IMPROVEMENT IN FPGA IMPLEMENTATION OF LDPC DECODER

A Project Report

submitted by

OJASVI GARG

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

MASTER OF TECHNOLOGY

DEPARTMENT OF ELECTRICAL ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY MADRAS.

JUNE 2014

THESIS CERTIFICATE

This is to certify that the thesis titled **IMPROVEMENT IN FPGA IMPLEMENTATION OF LDPC DECODER**, submitted by **Ojasvi Garg**, to the Indian Institute of Technology, Madras, for the award of the degree of **Master of Technology**, is a bona fide record of the research work done by him under our supervision. The contents of this thesis, in full or in parts, have not been submitted to any other Institute or University for the award of any degree or diploma.

Prof.Nitin Chandrachoodan
Research Guide
Associate Professor
Dept. of Electrical Engineering
IIT-Madras, 600 036
Place: Chennai

Date : 20th June 2014

ACKNOWLEDGEMENTS

I express my sincere thanks to Dr. Nitin Chandrachoodan and Dr.Andrew Thangaraj for their guidance and constant encouragement throughout the project work. I am grateful to them for providing me their valuable time through various sessions to discuss the project work which enabled me to take this project to fruitful completion.

Special thanks to Dr.Nitin for explaining me concepts of Digital Design and for helping me in Improvement of hardware implementation of LDPC Decoder. I am greatly indebted to him for the knowledge on VLSI design that I gained during the course of my project work.

My sincere gratitude to Dr. Andrew for explaining the basics of LDPC codes during the initial phase and for helping me understanding algorithm issues related to LDPC decoder.

I would also like to thank Lab mates who helped me in lab during the course of the project work.

My sincere thanks to all my friends at IIT Madras, who supported me during my stay in the campus and made it really enjoyable and memorable.

Finally, I thank my family for their support and constant encouragement.

ABSTRACT

This thesis presents a multi-rate LDPC decoder architecture designed for parallelism equal to expansion factor. The LDPC architecture is designed for OFDM receiver. All improvement in the code has been done by considering the requirement of OFDM receiver. Parallelism equal to expansion factor improves the computation time for suitable no. of iteration for OFDM receiver. According to high bit rate requirement of OFDM receiver, LDPC is designed for multiple code rate. LDPC decoder is implemented for Parallelism P=8,P=32 and P=96. It presents a suitable choice between different parallelism for OFDM receiver. It also presents optimization of resources for improving the area. Analysis for FIFO as storing the input data is also present for parallelism P=96. It provides extra clock cycle for processing the one set of data. Detail analysis with different depth is present for clear understanding.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABBREVIATIONS

IITM	Indian Institute of Technology, Madras
RTFM	Read the Fine Manual

NOTATION

r	Radius, m
α	Angle of thesis in degrees
β	Flight path in degrees

CHAPTER 1

INTRODUCTION

Error correcting codes are widely used in communication system for error free transmission of data. LDPC codes and turbo codes are two best known codes for achieving performance near to the Shannon limit of channel capacity. When compared to turbo codes, LDPC decoding algorithm is parallel and simpler computational requirement. Due to its excellent error correcting capability and low implementation complexity, LDPC codes are preferred for FPGA implementation.

Scope of this project is to improve computation time for make it compatible for OFDM receiver chain. LDPC codes are characterized by Low density parity check matrix. Using this matrix decoding of received codeword is done. Min-sum decoding algorithm is used for decoding the incoming codeword. Based on implementation, LDPC code is divided into three category, fully parallelism,partly parallelism and serial decoder. To improve computation time, LDPC code is processed by using Different parallelism. Parallelism means that it can process more than one row of H matrix at a time. For serial decoder,it process one row at a time. Serial decoder takes more computational time than partly or fully parallelism LDPC decoder. Serial LDPC decoder occupies less area than parallel processed LDPC. Hardware resources increase with parallelism. So partly parallelism may be good choice for both computation time and area.

LDPC code can be used for different code rates like 2/3,3/4,1/2 and 5/6. According to requirement LDPC code is used for any of the code rates. There are different matrix for different code rates so LDPC decoder is implemented in hardware in such a way that it can be processed for different code rates.The proposed architecture of LDPC decoder for parallelism(P)= expansion factor converts base parity check matrix into Z*Z circularly right shifted identity matrix.

By using parallelism and different code rates ,it is occupying a lot of FPGA resources and RAM. Resources optimization is being done by changing the logic of different module. By removing temporary memory, memory optimization is being done.

1.1 Scope of the Problem

LDPC decoder is part of OFDM receiver chain which play a crucial play in error correction. OFDM sends data in 384 clock cycle and 192 blank clock and repeat again this process. It requires that LDPC which can capture and process in this available time. So in this project,we processed LDPC for parallelism P=8,P=32 and P=96 and try to find out which parallelism will be suitable for OFDM receiver in terms of speed and area. We have discussed comparison of different parallelism with respect to area,speed and performance(no. of iteration) by keeping compatibility with OFDM receiver.OFDM receiver may require different code rate for high bit rate so LDPC is converted for multiple code rate for this requirement.It is processed for 2/3,1/2,3/4 and 5/6 code rate for different bit rate.Now OFDM can switch to any code rate according to requirement of bit rate and can choose parallelism according to availability of hardware and performance required for operation.

1.2 Organization of the Thesis

Chapter 2 : This chapter gives a brief introduction to LDPC codes and LDPC decoding algorithm. This chapter includes existing hardware architecture of LDPC decoder and memory organization.
Chapter 3 : This chapter includes arrangements of input data for different parallelism. In this chapter, it explains computation time for different parallelism and resources required for different parallelism.
Chapter 4 : This chapter explains decoder architecture for P=96 . It explains multi-rate decoder architecture, control block state machine, optimization for resources and analysis with FIFO.
Chapter 5 : This chapter explains implementation result and synthesis result for different parallelism. It also shows different plot between different parameter for parallelism.

CHAPTER 2

LDPC CODES

2.1 Representation of LDPC code

LDPC codes are represented by parity check sparse matrix H having n rows and m columns.The length of codeword is n bits and message length is k= n-m where m is number of parity bits.The code rate is also defined by k(code rate) = n/m .

Based on construction of parity check matrix LDPC codes are classified in two types,random codes and structured codes.In random LDPC codes ,1's of H matrix is distributed randomly which makes hardware implementation of it impractical. In structured LDPC, 1's of H matrix is properly arranged for good performance. Each element of H_b is replaced by Z*Z identity matrix.

Based on rows weight, LDPC codes are classified in two type, regular LDPC and irregular LDPC. In regular LDPC number of 1's in each are equal but in irregular LDPC ,number of 1's in each rows is not a constant. Generally regular LDPC is more practical for hardware implementation.

2.2 Code Description

LDPC code is described by matrix H having m rows and n columns.The matrix H is expanded form Base block matrix which is having m_b no. of rows and n_b no. of columns. Each element of H_b is defined as if $P_{ij} \geq 0$ then P_{ij} is represented by circularly right shifted Z*Z identity matrix .If P_{ij} < 0 then P_{ij} is represented by Z*Z zero matrix.

2.3 LDPC Decoding Algorithm

Two phase message passing algorithm is used as decoding algorithm for LDPC codes. In this algorithm soft information is exchanged between check nodes and variable nodes.Decoding algorithm is as follows

Step1. Initialization:
All variable nodes are initialized with channel information obtained by the received bits and all check to variable messages are initialized to zero

$$L_n^{(0)} = y_n, R_{m,n}^{(0)} = 0$$

Step2.Check node processing:
At i_th iteration, check node m receives the variable to check messages from neighboring variable nodes and corresponding check to variable messages are computed.

$$R_{m,n}^{(i)} = S_{m,n}^{(i)} max(M_{m,n}^{(i)}, 0)$$

where $S_{m,n}^{(i)} = \prod\limits_{j \in V_{n/m}} \alpha_{j,m}$,$M_{m,n}^{(i)} = min_{j \in V_{m/n}} \beta_{j,m}$ and $\gamma isapositiveconstantdependentoncode. \alpha_{n,m}$ is the sign of $Q_{n,m}^{(i-1)}$ and $\beta_{n,m}$ is the magnitude of $Q_{n,m}^{(i-1)}$

Step3.Variable node processing:
At i_th iteration, variable node n receives check to variable messages from neighboring check nodes and corresponding variable to check messages are computed as

$$Q_{n,m}^{(i)} = L_n^{(0)} + \sum_{j \in C_{n/m}} R_{j,n}^{(i-1)}$$

where $L_n^{(0)}$ is the channel a prior information of the current bit. At the same time updated values of variable nodes also known as Soft Output(SO) is computed as

$$L_n^{(i)} = L_n^{(0)} + \sum_{j \in C_n} R_{j,n}^{(i-1)}$$

Step4. Final decision:
Steps 2 and 3 are repeated until all parity checks are satisfied i.e $c.H^T = 0$ or maximum number of iterations is reached. Once the decoding process is finished, hard decision is made as follows

$$c_n = \left\{ \begin{array}{ll} 0 & \text{if } L_n \geq 0 \\ 1 & \text{otherwise} \end{array} \right\}$$

Whereas variable is as follows

V_m : set of all variable nodes connected to check node m.

$V_{m/n}$:set of variable nodes connected to c-node m excluding variable node n.

C_n : set of all check nodes connected to variable node n.

$C_{n/m}$: set of check nodes connected to variable node n excluding check node m.

$Q_{n,m}$ is a message sent by the variable node n to the check node m.

$R_{m,n}$ is a message sent by the check node m to the variable node n.

2.4 HARDWARE ARCHITECTURE

Existing hardware architecture is explained in this chapter. This will explain some introduction to Basic LDPC decoder architecture.

2.4.1 Basic Decoder Architecture

The basic block diagram of partly parallel decoder architecture implementing layered de-coding is shown in figure 2.1 . This figure shows basic architecture of LDPC decoder. The input from OFDM module is stored in SO RAM and LLR_mag LLR_sign memory is initially reset to zero. The Node processor unit updates the value of SO RAM and LLR RAM after processing the data according to algorithm. The circular right shifter is used for shifting the SO RAM data to align SO messages with the order of node processors according to Shift value which is stored in Positional ROM. Same shift value is used to shift updated SO value in Left direction to bring it in previous position.

