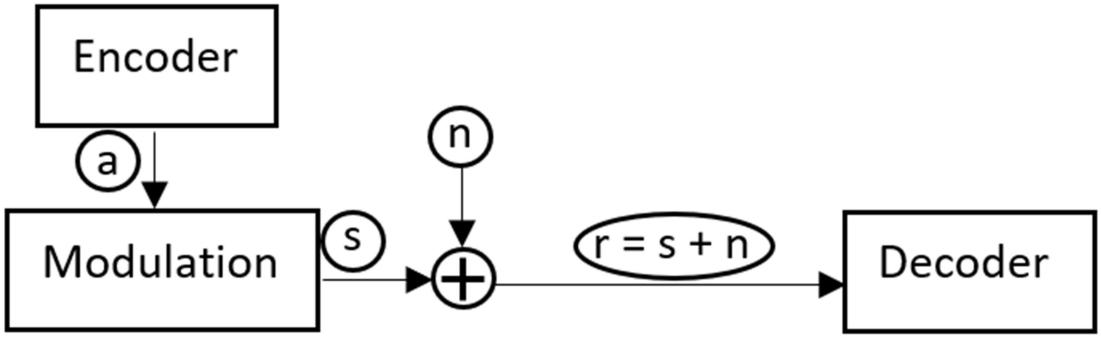

Fig 6 – Proposed Design

3.1 LDPC ENCODING AND TRANSMISSION

3.1.1 ENCODING

Encoding LDPC codes in 5G standard is slightly different from encoding the Hamming Codes. We use Parity Check Matrix (PCM) instead of Generator matrix for the encoding procedure. Consider a toy-example of encoding using PCM which can be quickly generalized to the encoding of LDPC Codes for 5G communications.

Consider an input say, a (6, 3) code with parity check matrix (H) as given below,

$$\text{PCM (H)} = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 \end{bmatrix}$$

A message vector say M [m1, m2, m3] and the codeword will be [m1 m2 m3 p1 p2 p3] where p1, p2, p3 are parity bits. These three parity bits are computed using parity check matrix (H) and message vector (M) i.e. equation (i)

$$M = [m1 \quad m2 \quad m3]$$

$$C = [m1 \quad m2 \quad m3 \quad p1 \quad p2 \quad p3]$$

$$H*C^T = 0 \qquad \text{--- (i)}$$

$$\begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} m1 \\ m2 \\ m3 \\ p1 \\ p2 \\ p3 \end{bmatrix} = 0$$

Solving the above equation, gives

$$p1 = m1 + m2$$

$$p2 = m2 + m3$$

$$p3 = m3 + m1$$

This is how the encoding of the message bits carried out. Now this parity check matrix is generalised to 5G standard. Encoding to the 5G standard is different from the basic encoding process. As mentioned in the previous chapter, there are two different base matrices of size 46*68 (BG1) and 42*56 (BG2) authorized by 3GPP. Each element of the base matrix is replaced by an identity matrix of size z*z where z is the expansion factor of the transmission. A sample base matrix is considered below. 10 rows and 20 columns from BG2 base matrix which has an expansion factor of 48.

24	14	23	37	−1	−1	47	−1	−1	8	1	0	−1	−1	−1	−1	−1	−1	−1	−1
5	−1	−1	12	19	12	19	8	29	31	−1	0	0	−1	−1	−1	−1	−1	−1	−1
8	35	−1	46	47	−1	−1	−1	43	−1	0	−1	0	0	−1	−1	−1	−1	−1	−1
−1	41	6	−1	36	28	28	14	12	37	1	−1	−1	0	−1	−1	−1	−1	−1	−1
8	16	−1	−1	−1	−1	−1	−1	−1	−1	−1	5	−1	−1	0	−1	−1	−1	−1	−1
41	42	−1	−1	−1	26	−1	27	−1	−1	−1	1	−1	−1	−1	0	−1	−1	−1	−1
27	−1	−1	−1	−1	7	−1	31	−1	30	−1	17	−1	−1	−1	−1	0	−1	−1	−1
−1	7	−1	−1	−1	13	−1	9	−1	−1	−1	6	−1	37	−1	−1	−1	0	−1	−1
3	43	−1	−1	−1	−1	−1	−1	−1	−1	−1	−1	8	−1	−1	−1	−1	−1	0	−1
−1	2	−1	−1	−1	−1	−1	−1	30	−1	40	35	−1	−1	−1	−1	−1	−1	−1	0

Fig 7 – BG2 example for illustrating Encoding

As the expansion factor is 48, the entries of the base matrix are from -1,0,1 to 47. Each entry will expand to a 48*48 identity matrix. Identity matrix is cyclically right shifted by k times where k is the particular entry of the base matrix. If the entry is -1, it is replaced by a zero matrix of size z*z. If the entry is 0, it is replaced by the Identity matrix. If it is greater than 0, it is replaced by an identity matrix that is shifted that many times.

The resultant matrix is the parity check matrix, which is used to find the parity bits in turn leads to determination of codeword. The aim of encoding is to generate a codeword for the given input bits. The generated codeword is modulated using BPSK and transmitted using AWGN channel.

Double diagonal structure of the base matrix helps in faster calculation of parity bits for the codeword. The double diagonal of the above considered matrix is shown in Fig 8.

$$H = \begin{bmatrix} I_1 & 0 & I_3 & I_1 & I_2 & I & 0 & 0 \\ I_2 & I & 0 & I_3 & 0 & I & I & 0 \\ 0 & I_4 & I_2 & I & I_1 & 0 & I & I \\ I_4 & I_1 & I & 0 & I_2 & 0 & 0 & I \end{bmatrix}$$

Fig 8 – Double Diagonal Structure

The double diagonal structure is very interesting and is present in every 5G base matrix. We used this structure to do encoding in lower complexity. This made the encoding ease.

3.1.2 MODULATION

Signal modulation is very essential in wireless communications. Modulation helps signals to travel longer distances. There are different modulation schemes available for wireless communications. The modulation schemes supported by 5G are BPSK, QPSK, 16 QAM, 64 QAM and 256 QAM. For the sake of simplicity BPSK modulation scheme is considered for our proposed design.

Binary Phase Shift Keying where binary refers to two phase offsets. If the input digit is 0, it has 0^o degree phase shift and 180^0 phase shift if it is 1. Hence, if the input binary digit is a then the modulated bit is 2a-1.

For a = 1, modulated output is 1.

a = 0, modulated output is -1.

3.1.2.1 ADVANTAGES OF BPSK MODULATION SCHEME

- The binary 1 and 0 are separated by 180^{o} phase shift and hence it is the robust modulation.
- Due to this, the modulated signal can travel longer distances i.e. when travelling from base stations to subscriber stations.
- When demodulating, the decision must be chosen around only two values. Hence, there is lesser ambiguity while recreating the original binary sequence.

3.1.3 AWGN Channel

Any transmitted signal is high likely to prone to distortion due to noise. So, we considered AWGN channel for our transmissions. Here we monitored SNR of the transmitted signal.

3.1.3.1 CALCULATING SNR

Let the signal power be p. The noise spectral density of an AWGN channel is $No/2$. To calculate the SNR, of the proposed BPSK for this work,

Consider a simple BPSK signal as shown in Fig 9.

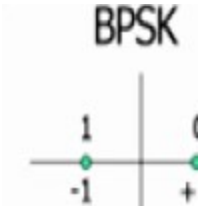

Fig 9 – Simple BPSK Waveform

Energy of the Signal is mean of square of symbols = $E_S = (-1)^2 + (1)^2 /2 = 1$

Noise power $E_N = \sigma^2$

SNR = $E_S / E_N = 1/\sigma 2$

Now that we have SNR of the signal, the no of bit errors can be calculated by,

$$BER = Q(1/\sigma) = Q(sqrt(SNR))$$

Where Q is the Q function,

$Q(x) = 0.5*\text{erfc}(x/sqrt(2))$

3.2 PROPOSED DECODING ALGORITHM

In the previous chapter, several decoding techniques were specified. We have chosen the Min-Sum decoding algorithm and modified it for achieving the higher throughputs. The decoder is the heart of the success of the LDPC codes. This decoding exploits the iterative decomposition of bit node and check-node combined-processing units which makes us to achieve area optimization and lower hardware complexity. The proposed decoding process is shown in Fig 10,

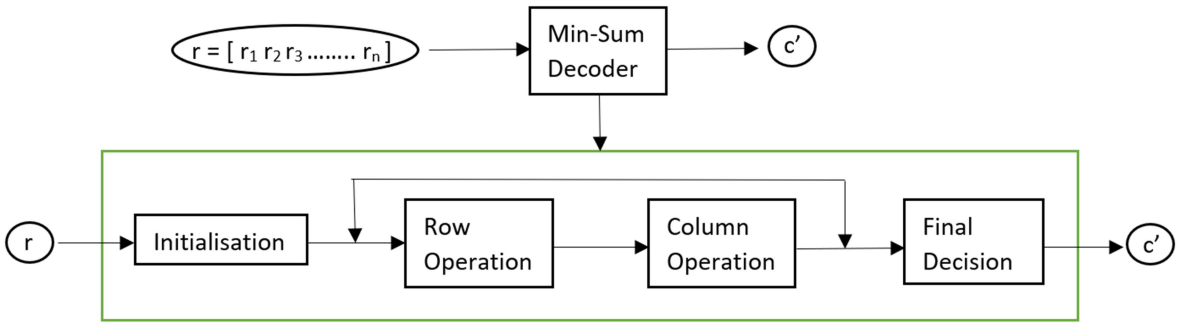

Fig 10 – Proposed Decoding algorithm

The proposed decoder is Min-Sum Iterative message passing decoder. It is an iterative decoder which works in multiple rounds of iterations and it actually consists of reasonably simple operations and the memory structure is what is important, because we need to store a lot of things in the memory and move it around, access it etc.

The decoder of the LDPC is Soft-input, Soft-Output. It deals with the belief propagation and Log-Likelihood ratio for the received values and for the bits in the codeword and then we improved using some calculations. It is iterative in the sense that it does not try to decide the entire code at once. It will use some partial information from the code and then do some decoding and then pass messages

around. So, that is sort of the idea in this, there is an approximation involved which is called min-sum approximation

So, the setting is, we have BPSK over AWGN, the noise standard deviation is, the code word c., the received vector r.

$\mathbf{c} = [c_1\ c_2\ \ldots c_n]$

$\mathbf{r} = [r_1\ r_2\ \ldots.\ r_n]$

$\mathbf{r_i = S_i + noise}$

where S_i is the output of BPSK modulation.

If the codeword bit is c_i, then

$$S_i = 1 - 2*c_i$$

The received vector is not a binary digit, it will be prone to channel noise. Eventually these binary digits will all be integer values after suitable scaling and quantisation. LLR comes into picture now, LLR is defined as the log of the ratio of probabilities of C_i given a particular bit.

$$L_i = \log [\text{prob}\ \{C_i = 0|\ r_i\}/\text{prob}\ \{C_i=1|r_i\}]$$

$$= \log[p(r_i/C_i=0)/\ p(r_i/C_i=1)]$$

For BPSK over AWGN,

$$L_i = {}^{2 * ri}/_{\sigma * \sigma}$$

Here we seek output LLR where we considered the whole received vector **r**.

$$L_i = \log[\text{prob}\{C_i = 0|\ \mathbf{r}\}/\text{prob}\{C_i=1|\mathbf{r}\}]$$

We will slowly improve the LLR, using more and more information form remaining codewords like this L_i uses only information from R_i. Here we use other bits of information slowly in an iterative manner and expand the belief that we have, making the belief that we have a bit better and better. Let us see a small example for our

proposed decoding algorithm. Consider a part of the LDPC matrix as shown in the Fig 11.

