A

Project Report

On

"DESIGN OF TURBO ENCODER FOR IN-VEHICLE SYSTEM"

A Dissertation Submitted in Partial Fulfillment of the Requirement for the Award of Bachelor Degree in

ELECTRONICS AND COMMUNICATION ENGINEERING

Submitted By

MIRYALA SIDDARTHA	**20K91A04B4**
MUTHINENI SAI JAGADEESH	**20K91A04C0**
NALLAGATLA NIVAS	**20K91A04C5**

Under the guidance of

Ms. G. ANITHA CHOWDARY

Associate Professor, ECE Dept.

2023-2024

TKR COLLEGE OF ENGINEERING AND TECHNOLOGY

AUTONOMOUS

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

(Accredited By NBA, Approved by AICTE)

Accredited by NAAC with **"A+"** grade

(MEDBOWLI, MEERPET, SAROORNAGAR, HYDERABAD-500097)

TKR COLLEGE OF ENGINEERING AND TECHNOLOGY
(Sponsored by TKR Educational Society, Approved by AICTE, Affiliated by JNTUH)
Autonomous, Accredited by NAAC with 'A' Grade. Accredited by NBA
Medbowli, Meerpet, Saroornagar, Hyderabad - 500 097
Phone: 9100377790, e-mail: info@tkrcet.ac.in website: www.tkrcet.ac.in

College Code : K9

CERTIFICATE

This is to certify that the Project Report entitled **"DESIGN OF TURBO ENCODER FOR IN-VEHICLE SYSTEM"** submitted by

MIRYALA SIDDARTHA	**20K91A04B4**
MUTHINENI SAI JAGADEESH	**20K91A04C0**
NALLAGATLA NIVAS	**20K91A04C5**

is a bonafide work carried out by them during the academic year 2023-2024 under the guidance and supervision of

Ms. G. ANITHA CHOWDARY
Associate Professor
Internal Guide

Co-Ordinator

Dr. M. MAHESH
Head of the Department
ECE

EXTERNAL EXAMINER

TKR COLLEGE OF ENGINEERING AND TECHNOLOGY
(Sponsored by TKR Educational Society, Approved by AICTE, Affiliated by JNTUH)
Autonomous, Accredited by NAAC with 'A' Grade. Accredited by NBA
Medbowli, Meerpet, Saroornagar, Hyderabad - 500 097
Phone: 9100377790, e-mail: info@tkrcet.ac.in website: www.tkrcet.ac.in

College Code : K9

PLAGIARISM REPORT

This is to certify that the Project Report entitled **"DESIGN OF TURBO ENCODER FOR IN-VEHICLE SYSTEM"** submitted by

1. MIRYALA SIDDARTHA 20K91A04B4
2. MUTHINENI SAI JAGADEESH 20K91A04C0
3. NALLAGATLA NIVAS 20K91A04C5

Is Checked for Plagiarism and similarity obtained is__27%

Ms. G. ANITHA CHOWDARY
Associate Professor
Internal Guide

Co-Ordinator

Dr. M. MAHESH
Head of the Department
ECE

PLAGIARISM DOCUMENT

DrillBit

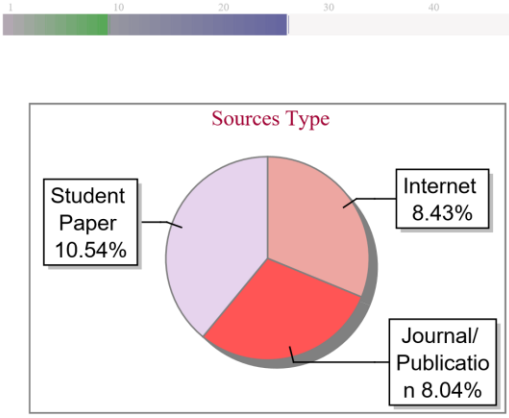
The Report is Generated by DrillBit Plagiarism Detection Software

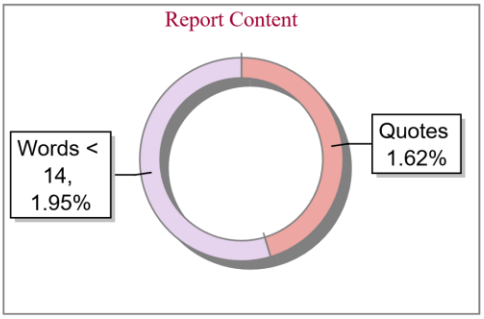
Submission Information

Author Name	ECE IV-C BATCH NO 10
Title	DESIGN OF TURBO ENCODER FOR IN-VEHICLE SYSYTEME
Paper/Submission ID	1623274
Submitted by	COE@TKRCET.AC.IN
Submission Date	2024-04-10 13:31:03
Total Pages	65
Document type	Project Work

Result Information

Similarity **27 %**





Exclude Information

Quotes	Not Excluded	Language	English
References/Bibliography	Not Excluded	Student Papers	Yes
Sources: Less than 14 Words %	Not Excluded	Journals & publishers	Yes
Excluded Source	**0 %**	Internet or Web	Yes
Excluded Phrases	Not Excluded	Institution Repository	Yes

Database Selection

A Unique QR Code use to View/Download/Share Pdf File

ACKNOWLEDGEMENT

Any attempt at any level can't be satisfied completely without the report and guidance of learned people. These words are not enough to show our gratitude towards them. We would like to express our token of thanks to them.

We would like to express our gratitude to **Ms. G. ANITHA CHOWDARY,** Associate Professor, ECE for guiding and correcting various documents with a lot of attention and care.

We owe our gratitude to our Coordinators **Dr. P. VENKATA LAVANYA,** Associate Professor, Department of ECE, **Dr. P. GAYATHRI,** Associate Professor, Department of ECE, and **Ms. CH. DIVYA,** Assistant Professor, Department of ECE, who took keen interest on our project and guided us all along, till the completion of our project by providing all the necessary information for developing a good project.

We would like to convey our sincere thanks to **Dr. M. MAHESH**, HOD, ECE department for his support and encouragement in completion of Our project.

We express our sincere gratitude to the Principal **Dr. D.V. RAVI SHANKAR** for the conductive environment created by him in the college for effective completion of the project undertaken by us.

We would also like to thank our faculty members without whom this project would have been a distant reality.

DECLARATION

We here by declare that the project **"DESIGN OF TURBO ENCODER FOR IN-VEHICLE SYSTEM"** is original and bonafide work carried out by us for the award of degree of **BACHELOR OF TECHNOLOGY** under the guidanceof **Ms. G. ANITHA CHOWDARY,** Associate Professor.

Submitted By

MIRYALA SIDDARTHA **(20K91A04B4)**

MUTHINENI SAI JAGADEESH **(20K91A04C0)**

NALLAGATLA NIVAS **(20K91A04C5)**

ABSTRACT

This project study design of the Turbo encoder in the in-vehicle system (IVS). Developing the parallel computation method using carry increment adder, it is shown that both chip size and processing time are improve. The logic utilization is enhance by reduce area. The Turbo encoder module designing, simulating, and synthesing using Xilinx tools. Xilinx vertex low power is employ. The Turbo encoder module design to be a part of the IVS chip on a single programmable device.

The EU eCall system, mandated since March 2018, facilitates immediate communication between vehicles and emergency centers post-accident. Key components include the in-vehicle system (IVS), public safety answering point (PSAP), and cellular communication channel. The IVS, equipping with a Turbo encoder, automatically activates a data channel upon a collision, sending crucial data like GPS coordinates and VIN number to the nearest PSAP within 4 seconds. The Turbo encoder, employing a parallel concatenated convolutional code (PCCC), uses two constituent encoders with eight states and a 1/3 code rate to enhance digital communication reliability. The encoded data structure involves 1148 input bits, generating 3456 output bits with the influence of the Turbo encoder's thrills structure. The PCCC incorporates a 3GPP-designed interleaver technique for improved performance.

Language Used: Verilog

Tool Used: Xilinx

INDEX

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 Introduction

In 1948, Claude Shannon established a fundamental principle in information theory, demonstrating that every noisy communication channel has a maximum data transmission rate, known as the Shannon limit. He also proposed that error-correcting codes, if unbounded in length, could approach this limit. Over the ensuing six decades, coding theorists relentlessly pursued practical codes capable of nearing this theoretical limit. Among these efforts, turbo codes emerged as a groundbreaking advancement in the 1990s. Introduced by Berrou in 1993, turbo coding introduced a concatenated forward error correction (FEC) scheme that revolutionized channel coding. By leveraging iterative decoding mechanisms, recursive systematic encoders, and large interleavers, turbo codes showcased remarkable error correction performance, often outperforming previous coding techniques. Their efficacy was particularly notable in the realm of 3G standards, where turbo codes became a cornerstone technology.