2.4.2 Node Processor Unit

This unit performs functionality of LDPC algorithm. It performs the subtraction of SO output and LLR memory output. After subtraction, it calculates minimum1,minimum2 and index of minimum1 for each row. After using min1,min2 and index of min1, it will update the SO memory and LLR memory. It works serially for calculating minimum1,minimum2 and index of min1. Initially it keeps register1,register2 at maximum value which is 15. As data will come every clock and it compares these register1 value

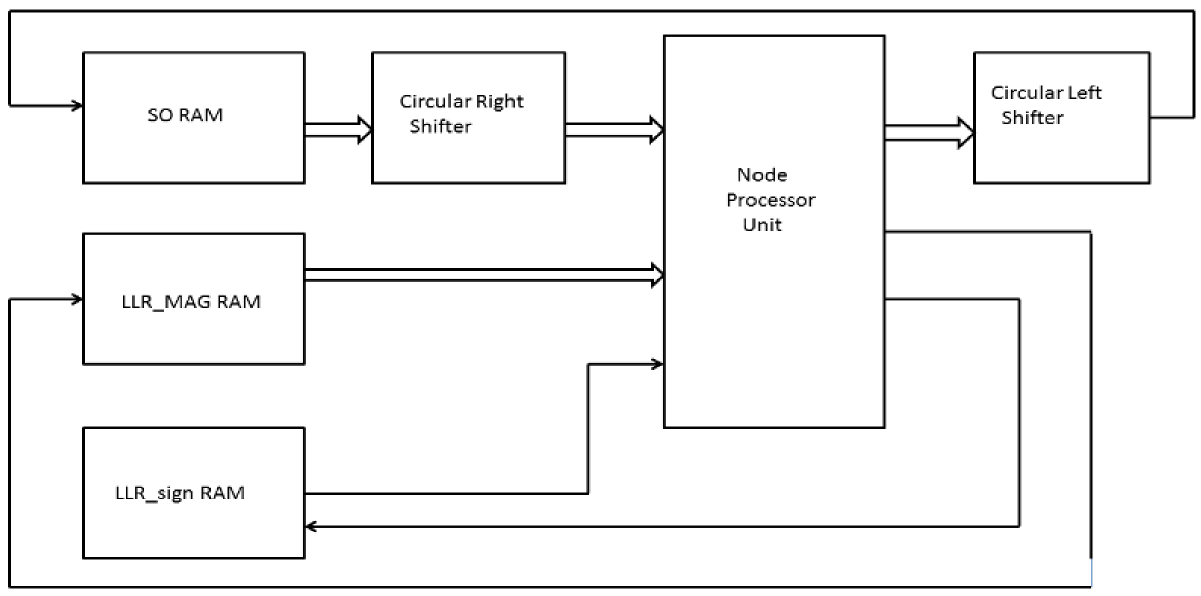


Figure 2.1: Basic Decoder Architecture

with incoming data. If data is lesser than register1 value then it will assign the value of incoming data to register1 and update the index of minimum1. If it is greater than register1 then it will compare this value with register2. If incoming data is lesser than register2 then it will assign data value to register2. It will continue this procedure for all incoming data corresponding to 1's in the row of H matrix.

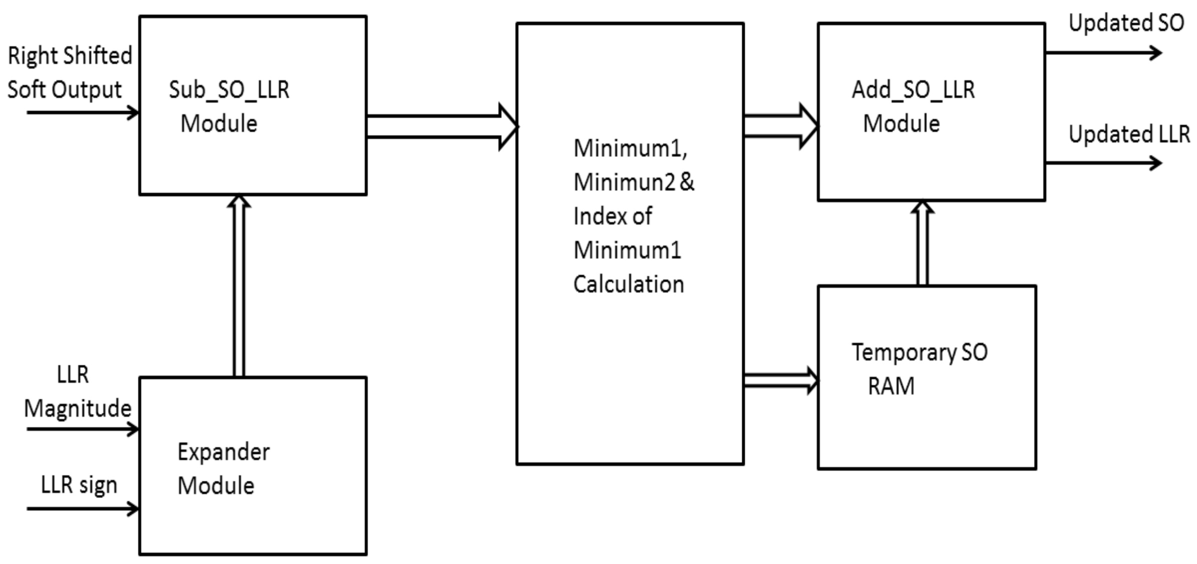


Figure 2.2: Node Processor Architecture

2.4.3 Memory Description

LDPC code used memory to store Soft output data of variable node,LLR magnitude and LLR sign,Positional ROM,S1 ROM,S2 ROM,Temporary memory and min_mem memory for the requirement of the algorithm. Below section will describe each memory in detail.

2.4.4 SO RAM

It stores soft output messages of variable node. Each word contains P SO messages to support parallelism of p. Each SO message contains q+2 bits, where q is number of quantization bits and additional 2 bits are to avoid over ow because of internal computations. Width of this memory is (q+2)*P. SO RAM should be two port RAM with depth 24*Z/P to support pipe-lined operation where 24 is the number of columns in base parity check matrix. Instead SO RAM is made of two single port RAM's each of depth 12*Z/P.

2.4.5 LLR Magnitude RAM

LLR magnitude RAM stores magnitude part of LLR messages in compressed format i.e min1,min2,index of min1. Each word of this memory contains magnitude part of LLR messages corresponding to P row's. LLR magnitude RAM is a single port RAM of depth Nl where Nl is maximum number of row in base H matrices supported by decoder.

2.4.6 LLR Sign RAM

LLR sign RAM stores signs of individual LLR messages. Each word contains a signs of individuals LLR messages corresponding to P row's. Width of the memory is P.LLR sign RAM should be a two port RAM of depth N_{nz} * z/p where N_{nz} is maximum number of non negative elements in base H matrices supported by decoder.

2.4.7 Positional ROM

Position ROM stores the position of non negative elements in base H matrix. Number of positional ROMs required is equal to number of rates supported by decoder.Each Positional ROM has width of 5 i.e ceil(log2 24), since number of columns in base H matrices of Wimax is 24. Depth of each positional ROM is N_{nz} , where N_{nz} is the number of non negative elements in base H matrix corresponding to that rate.

2.4.8 S1 ROM

S1 ROM stores S1 values which represent position of a smaller sub block with in a sub block corresponding to sub layer. Number of S1 ROMs required is equal to number of rates supported by decoder. Each S1 ROM is a single port ROM of width $log_2(Z/P)$ and depth N_{nz}.

2.4.9 S2 ROM

S2 ROM stores S2 values which represents shift values required for permuter. Number of S2 ROM required is equal to number of rates supported by decoder. Each S2 ROM is a single port ROM of width $log_2 P$ and depth N_{nz}.

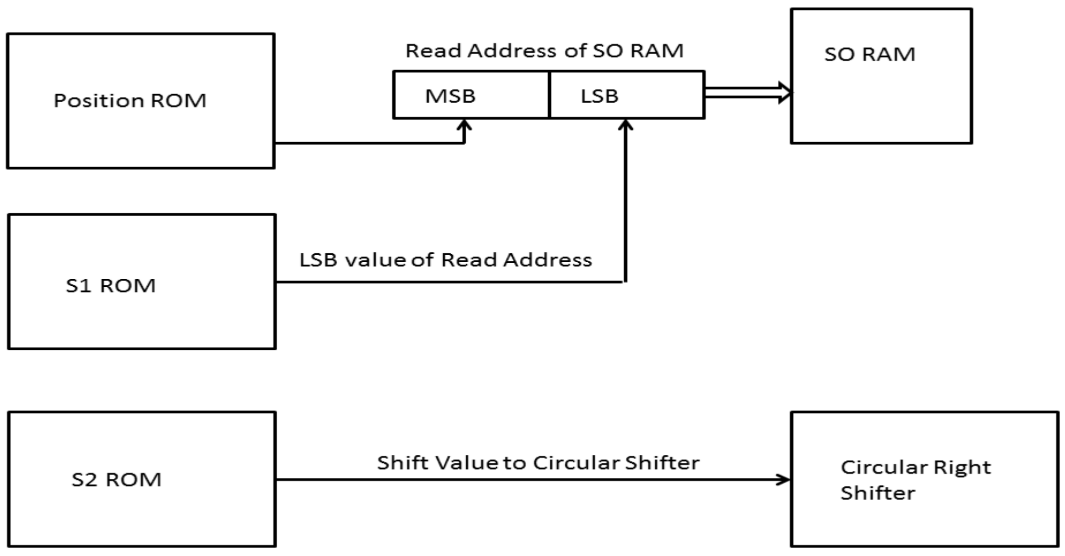


Figure 2.3: Memory Organization

2.4.10 Sign-bit RAM

Sign-bit RAM is used in serial CNU to store signs of incoming variable to check messages. Sign-bit RAM is a two port RAM of width p.

2.4.11 Temporary S1 and S2 RAMs

These RAMs are used to store S1 and S2 values temporarily and used for address generation necessary for processing sub layers. Width of temporary S1 RAM is $log_2(Z/P)$ and width of temporary S2 RAM is $log_2 P$.

CHAPTER 3

Choice between different parallelism

LDPC is used for different parallelism according to the requirement of the system. In this chapter Parallelism P = 8, P= 32 and P = 96 will be discussed in every aspect of application. It will discuss all the changes being done in hardware for different parallelism. The trade-off of different parallelism and effect on computation time will be discussed in this chapter.