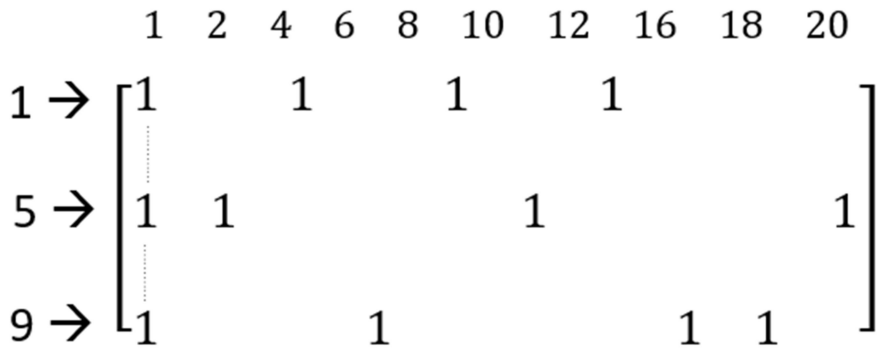

Fig 11 – Example for Decoding illustration

The first column has three ones, everything else is zero. Ones are in the first row, fifth row, and ninth row. First row has ones in the fourth, eighth and twelfth columns. This is a sparse matrix so there are a lot of zeros and few ones. This is the sort of local structure. This notion of the local structures is very important in this approximate iterative message passing decoder. Through R1, we found the first estimate. We exploited the local structure, the connection between row one, five and nine. Every row of a parity check matrix defines a single parity check code. It gives the parity check constraint that is satisfied by one subset of the bits of the codeword. So, using the SPC decoder, the second estimate is formed by the single parity check decoder. The second estimate is the LLR of the fourth, eight and twelfth rows. We did the min-sum approximation or even the exact computation with the log tan hyperbolic function etc; we computed the extrinsic estimate from this. Likewise, the other estimates are also calculated.

So, the local structure has already been given. The first estimate comes from the channel itself, second comes from the first and so on. So, every parity check in the local structure gives us more estimates for that particular bit. So, we just need this single parity check decoder which will be doing seesaw calculations

To do it more efficiently, we used other received values. Using message passing, each code will use the local structure and then pass messages to each other and improve each other in an iterative nice fashion.

3.2.1 EFFICIENT CALCULATION OF CONDITIONAL LLR

We know, if there are two binary random variables XY, modulo 2 addition of Z is X+Y. Then 1-2Z is the product of 1-2X and 1-2Y. This is sort of after BPSK, the XOR becomes multiplication. To take expectations, these binary random variables are assumed to be independent or atleast uncorrelated for this proposed, so if the expectation is taken, expected value of binary random variable being equal to 1 is the same as probability that is equal to 1, so that another little quick result we derived.

Suppose $z = x + y$;

x, y, z: binary RVs

$(1-2*z) = (1-2*x)(1-2*y)$

$(1-2*p_z(1) = (1-2*p_x(1))(1-2*p_y(1))$

$\tanh(I_z/2) = \tanh(I_x/2)* \tanh(I_y/2)$

$\text{sgn}(I_z) = \text{sgn}(I_x)* \text{sgn}(I_y)$

$\text{abs}(\tanh(| I_z|/2)) = \text{abs}(\tanh(| I_x|/2))* \text{abs}(\tanh(| I_y|/2))$

suppose $z = x_1 + x_{2+} x_{3+ \ldots +} x_w$

$\text{sgn}(I_z) = \text{sgn}(I_1)* \text{sgn}(I_2)* \text{sgn}(I_3)\ldots. \text{sgn}(I_w)$

$| I_z| = \text{abs}(f [f(I_1) + f(I_2) + \ldots + f(I_w)])$

Where $f(x) = \log \tanh (|x|/2)$

3.2.2 MIN-SUM APPROXIMATION

To approximate this, there is a useful reason. It is good to have lesser computations and it is close enough to actual the exact completion. So, the min-sum approximation is the best thing to do here. We saw this before; the absolute value is

dominated by the minimum of these values. So, let us say if the computation of non-linear function has really no loss in practice, so usually we can do a little offset as well, we will implement it in the decoder.

$$| L_z | = \text{abs} (f [f(L_1) + f(L_2) + \ldots + f(L_w)])$$

$$f(x) = \log \tanh(|x| / 2)$$

$$| L_z | = \min\{\text{abs}(L_1), \text{abs}(L_2), \ldots., \text{abs}(L_w)\}$$

3.2.3 ILLUSTRATIONS USING TANNER GRAPHS

This is the tanner graph representation for the above considered parity check matrix as an example. The local structure computation is successfully captured using Tanner graphs.

3.2.3.1 FIRST ITERATION

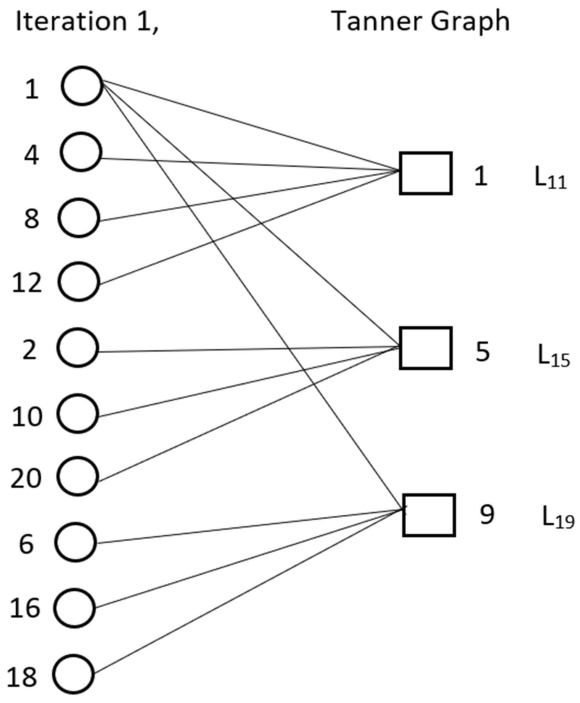

Fig 12 – First Iteration using Tanner Graph

The first estimate L_1 comes from the channel. L_i is passed from bit node I to all neighbouring check nodes. Estimates for bit 1, L_{11}, L_{15}, L_{19} are calculated at check

nodes one, five and nine as shown in Fig 12. Estimates passed from check nodes to neighbouring bit nodes. This is done for all nodes in parallel.

3.2.3.2 SECOND ITERATION

Bit to Check operation is shown in Fig 13

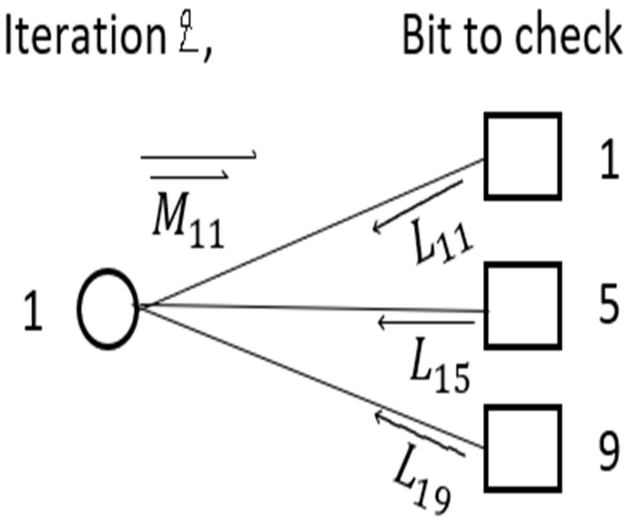

Fig 13 – Bit to Check operation in Iteration 2

Iterations are done to involve more bits and checks in the decoding. Similar rule for other bit nodes.

$m_{11} = L_1 + L_{15} + L_{19}$

$m_{15} = L_1 + L_{11} + L_{19}$

$m_{19} = L_1 + L_{11} + L_{15}$

Check – to –Bit operation is shown in Fig 14

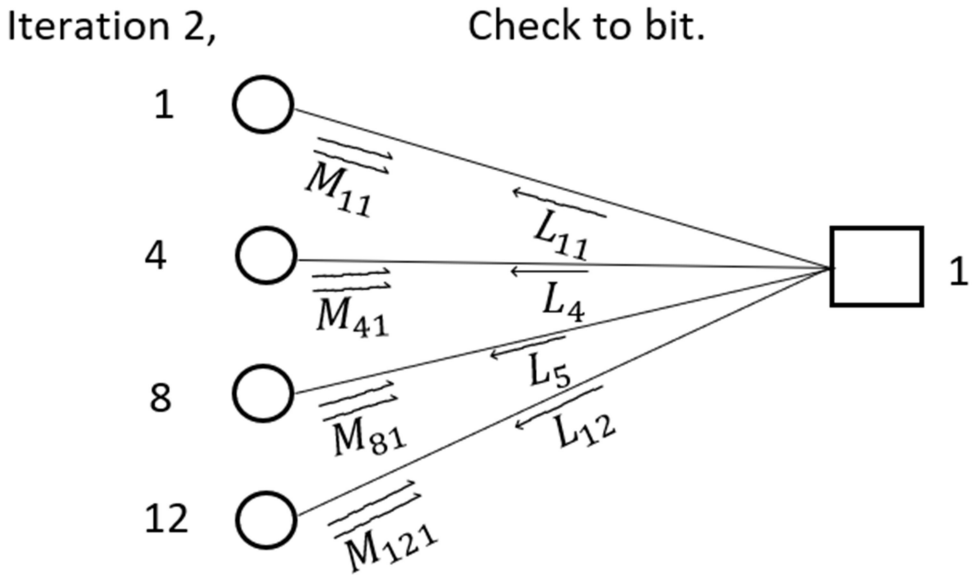

Fig 14 – Check to Bit operation in Iteration 2

Repeat computation of $L_{11,}$ $L_{15,}$ L_{19} using m_{41}, m_{81} and m_{121}. Same process for all the other check nodes.

In row iteration, Check node computation and computation of conditional LLRs for one bit happens. In column iteration, Bit node computation and consolidation of conditional LLRs happens. The modifications for high throughputs are specified below.

3.2.4 TECHNIQUES FOR HIGH THROUGHPUT

In the previous chapter, we have specified the working of the proposed LDPC decoder for 5G wireless communications standard. We have described how we used the local structure very efficiently to do operations and improve our beliefs and how we combined different messages coming from the local structure iteratively in a very smart way using very extensive information to more and more iterations. So these are the two ideas which give really powerful performance.

3.2.4.1 LAYERED DECODING

The most important modification is this layered decoding. Before this row and column operations were specified. In layered decoding, we can update the column more aggressively and need not wait for all the rows to finish before doing the

column operation. We took a set of rows or a layer of the PCM once when we finished processing the layer, we immediately updated the column. So it turns out that this has faster conversions, less iteration. We designed our code in a way which supported layering that is depicted in Fig 15. Let us see a small protograph example for the proposed idea.

Layered Decoding work by grouping of the rows of the PCM. Operations are done layer to layer. So this sort of idea : When the column weight of each layer is usually kept as 1 it is possible to increase that also and do some more complicated arithmetic. In the 5G LDPC matrix specifications we think of each layer as 1 block row in the standard in the 5G LDPC codes. For the sake of simplicity, consider a small toy example as shown in Fig 16.

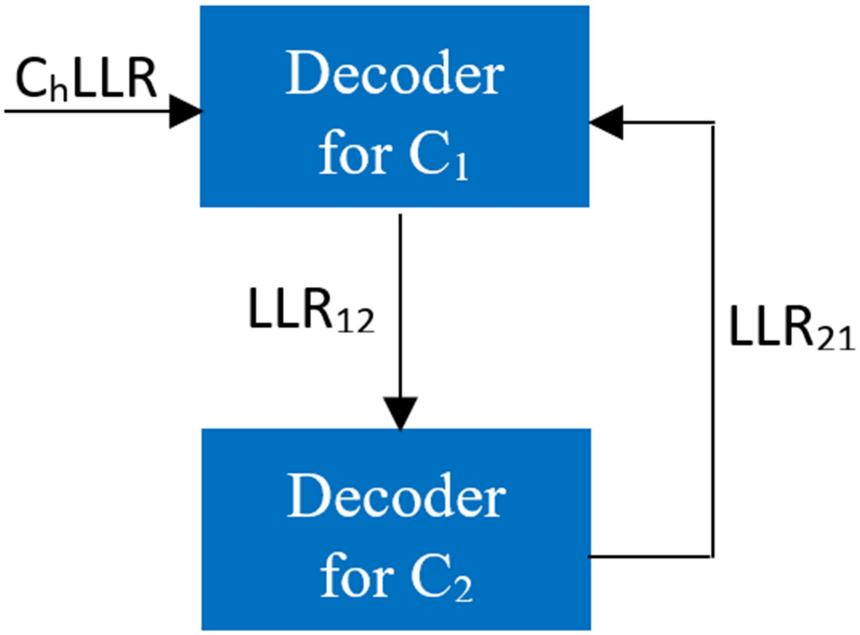

Fig 15 – Layered Decoding

$$H = \begin{bmatrix} 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \end{bmatrix}$$

$$r = [0.2 \quad -0.3 \quad 1.2 \quad -0.5 \quad 0.8 \quad 0.6 \quad -1.1]$$

$$L = \begin{bmatrix} 0.2 & -0.3 & 1.2 & & 0.8 & \\ & & & -0.5 & & 0.6 & -1.1 \\ - & - & & - & & - & \\ & & - & & - & - & \end{bmatrix}$$

Fig 16 – Toy example for layered decoding

Split the PCM into two layers, layer 1and layer 2 as shown in Fig-. The first layer has just a single 1 in each column, the column weight is 1. Similarly, the second layer also has the column weight 1. These two together specify a parity check matrix. Consider a length 7 code for this example. When we actually implement the 5G code, code is much larger, but the steps are the same.