Utilizing turbo codes significantly enhances the efficiency of data transmission within digital communication systems. The development of turbo codes has been driven by theoretical investigations into various aspects, including polynomial selection, interleaver designs impacting weight distributions, decoder error floors, and thresholds for iterative decoding. A standardized family of turbo codes has been established and is currently employed by numerous spacecraft. JPL's LDPC codes, constructed from protographs and circulants, also contribute to this advancement. Turbo codes facilitate reliable communication over channels with limited power, approaching the theoretical limit proposed by Shannon. However, achieving this high level of reliability often demands a substantial number of iterations, leading to increased latency. Balancing the need for error correction with real-time constraints remains a focal point of ongoing research in the implementation of turbo codes.

The process of designing a channel code inherently involves striking a delicate balance between energy efficiency and bandwidth efficiency. Codes with lower rates, meaning higher redundancy, typically possess greater error correction capabilities. This enhanced error correction capacity enables communication systems to function with reduced transmit power, facilitating transmission over extended distances, greater tolerance to interference, utilization of smaller antennas, and the ability to transmit at higher data rates. These attributes collectively contribute to the energy efficiency of the code. However, it's important to note that lower-rate codes also entail a larger overhead, thereby consuming more bandwidth. This tradeoff highlights the intricate relationship between energy efficiency and bandwidth efficiency in the design of channel codes.

Decoding complexity poses a significant challenge in channel coding, as it escalates exponentially with the length of the code. Long codes, particularly those with lower rates, impose substantial computational demands on conventional decoders. As highlighted by Viterbi, this discrepancy between the ease of encoding and the difficulty of decoding constitutes a central issue in channel coding. The European eCall system, mandated for implementation by March 2018, serves as a crucial telematics solution aimed at enhancing emergency response in vehicle accidents. This governmental initiative ensures the establishment of an immediate voice and data channel between vehicles and emergency centers post-accident. Comprising essential components such as the in-vehicle system (IVS),

1.2 In Vehicle System (IVS):

Public safety answering point (PSAP), and cellular communication channel, the eCall system operates seamlessly to facilitate rapid response in critical situations. Upon detecting a vehicular accident, the IVS automatically triggers the data channel, swiftly collecting vital information known as the minimum set of data (MSD). This includes GPS coordinates, vehicle identification number (VIN), and other pertinent details necessary for prompt emergency assistance. Within a mere 4 seconds, the IVS transmits the MSD via cellular communication to the nearest PSAP. Subsequently, the PSAP coordinates the dispatch of emergency teams to the accident location, ensuring swift and effective response to mitigate potential harm.

Within the IVS modem, multiple modules are dedicated to processing the MSD signal. This intricate system, illustrated in Figure 1, is designed to optimize the handling of data transmissions. Notably, a Turbo encoder serves as a crucial component, functioning as a forward error correcting (FEC) mechanism. Leveraging advanced digital data encoding techniques, the Turbo encoder plays a pivotal role in enhancing the bit error rate (BER) within digital communications. Among the various modules integrated into the IVS, several key functions are encapsulated within the IVS chip itself. These include the cyclic redundancy check (CRC), modulator, and demodulator-decoder modules, all meticulously crafted to operate seamlessly within the IVS architecture. To ensure efficiency and reliability, these modules are meticulously projected and implemented on a dedicated FPGA device, offering robust embedded solutions for the IVS's signal processing requirements.

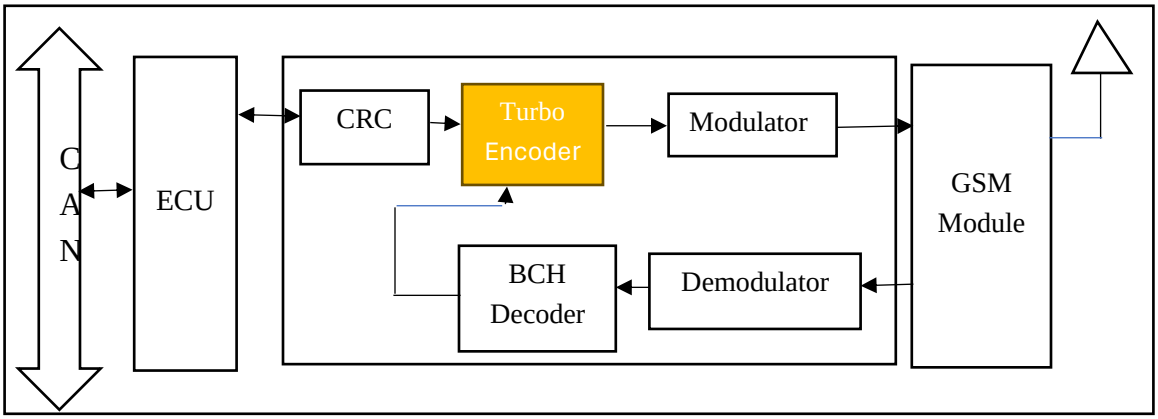


Fig 1.2.1 IVS Block Diagram

This research delves into the hardware evolution of the Turbo encoder, focusing on its integration as a fundamental component within the IVS modem. Employing cutting-edge FPGA technologies, the study intricately explores the development process, emphasizing novel methodologies to enhance Turbo encoder functionality. Central to the investigation are discussions surrounding serial and parallel computation techniques tailored specifically for the Turbo encoder. Beyond mere design and implementation, the research presents innovative solutions aimed at optimizing Turbo encoder performance. Notably, considerable attention is devoted to addressing challenges related to chip size and processing time, with particular emphasis on leveraging parallel computation techniques

to achieve significant improvements. Through meticulous experimentation and analysis, the research underscores the tangible benefits derived from the proposed advancements in Turbo encoder implementation. This includes notable enhancements in chip size efficiency and processing speed, showcasing the potential for transformative breakthroughs in hardware development for digital communication systems.

1.3LITIRATURE SURVEY

The evolution of decoding systems prompted a deeper investigation into the efficacy of turbo codes. This segment delves into the structural intricacies of turbo codes, tracing their development, the evolution of understanding, and subsequent enhancements in turbo code design. The genesis of turbo encoding can be attributed to seminal works by [JOE94] and later, [ROB94]. These scholars underscored the critical importance of trellis termination, a facet initially overlooked. Unlike non-systematic convolutional codes, which could be reset to an all-zero state with mere zeros, it became apparent that Recursive Systematic Convolutional (RSC) codes necessitated distinct termination bits appended based on the encoder's final state. While the optimal approach would terminate both component trellises, the presence of an interleaver within the turbo encoder complicated matters. Tail bits appended to the data stream to terminate the first encoder trellis at an all-zero state were unlikely to effectively terminate the second trellis. In addressing this conundrum, [ROB94] proposed a practical albeit sub-optimal solution: terminating the first component encoder to ensure the trellis concluded at its starting state.

The paper titled "eCall Data Transfer: In-Band Modem Solution; General Description," authored by 3GPP, presents the design and implementation of the in-vehicle system (IVS) for the European Union's emergency call (eCall) system. The IVS modules are meticulously developed and realized on a field-programmable gate array (FPGA) device. Through simulation, synthesis, and optimization, these modules are tailored to be loaded onto a reconfigurable device, effectively creating a system-on-chip (SoC) for the IVS electronic device. Benchtop testing is conducted to validate and verify the functionality of the developed modules. The paper also delves into the hardware architecture and interfaces of the IVS, providing insights into its signal

processing time across various frequencies. It proposes a range of suitable frequencies and outlines two hardware interfaces. Employing a state-of-the-art FPGA design as the primary implementation approach for the IVS prototyping platform, this work lays the groundwork for potentially integrating all IVS modules onto a single SoC chip in the future.