3.1 LDPC Part of OFDM Receiver

LDPC decoder is a part of OFDM(Orthogonal Frequency Division Multiplexing) receiver chain. OFDM is a method of encoding digital data on multiple carrier frequencies.First incoming data is received by Timing and Frequency Estimation and it will check for the preamble over the received data. Then it does Frequency correction for estimated frequency offset in previous stage.

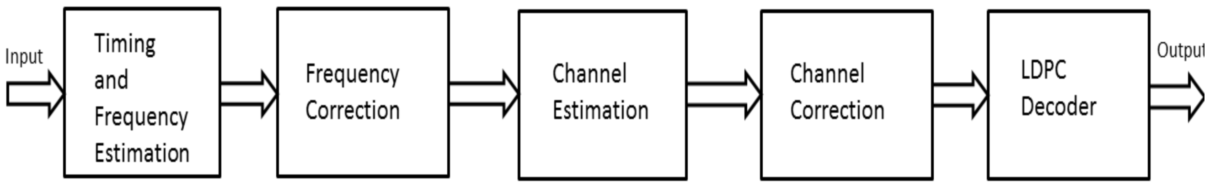


Figure 3.1: OFDM receiver chain

Then it performs Channel estimation and Channel correction. From the received data it detects the pilot and null values from the known data locations values. It estimates the offset in the pilot values and raises the flag for the next stage. This offset value is used in obtaining the channel corrected data of the received input and raises a flag for the next stage. After LDPC decoder receives the flag and start getting data for decoding. It is used for correction error present in the codeword.

3.2 Interface With OFDM receiver

LDPC decoder gets data from Channel Estimation module of OFDM receiver chain. It receives 16 bit real and 16 bit imaginary data in one clock cycle. There is pattern for incoming data to LDPC module. It receives a start signal which will be high for 384 clock cycle then it starts receiving 384 data in 384 clock cycle. After 384 cycle, it will not receive any data in next 192 clock cycle. For these 192 cycle, start signal will be low. Again it will receive 384 data and this sequence will be continue for receiving the data.

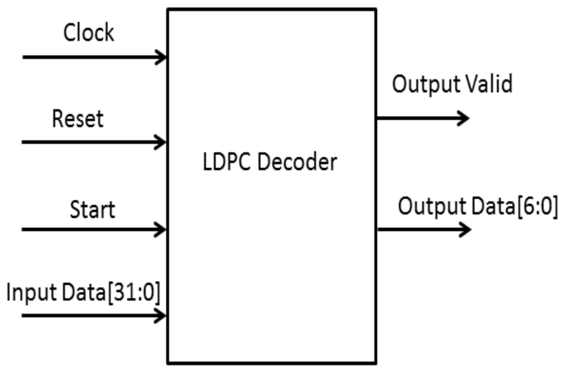


Figure 3.2: Interface between Channel Estimation and LDPC decoder

Due to less time availability of time between two set of data, we have to instantiate more than one module of LDPC for doing more number of iteration. It will try with different parallelism which will be more suitable in terms of area and computation time. We need to choose best case for OFDM receiver.

3.3 Arrangement of input data for different parallelism

For the parallelism P = 8, P =32 and P=96 ,it needs to arrange input data in some particular manner. LDPC decoder is using H matrix which is having m rows and n columns. H matrix is expanded form of Base block matrix. According to algorithm, LDPC processed input code word for every row. For each row, it calculates minimum1,minimum2 and index of minimum1. So LDPC arranged data in such a manner so that it can calculates min1 and min2 for more than one rows in single clock cycle.

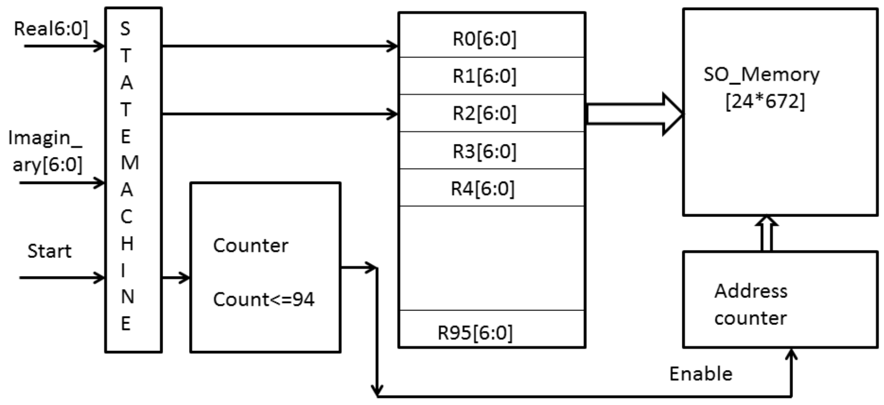


Figure 3.3: Architecture for Arrangement of Input Data

For arranging data in this particular manner LDPC uses a state machine. It uses architecture for arranging the data as shown in fig. 3.3. In the First state of state machine, it will arrange incoming real and imaginary data in a array of registers. The array of registers is arranged in some particular manner for different parallelism. After getting one frame of data which consists of 384 real and imaginary data, it will start arranging data into the SO memory.In the second state of state machine, it increments the address of SO memory and enable the writing signal for writing into the memory. After 192 clock cycle, OFDM module will send the input data, so it will wait in the first state for 192 clock cycle. After that it will again repeat the process for arranging data into the SO memory. As it will arrange complete input code word,it will enter into 3rd state for start decoding.

3.3.1 Arrangement for P=8

For P=8, LDPC will process 8 rows at a time. In this case Z/P = 96/8=12,so it will arrange all columns having modulus 12 as zero i.e. 0,12,24,36,48—-84 then it will arrange columns having modulus 12 as 1 i.e. 1,13,37,49—-85 and so on up to modulus 12 is 11 . It will arrange rows also in this manner. Using this approach LDPC will arrange data in the array of registers .

$$Array = \begin{bmatrix} r[0] & r[12] & r[24] & r[36] & r[48] & r[60] & r[72] & r[84], \\ r[1] & r[13] & r[25] & r[37] & r[49] & r[61] & r[73] & r[85], \\ r[2] & r[14] & r[26] & r[38] & r[50] & r[62] & r[74] & r[86], \\ r[3] & r[15] & r[27] & r[39] & r[51] & r[63] & r[75] & r[87], \\ r[4] & r[16] & r[28] & r[40] & r[52] & r[64] & r[76] & r[88], \\ r[5] & r[17] & r[29] & r[41] & r[53] & r[65] & r[77] & r[89], \\ r[6] & r[18] & r[30] & r[42] & r[54] & r[66] & r[78] & r[90], \\ r[7] & r[19] & r[31] & r[43] & r[55] & r[67] & r[79] & r[91], \\ r[8] & r[20] & r[32] & r[44] & r[56] & r[68] & r[80] & r[92], \\ r[9] & r[21] & r[33] & r[45] & r[57] & r[69] & r[81] & r[93], \\ r[10] & r[22] & r[34] & r[46] & r[58] & r[70] & r[82] & r[94], \\ r[11] & r[23] & r[35] & r[47] & r[59] & r[71] & r[83] & r[95], \end{bmatrix}$$

3.3.2 Arrangement for P=32

For P=32, LDPC will process 32 rows at a time. In this case Z/P = 96/32=3,so it will arrange all columns having modulus 3 as zero i.e. 0,3,6,9,12—-93 then it will arrange columns having modulus 3 as 1 i.e. 1,4,7,10—-94 and so on up to modulus 3 is 2 . It will arrange rows also in this manner. Using this approach LDPC will arrange data in the array of registers.

$$Array = \begin{bmatrix} r[0] & r[3] & r[6] & r[9] & r[12] & --- & --- & r[93], \\ r[1] & r[4] & r[7] & r[10] & r[13] & --- & --- & r[94], \\ r[2] & r[5] & r[8] & r[11] & r[14] & --- & --- & r[95], \end{bmatrix}$$

. This arrangement will be followed for remaining rows of block parity matrix.In this manner it will be arranged in SO memory.

3.3.3 Arrangement for P=96

For P=96, There is no need of arrangement of data in some particular fashion.It will first 96 data in the first row of SO memory and then 2nd 96 data in the 2nd row of SO memory.This makes it easier for implementation in hardware. Z= 96 is the expansion factor, that is why there is no need of arrangement.

$$Array = \begin{bmatrix} r[0] & r[2] & r[3] & r[4] & r[5] & --- & --- & r[95], \end{bmatrix}$$

.

3.4 Comparison for speed

LDPC decoder is part of OFDM receiver chain which is working at 20 MHZ frequency. LDPC should work at such speed so that LDPC decoder can do more no. of iteration. OFDM module gives data in some particular sequence, so LDPC should capture data correctly and process the data before coming the next data. By considering this point in mind , we need to use more than one module of LDPC for OFDM receiver. The number of module is depend on parallelism and it will reduce with increase in parallelism. We will discuss computation time for different parallelism.

3.4.1 Computation Time for Parallelism=8

For Parallelism P=8 ,It processed 8 rows at time. As we know one row of block parity matrix will be converted into 8x8 circularly right shifted identity matrix. It takes 11 clock cycle to process 8 rows at a time. So total clock cycle required for one row of block parity matrix is (12*11= 132). For complete matrix it will take (132*8)=1056 clock cycle for one iteration. For reducing error bit ,it is required to process more number of iteration, at least 8 iteration is required. For that it will take 8448 clock cycle to complete the process.

OFDM sends 384 data in 384 clock cycle and then 192 blank clock cycle. So if OFDM uses one module of LDPC then after receiving first 2304 data ,LDPC has only

192 clock cycle to process the data but this is impossible to process data in 192 clock cycle. So OFDM needs to use more than one module of LDPC. With 2 module of LDPC, first set of 2304 data will be received by 1st module and 2nd set of data will be received by 2nd module and again 3rd set of data will be received by 1st module and it will goes on further. After receiving first set of data by 1st module, it has time (192+384+192+384+192+384 + 192 = 1920) clock cycle. In this time it can process only 1 iteration because it will take 1056 clock cycle for 1 iteration and 320 clock cycle for getting the output from SO memory. It will take total time (1056 + 320 = 1376) which is possible in given available time but 2 iteration is not possible.