When the column weight is assumed to be at most 1, there is no need to update the same bit twice in the same layer. There may be a column with weight 0. But in this particular example it isn't there. The incoming received vector r is r. Easy way to deal with layered decoding is, start with the incoming received vector and start going down the PCM row by row. As we process through each row, we keep updating this received vector. That is the simple of visualising the decoder. In each layer some processing occurs and the received vector gets updates.

Wherever there are 1's we store something and wherever there are 0's in the PCM we need not store anything, so it is represented by blue solid in the Fig17. We only use the first layer in the beginning as an initialisation step. So the initialisation just

goes down and every position is initialized with the received vector. In later iterations there may be already some value stored in L, in that case, we do something different.

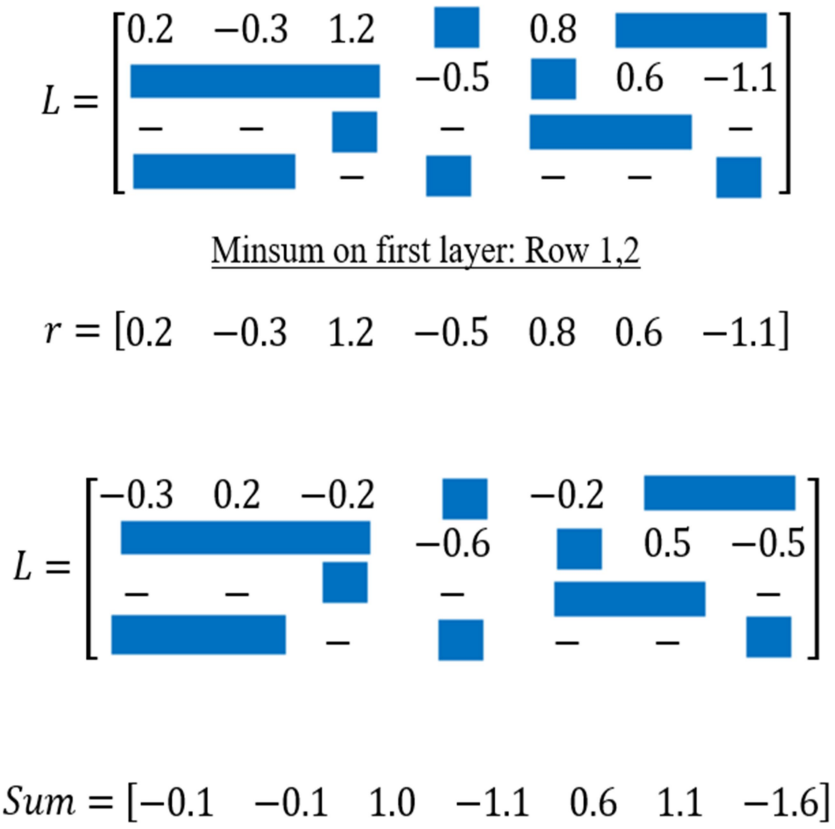

Fig 17 – Layer 1 operation

So after we initialise we move to iteration 1 as shown in Fig-. Now we do row processing. In the first row, mark min 1 and min 2. An absolute value and overall parity is not satisfied, so we replace min 1 with the absolute value of min 2 and then all other positions get replaced by -1, all signs get flipped. The same thing is done in the next row. Then we immediately update r. Then we move to layer 2 of iteration 1. The same process is repeated and the r is updated as shown in Fig 18.

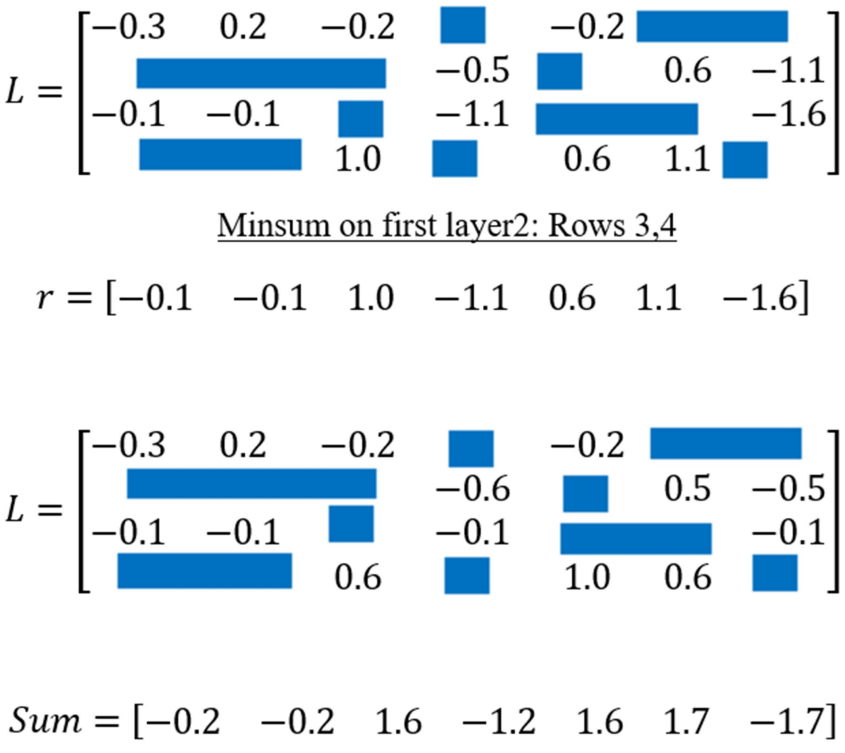

Fig 18 – Layer 2 operations

In the next iteration, the first thing we need to do is subtraction. Where the incoming belief is subtracted from layer 1. We have an incoming belief and before we start processing layer 1, we have to correct it. We are correcting it because the incoming belief actually had some input from layer 1 itself in the previous iteration. So, when it comes back again, we have to subtract that influence before doing the fresh processing. This is the most important step. If not done correctly, the decoder loses its stability. So the next step of this iteration is min-sum. We have incoming beliefs and layers along with r. Min-sum is exactly the same as before, we identify min 1 and min 2 and overall parity is -1. So once you finish processing layer 1, the next step is just updating. Add the row 1 elements with that of elements in sum vector. Repeat the above things for layer 2 as well. The r after two iterations is

r = [-0.4 -0.4 1.8 -1.9 1.8 1.9 -1.9]

So, this is the idea where row processing is done first and then the column operation in layer wise fashion. The structure of the storages is very small. To store and read these storages is the most challenging one. We have done it using HLS pragma features. Those features of Vivado HLS made our work easy. To generalise to 5G

standard, there is a base matrix which will be expanded to size z*z. This expansion is considered as a single layer of z rows. We did the exact above-mentioned procedure for 5G transmission as well.

3.2.4.2 INTRODUCING OFFSET

So, it turns out min-sum is pretty useful approximation, now need to improve approximation. Instead of just taking the minimum of two values, we take some constants a and b multiply with that and add b. typically scaling is not preferred. We just used minimum minus some offset. Subtracting with some offset makes it more accurate. Offset min sum approximation is the added modification to the above algorithm. Our implementation used a small offset min-sum with layering on our proposed decoder.

3.2.5 DECISION MAKING

Decision making is the final step of decoder. Where we take the updated received vector **r**after any iteration, and decide which bit is transmitted. The decision making in BPSK is quite well known. We assume a threshold value and make a decision for each bit of r.

If ri > 0, Decision on bit i = 0

If ri < 0, Decision on bit i = 1

It is based on assuming BPSK 0 -> +1 and 1 -> -1

After two iterations,

The decoded code word for the above considered toy example is

Decoded c’ = [1 1 0 1 0 0 1]

CHAPTER 4
IMPLEMENTATION

IMPLEMENTATION

4.1 CHANNEL CODING USING LDPC

In the previous chapter, LDPC encoding and decoding approach illustrates the necessity of high throughput and low latency. Our objective is to design architecture for the 5G channel that promises higher throughputs with lower latency. The proposed architecture reflects the advantages offered by HLS.

A partial parallel and full pipelined is used for the proposed decoding technique. This design exploits the advantages of FPGA. Partial parallel means that this design starts updating multiple rows and columns at the same time. Full pipeline refers to the ability of HLS to update a row and a column in a single clock cycle.

4.1.1 HLS PRAGMAS

The hardware kernel must be synthesised from C++, C or OpenCL into RTL which can be embedded into FPGA Device. Vivado HLS offers ‘pragmas’ which are used to optimise the design improve throughput and reduce latency. Pragmas also helps in reducing the area and resource utilization of the synthesized RTL code. Vivaldi HLS provides various optimization types such as Kernel Optimization, Function In-lining, Interface Synthesis, Loop Optimization etc. We used Interface, Unroll, and Pipeline pragmas for the proposed architecture.

4.1.1.1 PRAGMA HLS UNROLL

Pragma HLS Unroll converts loops by creating multiple copies of data, which allows them to execute multiple loop iterations in parallel. Unroll pragma creates multiple independent operations rather than a single independent operation. We can unroll loops to increase data access and throughput using this pragma.

4.1.1.2 PRAGMA HLS PIPELINE

Pragma HLS Pipeline allows the operations of a loop to be executed in a concurrent manner. Using this pragma, we reduced the initiation interval for a loop and function by allowing the concurrent execution of its operations.

4.1.1.3 PRAGMA HLS INTERFACE

All input and output operations in any RTL design must be performed through a port. The pragma HLS Interface describes how RTL ports are generated from function definition during Interface Synthesis. These ports operate using a specific I/O protocol. Our design uses ports that are derived from Function-Level protocol and Function Arguments.

4.1.1.4 TECHNIQUES FOR HIGH THROUGHPUT

4.1.1.4.1 PIPELINING

Pipelining is used to split a process into multiple stages and executes the stages in an overlapping fashion. It enables concurrency. Multiple inputs can be processed at the same time. Pipelining increases the throughput of the design. Pipelining is essential for any design which promises lower latency. Pipelining increases the efficiency of resource utilization. Pipeline structure can execute two operations in a single clock cycle.

4.1.1.4.2 ARRAY PARTITIONING

When there is a bottleneck in bandwidth Array Partitioning is used. Array partitioning means that a large array is divided into small subarrays in the lane of internal FPGA memory block which is called Block Random Access memory (BRAM). This increases the memory bandwidth. When different resources access a data at the same time, data access conflict occurs. Array Partitioning reduces this conflict.

CHAPTER 5
RESULTS AND ANALYSIS

RESULTS AND ANALYSIS

5.1 THIS WORK (PROPOSED TECHNIQUE)

Specific Verification Example:

Base Graph: BG1 (46*68)

Code word size, Information bits, Expansion factor

(N, K, Z) = (1536, 1024, 64)

Number of Iterations = 50

Targeted Device: Xilinx Kintex 7 - xc7k160t fbg484 1

C Simulation is done and Zero errors are returned as shown in Fig 19.

```
INFO: [SIM 2] *************** CSIM start ***************
INFO: [SIM 4] CSIM will launch GCC as the compiler.
make: 'csim.exe' is up to date.
Error cnt : 0
INFO: [SIM 1] CSim done with 0 errors.
INFO: [SIM 3] *************** CSIM finish ***************
```

Fig 19 – C Simulation Result

C Synthesis is done and report is as shown in Fig 20.

Synthesis Report for 'ldpcDec'

General Information

Date: Sun Feb 28 11:09:43 2021
Version: 2020.1 (Build 2897737 on Wed May 27 20:21:37 MDT 2020)
Project: trail4
Solution: solution1
Product family: kintex7
Target device: xc7k160t-fbg484-1

Performance Estimates

- **Timing**
  - **Summary**

Clock	Target	Estimated	Uncertainty
ap_clk	4.00 ns	3.444 ns	0.50 ns

- **Latency**
  - **Summary**

Latency (cycles)		Latency (absolute)		Interval (cycles)		Type
min	max	min	max	min	max	
10020	10020	40.080 us	40.080 us	10020	10020	none

Fig 20 – C Synthesis Report

Utilization Estimates for the above example is as shown in Fig 21.