In their paper titled "Design and Implementation of CRC Module of eCall In-Vehicle System on FPGA," M. Nader and J. Liu address the pivotal role of the EU emergency call (eCall) system in mitigating vehicular incidents and saving lives. The focus of their work lies in crafting and deploying a CRC module within the in-vehicle system (IVS) of the eCall system using FPGA technology. Highlighting the significance of CRC in detecting bit errors during transmission, the paper meticulously details the hardware design procedures involved in developing the CRC module. The system is engineered to process 1120 serial input bits of the Minimum Set of Data (MSD), compute 28 bits of CRC parity, and produce an output signal comprising 1148 serial bits, consisting of the MSD appended with CRC. Utilizing Verilog HDL, the system undergoes compilation, synthesis, and simulation across various MSD scenarios, with results meticulously analyzed. The paper also provides a comprehensive illustration and discussion of the implemented algorithm through a flowchart. Extensive testing and verification are conducted across different frequencies to ascertain the design's applicability range. Notably, the authors observe a correlation between clock frequency and signal distortion, with higher frequencies leading to increased distortion. The paper scrutinizes the signals generated at clock frequencies of 50 kHz, 5 MHz, and 10 MHz, offering insights into their characteristics. Furthermore, the performance of the module is evaluated through a comparison of simulated MSDs and FPGA-generated signals across multiple scenarios. Additionally, the authors employ a CRC module developed in C code for IVS to further validate the module's performance, enhancing the robustness of their analysis.

The document titled "Technical Specification Group Radio Access Network; Multiplexing and channel coding (FDD)," authored by 3GPP and identified as Tech. Rep. TS22.212, delves into the creation and integration of a Turbo encoder as an embedded module within the in-vehicle system (IVS) chip. Utilizing a field-

programmable gate array (FPGA), the Turbo encoder module is meticulously crafted, exploring both serial and parallel computation methods for encoding. The paper meticulously compares and contrasts the two design approaches, highlighting the advantages of parallel computation. Through the development of the parallel computation method, significant improvements in both chip size and processing time are demonstrated. Notably, the logic utilization is enhanced by 73%, while the processing time sees a remarkable reduction of 58%. The Turbo encoder module undergoes comprehensive design, simulation, and synthesis using Xilinx tools, with a particular focus on employing the Xilinx Zynq-7000 FPGA device for implementation. The ultimate aim is to seamlessly integrate the Turbo encoder module into the IVS chip, consolidating its functionalities onto a single programmable device. This endeavor represents a concerted effort towards enhancing the efficiency and effectiveness of the IVS system, leveraging cutting-edge FPGA technology for optimal performance.

1.4 Interlever Design

The concept of designing the "optimal" interleaver for turbo codes has been explored extensively in the literature, with varying degrees of complexity. Early works, such as those by [BER93b] and [JUN94], initially relied on trial and error approaches to devise interleavers, which served as valuable starting points in demonstrating potential improvements achievable through interleaving.

Subsequent research, exemplified by [HOS00] and [DAN99], delved deeper into interleaver design, presenting more intricate methodologies. While some researchers focused on defining specific interleaver designs, others elucidated systems capable of generating high-quality interleavers.

For instance, [DIV95a] introduced the concept of "S-random" interleavers, where S represents half the size of the interleaver (S = N / 2, with N being the interleaver size). This method involves selecting random integer values within the interleaver size while ensuring that each chosen integer is sufficiently distant from previously selected ones. Specifically, if a chosen integer falls within a certain proximity (+/- S positions) of a

previously selected integer, it is discarded, and the selection process continues until all integers have been chosen.

Overall, interleaver design for turbo codes has evolved from rudimentary trial and error approaches to more sophisticated methodologies, with researchers continuously striving to optimize interleaving techniques to enhance the performance of turbo codes in various applications.

In the realm of turbo code interleaving, various innovative techniques have been proposed to enhance performance and mitigate error floors. [YUA99] introduced the concept of the "Code Matched Interleaver" (CMI), which optimizes interleaver design by analyzing the weight spectrum of lower weight codewords. By identifying patterns that contribute significantly to error probability at high signal-to-noise ratios, the interleaver is tailored to eliminate these patterns post-interleaving. Building upon this, [BYU99] developed the "swap interleaver" as an enhancement to the S-random interleaver proposed by [DIV95a]. This method involves iteratively swapping random positions within the interleaver and ensuring compliance with a specified S value, resulting in improved efficiency and performance, particularly for large frame sizes.

Additionally, [HO98b] addressed the challenges of punctured turbo codes, which often suffer from uneven parity bit protection. To address this issue, they introduced the mod-k interleaver, an extension of the odd-even interleaver described previously. By permuting data bits to ensure each is paired with a parity bit after puncturing, the mod-k interleaver enhances performance. Moreover, [HO98b] proposed symmetric interleavers to mitigate the need for separate de-interleavers, thereby optimizing storage utilization. Experimental results demonstrated the efficacy of mod-k interleavers, especially when combined with symmetric interleaving techniques, showcasing significant performance improvements compared to traditional interleaver designs.

In summary, these advancements in interleaver design offer promising avenues for improving turbo code performance, reducing error floors, and optimizing resource utilization in communication systems.

1.5 Existed Methodology

In this module there are implementation of the Turbo encoder to be an embedded module withinside the in-automobile system (IVS) chip. Field programmable gate array (FPGA) is hired to increase the Turbo encoder module. Both serial and parallel computations for the encoding technique are studied. The layout techniques are provided and analyzed. Developing the parallel computation method, it's miles proven that each chip length and processing time are improved. The logic utilization is enhanced by 73% and the processing time is reduced by 58%. The Turbo encoder module is designed, simulated, and synthesized the use of Xilinx tools. Xilinx Zynq-7000 is hired as an FPGA tool to enforce the advanced module. The Turbo encoder module is designed to be part of the IVS chip on a unmarried programmable device.

1.6 Proposed Methodology

This module focuses on designing a Turbo encoder for use in in-vehicle systems (IVS). By implementing a parallel computation method using a carry increment adder, both chip size and processing time are improved, enhancing logic utilization and reducing area. The Turbo encoder is designed, simulated, and synthesized using Xilinx tools, specifically targeting the Xilinx Vertex low power platform. This Turbo encoder module is intended to be integrated into the IVS chip on a single programmable device.

The Turbo encoder plays a vital role in the EU eCall system, mandated since March 2018, facilitating immediate communication between vehicles and emergency centers post-accident. It automatically activates a data channel upon collision, transmitting crucial data such as GPS coordinates and VIN number to the nearest public safety answering point (PSAP) within 4 seconds. Employing a parallel concatenated convolutional code (PCCC) with two constituent encoders featuring eight states and a 1/3 code rate, the Turbo encoder significantly enhances digital communication reliability. The encoded data structure involves 1148 input bits, generating 3456 output bits, utilizing a thrills structure. Additionally, the PCCC incorporates a 3GPP-designed interleaver technique for improved performance.

CHAPTER 2

TURBO ENCODER MODULE

2.1 Introduction:

The Turbo encoding method stands out as a robust forward error correction (FEC) technique in the realm of digital communication. Integrated within the In-Vehicle System (IVS), Turbo encoder modules are pivotal components operating at a 1:3 code rate. These modules adhere to the specifications outlined in the third generation partnership project (3GPP) standards, offering a standardized framework. Visualized in Figure 1, the Turbo encoder's input signal comprises Main System Data (MSD) appended with CRC parity bits, all represented in binary format. The MSD data, with a block length of 1148 bits, undergoes encoding, resulting in a binary output of MSD-encoded data. Notably, when implementing Turbo coding with a 1:3 coding rate, the output length extends to 3456 bits. This expansion is influenced by the structure of the Turbo encoder, particularly the incorporation of "thrills bits. Operating on a parallel concatenated convolution code (PCCC) scheme, the Turbo encoder integrates two constituent encoders, each featuring eight states, as depicted.