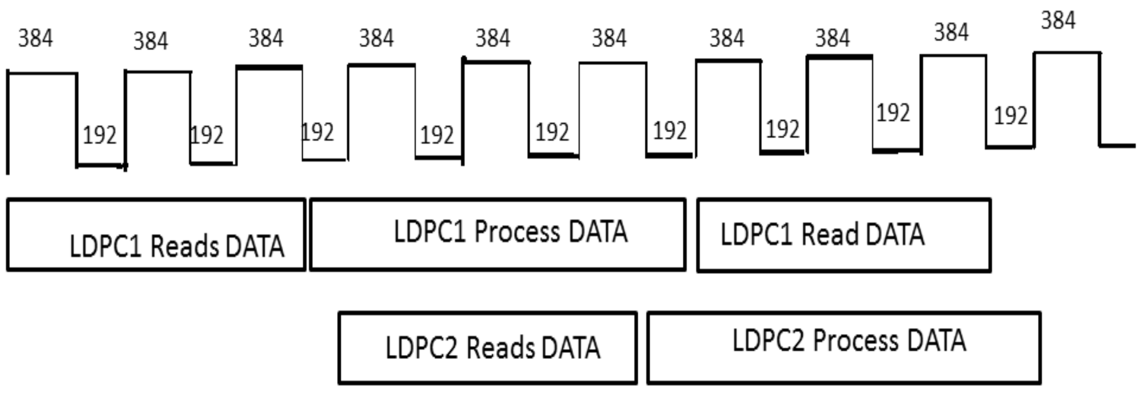


Figure 3.4: Instantiation of Two Module of LDPC

With three module LDPC,it has available time(1920 + 3*384 +3*192 = 3648)clock cycle. For one iteration it takes 1056 clock cycle and for 3 iteration it will take 3*1056 = 3168 clock cycle and 320 clock for getting the output data. So total cycle required for three iteration is 3488 which is less than available clock cycle for 3 iteration. So 3 iteration is possible with 2 module of LDPC with parallelism = 8. With four module LDPC, it has available time (3648 + 3*384+3*192 = 5376)clock cycle. For 5 iteration it will take 1056*5 =5280 clock cycle and 320 clock cycle for taking output. So total clock(5280 + 320 = 5600)cycle are greater than available clock cycle.So 4 iteration is possible with 4 module of LDPC.

3.4.2 Computation Time for Parallelism=32

For parallelism P= 32, it processed 32 rows at a time .In this every row of block parity matrix will be converted into 32x32 matrix. So it takes 11 clock cycle for every row of

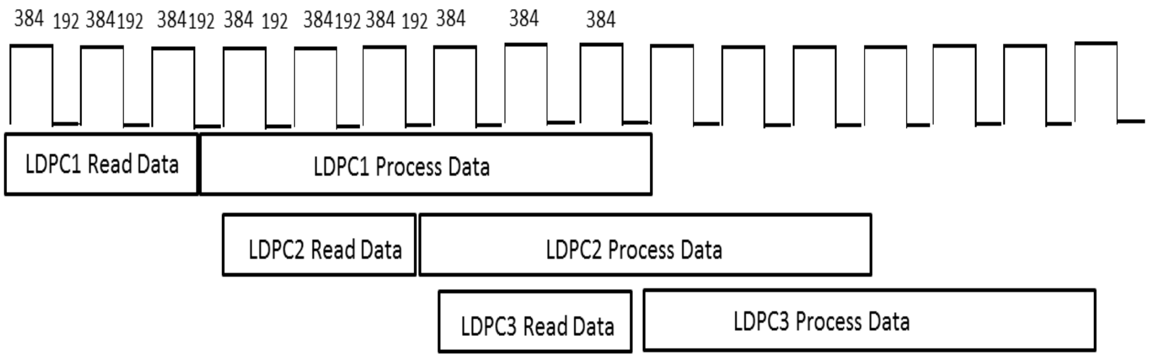


Figure 3.5: Instantiation of Three Module of LDPC

Table 3.1: No. of LDPC module required for OFDM with P=8

NO. of Instantiate module	Available clock cycle	iteration	required clock cycle
One module	192	Not Possible	—-
Two module	1920	1 iteration	1376
Three module	3648	3 iteration	3488
Four module	5376	4 iteration	4544
Five module	7104	6 iteration	6656

expanded matrix. It will take 33 clock cycle for one row and another 11 clock cycle for updating the so memory. It will take 44 clock cycle for one row. It will take (44*8 = 352) clock cycle for one iteration. If it will use one module of LDPC for parallelism =32 ,it will loose the incoming data. For proper working of OFDM receiver, it is required to use more than one module of LDPC.

With 2 module of LDPC for P=32,it has available time 1728 clock cycle.For one iteration it takes 352 clock cycle and 560 clock cycle for taking the output. So it has remaining clock cycle is (1920 - 560 = 1360).So It can process 3 iteration in this time which is not sufficient for high performance. With 3 module of LDPC ,it has available time 3648 clock cycle. The time available for performing process is 3648 - 560 = 3088 clock cycle. In this time, It can perform 8 (3088/352 = 8.77) iteration easily. With Four module of LDPC, it has available 5376 clock cycle. So remaining clock cycle for processing is (5376 - 560 = 4816) clock cycle. In these clock cycle no . of iteration = 4816/352 = 13.68 so 13 iteration is possible with 4 module of LDPC.

Table 3.2: No. of LDPC module required for OFDM with P=32

NO. of Instantiate module	Available clock cycle	iteration	required clock cycle
One module	192	Not possible	—-
Two module	1920	3 iteration	1616
Three module	3648	8 iteration	3376
Four module	5376	13 iteration	5136

3.4.3 Computation Time for Parallelism=96

For parallelism P=96,it process one row of block parity check matrix in just 11 clock cycle. Due to overlapping between consecutive layer of matrix,it uses one extra state for updating the SO memory. So it will take 22 clock cycle for one row processing of block parity matrix. It will take 22*8 = 176 clock cycle to processing of complete matrix. This is most suitable for OFDM receiver which can process more iteration easily without any loss of data. For making it compatible with OFDM receiver ,it may use more no. module of LDPC.

With 1 module of LDPC for P=96, it has 192 clock cycle available for processing. For 1 iteration,it takes 176 clock cycle and 208 clock cycle for getting the output from SO memory. It will take 384 clock exact for 1 iteration. With 2 module, it has available time is 1728 clock cycle. Remaining clock cycle for processing is (1920 -208 = 1712)clock cycle. The no. of iteration is (1712/176 = 9.72) 9 . SO it can process 9 iteration with 2 module. With 3 module of LDPC, it has 3648 clock cycle for processing the data. So it has remaining (3648 -208 = 3440) clock cycle for processing data. The no. of iteration is (3440/176 = 19.54) 19 performed with 3 module of LDPC for P= 96. With 4 module of LDPC, it has 5376 clock cycle for processing. So in these clock cycle, it can perform (5168/176 = 29.36) 29 iteration easily. So it is possible more iteration with LDPC with parallelism P=96.

From all above discussion, it is clear that LDPC with parallelism P=96 is most suitable for OFDM receiver chain. It can process more iteration with less no. module. For good performance ,it can process 20 iteration with only 3 LDPC module.

Table 3.3: No. of LDPC module required for OFDM with P=96

NO. of Instantiate module	Available clock cycle	iteration	required clock cycle
One module	192	not possible	384
Two module	1920	9 iteration	1792
Three module	3648	19 iteration	3552
Four module	5376	29 iteration	5312

Table 3.4: computation time table.

Parallelism	Computation Time(1 iteration)
P=8	1056 clock cycle
P=32	352 clock cycle
P=96	176 clock cycle

3.5 Comparison for Area

Area is also a main issue for selection between parallelism. As parallelism will increase, area will also increase. Due to increase in parallelism, Node processor unit will also increase in same proportional. Due to limited FPGA resources we need to take care of resources as well as computation time. In this discussion we will compare Different parallelism with respect to area.

There is node processor unit, circular left shifter,circular right shifter and control block which are using most of the resources of FPGA. These blocks are directly proportional with parallelism.As parallelism increases, resources will also increase.

3.5.1 Effect of parallelism on Node Processor unit

Node processor unit is the central core of LDPC decoder . It implements MIN SUM decoding algorithm in hardware. It includes adder,sub tractor,comparator,memory and mux. The main function of this unit is to calculates the minimum1,minimum2 and index of minimum1.There are other three unit which includes sub_so_llr ,add_so_llr and expander unit.The sub_so_llr unit subtracts SO memory data llr memory data. The Add_so_llr unit adds So memory and llr memory data.

In this figure you can see that parallelism directly replicate the copy of node proces-

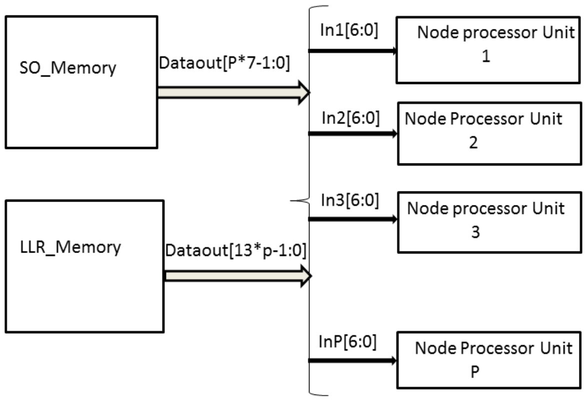


Figure 3.6: Parallelism effect on Node processor Unit

sor unit. It makes parallelism equal copy of node processor unit. Due to this Resources increases with parallelism.

3.5.2 Effect on circular shifter

Circular left and right shifter used for shifting the incoming SO message to align the data with node processor unit. It is uses barrel shifter for shifting operation. Due to this,resources of circular shifter depends on Incoming data Width. Data width increases with parallelism and it is equal to [P*qs-1] where P is parallelism and qs in no. of bits per data.So we can see, as parallelism will increase, Width[P*qs-1] of incoming data will also increase. Due to this ,consumption of resources will also increase.

3.5.3 Effect on Control block

Control blocks includes state machine for controlling the complete hardware,So memory,LLR_mag memory,LLR_sign memory and temporary memory.In state machine,it uses adders, subtractor,comparator and counters. The sizes of subtractor and adder is directly dependent on parallelism. As parallelism increases, Width of most of the register will increase. So it will use higher bits adder and subtractor to perform the operation in state machine. The incoming data width for every memory is directly proportional to parallelism,as width will increase,the resources related to memory will also increase . So there is directly effect of parallelism on resources of Control block.