Utilization Estimates

- **Summary**

Name	BRAM_18K	DSP48E	FF	LUT	URAM
DSP	-	-	-	-	-
Expression	-	-	0	146	-
FIFO	-	-	-	-	-
Instance	0	-	34846	44280	-
Memory	104	-	48	192	0
Multiplexer	-	-	-	6327	-
Register	0	-	189	64	-
Total	104	0	35083	51009	0
Available	650	600	202800	101400	0
Utilization (%)	16	0	17	50	0

Fig 21 – Utilization Estimates

RTL/Co-Simulation is passed as shown in the Fig 22.

Cosimulation Report for 'ldpcDec'

Result

		Latency			Interval		
RTL	Status	min	avg	max	min	avg	max
VHDL	NA	NA	NA	NA	NA	NA	NA
Verilog	Pass	10020	10020	10020	10021	10021	10021

Export the report(.html) using the Export Wizard

Fig 22 – Co-simulation Report

Wave form is dumped and seen through Open Wave Viewer in Vivado which is shown in Fig 23.

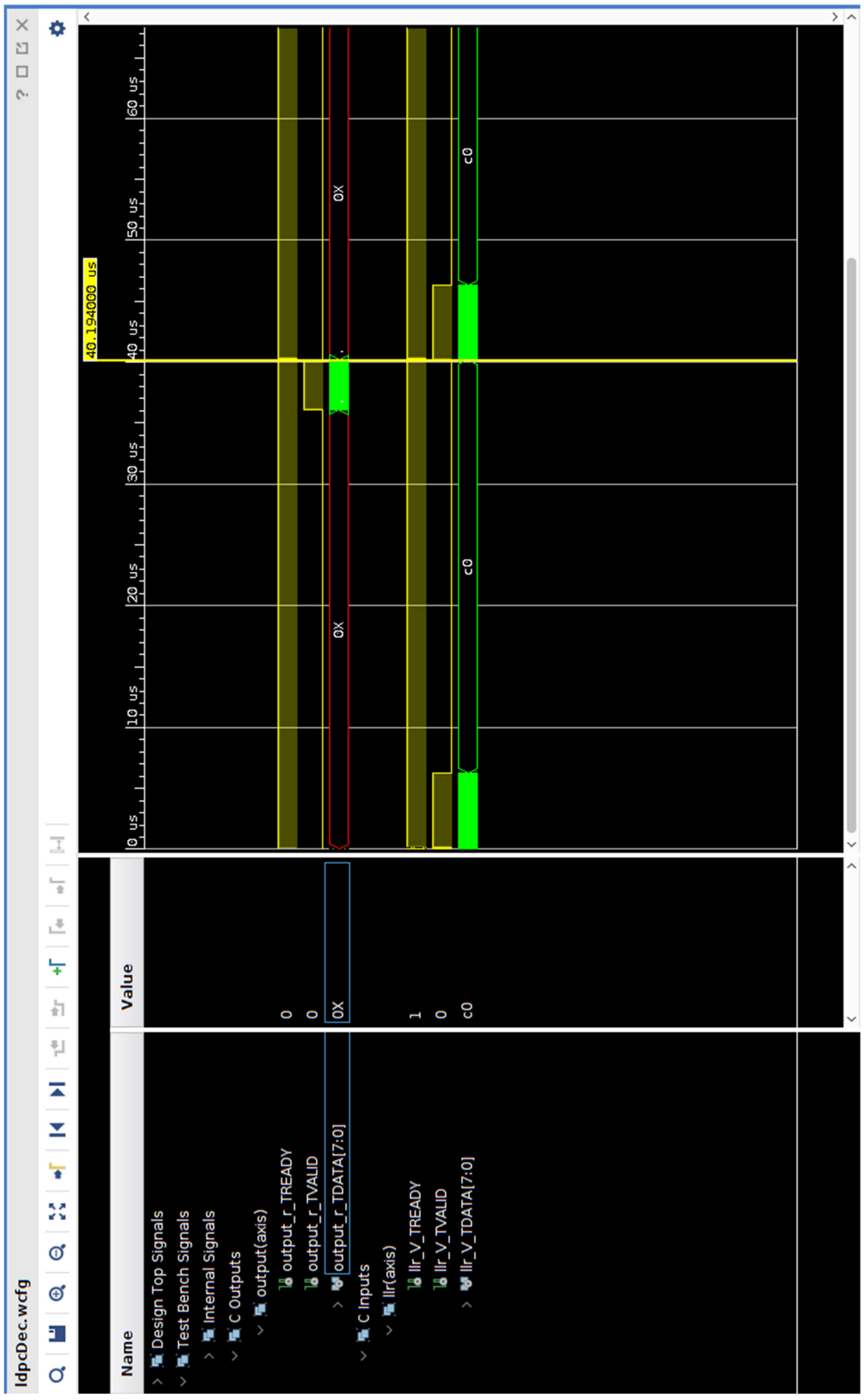

Fig 23 – Waveform

RTL Schematic for the above example is shown in the Fig 24.

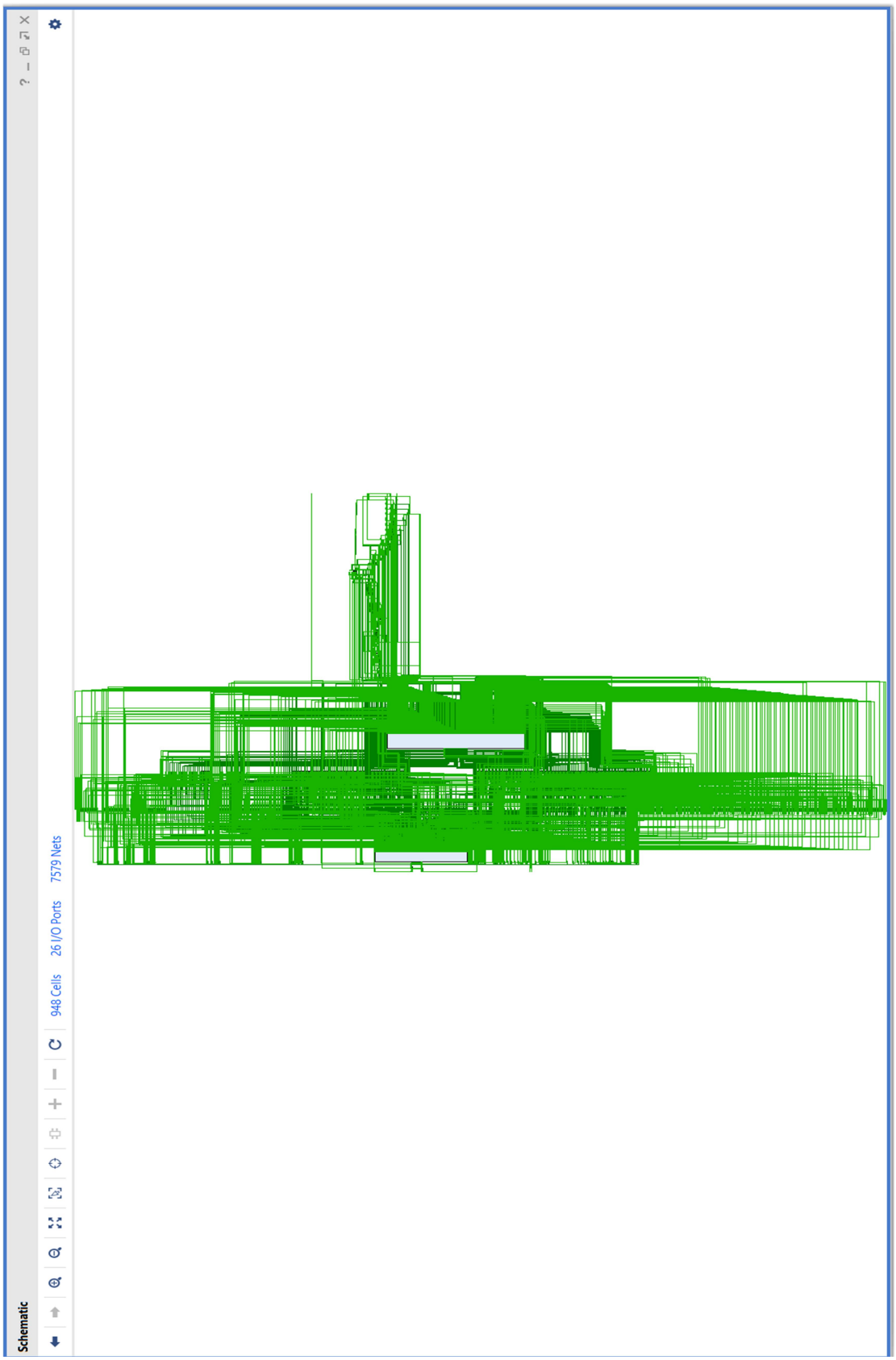

Fig 24 – Implemented module Design

A much closer look of the above schematic is shown in Fig 25.

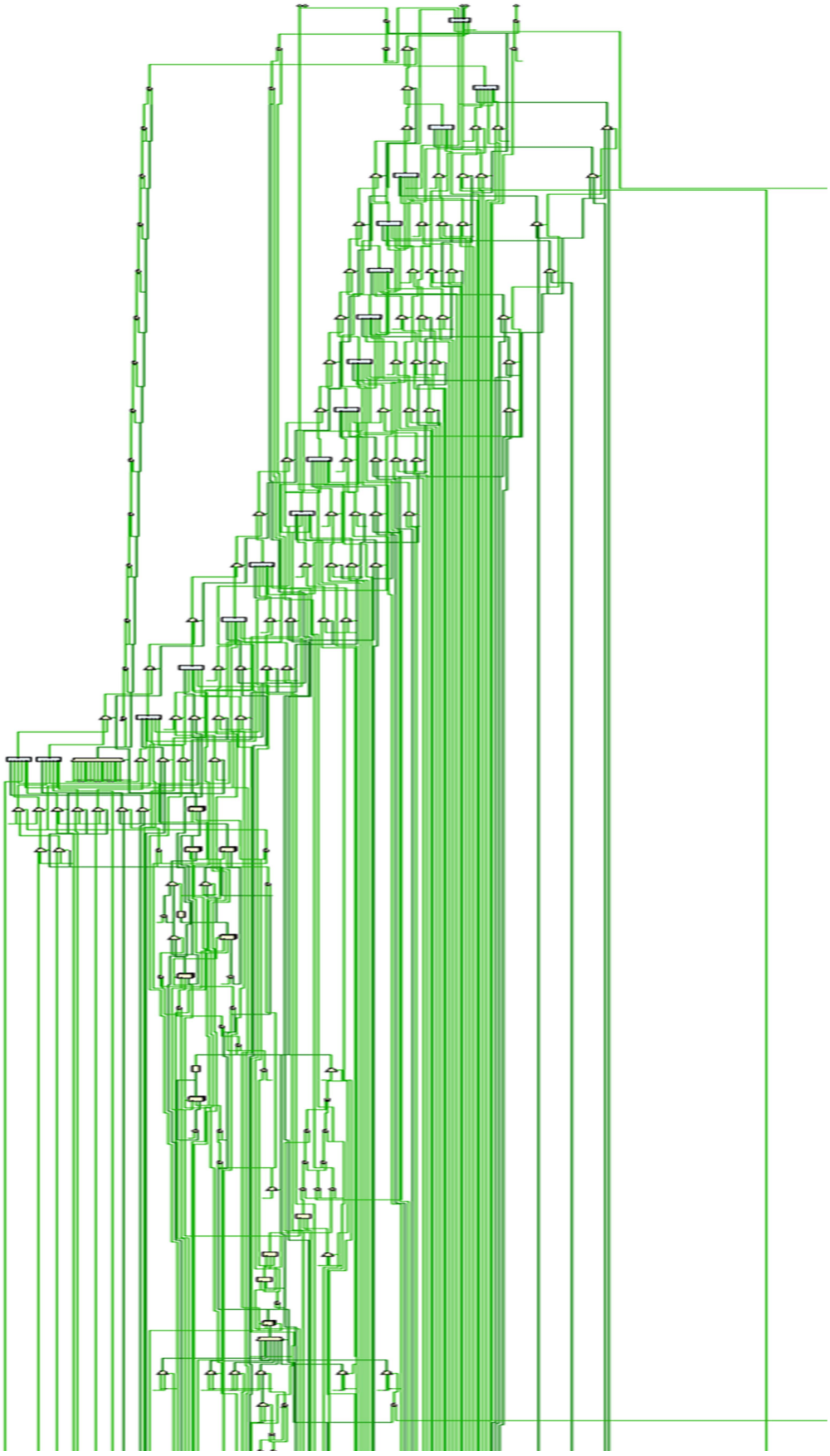

Fig 25 – Closed look of Implemented module Design

RTL Implementation Summary obtained in Vivado is shown in Fig 26.