Initializing with zeros, the encoder's registers prime for operation. The first constituent encoder processes MSD bits, employing a convolutional technique to generate parity bits, working on a bit-by-bit basis. In parallel, the second constituent encoder employs a similar methodology but introduces an interleaving step using a 3GPP-designed technique before processing the MSD bits. In summary, the Turbo encoder within the IVS employs a sophisticated coding strategy, leveraging parallel concatenated convolution

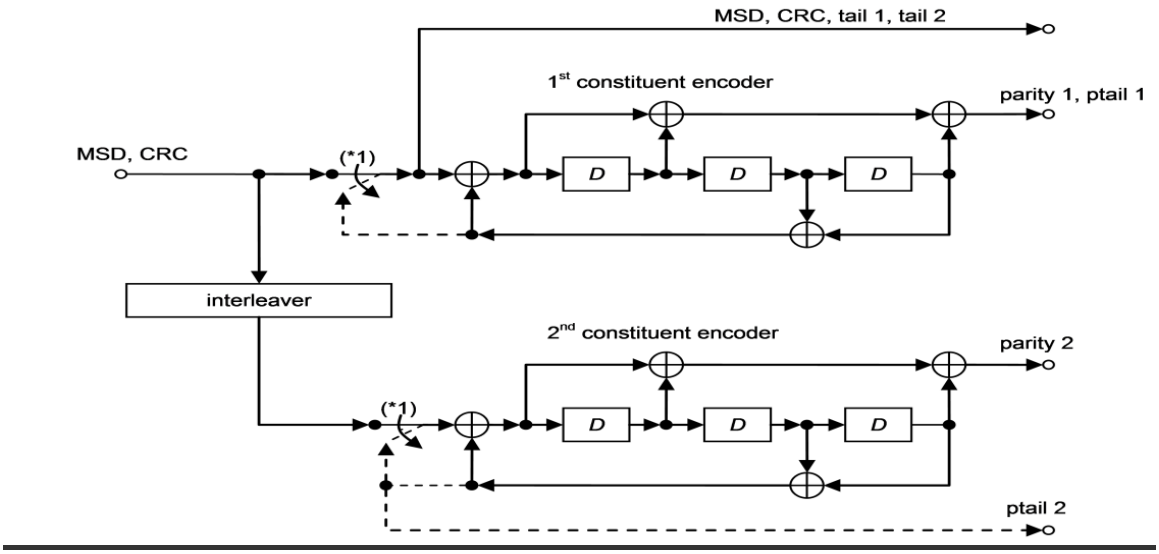


Fig 2.1.1 The structure of the Turbo encoder

The Turbo encoder outputs encoded data with a structure that includes tail bits, which are derived from feedback in the shift register. These tail bits consist of 12 bits and are applied at the endpoints between the encoded data blocks. In total, the input data, along with the two sets of parity bits (parity1 and parity2), spans 1148 bits.

MSD+CRC	$tail_1$	$tail_2$	Parity 1	$ptail_1$	Parity 2	$ptail_2$

Fig 2.1.2 The output buffer of the Turbo encoder

2.2 Interleaver

Denote the transfer function of the employed PCCC as:

$$G(D) = \left(1, \frac{g_1(D)}{g_0(D)}\right) \qquad (1)$$

where

$$g_1(D) = 1 + D^2 + D^3$$

$$g_0(D) = 1 + D + D^3$$

The Turbo encoder takes a sequence of input bits and processes them through two constituent encoders, resulting in three sets of output bits. These output bits are organized into code blocks, denoted as X 1 , X 2, and X 3, where K represents the number of input bits.

The encoder output is expressed as:

$$d_K^{(0)} = x_K, d_K^{(1)} = z_K, d_K^{(2)} = z'_K$$

where $K = 0, 1, ..., K - 1.$

The code blocks $d_K^{(0)}$, $d_K^{(1)}$, and $d_K^{(2)}$ are separated by trellis bits, which are derived from the tail bits of the shift registers after encoding all input bits. Specifically, when the upper switch is lowered and the second constituent is inactive, the termination of the first constituent is guided by three tail bits. The resulting output bits of the Turbo encoder, including the trellis bits, can be represented as

$$d_K^{(0)} = x_K, \quad d_{K+1}^{(0)} = z_{K+1}, \quad d_{K+2}^{(0)} = x'_K, \quad d_{K+3}^{(0)} = z'_{K+1}$$

$$d_K^{(1)} = z_K, \quad d_{K+1}^{(1)} = x_{K+2}, \quad d_{K+2}^{(1)} = z'_K, \quad d_{K+3}^{(1)} = x'_{K+2}$$

$$d_K^{(2)} = x_{K+1}, d_{K+1}^{(2)} = z_{K+2}, d_{K+2}^{(2)} = x'_{K+1}, d_{K+3}^{(2)} = z'_{K+2}$$

where $K = 0, 1, ..., K - 1.$

The internal interleaver of the 3GPP Turbo encoder is designed to generate a systematic relationship between x_K and x'_K for any $40 \leq K \leq 5114$. There is a specific approach to design an internal interleaver for the employed Turbo encoder that is detailed in. This work employs the 3GPP standard approach to design the internal interleaver for the employed Turbo encoder.

First, the input bits of the Turbo encoder is re-arranged in a matrix form that has column, C, and row, R. The rows are labeled as 0,1,…,R - 1 and the columns are organized as 0,1,…,C-1. The numbers of rows and columns are determined according to the 3GPP standard for Turbo encoder interleavers. Then the input bits $x_1, x_2, ..., x_K$ are re-

organized in a matrix where $y_k = x_k$ for k = 1,2,…,K and $y_k = 0$ for the elements that $R \times C > K$:

$$\begin{bmatrix} y_1 & y_2 & y_3 & \cdots & y_C \\ y_{(C+1)} & y_{(C+2)} & y_{(C+3)} & \cdots & y_{(C+C)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ y_{((R-1)C+1)} & y_{((R-1)C+2)} & y_{((R-1)C+3)} & \cdots & y_{(R\times C)} \end{bmatrix}$$

Then an intra-row and inter-row permutation is performedon the R X C matrix. This work employs the 3GPP standard approaches for the intra-row and inter-row.

Denote,

$$s(j) = (v \times s(j-1)) \mod p \qquad (2)$$

Where $\langle s(j) \rangle$ for $j \in 1, 2, ..., p-2$ and p(0) = 0 is the intarow permutation sequence and v is associated primitive root for the specified p from table ??, and use table I to choose the appropriate pattern to compute the inter-row permutation, T(i) for $i \in 0, 1, ..., R-1$. i is the row number index of R X C matrix, and j is the column number index of the matrix. Also the minimum prime integer (qi) is determined in the sequence q(i) for $i \in 0, 1, ..., R-1$ such that qi > q(i-1), qi > 6 and g.c.d(qi, p - 1) = 1, where g.c.d is the greater common divisor. Then one can build a sequence of the permuted prime integers D r(i) for $i \in 0, 1, ..., R-1$ such that,

Table 2.2.1: 3GPP inter-row permutation pattern

K	*R*	Inter-row permutation patterns <*T*(0), *T*(1), …, *T*(*R* - 1)>
(40 ≤ *K* ≤ 159)	5	<4, 3, 2, 1, 0>
(160 ≤ *K* ≤ 200) or (481 ≤ *K* ≤ 530)	10	<9, 8, 7, 6, 5, 4, 3, 2, 1, 0>
(2281 ≤ *K* ≤ 2480) or (3161 ≤ *K* ≤ 3210)	20	<19, 9, 14, 4, 0, 2, 5, 7, 12, 18, 16, 13, 17, 15, 3, 1, 6, 11, 8, 10>
K = any other value	20	<19, 9, 14, 4, 0, 2, 5, 7, 12, 18, 10, 8, 13, 17, 3, 1, 16, 6, 15, 11>

Denote the pattern of i _ th row Intra-row permutation as,

$$U_i(j) = s((j \times r_i) \mod (p-1)) \quad (3)$$

where $j = 0, 1, ..., (p-1)$ and $U_i(p-1) = 0$.
if $(C = p+1)$ then,

$$U_i(j) = s((j \times r_i) \mod (p-1)) \quad (4)$$

where $j = 0, 1, ..., (p-1)$, $U_i(p-1) = 0$, and $U_i(p) = p$.
if $(C = p-1)$ then,

$$U_i(j) = s((j \times r_i) \mod (p-1)) - 1 \quad (5)$$

where $j = 0, 1, ..., (p-1)$.

And then the inter-row permutation is implemented on the R X C matrix by using the sequence pattern T(i) for i = 0,1,2…, R-1. After the permutations, the elements of the R X C matrix is denoted by $y'_k = y_k$ such that:

$$\begin{bmatrix} y'_1 & y'_{(R+1)} & y'_{(2R+1)} & \cdots & y'_{((C-1)R+1)} \\ y'_2 & y'_{(R+1)} & y'_{(2R+2)} & \cdots & y'_{((C-1)R+2)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ y'_R & y'_{(2R)} & y'_{(3R)} & \cdots & y'_{(R\times C)} \end{bmatrix}$$

The interleaver takes a sequence of K bits, denoted as 1, 2, … , B 1 ,B 2 ,…,B K , where = 1148 K=1148, and reorganizes them systematically. This reorganization is achieved through an interleaver matrix, which is a rectangular matrix with R rows and C columns. For this specific case, the interleaver matrix size is 20 × 58 20×58, designed to handle an input data block of 1148 bits. The interleaver operates by reading out bits from the interleaver matrix column by column, starting with the top left element and ending with the bottom right element. Any appended zero bits, which may occur due to the difference between the total number of bits and the size of the matrix (×R×C), are removed from the output. The interleaver matrix consists of 1148 elements, representing the 1148 bits of the original data block. These bits are rearranged systematically within the matrix using intra-row and inter-row techniques. The systematic order ensures that the bits are efficiently spread out across the matrix, optimizing the interleaving process.