3.5.4 comparison with different parallelism for Area

Table 3.5: No. of LUTs consumption for different parallelism

Module Name	Parallelism P=8	Parallelism P=32	Parallelism P=96
Min.v	72*8=576	72*32=2304	72*96=6912
Add_so_llr.v	14*8=112	14*32=448	14*96=1344
Sub_so_llr.v	18*8=144	576	1728
Expander.v	10*8=80	10*32=320	10*96=960
SO memory	57 and 1 RAM	225 and 4 RAM	673 and 10 RAM
LLR_mag memory	288 and 2 RAM	1102 and 9 RAM	3279 and 24 RAM
LLR_sign memory	35 and 1 RAM	85 and 1 RAM	207 and 2 RAM
Circular right shifter	416	728	4704
Circular left shifter	416	728	4704
Control block	1134	3172	3609
Total	3258	10125	27986

From this table, it is clear that Parallelism P=96 is taking 2.5 times greater LUTs than P= 32 and 8 times greater LUTs than P=8. It is obvious from above analysis that Most of the resources are consumed by node processor unit(min.v,Add_so_llr.v,sub_so_llr.v,expander.v For checking compatibility with OFDM receiver, we need to consider area occupied by modules of LDPC with the speed of Modules. It is quite clear from speed point of view that LDPC P=96 is better than other Parallelism.But in terms of area,it is taking more resources than other parallelism so we need to make arrangement to resolve this problem.

3.6 Comparison between area and performance

After getting the area and computation time detail,we need to see which parallelism will be better for OFDM receiver. For compatibility with OFDM receiver,it should process the incoming data with high performance without losing the data. Due to limited resources availability on FPGA, area is also main issue with parallelism. In this section we will try to choose which parallelism will be good for OFDM. In following table,you can see details about every parallelism in terms of how many module required to perform sufficient iteration and how much area it will occupy. With this information picture will be more clear.

Table 3.6: comparison for P=8

Parallelism	no. of module	No. of iteration	total resources(LUTs)
P=8	2	1 iteration	6516
P=8	3	3 iteration	9774
P=8	4	4 iteration	13032
P=8	5	6 iteration	16290
P=8	6	8 iteration	19548

This table shows that with parallelism P=8,it requires 6 module to perform 8 iteration and it occupies area is 19548. It is somehow difficult to arrange 6 module because it will require a state machine which will arrange data in these module. So it is not good practice to use 6 module.

Table 3.7: comparison for P=32

Parallelism	no. of module	No. of iteration	total resources(LUTs)
P=32	2	3 iteration	20250
P=32	3	8 iteration	30375
P=32	4	13 iteration	40500

With parallelism P=32,it is possible to perform 8 iteration with 3 module and 13 iteration with 4 module. For 8 iteration it is taking more area than parallelism P=8,but it is using only 3 module of LDPC which is easy for arrangement of data.

Table 3.8: comparison for P=96

Parallelism	no. of module	No. of iteration	total resources(LUTs)
P=96	1	1 iteration	27986
P=96	2	10 iteration	55972
P=96	3	20 iteration	83958
P=96	4	30 iteration	111944

With Parallelism P=96, it is possible high performance of LDPC. With 2 module it can process 10 iteration but it takes more hardware as compared to parallelism P=32. So there is conflict between P=32 and P=96 between performance and area. To resolve this issue we will try FIFO for parallelism P=96 to resolve this issue.

CHAPTER 4

Architecture for LDPC P=96

This chapter explains the architecture for LDPC P=96 which is somewhat different from other parallelism. In this section, it will explain various changes in various block to reduce resources and complexity of hardware. We will discuss optimization and effect of different clock frequency rate on computation time and performance.

4.1 Clipping Architecture

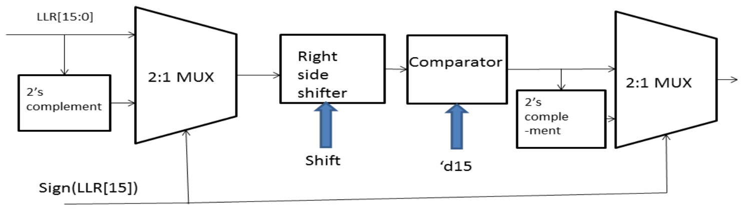


Figure 4.1: Clipping module

The input to LDPC is coming from channel estimation module. It gives 16 bits LLR outputs to LDPC. To convert 16 bits LLR to 7 bit LLR, this input is feeded to clipping module. This module clips the input in such a way so that it lies in the range of [-16 to 15]. For this purpose, it checks the sign of incoming signal. If input is negative, it will take 2's complement of input. After this magnitude is shifted towards right side according to shift value. Now this number is compared with '15' to clip the signal within the range of [-16 to 15]. Again by using the sign signal,if it is positive then there will be no change. But if it is negative then it will take 2's complement to give output.

4.2 Arrangement for P=96

For P=96, It is simple to arrange data in to the SO memory. Out of 2304 input data , first 96 will be arranged in the 1st row and 2nd 96 data in the second row and so on upto 24th row of SO memory. Incoming data will be arranged in a array of register which is arrange in the following manner, when data[94] and data[95] will come ,this array of register will be transferred to SO memory and address of the memory will be incremented.This process will be going on for first set of data. This process is being done using a state machine. OFDM module will send a start signal to state machine to start the arrangement of incoming data.This start signal will be high for 384 clock cycle so in this time it will take data and arrange the data in the SO memory. You can understand from following diagram.

$$Array = \begin{bmatrix} r[0] & r[2] & r[3] & r[4] & r[5] & --- & --- & r[95], \end{bmatrix}$$

.

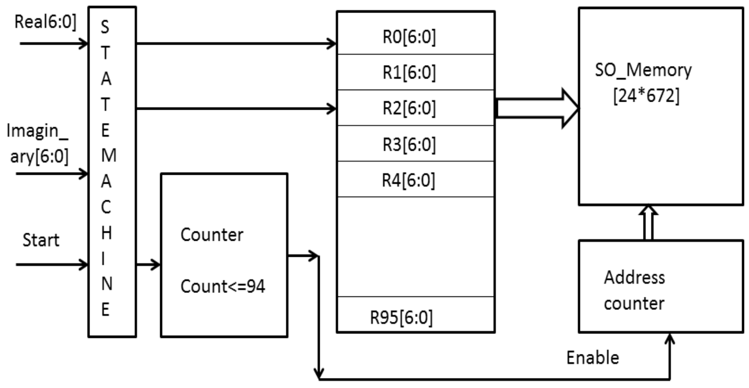


Figure 4.2: Arrangement module

4.3 Addressing Scheme

Position of non negative element is stored in Position ROM. The output of this memory will be used for calculating the address for reading the data from SO memory. The output of Position ROM is equal to SO memory reading address.For Due to parallelism P=96 which is equal to expansion factor,there is no need of use of S1 ROM. S2 ROM is used for storing non negative element of Block H matrix. The output of S2 Rom is used for circularly right and left shifting the output data of SO memory.

There is no need of using temporary memory for storing S2 ROM data and Position ROM data. Temporary memory are used to give value of position and shifting for updating the SO memory. In this case ,it is processing complete one of block H matrix at a time so it will read again S2 ROM and position ROM for updating the SO memory. It will use a pointer which will track the address of S2 ROM and Position ROM. Before updating the SO memory, this pointer brigs back to initial address of that row for which Updating is going to happen.

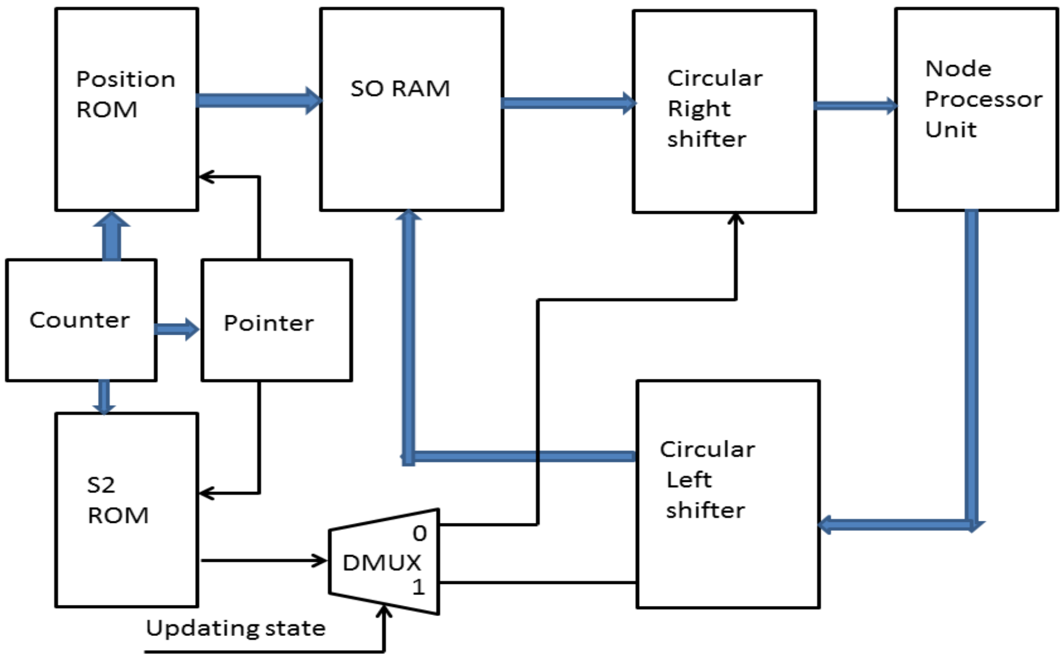


Figure 4.3: Addressing Scheme

4.3.1 Single ROM Representation

Addressing scheme for S12 ROM and Position ROM is almost same,so it can be converted into single ROM Shift_addra which will contain both Position ROM and S2 ROM. The 5 bit MSB of output of Shift_addra will represent the position of non negative element and 7 bit of LSB will represent the value of non negative element in block H matrix. Due to this State will be simplified and register consumption also reduced.

4.4 Control Block State Machine

In control block,State machine is used to control all processing of LDPC Decoder. This state machine control all tasks from arrangement to taking output.To understand the LDPC decoder implementation, it is necessary to understand the each state of state machine.