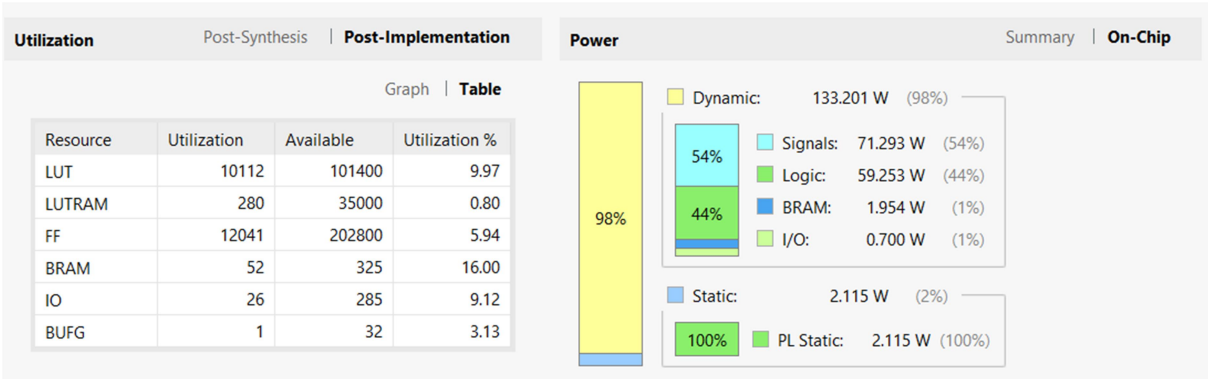

Fig 26 – RTL Implemented Summary

The Design Timing summary of the above example is shown in Fig 27.

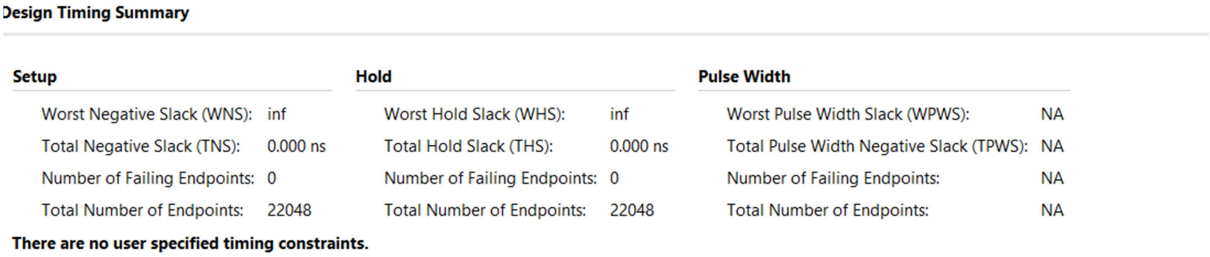

Fig 27 –Design Timing Summary

The specified example takes 10200 clock cycles for the execution as shown in the synthesis report Fig. 20. Utilization estimates is depicted in Fig. 21. Then the code is executed through RTL simulation and its results are specified in Fig. 22. Fig. 23 is the waveform of the above example and we can validate that it returns o errors (output T_Data Signal in the waveform) as shown in Fig. 19. The RTL simulation of the above simulation is specified in Fig.24 and Fig. 25. It shows that we need 948 cells and 26 I/O ports for the proposed design. The implementation summary is depicted in Fig. 26 and Fig. 27 The throughput and Latency of overall execution is specified in Table 5.

The device schematic diagram of the above example is shown in Fig 28.

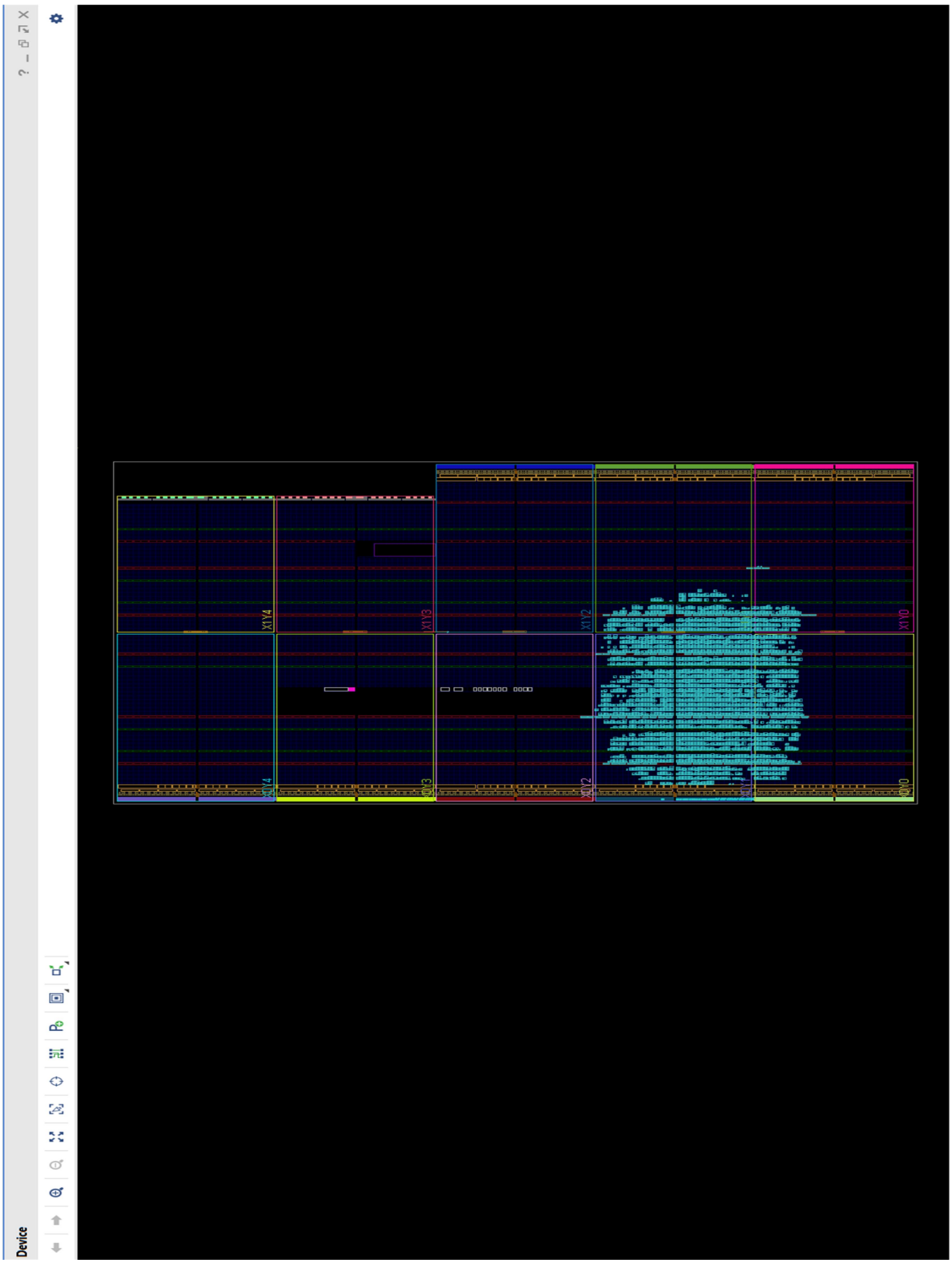

Fig 28 – Implemented Device Design

Further Simulations,

We have considered 19 different code word sizes for 4 different code rates. Results are tabulated below.

Targetted device - Xilinx Kintex 7 - xc7k160t fbg484 1

Bit errors and Bit error % results for 19 different code word sizes for 4 different code rates is shown in Table 3.

S NO	CODE WORD LENGTH (N)	Z FACTOR	INFORMATION LENGTH (K)	RATE	BIT ERRORS	BIT ERROR %
1	576	24	288	½	0	0
			384	2/3	0	0
			432	¾	401	0.029
			480	5/6	33	0.002
2	672	28	336	½	224	0.017
			448	2/3	0	0
			504	¾	13	0.005
			560	5/6	29	0.0015
3	768	32	384	½	256	0.0104
			512	2/3	0	0
			576	¾	15	0.006
			640	5/6	38	0.0015
4	864	32	432	½	0	0
			576	2/3	0	0
			648	¾	592	0.02
			720	5/6	34	0.0012
5	960	36	480	½	0	0
			640	2/3	0	0
			720	¾	12	01.73
			800	5/6	80	0.002
6	1056	44	528	½	0	0
			704	2/3	0	0
			792	¾	017	0.003
			880	5/6	21	0.004
7	1152	48	576	½	0	0
			768	2/3	0	0
			864	¾	794	1.4
			960	5/6	55	0.009
8	1248	52	624	½	0	0
			832	2/3	0	0
			936	¾	17	0.02
			1040	5/6	874	1.3
9	1344	56	672	½	0	0

			896	2/3	0	0
			1008	¾	34	0.04
			1120	5/6	46	0.06
10	1440	60	720	½	480	0.55
			960	2/3	0	0
			1080	¾	28	0.03
			1200	5/6	1032	1.19
11	1536	64	768	½	0	0
			1024	2/3	0	0
			1152	¾	1064	1.08
			1280	5/6	80	0.08
12	1632	68	816	½	0	0
			1088	2/3	0	0
			1224	¾	40	0.03
			1360	5/6	71	0.06
13	1728	72	864	½	0	0
			1152	2/3	11	0.0008
			1296	¾	12	0.0008
			1440	5/6	68	0.05
14	1824	76	912	½	60	0.04
			1216	2/3	12	0.0008
			1368	¾	0	0
			1520	5/6	18	0.012
15	1920	80	960	½	0	0
			1280	2/3	0	0
			1440	¾	41	0.026
			1600	5/6	13	0.0008
16	2016	84	1008	½	0	0
			1344	2/3	13	0.0076
			1512	¾	3	0.00002
			1680	5/6	14	0.0076
17	2112	88	1056	½	0	0
			1408	2/3	0	0
			1584	¾	47	0.025
			1760	5/6	94	0.05
18	2208	92	1104	½	73	0.03
			1472	2/3	0	0
			1656	¾	15	0.0007
			1840	5/6	98	0.048
19	2304	96	1152	½	0	0
			1536	2/3	153	0.069
			1728	¾	18	0.0008
			1920	5/6	94	0.04

Table 3 – Bit Errors and Bit Error %

Resource Utilization for the above considered inputs are shown in Table 4.

S No	N	K	Z	Rate	BRAMs	LUTs	FFs
1	576	288	24	½	104	52670	366
2	672	504	28	¾	104	52676	36517
3	768	640	32	5/6	104	50821	34661
4	864	432	36	½	104	52377	35891
5	960	640	40	2/3	104	52035	35078
6	1056	880	44	5/6	104	52086	35092
7	1152	864	48	¾	104	51978	34857
8	1248	624	52	½	104	51973	34866
9	1344	896	56	2/3	104	51976	34874
10	1440	1080	60	3/4	104	51980	34886
11	1536	1024	64	2/3	104	51009	35083
12	1632	1360	68	5/6	104	52052	35096
13	1728	1296	72	¾	104	52064	35159
14	1824	1216	76	2/3	104	52079	35177
15	1920	960	80	½	105	52084	35196
16	2016	1680	84	5/6	105	52096	35198
17	2112	1408	88	2/3	105	52169	35199
18	2208	1840	92	5/6	105	52256	35027
19	2304	1152	96	½	105	52394	35215

Table 4– Resource Utilization

The above table shows the resource utilization of the above considered examples. It says that the proposed design uses a maximum of 52676 LUTs of the available LUTS in the targeted device, which is exactly 17 % of available. The BRAMs usage is 16% and the Flip Flops usage is 51% of the available FFs in the targeted device.

Latency and Throughput of the above inputs is shown in Table 5.