The process of generating the interleaver matrix for the Turbo encoder, adhering to 3GPP standards, involves several steps:

1**. Input Data Preparation**: The input data bits, denoted as 1, 2 , … ,b 1 ,b 2 ,…,b K , where = 1148 K=1148, are organized into a sequence. Any remaining elements beyond K are padded with zeros to ensure a consistent size.

2. **Intra-row and Inter-row Permutation**: The interleaver performs both intra-row and inter-row permutation techniques, as specified by 3GPP standards. These techniques systematically rearrange the input bits within the interleaver matrix.

3.**Hexadecimal File Generation:** The calculated elements of the interleaver matrix, excluding the padded bits, are stored in a file in hexadecimal format. This file serves as the input data for the Turbo encoder, facilitating seamless integration with the Verilog HDL implementation.

The Turbo encoder module is developed in Verilog HDL, operating at the register transfer level (RTL). Multiple registers are defined within the module to handle input, output, and other necessary parameters for Turbo encoding. Two methods of encoding, namely serial computation and parallel computation, are studied to optimize the encoding process.

The Verilog HDL implementation of the Turbo encoder utilizes the hexadecimal file containing the interleaver data, enabling efficient utilization of the interleaver matrix during the encoding process. This integration ensures compatibility and ease of implementation within the Turbo encoder module.

2.3 The serial computation method

The serial computation method within the Turbo encoder module operates on a bit-by-bit basis, processing one bit per clock cycle. This method involves sequentially reading the input data of the most significant digit (MSD) and constructing input and output registers. Subsequently, it calculates parity1, parity2, and the tail bits in a serial fashion. Once the encoding process is complete, the method generates the output bits. Despite its implementation, it's observed that this serial computation method entails a lengthy

processing time. This extended duration may introduce delays that could potentially overlap with other concurrent processes within the module. Depicts the pseudocode outlining the steps involved in the serial computation of the Turbo encoder module. This pseudocode provides a structured representation of the sequential operations carried out during the encoding process, facilitating comprehension and implementation of the serial computation method.

Pseudocode code for Serial Computation of Turbo encoder

```
module TURBO_SERIAL ( inputs, outputs;)
        define REGISERS and PARAMETERS;
      always @(posedge clock, posedge reset)
        begin
         If (reset) output=0;
         else begin
            repeat1 (1148)  {
                        Read MSD input data}
          If (repeat1 is done)
             repeat2 (1148)  {
                        Build output register for MSDinput;
                        Build output register for Parity1;
                        Build output register for Parity2; }
          If (repeat2 is done)
             repeat3 (1148) {
                        Process Parity1 bits;  }
          If (repeat3 is done)
             Repeat4 (3) {
                        Process tail1 bits;
                        Process ptail1 bits;     }
          If (repeat4 is done)
             Repeat5 (1148) {
                        Process Parity2 bits;  }
          If (repeat5 is done)
             Repeat6(3) {
                        Process tail2 bits;
                        Process ptail2 bits;     }
          If (repeat6 is done)
             Repeat7(3456) {
                        Generate output bits     }
           end   end
endmodule;
```

Fig 2.3.1: The pseudo code for serial computation of the Turbo encoder.

The processing time T_s of the Turbo encoder module using the serial computation method can be expressed as the sum of several individual processing times:

T_s = parity1+ tail + T s =T r +T b +T parity1 +T tail +T w

$$T_s = 1148 + 1148 + 1148 + 3 + 1148 + 3 + 3456 = 8054$$

Where:

T r represents the time required to read the 1148 bits of the most significant digit (MSD) data.

T b is the processing time needed to construct the output register.

T parity1 denotes the time taken for processing parity1 bits.

T tail signifies the processing time for tail bits.

T w indicates the time required for generating output bits.

By summing up these individual processing times, the total processing time T s for the serial computation method of the Turbo encoder module is determined. This breakdown allows for a detailed analysis of the time consumption at each stage of the encoding process, aiding in performance optimization and system design.

2.4 The parallel computation method

Utilizing a parallel computing technique in the implementation of the Turbo encoder in Verilog brings substantial reductions in processing time by concurrently executing multiple tasks that were previously performed sequentially in the serial computation method. This parallel approach involves the integration of two functions capable of handling nearly all facets of the encoding process. Unlike its serial counterpart, which processes data bit by bit, the parallel technique capitalizes on the entirety of the MSD data to streamline encoding operations.

The pseudocode provided in outlines the methodology adopted for the parallel technique, facilitating its seamless integration into the Verilog design. Both serial and

parallel pseudocodes are developed in Verilog, ensuring compatibility and ease of incorporation into the overall system.

The processing time of the Turbo encoder module is denoted by Ts for the serial computation method and T p for the parallel technique. These durations are influenced by several factors:

$$T_s = T_r + T_b + T_{parity1} + T_{tail1} + T_{parity2} + T_{tail2} + T_w$$

$$T_s = 1148 + 1148 + 1148 + 3 + 1148 + 3 + 3456 = 8054$$

T r: Time taken to read the 1148 bits of the MSD.

T b: Processing time for constructing the output register.

T parity1: Duration required for processing parity bits.

T tail : Processing time allocated for tail bits.

T w : Time needed for generating output bits.

In the parallel computing approach, the sum of T r +T b +T parity1 +T tail1 +T parity2 +T tail2 is completed within a single clock cycle. This concurrent processing dramatically reduces the overall processing time compared to the sequential serial method.

The adoption of parallel computing enhances the efficiency and throughput of the Turbo encoder, rendering it well-suited for applications requiring high-speed data transmission capabilities.

$$T_p = 1 + T_w = 1 + 3456 = 3457 \qquad (2)$$

Then one has,

$$T_p = 0.42T_s \qquad (3)$$

```
Pseudocode code for Parallel Computation of Turbo encoder
module TURBO_PARALLEL ( inputs, outputs;)
        define REGISERS and PARAMETERS;
 function [3455:0]  codedMSD1;
              input [1147:0] MSDdata;
                 begin
                    repeat1 (1148)  {
                        Build output register for MSDinput;
                        Build output register for Parity1;
                        Build output register for Parity2; }
                  If (repeat1 is done)
                      repeat2 (1148) {
                       call   CodedMSD2 (codedMSD1)  }
                  end
       endfunction
 function [3455:0]  codedMSD2;
               input [3455:0] codedMSD1;
                 begin
                 repeat3 (1148) {
                        Process Parity1 bits;  }
                   If (repeat3 is done)
                     repeat4 (3) {
                        Process tail1 bits;
                        Process ptail1 bits;     }
                If (repeat4 is done)
                    repeat5 (1148) {
                        Process Parity2 bits;  }
               If (repeat5 is done)
                    repeat6(3) {
                        Process tail2 bits;
                        Process ptail2 bits;     }
                  end
       endfunction
   always @(posedge clock, posedge reset)
       begin
        if (reset) output=0;
        else begin
           repeat1 (1148)  {
                      Read MSD input data     }
              If (repeat1 is done) {
                  call   CodedMSD1 (MSDdata)   }
             Repeat7 (3456) {
                        Generate output bits     }
           end  end
endmodule;
```

Fig 2.4.1: The pseudo code for parallel computation of the Turbo encoder.

CHAPTER 3

ADDER DESIGNS

3.1 Adders:

Addition lies at the heart of many computational processes, and in digital circuits, adders play a fundamental role in arithmetic operations. These circuits, often integrated within Arithmetic Logic Units (ALUs), are essential components in processors, responsible for performing addition operations on binary numbers.

Binary representation is the most common format for adders, allowing for straightforward arithmetic operations. While two's complement or ones' complement representation easily lends itself to addition and subtraction, other signed number representations require additional logic to manipulate effectively.

A basic Binary Adder circuit, constructed using AND and XOR gates, facilitates the addition of two single-bit binary numbers, producing both a SUM and a Carry-out (COUT) bit as outputs. This circuit forms the building block for more complex arithmetic and counting circuits, enabling efficient computation of binary numbers.