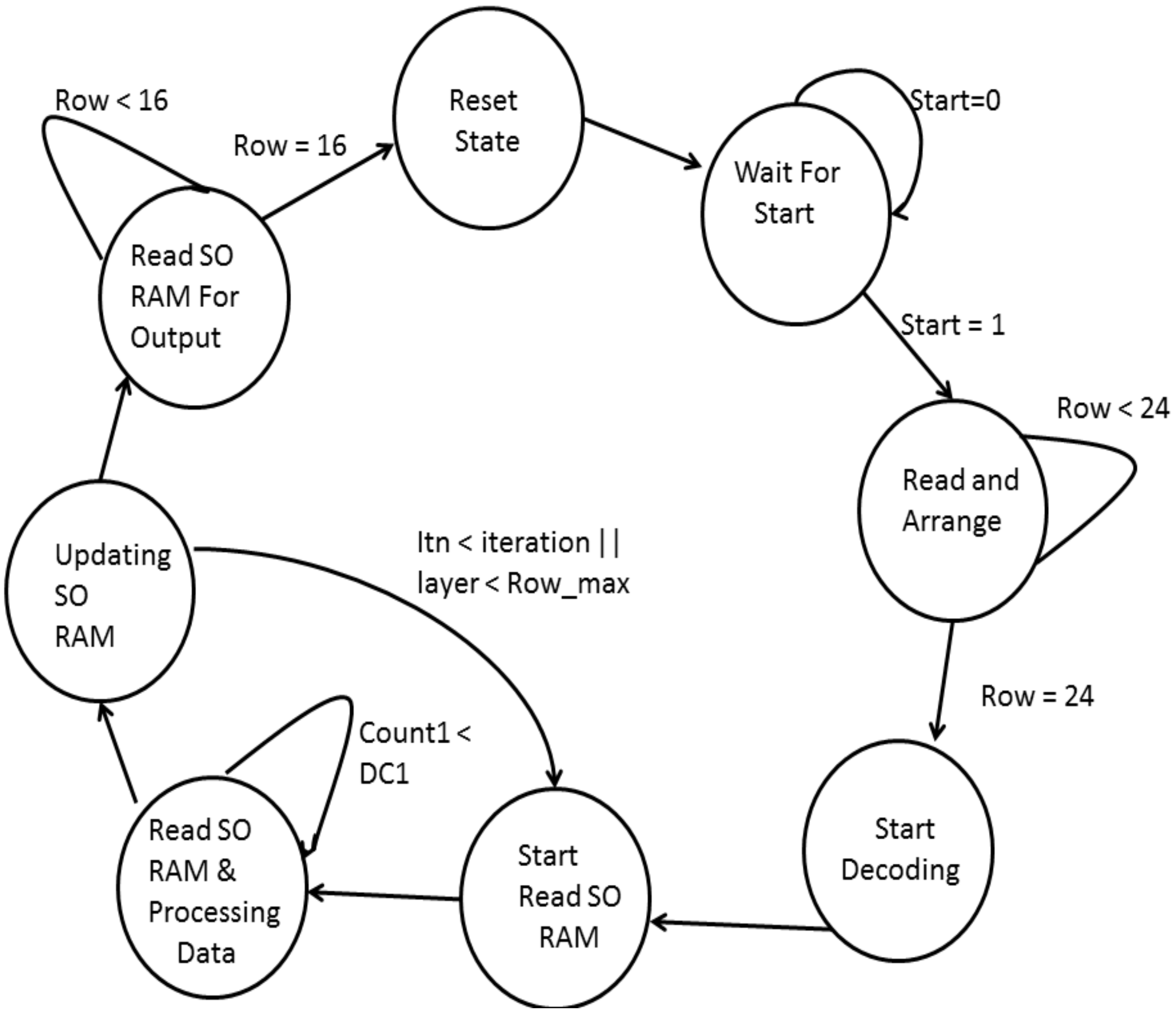


Figure 4.4: State Machine

First State is reset state which reset all register value and it also reset the memory. It will stay for some clock cycle in this state to reset all data of memory. In wait_for_start state, it will wait for start signal which will come from OFDM module. Start signal is indication that OFDM module is start for giving input data for LDPC module. As start signal will come, it will enter next state which is read and arrange state. In Read_and_arrange state, it will read and arrange data into array of register. After getting 96 data, it will transferred array of register to SO memory. OFDM module sends data in the frame of 384 data. After arranging first 384 input data ,it will go back to wait_for_start state to wait again for start signal. After getting start signal, it will follow same procedure for arranging the data. For one set of data which is 384, it require 3 time 384 data which include real and imaginary data. After arranging total 2304 data, it will go to start_decoding state for starting the decoding.

In start_decoding state, it enable the reading signal of Positional ROM and S2 Rom so that it can read data from SO memory in next state.After this, it will enter Start Read_SO_RAM state. In this state, it will enable LLR_mag memory and LLR_sign memory and reads first data from Position ROM and S2 ROM. Then it will enter Read_SO_RAM_Processing state.

In Read_SO_RAM_Processing state,It will enable signal(min_ren) for node processor unit to start processing for incoming data.The min_mem is used for storing the no. of 1's in each row. So in this state, it will initialize a counter which will count upto no. of 1's in that row which is equal to min_mem memory output. If no. of 1's in each row is 10 then counter will run for 10 time. Each time increment in counter will also do increment in address of Position ROM and S2 ROM. According the output of Position ROM SO memory will give output and circular right shifter will shift according to S2 ROM output. For reading LLR_mag and LLR_sign memory, it also increments address of these memory to get data in sequence with counter. In this state, node processor unit will calculate minimum1,minimum2 and index of minimum1 according to incoming SO memory data, LLR magnitude memory and LLR sign memory. All these three data plays crucial role in the algorithm of LDPC, so the order and sequence should be according to algorithm. In this state ,node processor will finish processing for incoming data. As counter will become equal to DC1 which is no. of 1's in each row. Then it will enter into next state which is Updating_SO_RAM state.

In Updating_SO_RAM state, node processor unit will start updating data using the minimum1,minimum2 and index of minimum1. In this state, it will track back the initial address of each row in Position ROM and S2 ROM. For updating the same location,it needs to read again S2 ROM and Position ROM. For this purpose it will use a counter. In this state, it will check for row and if row value is maximum then it will check for iteration. If iteration is less maximum iteration then again it will go to start_decoding state. Otherwise it will go to Read_SO_RAM_Output state.

In Read_SO_RAM_Output state, It will read SO memory to get the message. Message bits are the sign bits of SO memory data[6:0].So every clock cycle it will read one row at a time . For reading 16 rows which is the message length for rate 2/3, will require 16 clock cycle. It sends output data to FIFO, from message can be taken out. After this state, it will go back to reset state.

4.4.1 Change in State machine compared to Other parallelism

For Parallelism P=8 there is less overlapping between consecutive rows of LDPC block matrix. In this state machine,for updating the SO memory there is no requirement of extra state,so it will reserve clock cycle required for updating.

For parallelism P=32 & P=96, there is more overlapping between consecutive rows of LDPC block parity matrix so in this case it require one extra state for updating the SO memory separately. It will take some extra clock cycle for updating the SO memory.

For parallelism P=8 and P=32,it was using temporary memory for storing address and position for sub-layers. But in the case of P=96,there is no sub-layer. It is complete one row of Block H matrix at a time so there is no need of Temporary memory. State will be simplified after removing the temporary memory address line and enable signal.

4.5 Multiple-Rate LDPC

LDPC is described for code rate 2/3 . It can be used for different code rate like 3/4,5/6 and 1/2. Without changing much hardware it should work for different code rate as well. For different rates, it will use different Block H matrix. These matrix will be different in terms of no. of 1's in each row and number of rows in the matrix. The following diagram will explain about the architecture for multiple rate LDPC.

For multiple code rate, LDPC will use different Addra_shift memory related to different code rates. It will use a mux to select the matrix. The select line will be rate which will select code rates.

LDPC is using min_men memory to store number of 1's in each row. Due to dif-

Table 4.1: Rate selection table.

Code rate	Select_Line
Rate 2/3	0
Rate 1/2	1
Rate 3/4	2
Rtae 5/6	3

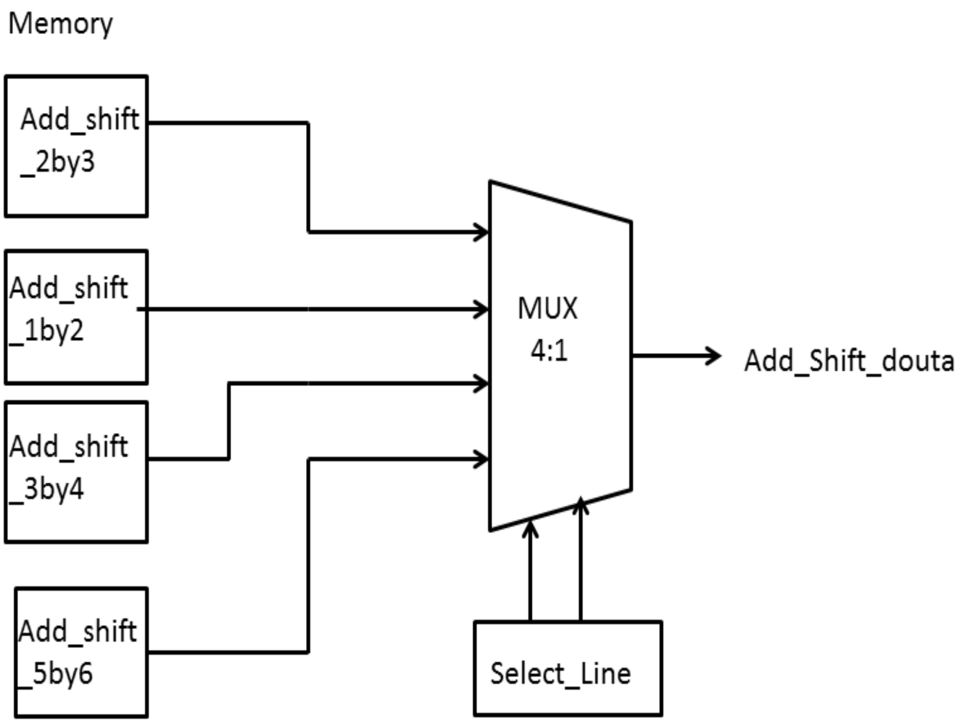


Figure 4.5: Selection of the Addra_shift Memory

ferent matrix for different code rates, it has to use different min_mem memory for each matrix. The selection of min_mem memory will be same as Addra_shift memory. It will use mux to select min_mem memory using rate as a select line. By using this it will information about no. of 1's in each row.The counter of state Read_SO_processing will be compared with the output of min_mem memory output.