S NO	N	Z	K	RATE	LATENCY (µS)	THROUGHPUT (GBPS)
1	576	24	288	½	19.88	2.113
			384	2/3	20.42	1.584
			432	¾	21.86	1.408
			480	5/6	23.08	1.267
2	672	28	336	½	22.12	1.389
			448	2/3	23.02	1.046
			504	¾	24.17	0.930
			560	5/6	26.12	0.837
3	768	32	384	½	22.156	1.066
			512	2/3	23.806	0.799
			576	¾	24.098	0.711
			640	5/6	26.156	0.639
4	864	32	432	½	26.596	0.71
			576	2/3	27.561	0.526
			648	¾	28.276	0.467
			720	5/6	30.946	0.421
5	960	36	480	½	28.836	0.568
			640	2/3	29.667	0.426
			720	¾	31.127	0.379
			800	5/6	32.236	0.341
6	1056	44	528	½	31.076	0.402
			704	2/3	32.092	0.301
			792	¾	33.054	0.268
			880	5/6	34.326	0.241
7	1152	48	576	½	31.116	0.337
			768	2/3	32.764	0.253
			864	¾	33.562	0.225
			960	5/6	34.165	0.202
8	1248	52	624	½	33.356	0.268
			832	2/3	34.25	0.201
			936	¾	36.356	0.178
			1040	5/6	37.629	0.160
9	1344	56	672	½	35.596	0.229
			896	2/3	37.524	0.172
			1008	¾	38.436	0.153
			1120	5/6	39.634	0.137
10	1440	60	720	½	37.836	0.177
			960	2/3	38.54	0.133
			1080	¾	39.413	0.118
			1200	5/6	37.8	0.106
11	1536	64	768	½	40.080	0.155
			1024	2/3	41.568	0.116
			1152	¾	42.028	0.103
			1280	5/6	44.087	0.093

12	1632	68	816	½	42.320	0.137
			1088	2/3	43.814	0.103
			1224	¾	45.006	0.091
			1360	5/6	46.205	0.082
13	1728	72	864	½	44.560	0.122
			1152	2/3	45.953	0.091
			1296	¾	46.569	0.816
			1440	5/6	47.214	0.735
14	1824	76	912	½	46.800	0.110
			1216	2/3	47.521	0.082
			1368	¾	48.856	0.073
			1520	5/6	49.483	0.061
15	1920	80	960	½	49.237	0.094
			1280	2/3	50.04	0.074
			1440	¾	52.045	0.063
			1600	5/6	53.516	0.059
16	2016	84	1008	½	51.007	0.092
			1344	2/3	52.84	0.069
			1512	¾	53.864	0.061
			1680	5/6	54.627	0.057
17	2112	88	1056	½	52.520	0.084
			1408	2/3	53.385	0.063
			1584	¾	54.562	0.056
			1760	5/6	55.320	0.050
18	2208	92	1104	½	55.76	0.082
			1472	2/3	56.605	0.061
			1656	¾	57.224	0.054
			1840	5/6	58.76	0.049
19	2304	96	1152	½	58.00	0.046
			1536	2/3	60.046	0.058
			1728	¾	63.853	0.051
			1920	5/6	66.247	0.046

Table 5 – Latency and Throughput

5.2 COMPARISON WITH XILINX LDPC

To compare with Xilinx LDPC results with our results, we requested Xilinx Lounge to provide their results.

The comparison is for 10 different code words sizes with varying information bits length and expansion factor for both the base graphs.

No of Iterations = 8

Targeted device: Kintex 7 – xc7k160t fbg484 1

The analysis of sub – chapter 5.2 and 5.3 is specified in inference sub – section 5.5.

For Base Graph 1, the comparison results are shown in Table 6.

S No	BG	N	K	Z	LATENCY (µS)		THROUGHPUT (Gbps)	
					Xilinx	This Work	Xilinx	This Work
1	BG1	10368	8448	384	15.16	46.728	1.775	0.831
2	BG1	6912	5632	256	10.53	36.056	1.742	1.037
3	BG1	3456	2816	128	5.85	28.304	1.704	1.126
4	BG1	1728	1408	64	3.6	15.048	1.513	1.311
5	BG1	864	704	32	2.71	7.876	1.095	1.467
6	BG1	26112	8448	384	39.41	86.358	0.417	0.609
7	BG1	17408	5632	256	36.05	58.086	0.432	0.730
8	BG1	8704	2816	128	26.2	39.384	0.424	0.979
9	BG1	4352	1408	64	21.42	25.063	0.265	1.092
10	BG1	2176	704	32	19.15	17.876	0.149	1.123

Table 6 – BG1 results comparison with Xilinx

For Base Graph 1, the comparison results are shown in Table 7.

SNo	BG	N	K	Z	LATENCY (µS)		THROUGHPUT (Gbps)	
					Xilinx	This Work	Xilinx	This Work
1	BG2	6258	3840	384	10.54	54.056	1.225	0.721
2	BG2	4352	2560	256	7.48	29.384	1.218	0.865
3	BG2	2176	1280	128	4.29	15.048	1.208	0.983
4	BG2	1088	640	64	3.18	9.035	0.848	1.139
5	BG2	544	320	32	3.03	7.806	0.445	1.176
6	BG2	19968	3840	384	33.8	89.713	0.315	0.237
7	BG2	13312	2560	256	29.31	72.472	0.313	0.463
8	BG2	6656	1280	128	19.33	58.800	0.263	0.813
9	BG2	3328	640	64	17.79	18.632	0.145	0.947
10	BG2	1664	320	32	16.71	11.124	0.077	1.073

Table 7 – BG2 results comparison with Xilinx

5.2.1 COMPARISON WITH KINTEX-7 FAMILY

ATTRIBUTE	XILINX	THIS WORK
Target Device	xc7k160t ffv676 2	xc7k160t ffv676 2
(N,K,Z)	-	(2304,1152,96)
Algorithm	Normalized Min-Sum	Min-Sum BP
Clock Input	core_clk	ap_clk
Fmax (MHz)	234	289.52
LUTs	53997	52068
FFs	56871	35215
DSPs	0	0
36K BRAMs	109	-
18K BRAMs	20	104

Table 8 – Comparison with Xilinx Kintex 7 family

5.2.2 COMPARISON WITH KINTEX ULTRASCALE FAMILY

ATTRIBUTE	XILINX	THIS WORK
Target Device	xcku115 flva1517 -2	xcku115 flva1517 -2
(N,K,Z)	-	(1536,1024,64)
Algorithm	Normalized Min-Sum	Min-Sum BP
Clock Input	core_clk	ap-clk
Fmax (MHz)	304	287.108
LUTs	52707	42395
FFs	57049	19225
DSPs	0	0
36K BRAMs	109	-
18K BRAMs	20	104

Table 9 – Comparison with Xilinx Kintex Ultrascale family

5.2.3 COMPARISON WITH KINTEX ULTRASCALE+ FAMILY

ATTRIBUTE	DEVICES			
	XILINX	THIS WORK	XILINX	THIS WORK
Target Device	xcku13p ffve900 -1LV	xcku5p sfvb784 -1LV	xcku13p ffve900 -1	xcku5p sfvb784 -1
(N,K,Z)	-	(1728,1440,72)	-	(1728,1440,72)
Algorithm	Normalised min-sum	Min-sum BP	Normalised min-sum	Min-sum BP
Clock Input	core_clk	ap-clk	core_clk	ap-clk
Fmax (MHz)	310	289.017	373	287.108
LUTs	52900	50979	53158	50986
FFs	57475	23272	58151	23744
DSPs	0	0	0	0
36K BRAMs	109	-	109	-
18K BRAMs	20	104	20	104

Table 10 – Comparison with Xilinx Kintex Ultra scale+ family - i

ATTRIBUTE	DEVICES			
	XILINX	**THIS WORK**	**XILINX**	**THIS WORK**
Target Device	xcku13p ffve900 -2LV	xcku5p sfvb784 -2LV	xcku13p ffve900 -2	xcku5p sfvb784 -2
(N,K,Z)	-	(1728,1440,72)	-	(1728,1440,72)
Algorithm	Normalised min-sum	Min-sum BP	Normalised min-sum	Min-sum BP
Clock Input	core_clk	ap-clk	core_clk	ap-clk
Fmax (MHz)	338	289.017	439	299.958
LUTs	52547	50980	49151	50580
FFs	58718	23734	56154	22376
DSPs	0	0	0	0
36K BRAMs	109	-	109	-
18K BRAMs	20	104	20	104

Table 11 – Comparison with Xilinx Kintex Ultra scale+ family - ii

5.2.4 COMPARISON SUMMARY WITH XILINX

ATTRIBUTE	XILINX		THIS WORK	
	BG1	BG2	BG1	BG2
Frequency (MHz)	400	400	290	290
Latency (µS)	15.16	10.54	7.876	7.806
Throughput (Gbps)	1.775	1.225	1.467	1.178

Table 12 – Comparison Summary with Xilinx

5.3 COMPARISON WITH RESULTS FROM [1]

ATTRIBUTE	RESULTS FROM [1]	THIS WORK
HLS Technology	LabVIEW	Xilinx Vivado HLS
Standard	IEEE Wi-Fi	-
(N,K,Z)	(1944,972,81)	(1944,972,81)
Decoding	Serial and Layered	Parallel & Pipelined
Throughput	608	937.73
Iterations	4	4
Development Time	~days	~6 months
FPGA Device	Kintex-7 K410T	Kintex-7 K410T

Fixed-Point Precision	10	8
Clock Rate (MHz)	200	290
LUT (%)	8.2	51
FF (%)	5.3	17
BRAM (%)	6.4	16
DSP (%)	5.2	0

Table 13 – Comparison with Results from [1]

5.3.1 COMPARISON SUMMARY RESULTS FROM [1]

ATTRIBUTE	RESULTS FROM [1]	THIS WORK
Frequency (MHz)	200	290
Latency (µS)	5.7	19.8
Throughput (Gbps)	0.608	2.113

Table 14 – Comparison Summary with Results from [1]

5.4 BER, FER PLOTS

For a random matrix, we have plotted the variations between two parameters, keeping the other parameters constant for both BG1 and BG2. The analysis of the plots is specified in the next sub-chapter 5.5.

i. BG1, Z = 384, Iterations = 20, Frames = 500, Rate = 2/3

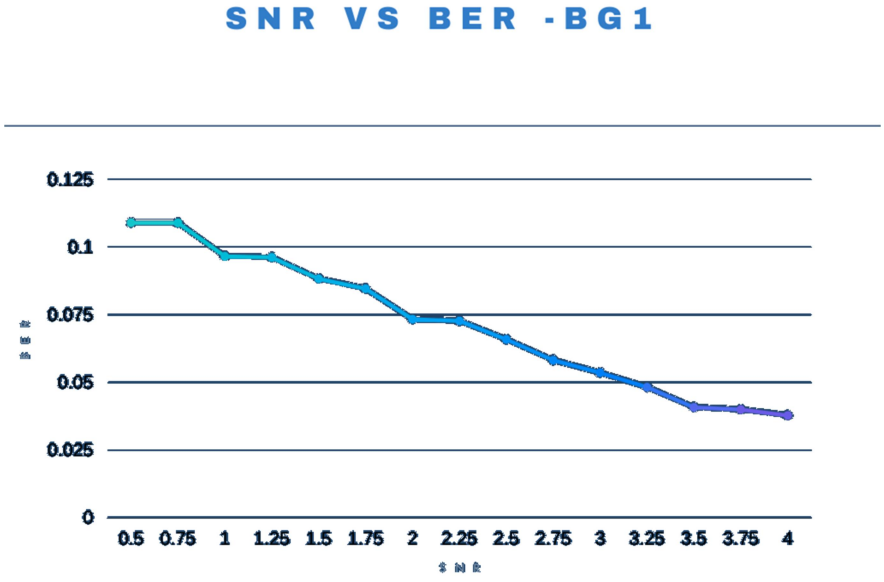

Fig 29 – SNR vs BER – BG1

ii. BG1, Z = 24 , Iterations = 20, Rate = ½

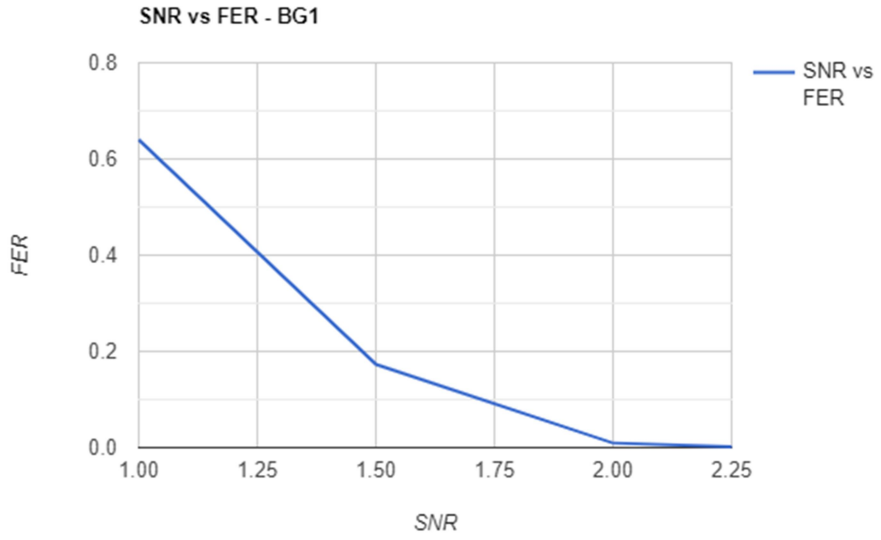

Fig 30 – SNR vs FER – BG1

iii. BG1, Z = 256, Frames = 200, SNR = 1.5, Iterations = 10.