In binary addition, each column represents a power of two, analogous to the weighted values assigned to each digit in decimal addition. The process of adding binary numbers mirrors that of decimal addition, with carry generation occurring when the sum in any column equals or exceeds the base number (2 in binary).

Just as in decimal addition, where carrying occurs when the sum reaches 10 or greater, binary addition triggers carry generation when the sum reaches 2 or greater. This fundamental principle underpins the operation of binary adders, ensuring accurate computation of binary numbers by managing carry propagation through successive columns.

123	A	
+ 789	B	(Addend)
912	SUM	

In essence, while the mechanics of binary addition may differ from decimal addition, the underlying principles remain the same. Binary adders embody these principles, providing the foundation for efficient arithmetic operations in digital systems.

3.2 2Binary Addition:

In binary addition, the principles remain akin to those of decimal addition, albeit with a simpler digit set consisting solely of "0" and "1". Unlike decimal addition, where carry-over occurs when the sum reaches or exceeds 10, in binary, carry-out is triggered when the sum equals or surpasses two, representing the addition of two ones.

Picture a scenario where we're adding two single-bit binary numbers: when both bits are "1", the sum becomes "10" in binary, necessitating a carry-out to the next column. This carry bit then contributes to the next addition, allowing for seamless propagation of carry-over throughout successive columns.

In essence, binary addition operates on the same fundamental principles as decimal addition, albeit with a simplified digit system. Understanding these principles is crucial for efficiently performing arithmetic operations in binary, ensuring accurate computation across digital systems.

Binary Addition of Two Bits

0	0	1	1
+0	+1	+0	+1
0	1	1 (carry)	1←0

When the two single bits, A and B are added together, the addition of “0 + 0”, “0 + 1” and “1 + 0” results in either a “0” or a “1” until you get to the final column of “1 + 1” then the sum is equal to “2”. But the number two does not exists in binary however, 2 in binary is equal to 10, in other words a zero for the sum plus an extra carry bit.

Then the operation of a simple adder requires two data inputs producing two outputs, the Sum (S) of the equation and a Carry (C) bit as shown.

3.3 n-bit Binary Adder:

When we need to add together two n-bit binary numbers, we employ a construct known as a Ripple Carry Adder. This adder comprises "n" 1-bit full adders interconnected or "cascaded" together, each representing a weighted column in a binary addition operation.

Visualize this adder as a chain of full adders, where each full adder processes one bit of the input numbers. As the addition progresses from the least significant bit (LSB) to the most significant bit (MSB), the carry signals propagate through the adder, akin to ripples spreading from right to left.

Consider adding two 4-bit numbers: the output of the first full adder delivers the least significant bit (LSB) of the sum along with a carry-out bit. This carry-out bit serves as the carry-in for the subsequent full adder in the chain.

Similarly, each subsequent full adder in the cascade produces a bit of the sum while generating another carry-out bit. By connecting the carry-out of each adder to the carry-in of the next, we can seamlessly add larger numbers, extending the chain of full adders as needed.

This interconnected arrangement enables the Ripple Carry Adder to efficiently compute the sum of multi-bit binary numbers, with the ripple effect of carry propagation ensuring accurate addition across all bits.

A 4-bit Ripple Carry Adder

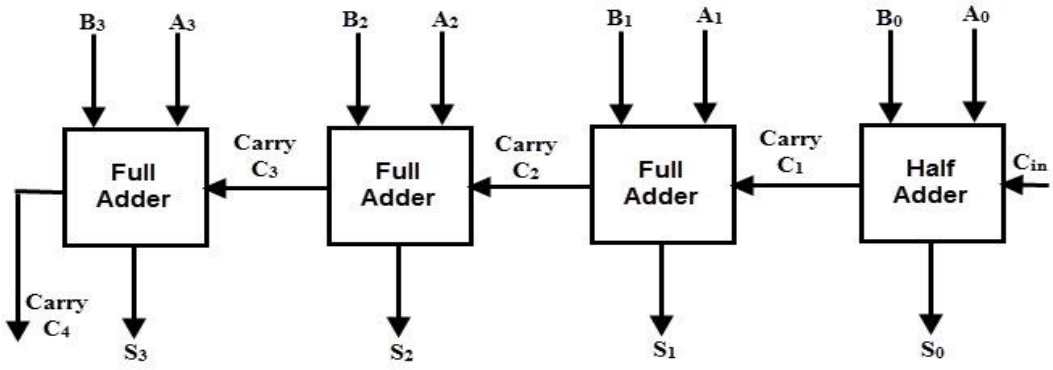
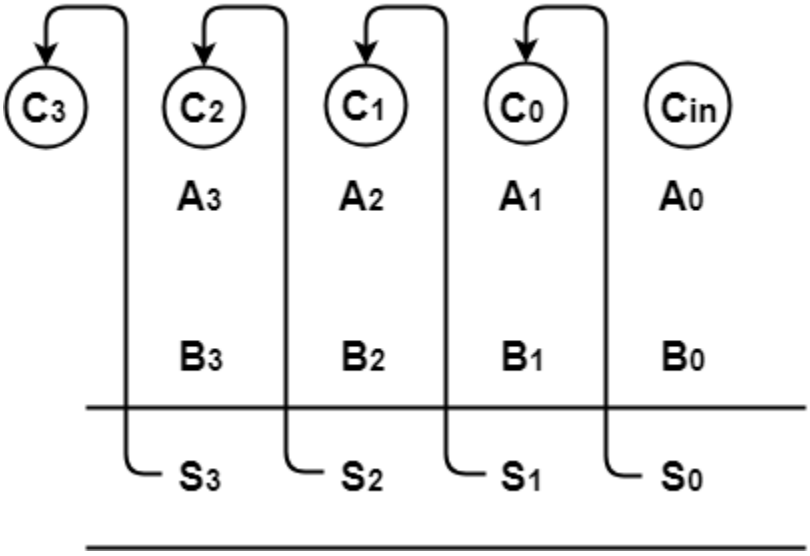


Fig 3.3.1: 4-bit binary parallel adder

- 4-bit ripple carry adder is used for the purpose of adding two 4-bit binary numbers.
- In mathematics, any two 4-bit binary numbers $A_3A_2A_1A_0$ and $B_3B_2B_1B_0$ will be added as-



Indeed, the Ripple Carry Adder operates in distinct stages, with each full adder producing a carry-out signal that serves as the carry-in input for the adjacent, more significant full adder. This sequential progression ensures that each full adder activates in turn as its carry-in becomes available.

The process unfolds as follows: Initially, the least significant full adder receives the input bits and produces a sum along with a carry-out signal. This carry-out signal then becomes the carry-in for the next full adder, prompting its activation. As this cascade continues, each subsequent full adder activates once its carry-in signal is received, allowing for the incremental computation of the binary addition operation.

By employing this staged approach, the Ripple Carry Adder effectively manages the flow of carry signals throughout the adder chain, facilitating accurate addition of multi-bit binary numbers. This systematic activation ensures that each full adder contributes to the overall sum calculation in a synchronized manner, culminating in the final result.

3.4 Working of a 4-bit Ripple Carry Adder-

Suppose we want to add two 4bit binary numbers 0101 ($A_3A_2A_1A_0$) and 1010 ($B_3B_2B_1B_0$). Using ripple carry adder, this addition will be carried out as explained below-

Stage-01:

When C_{in} will be fed as input to full Adder A, it will activate full adder A.

Then, At Full Adder A,

- $A_0 = 1$
- $B_0 = 0$
- $C_{in} = 0$

The sum bit and carry bit produced as output by full adder A will be calculated by full adder A as-

Calculating S_0–

$S_0 = A_0 \oplus B_0 \oplus C_{in}$

$= 1 \oplus 0 \oplus 0$

$= 1$

$\therefore S_0 = 1$

Calculating C_0–

$C_0 = A_0B_0 \oplus B_0C_{in} \oplus C_{in}A_0$

$= 1.0 \oplus 0.0 \oplus 0.1$

$= 0 \oplus 0 \oplus 0$

$= 0$

$\therefore C_0 = 0$

Stage-02:

Now, when C_o will be fed as input to full adder B by full adder A, it will activate full adder B.

Then, At Full Adder B,

- $A_1 = 0$
- $B_1 = 1$
- $C_0 = 0$

The sum bit and carry bit produced as output by full adder B will be calculated by full adder B as-

Calculating S_1–

$S_1 = A_1 \oplus B_1 \oplus C_0$

$= 0 \oplus 1 \oplus 0$

$= 1$

$\therefore S_1 = 1$

Calculating C_1–

$C_1 = A_1B_1 \oplus B_1C_0 \oplus C_0A_1$

$= 0.1 \oplus 1.0 \oplus 0.0$

$= 0 \oplus 0 \oplus 0$

$= 0$

$\therefore C_1 = 0$

Stage-03:

Now, when C_1 will be fed as input to full adder C by full adder B, it will activate full adder C.