4.6 FIFO for storing Output Data

Two FIFO is used for storing the output data. One FIFO will store the output data from SO memory and 2^{nd} FIFO will store message bits which are the sign bits of the output data. In state machine,the Read_SO_Ram will read each row of SO memory to get the data. This data will be stored in FIFO1. The store data in FIFO1 will be taken by FIFO2 to take sign bit of each data and stored it in 8bit of register. This 8 bit register will be stored in FIFO2 for giving final message. After getting read_enable signal from OFDM module, it will start giving data to OFDM module.

4.7 Optimization

The consumed resources are main concern with parallelism P=96. It is taking 3 times more hardware than Parallelism P=32. So optimization of LDPC for P=96 is necessary to reduce some hardware.

4.7.1 Optimization of control block

Control is the main block which is taking more number of LUTs. For other parallelism,it was using temporary memory for storing data of S2 ROM and Positional ROM. By using pointer in the state machine, it is possible to remove these temporary memory. It was using two memory for Position and value of non negative but it can replaced with single memory which will give value. Due to this,it is possible to reduce some adder and memory consumption in the hardware.

4.7.2 Optimization of Node Processor unit

Node processor unit have four unit which includes Min module, Add_so_llr,Sub_so_llr and Expander module . Out of this min module was taking 82 Luts for one module. For parallelism P=96, it will 96 copy of this module. So it will consume a lot of LUTs. For calculating min1,min2 and index of minimum1, it was using three stages. For optimization, LDPC have 2 stage for this task.

4.7.3 Optimization of Circular-Shifter

Circular shifter is also using more hardware in this code. It is using barrel shifter for implementing the circular shifter.As width of data will increase, barrel shifter will consume more hardware. For parallelism P=96,width of input data is 672 bits. So barrel shifter will shift 672 bits at a time.

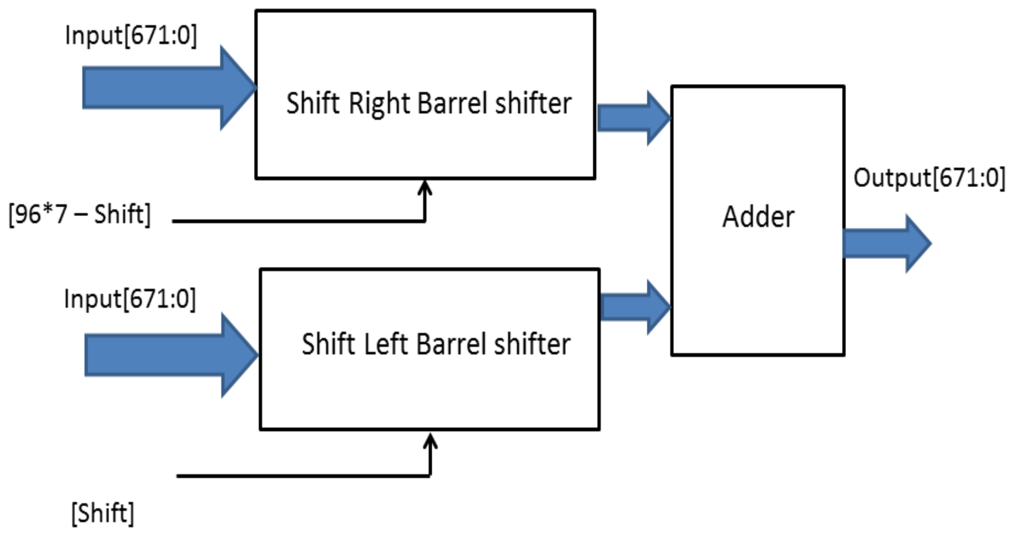


Figure 4.6: Circular Right Shifter

4.8 Analysis with FIFO

If FIFO is used for storing the input data coming from channel estimation module then it can provide some extra time to LDPC for processing the data. The increment in the time is also depend on depth of FIFO and aspect ratio between writing and reading width. If it will use long depth FIFO for storing the data then it can provide more time for processing but it will increase hardware. So we will analyze with different depth and aspect ratio FIFO.

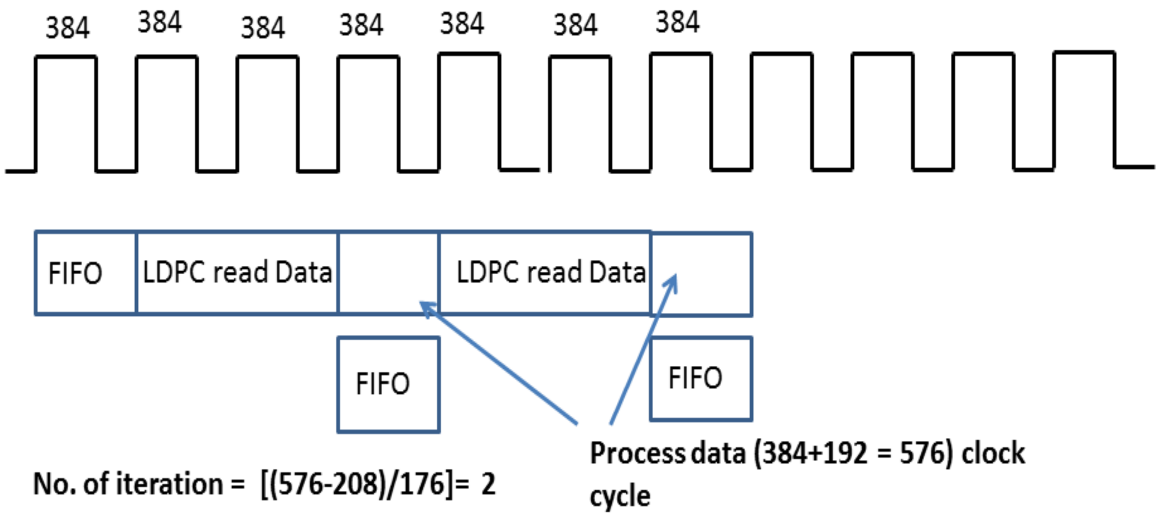

384
384
384
384
384
384
384
FIFO
LDPC read Data
LDPC read Data
FIFO
FIFO
No. of iteration = [(576-208)/176]= 2
Process data (384+192 = 576) clock cycle

Figure 4.7: FIFO depth 384 and LDPC one module

As you can see in figure 4.7 that FIFO with depth 384 provides some extra clock cycle for processing the data.First FIFO will store incoming data and LDPC will read FIFO.

- Available clock for processing data = 576
- no. of clock cycle required for 1 iteration = 176
- no. of iteration = [(576-208)/176] = 2

- **FIFO with Depth with 768 and aspect ratio 1:2**

As shown in figure 4.8 ,FIFO with depth 768 will provide more clock cycle available for process the data. For reading fast from the FIFO, it will use aspect ratio 1:2. For this it will try for different aspect ratio also to check which will be more suitable for this depth of FIFO.

- Available clock for processing data = (384*2 + 192*2 =1152)

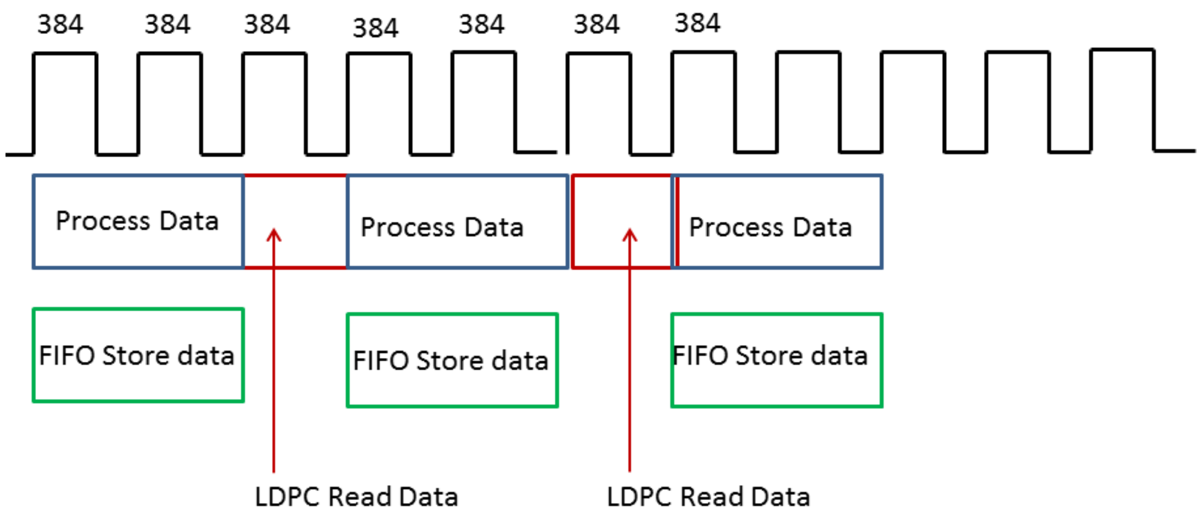


Figure 4.8: FIFO depth 768 and LDPC one module

- no. of clock cycle required for 1 iteration = 176
- no. of iteration = [(1152-208)/176] = 5
- **FIFO with Depth with 768 and aspect ratio 1:4**

With increase in aspect ratio, it will be able to read more in single clock cycle. It will provide some more clock cycle for processing the data.