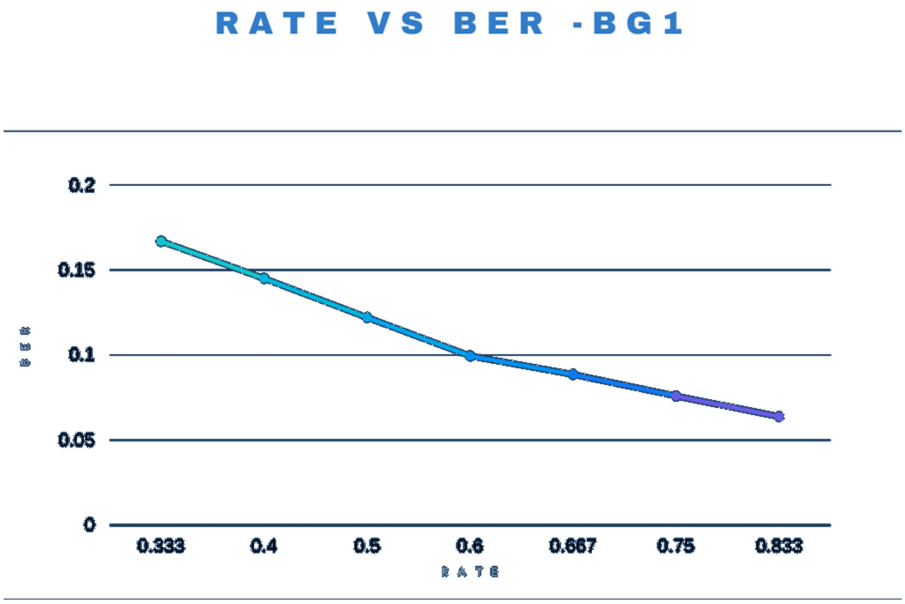

Fig 31 – Rate vs BER – BG

iv. BG1, Z = 64, SNR =1, Rate = 2/3, Iterations = 20

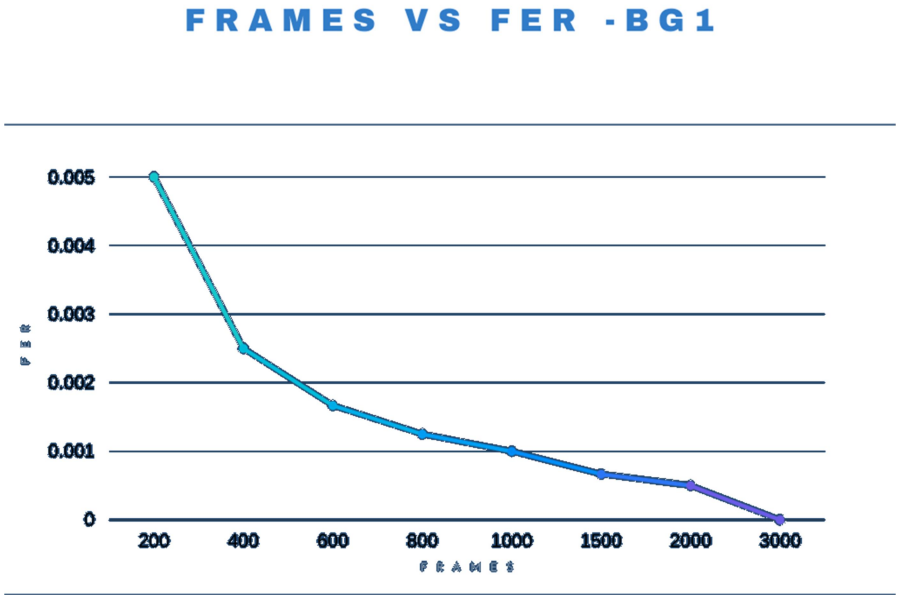

Fig 32 – Frames vs FER – BG1

v. BG2, Z = 192, Iterations = 20, Rate = ½, Frames = 300

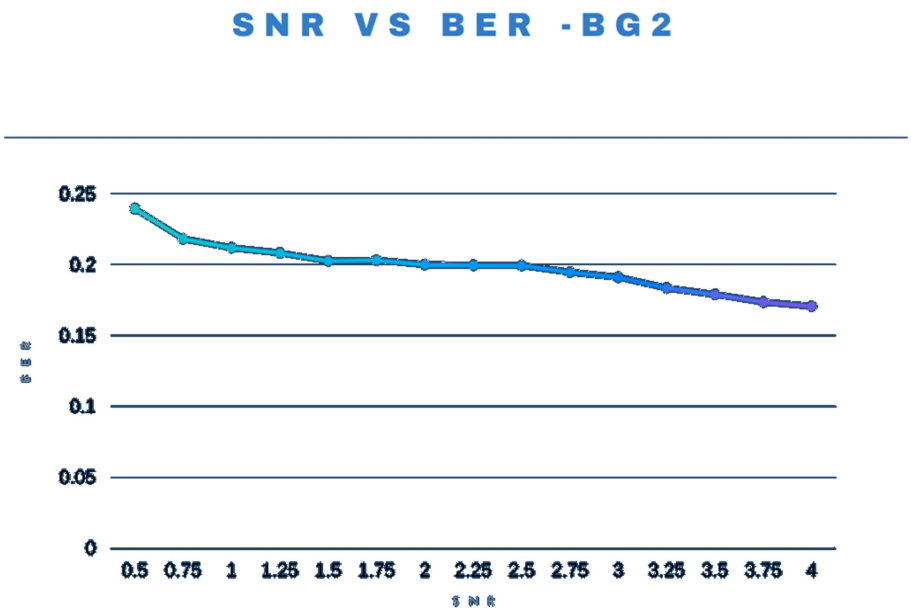

Fig 33– SNR vs BER – BG2

vi. BG2, Z = 52, Iterations = 20, Rate = ½

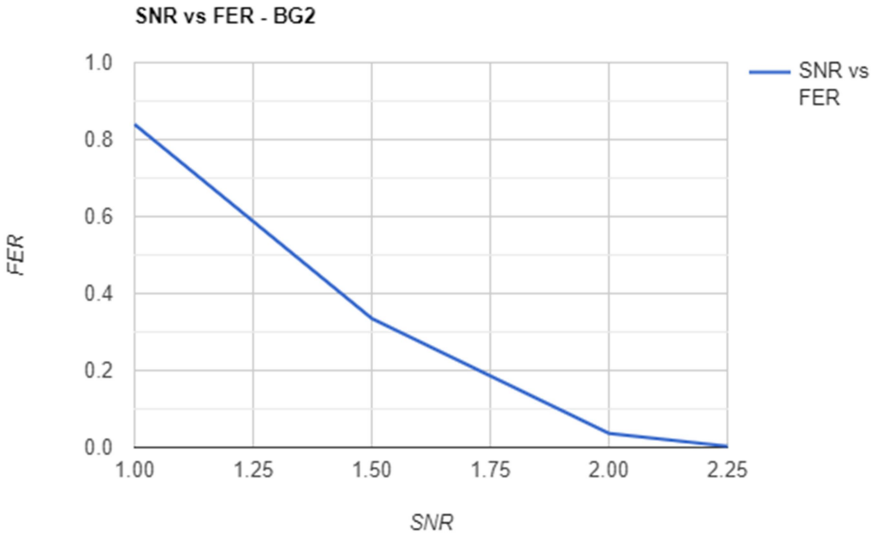

Fig 34– SNR vs FER – BG2

vii. BG2, Frames = 300, SNR = 1.5, Iterations = 10, Z= 160

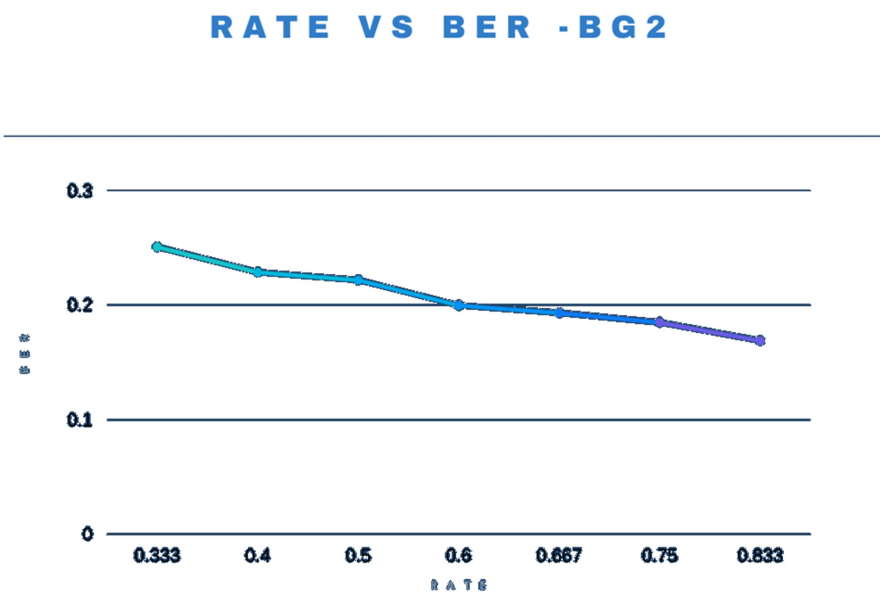

Fig 35– Rate vs BER – BG2

viii. BG2, Iterations = 15, Frames =100, SNR =1, Rate = 5/6

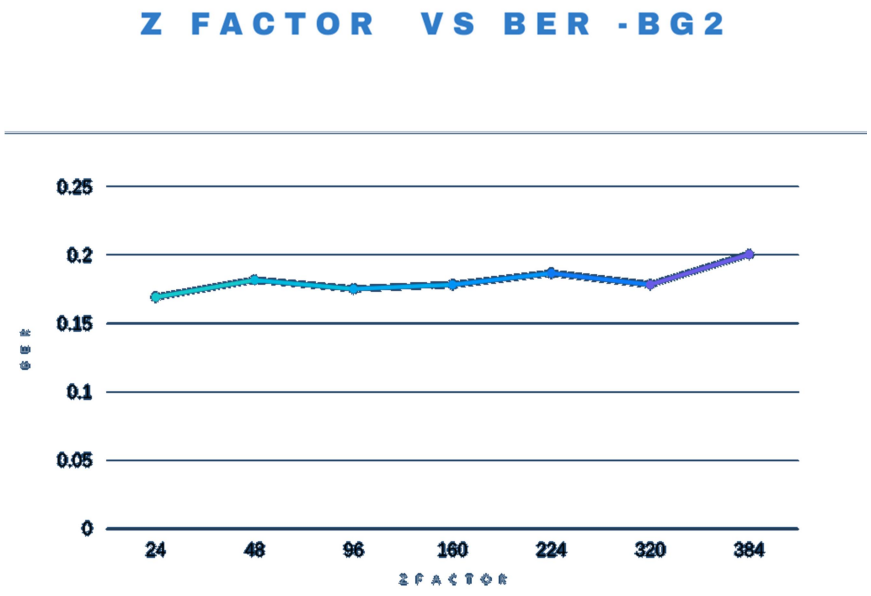

Fig 36 – Z vs BER – BG2

ix. BG2, Z = 224, Rate = ¾, SNR = 1.5, Frames = 400

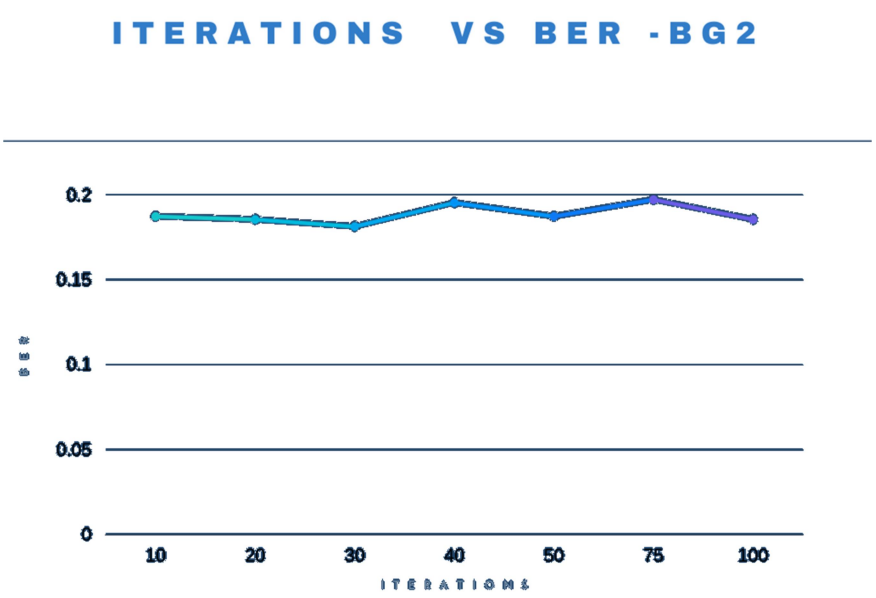

Fig 37– Iterations vs BER – BG2

x. BG2, Z = 80 Iterations = 10, SNR = 2, Rate = ½

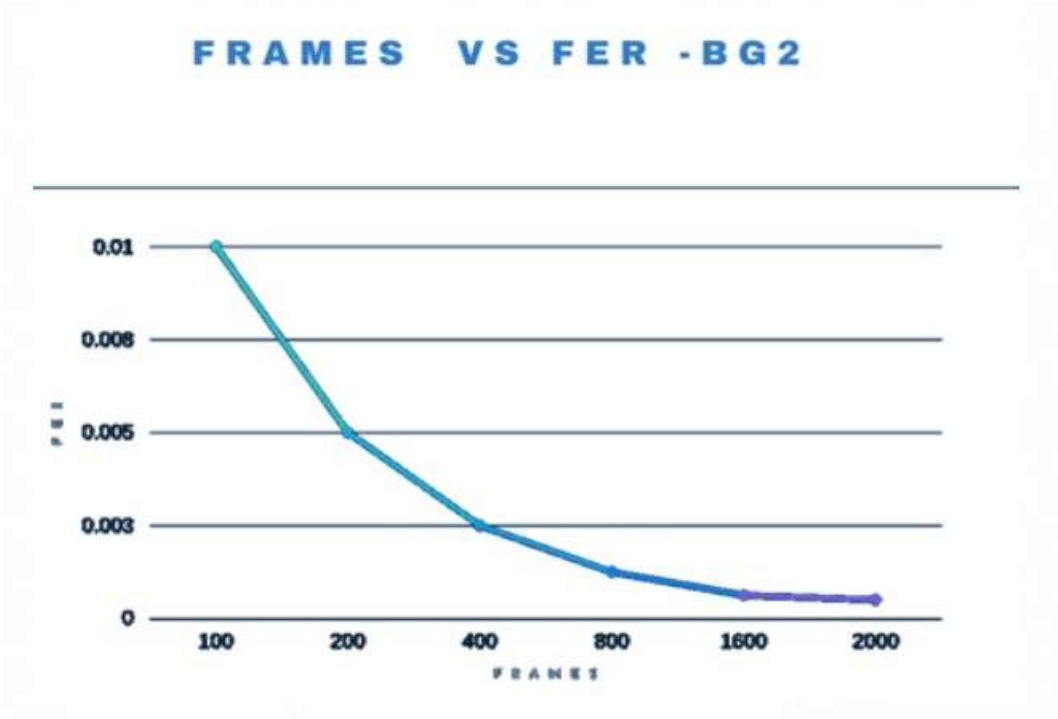

Fig 38– Frames vs FER – BG2

5.5 INFERENCE

In the previous sub-chapters, we have specified a specific verification example along with the simulation reports, wave form and RTL implementation summary. It shows that the proposed design is valid and can be implemented with various kinds of input. We also specified its working capability with different code word length, rates and expansion factors targeting different devices. Bit errors and their percentage, Utilization estimates, Throughput and latency are specified in Table 3, 4, 5 respectively. After closely observing the above obtained results, the peak throughput of the proposed design is 2.113 GB/s in 19.88 µs observed with a code word of size 576 and expansion factor of 24. 5G promises throughput range of 2 to 200 GB/s. We achieved peak throughputs which are suitable for 5G transmissions.

We wanted to compare with the existing algorithms to our work and evaluate how well and better is our work. Firstly we compared our results with Xilinx LDPC. Those comparisons are specified in the Tables 6 – 12. We have targeted different boards for comparison with Xilinx. Tables 6 and 7 shows the comparison with Xilinx LDPC for varying code rates, code word sizes and expansion factors for both BG1 and BG2. Table 8, 9 shows the comparison with Kintex 7 and Kintex 7 Ultra scale devices. These results are taken with same input with that of Xilinx LDPC. Comparison results with base paper [1], are specified in Table 13.

The comparison summaries of the results with Xilinx and base paper are specified in Table 12 and Table 14 respectively. It shows that our design achieves better results than our base paper [1]. The peak throughput of our base paper [1] is only 600 MB/s, whereas we achieved 2.113 GB/s. We can infer that our design far better than the results from base paper [1]. The comparison summary with Xilinx LDPC shows that this design is slightly better than their design. Their peak throughput is around 1.8 GB/s while we achieved 2.113 GB/s. From these comparisons we can infer that our design is more optimised and efficient.

We have plotted some BER and FER plots assuming various scenarios to ensure that this design is valid in all the conditions. These plots are depicted from Fig 29 – Fig 38. We have plotted the variations of Frames, rate, iterations and Z factor against BER and FER for both BG1 and BG2 assuming different conditions.

According to the fundamental laws of Wireless communication, the BER and FER should decrease with the increase in SNR of the channel. The variation with the expansion factor and Iterations purely depends on the particular instance of bit errors. The BER and FER should decrease with the increase in rate of the transmission as more number of information and parity bits are involved. The BER and FER should decrease with the increase in the number of frames. Higher the number of frames in the PCM lesser is the chance of ambiguity in the updating of row and column operations [5]. Hence the BER and FER should decrease. These variations are observed from our results as well.

These variations are valid as they follow the standard variations of wireless communications like SNR, frames, iterations, Z factor variations. From these plots we can infer that our design works fine in all conditions.

CONCLUSION AND FUTURE SCOPE

In this paper, we propose a novel high throughput LDPC encoder and decoder design for the 5G standard. Our novel encoder uses double diagonal structure of base graph to compute parity bits in lower complexity. Our novel decoder uses layered decoding and offset to achieve high throughputs. Based on the proposed algorithm design, obtained results are also specified. The obtained architecture exhibited lower complexity. With optimisation techniques such as pipelining, unrolling and array partitioning achieves high throughput in lower latency. The proposed architecture demonstrates a superior performance to the existing decoding algorithms. The proposed design achieves a peak throughput of 2.113 Gb/s in 19.8 μs. Moreover, the synthesis results of the proposed design satisfy the features offered by 5G wireless communications.