Then, At Full Adder C,

- $A_2 = 1$
- $B_2 = 0$
- $C_1 = 0$

The sum bit and carry bit produced as output by full adder C will be calculated by full adder C as-

Calculating S_2–

$S_2 = A_2 \oplus B_2 \oplus C_1$

$= 1 \oplus 0 \oplus 0$

$= 1$

$\therefore S_2 = 1$

Calculating C_2–

$C_2 = A_2B_2 \oplus B_2C_1 \oplus C_1A_2$

$= 1.0 \oplus 0.0 \oplus 0.1$

$= 0 \oplus 0 \oplus 0$

$= 0$

$\therefore C_2 = 0$

Stage-04:

Now, when C_2 will be fed as input to full adder D by full adder C, it will activate full adder D.

Then, At Full Adder D,

- $A_3 = 0$
- $B_3 = 1$
- $C_2 = 0$

The sum bit and carry bit produced as output by full adder D will be calculated by full adder D as-

Calculating S_3–

$S_3 = A_3 \oplus B_3 \oplus C_2$

$= 0 \oplus 1 \oplus 0$

$= 1$

$\therefore S_3 = 1$

Calculating C_3–

$C_3 = A_3B_3 \oplus B_3C_2 \oplus C_2A_3$

$= 0.1 \oplus 1.0 \oplus 0.0$

$= 0 \oplus 0 \oplus 0$

$= 0$

$\therefore C_3 = 0$

Thus finally,

- Output Sum = $S_3S_2S_1S_0$ = 1111
- Output Carry = C_3 = 0

3.5 Why Ripple Carry Adder is called so?

In the operation of a Ripple Carry Adder, the carry-out signal generated by each full adder becomes the carry-in input for the subsequent, more significant full adder in the chain. This sequential handoff of carry signals ensures the smooth progression of the addition process, with each full adder contributing to the cumulative sum calculation.

The term "Ripple Carry Adder" derives from the characteristic behavior of the carry signals as they propagate through the adder stages. Similar to ripples or waves spreading outward in water, the carry-out signal from one full adder ripples into the next stage, activating the subsequent full adder in the sequence. This ripple effect, where each carry signal triggers the activation of the next full adder, underscores the naming convention of the Ripple Carry Adder.

3.6 CARRY INCREMENT ADDER:

An 8-bit increment adder employs a combination of two 4-bit Ripple Carry Adders (RCAs) to achieve its operation. The first RCA handles the addition of the initial 4-bit inputs, producing partitioned sum and carry outputs. The carry-out signal from this first RCA serves as the carry-in input for the subsequent conditional increment block.

In the conditional increment block, the carry-out signal from the first RCA determines whether an increment operation is necessary. This block utilizes half adders to perform the increment operation based on the carry-out value from the first RCA. The second RCA, regardless of the output from the first RCA, then proceeds with the addition operation.

The input carry-in to the first RCA block is set to a low value consistently. The conditional increment block comprises half adders, which execute the increment operation based on the carry-out value from the first RCA. The output sum from the second RCA is obtained through the conditional increment block.

This design configuration ensures that the 8-bit increment adder efficiently handles addition operations while incorporating conditional increment logic based on the carry-out signal from the first RCA. The overall schematic of the Carry Increment Adder provides a comprehensive visualization of its operation and interconnection of components.

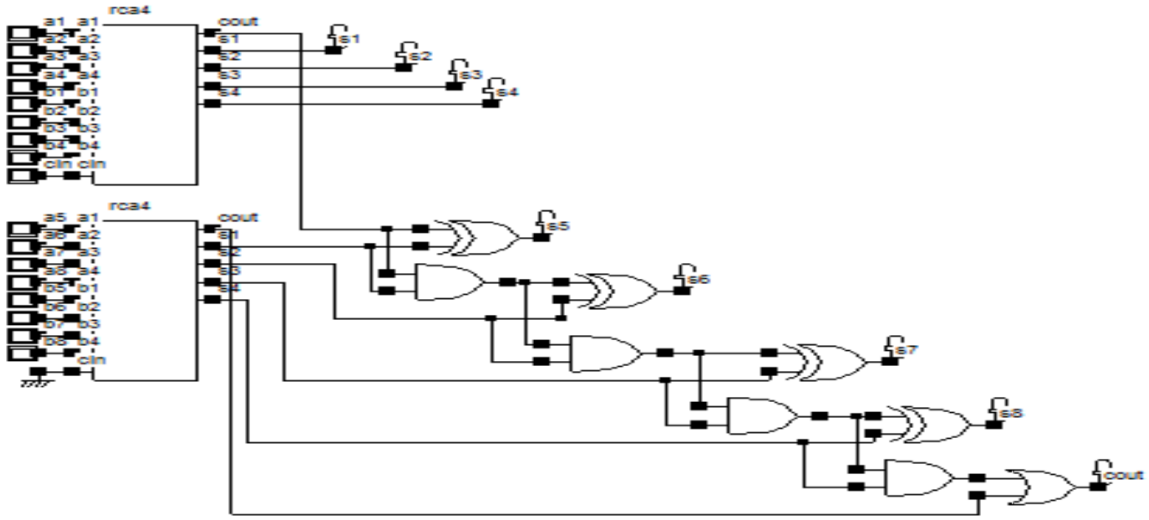


Figure 3.8.1: carry increment adder

Chapter 4

TOOLS USED

4.1Verilog

Verilog synthesis tools wield the alchemical power to transmute abstract behavioral descriptions into tangible hardware realities, sculpting logic-circuit structures that dance to the tune of selected technologies. From the humble beginnings of a simple combinational circuit to the grandeur of a complete microprocessor on chip, Verilog's versatile embrace allows designers to traverse the entire spectrum of hardware design with finesse.

Verilog HDL, a stalwart in the realm of hardware description languages, stands as a beacon of evolution and standardization. Its arsenal of features beckons designers with promises of ease and efficiency. Drawing parallels to the syntax of the venerable C programming language, Verilog extends a welcoming hand to those fluent in its code, offering a seamless transition into the world of hardware design.

In the tapestry of Verilog, abstraction finds harmony. Designers weave together switches, gates, RTL, and behavioral code, crafting intricate models with a singular language that accommodates diverse perspectives. With Verilog, the journey from stimulus to hierarchical design unfolds effortlessly, empowering designers with a unified toolset.

At the heart of Verilog's allure lies its widespread support among the pantheon of logic synthesis tools. This unanimous embrace solidifies Verilog's stature as the language of choice for discerning designers, offering a gateway to a plethora of possibilities.

In the labyrinth of chip design, Verilog serves as a guiding light, illuminating a path adorned with vendor libraries and post-logic synthesis simulations. Across the expanse of fabrication vendors, Verilog's ubiquitous presence ensures designers a cornucopia of options, enabling them to navigate the landscape with confidence.

Beneath the surface lies the hidden gem of Verilog's Programming Language Interface (PLI), a realm where customization thrives. With the PLI, designers wield the power to sculpt Verilog simulators to their whims, infusing them with bespoke functionality through the magic of custom C code.

In the symphony of hardware design, Verilog orchestrates a melody of innovation and possibility, inviting designers to unleash their creativity upon the canvas of logic and silicon.

4.1.1 History of Verilog HDL

In the swirling mists of technological evolution, a language was born. Verilog, whispered into existence by the minds at Gateway Design Automation Inc. around the year 1984, emerged as a clandestine fusion of the era's most revered digital incantations. Legend has it that it drew upon the mystical essence of HiLo, a revered High-Level Description Language, as well as the arcane runes of traditional programming tongues like C.

In those primordial days, Verilog roamed free, unshackled by standardization. With each passing moon, it shape-shifted, evolving through a kaleidoscope of revisions from 1984 to 1990, rewriting its own script with every cosmic dance.

In the year 1985, the first whispers of its power manifested in the Verilog simulator, a tool crafted by Gateway, which grew in potency through the crucible of 1987. Then, from the depths of innovation, emerged Verilog-XL, a phoenix reborn with enhancements and adorned with the enigmatic "XL algorithm," a key to unlocking the secrets of gate-level simulation with unprecedented efficiency.