- Available clock for processing data = (192+384*2 + 192*2 =1344)
- no. of clock cycle required for 1 iteration = 176
- no. of iteration = [(1344-208)/176] = 6
- **FIFO with Depth with 1152 and aspect ratio 1:4**

- Available clock for processing data = (96+192+384*2 + 192*2 = 1440 cycle)
- no. of clock cycle required for 1 iteration = 176
- no. of iteration = [(1440-208)/176] = 7

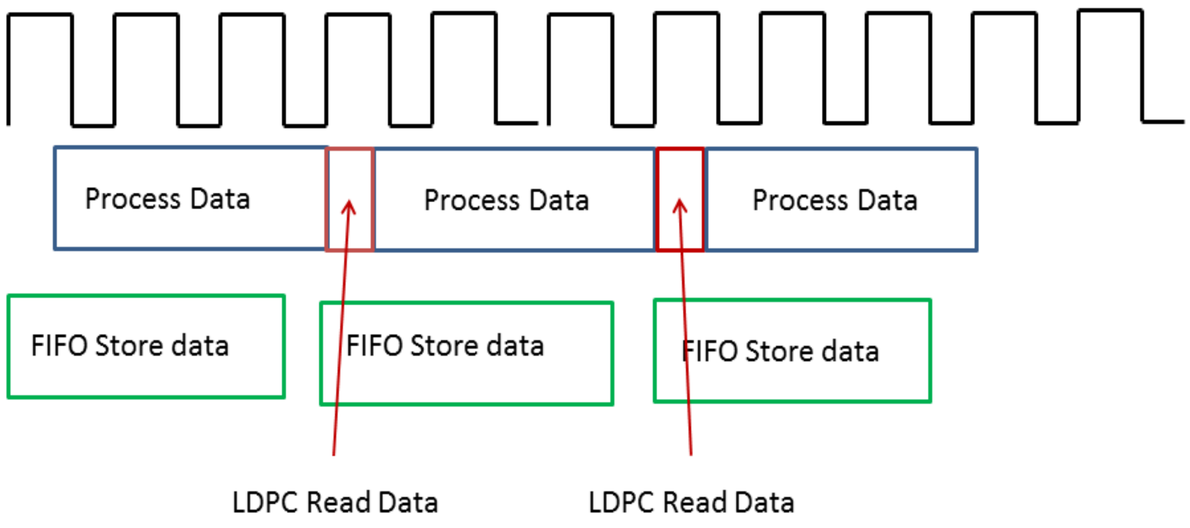


Figure 4.9: FIFO depth 1152 and LDPC one module

- **FIFO with Depth with 1152 and aspect ratio 1:8**

- Available clock for processing data = (240+192+384*2 + 192*2 = 1584 cycle)
- no. of clock cycle required for 1 iteration = 176
- no. of iteration = [(1584-208)/176] = 8

4.8.1 Extra Hardware utilization with FIFO

We did highly approximation of hardware utilization with FIFO. If FIFO depth aspect ration will increase, hardware utilization will increase.So you can see in table 4.2 that hardware increases with depth and aspect ratio.So for depth 768 aspect ratio 1:2 is good for providing 5 iteration with less hardware. Same for 1152, aspect ratio 1:4 is providing sufficient time to LDPC to process 7 iteration .According to availability of resources and requirement, we can make a choice between FIFO depth and Aspect ration. Table 4.2 will help to make such decision between these combination of FIFO depth and Aspect ratio.

Table 4.2: Hardware utilization with FIFO

FIFO depth	Aspect ratio	Iteration	Extra Hardware(Luts)
384	1:1	2	61
768	1:2	5	81
768	1:4	6	83
1152	1:2	5	92
1152	1:4	7	101
1152	1:8	8	110

CHAPTER 5

Implementation Result

This chapter presents implementation result of LDPC decoder for P=8,P=32 and P=96. It will present graphical analysis between speed,area,BER performance.

5.1 Synthesis result of LDPC decoder

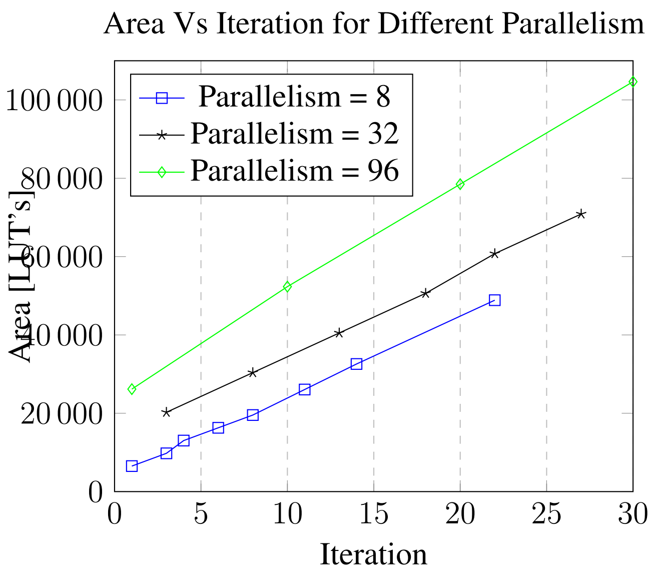
Hardware implementation of LDPC decoder was done in verilog and synthesis is also done in Xilinx Ise tool. Virtex 7 board is used for implementation platform. Chipscope is used for debugging the error in implementation. The following table consists of synthesis information about LDPC decoder for different parallelism. You can see the effect of parallelism on resources. Resources are increasing with parallelism.

Table 5.1: Synthesis Result.

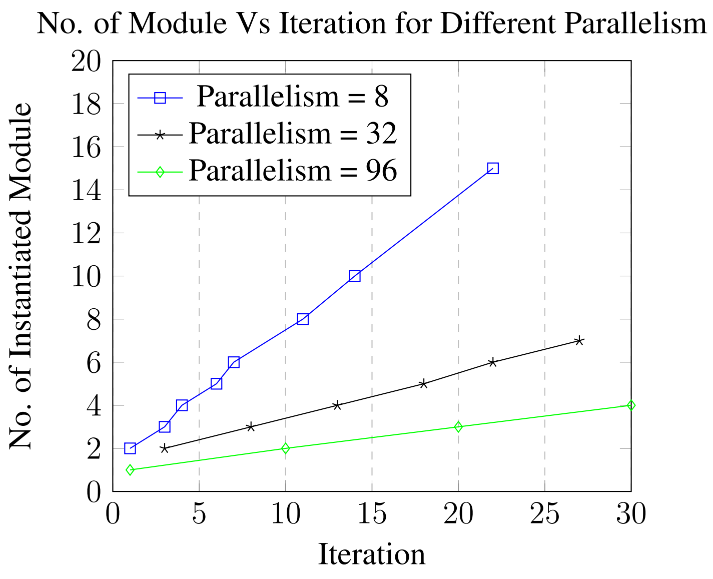
Parallelism	P=8	P=32	P=96
No. of slice register	1057	3314	5365
No. of slice LUTs	3258	10125	26164
No. of fully used LUT-FF pair	1153	2969	4437
No. of Block RAM	18	21	48

5.2 Comparison of Area and iteration

This graph will show a comparison between Area and iteration. It is clear from the graph that Area is varying linearly with iteration. It is possible to get more iteration with LDPC P=96 but it is taking more area. For getting required no. of iteration, LDPC P=8 will take more no. of module which is not a good practice with OFDM receiver to instantiate so many module to get desire performance.

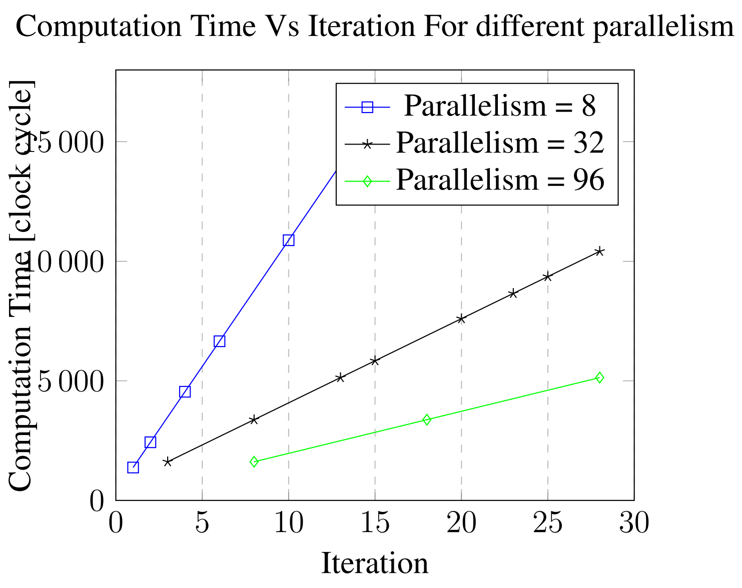


This graph will show relation between no. of module required for desired iteration.For P=8 ,more no. of module is required to get the reasonable amount of iteration which is not good practice for implementation. For P=32, it gives reasonable iteration with 3 or 4 module instantiation.



5.3 Comparison of Computation time and Iteration

This graph shows relation between time and iteration for different parallelism. Computation is varying linearly with iteration. The slope of the line for P=96 is very less than other parallelism. So it is quite clear P=96 takes very less time for more no. of iteration. If we want high performance of LDPC decoder,it is better to choose LDPC P=96. It is also suitable for OFDM receiver module.



CHAPTER 6

Conclusion

The aim of this project is to implement multiple rate LDPC decoder with parallelism equal to expansion factor. LDPC decoder is a part of OFDM receiver chain. For OFDM receiver, it is required to process sufficient iteration within available process time. So LDPC with parallelism equal to expansion factor, is able to process sufficient no. of iteration with 2 module of LDPC. For processing high bit rates, OFDM requires that LDPC should process data with different code rate. LDPC architecture implemented for code rate 1/2,2/3,3/4 and 5/6. It can process with any of these code rate. With parallelism equal to expansion factor, LDPC occupied a lot of resources so by optimization of resources, consumption of Luts is reduced by 7000 Luts. To reduces more consumption of resources, FIFO is used for storing the incoming data. With FIFO depth 1152, it is possible to do 8 iteration with single module of LDPC. Only one module of LDPC with parallelism equal to expansion factor is able to do 8 iteration with OFDM receiver chain. LDPC decoder architecture for parallelism P=96, is implemented on FPGA virtex7 with OFDM receiver module successfully.

CHAPTER 7

REFERENCES

[1] R. Gallager. **Low-Density Parity-Check Codes**, PhD thesis, MIT, 1963.

[2] Rovini, M. Rossi, F. Ciao, P. L'Insalata, N. Fanucci, **Layered Decoding of Non-Layered LDPC Codes** , 9th EUROMICRO Conference on Digital System De-sign: Architectures, Methods and Tools, 2006 Page(s):537 - 544.

[3] M. Cocco, J. Dielissen, M. Heijligers, A. Hekstra, and J. Huisken, **A scalable architecture for LDPC decoding**,in IEEE proceeding of DATE,2004.

[4] Gunnam, K. K. Choi, G. S. Yeary, M. B. Atiquzzaman, **VLSI Architectures for Layered Decoding for Irregular LDPC Codes of WiMax**, IEEE International Conference on Communications, 24-28 June 2007 Page(s):4542 - 4547

[5] Kazunori Shimizu, Nozomu Togawa, Takeshi Ikenaga,**Low Power LDPC Code Decoder Architecture Based on Intermediate Message Compression Technique**, IE-ICE Trans. Fundamentals, Vol.E91-A,No-4 April 2008