The base station can employ an adaptive modulation scheme for selecting the efficient modulation scheme according to the weather condition [4]. Base Station choosing modulation scheme maximizes the spectral efficiency. The idea is to employ a machine learning algorithm called Reinforcement learning. RL aims to find the best behaviour in a particular weather condition in order to maximize the accumulated reward. The future scope of this work is to deploy a RL agent to choose the most efficient modulation scheme.

REFERENCES

[1] Mhaske, Swapnil & Kee, Hojin & Ly, Tai & Aziz, Ahsan & Spasojevic, Predrag.(2017). FPGA-Based Channel Coding Architectures for 5G Wireless Using High-Level Synthesis. International Journal of Reconfigurable Computing. 2017. 1-23.10.1155/2017/3689308.

[2] Choi, Geon & Park, Kyeong-Bin & Chung, Ki. (2018). Optimization of FPGA-based LDPC decoder using high-level synthesis. ICCIP '18: Proceedings of the 4th International Conference on Communication and Information Processing. 256-259. 10.1145/3290420.3290441.

[3] H. Wu and H. Wang, "A High Throughput Implementation of QC-LDPC Codes for 5G NR," in IEEE Access, vol. 7, pp. 185373-185384, 2019, doi: 10.1109/ACCESS.2019.2960839.

[4] Y. Liu, P. M. Olmos and D. G. M. Mitchell, "Generalized LDPC Codes for Ultra Reliable Low Latency Communication in 5G and Beyond," in IEEE Access, vol. 6, pp. 72002-72014, 2018, doi: 10.1109/ACCESS.2018.2880997.

[5] G. Jo, H. Kim, J. Lee and J. Lee, "SOFF: An OpenCL High-Level Synthesis Framework for FPGAs," 2020 ACM/IEEE 47th Annual International Symposium on Computer Architecture (ISCA), Valencia, Spain, 2020, pp. 295-308, doi: 10.1109/ISCA45697.2020.00034.

[6] Y. Wang, W. Liu and L. Fang, "Adaptive Modulation and Coding Technology in 5G System," 2020 International Wireless Communications and Mobile Computing (IWCMC), Limassol, Cyprus, 2020, pp. 159-164, doi: 10.1109/IWCMC48107.2020.9148457.

[7] A. Verma and R. Shrestha, "A New Partially-Parallel VLSI-Architecture of Quasi-Cyclic LDPC Decoder for 5G New-Radio," 2020 33rd International Conference on VLSI Design and 2020 19th International Conference on Embedded Systems (VLSID), Bangalore, India, 2020, pp. 1-6, doi: 10.1109/VLSID49098.2020.00018

[8] H. Li, B. Bai, X. Mu, J. Zhang and H. Xu, "Algebra-Assisted Construction of Quasi- Cyclic LDPC Codes for 5G New Radio," in IEEE Access, vol. 6, pp. 50229-

50244, 2018,doi: 10.1109/ACCESS.2018.2868963.Bae, Jung & Abotabl, Ahmed & Lin, Hsien-Ping & Song, Kee-Bong & Lee, Jungwon. (2019). An overview of channel coding for 5G NR cellular communications. APSIPA Transactions on Signal and Information Processing. 8. 10.1017/ATSIP.2019.10.

[9] M. V. Patil, S. Pawar and Z. Saquib, "Coding Techniques for 5G Networks: A Review," 2020 3rd International Conference on Communication System, Computing and IT Applications (CSCITA), Mumbai, India, 2020, pp. 208-213, doi: 10.1109/CSCITA47329.2020.9137797.

[10] Arora, Komal, Randhawa, Yogeshwar PY - 2020/04/01 SP - 1 EP - 27 T1 - A survey on channel coding techniques for 5G wireless networks VL - 58 DO - 10.1007/s11235-019-00630-3 JO - Telecommunication Systems

[11] K. Zhang, X. Huang, and Z. Wang, "High-throughput layered decoder implementation for quasi-cyclic LDPC codes," *IEEEJournal on Selected Areas in Communications*, vol. 27, no. 6, pp.985–994, 2009

[12] Oh, J., & Ha, J. (2018). A two-bit weighted bit-flipping decoding algorithm for LDPC codes. *IEEE Communications Letters, 22*(5), 874–877

[13] Brink, S., Kramer, G., & Ashikhmin, A. (2004). Design of lowdensity parity-check codes for modulation and detection. *IEEE Transactions on Communications, 52*(4), 670–678.

[14] A. Tasdighi, A. H. Banihashemi, and M. Sadeghi, "Efficient search of girth-optimal QC-LDPC codes," IEEE Transactions on Information Theory, vol. 62, no. 4, pp. 1552–1564, April 2016.

[15] Nader Atteya, Sergey Maximov, Mohamed A. El-saidny,"5G NR," Mediatek, 2018

[16] N. Yang et al.,"Reconfigurable Decoder for LDPC and Polar Codes," 2018 IEEE International Symposium on Circuits and Systems (ISCAS), Florence, 2018, pp. 1-5.

[17] Bashirreza Karimi and Amir H. Banihashemi, "Construction of QCLDPC codes with low error floor by efficient systematic search and elimination of trapping sets," CoRR, vol. abs/1902.07332, 2019.

[18] Amin A A , Basak D , Khadem T , et al. Analysis of modulation and coding scheme for 5th generation wireless communication system[C]// International Conference on Computing. IEEE, 2017.