But fate, it seems, had other plans. As the clock struck the late hours of 1990,

Cadence Design System, a titan in the realm of digital design, cast its gaze upon Gateway's treasures. In a grand convergence of destinies, Cadence acquired Gateway, seizing hold of the Verilog language and its attendant powers. Thus, under the banner of Cadence, Verilog found new purpose, both as a language and a simulator, heralding a new age of synthesis.

Meanwhile, across the digital horizon, Synopsys, a master of the esoteric art of top-down design, saw in Verilog a conduit to boundless creation. Together, they forged a pact, blending top-down methodology with Verilog's incantations, unleashing a tempest of innovation upon the world.

And so, the saga of Verilog unfolded, a tale woven with threads of invention, acquisition, and collaboration, shaping the very fabric of the digital cosmos.

In the tapestry of technological evolution, Cadence stood at a crossroads in 1990, gazing into the abyss of standardization. With foresight as their lantern, they realized that Verilog, their prized creation, could not thrive in isolation. In a bold stroke, they unfurled the banner of Open Verilog International (OVI), offering the language's documentation to the winds of change in 1991, heralding an era of openness.

Under OVI's stewardship, Verilog underwent a metamorphosis, shedding its proprietary shackles and donning the mantle of vendor-independence. The Language Reference Manual (LRM) became the crucible of refinement, honing Verilog's syntax into a gleaming beacon of universality.

Yet, amidst the euphoria of openness, a specter loomed on the horizon. The fear that Verilog's proliferation would birth a cacophony of dialects, eroding its very essence, spurred action. Thus, in the crucible of 1994, the IEEE 1364 working group emerged, forging Verilog into the crucible of standardization. In December 1995, Verilog ascended to the pantheon of IEEE standards, a testament to its enduring legacy.

As Verilog spread its wings, a chorus of simulators echoed across the digital landscape. Chronologic Simulation's VCS emerged as a titan, wielding the power of true compilation to unlock unprecedented simulation speeds. The tide turned, and Verilog's allure eclipsed that of its federal counterpart, VHDL, casting a spell upon the hearts of designers worldwide.

The clamor for standardization crescendoed, prompting OVI's directors to beseech IEEE's aid. Thus, in the midsummer of 1993, the hallowed halls of IEEE bore witness to the birth of Working Committee 1364, a beacon of hope for Verilog's universal acceptance.

In May 1995, Verilog was enshrined in the annals of history as IEEE Std. 1364-1995, a testament to its enduring journey from the crucible of innovation to the pinnacle of standardization.

But the story did not end there. In the ebb and flow of time, Verilog evolved, shedding its old skin to emerge anew in the form of Verilog 2001. This iteration, baptized as IEEE Std. 1364-2001, heralded a new dawn, casting aside the shadows of the past and embracing a future ripe with possibilities.

4.1.2 Program Structure:

The building blocks of Verilog reside within the sanctum of the "MODULE," encapsulating the essence of digital incantations. Declarations within the module delineate inputs, outputs, and signals, while the heart of the module orchestrates operations, weaving a tapestry of logic transcending mortal comprehension. Identifiers serve as linguistic anchors, subject to stringent guidelines, allowing for clarity and functionality within programmatic syntax. Whitespace emerges as the silent conductor of clarity and coherence, guiding the eye through the labyrinthine corridors of code, enhancing the legibility of the narrative. Commentary stands as the cornerstone of understanding, elucidating the arcane and guiding the uninitiated through the labyrinth of logic. Port declarations define communication channels between modules, facilitating data flow within the system.

4.1.4 Logic System:

In the realm of Verilog, a 4-value logic system forms the bedrock of its operational framework. Within this system, a single-bit signal possesses the capability to embody one of four distinct states:

- 0: Represents the logical value zero, denoting a clear absence of signal or logical low.
- 1: Signifies the logical value one, indicating a presence of signal or logical high.
- X: Stands for the unknown logical value, implying uncertainty or indeterminacy in the signal's state.
- Z: Symbolizes high impedance, suggesting that the signal is effectively disconnected from the circuit or lacks a defined logical value.

This comprehensive logic system empowers Verilog designs to capture a wide range of scenarios, from deterministic binary states to more nuanced situations where signal behavior may be ambiguous or undefined. It provides a robust foundation for modeling complex digital circuits and facilitates accurate simulation of real-world hardware behavior.).

4.1.5 Operators:

Verilog, as a versatile hardware description language, equips designers with an array of operators catering to diverse needs across arithmetic, relational, bitwise, logical, reduction, shift, concatenation, replication, and more. These operators serve as fundamental building blocks for expression evaluation and manipulation, enabling the creation of intricate digital designs with precision and efficiency.

Arithmetic Operators:

Facilitate mathematical operations on operands.

- Addition (+)
- Subtraction (-)
- Multiplication (*)
- Division (/)
- Modulus (%)

Relational Operators:

Compare operands and yield a boolean result.

- Less than (<)
- Less than or equal to (<=)
- Greater than(>)
- Greater than or equal to(>=)
- Equal to (==)
- Not equal to (!=)

Bitwise Operators:

Perform operations on individual bits of operands.

- Bitwise NOT (~)
- Bitwise AND (&)
- Bitwise OR (|)
- Bitwise XOR (^)
- Bitwise XNOR (^ or ^)

Logical Operators:

Evaluate logical expressions and return boolean results.

- Logical NOT (!)
- Logical AND (&&)
- Logical OR (||)

Reduction Operators:

Operate on all bits of an operand vector and yield a single-bit result.

- Reduction AND (&)
- Reduction OR (|)
- Reduction NAND (~&)
- Reduction NOR (~|)
- Reduction XOR (^)
- Reduction XNOR (^ or ^)

Shift Operators:

Perform left or right shifts on operands.

- Shift left (<<)
- Shift right (>>)

Concatenation Operator:

Combines multiple operands to form a larger vector.

- Concatenation ({ })

Replication Operator:

Generates multiple copies of an item.

Replication ({n{item}})

These operators, when utilized effectively, enable designers to express complex logic and manipulate data efficiently within Verilog designs. They form the backbone of expression evaluation, paving the way for the creation of sophisticated digital systems with precision and reliability.

4.1.6 Dataflow Design Elements:

Continuous assignment statements describe combinational circuits in terms of data flow and operations, allowing for dataflow design or description. The fundamental syntax involves assigning expressions to nets or net concatenations.

4.1.7 Structural Design (Or) Gate-Level Modeling Structural:

Design entails describing circuits by instantiating and connecting individual gates and components using nets. Verilog offers built-in gate types, allowing for the instantiation of gates and interconnection to form complex circuits.

4.1.8 Behavioral Modeling:

Behavioral modeling describes circuit functionality at an abstract level, resembling C programming more than traditional circuit design. Structured procedures such as always and initial statements form the foundation of behavioral modeling, allowing for the sequential execution of procedural statements. Conclusion Verilog, with its rich history, robust program structure, versatile logic system, and

comprehensive design elements, remains a cornerstone in the realm of hardware description languages. Its evolution and standardization have paved the way for innovation, empowering designers to craft intricate logic and silicon designs with finesse and creativity.

4.2 Xilinx

Xilinx software stands as a pivotal tool in the arsenal of VHDL/Verilog designers, facilitating the crucial synthesis operation. This process marks a transformative phase wherein high-level HDL code metamorphoses into a gate-level netlist, a fundamental step in contemporary design workflows. The significance of synthesis cannot be overstated, as it serves as the bridge between abstract hardware descriptions and tangible FPGA configurations.

With Xilinx software, designers wield the power to translate simulated code directly into hardware configurations deployable on Field-Programmable Gate Arrays (FPGAs). This capability underscores the versatility and efficiency of Xilinx tools in enabling seamless integration between design and implementation phases.

Synthesis, within the realm of Xilinx software, encapsulates the essence of digital alchemy, where complex behavioral descriptions are distilled into concrete logic structures. It embodies the convergence of theoretical design concepts with practical hardware realities, paving the way for robust and efficient FPGA implementations.

Moreover, Xilinx's synthesis capabilities empower designers to optimize their designs for performance, power consumption, and area utilization, thereby ensuring the realization of high-performance and resource-efficient hardware solutions. This emphasis on optimization underscores Xilinx's commitment to pushing the boundaries of FPGA design possibilities.

In essence, Xilinx software serves as a catalyst for innovation, enabling VHDL/Verilog designers to transcend conceptualization and bring their designs to life in the form of synthesized gate-level netlists. It epitomizes the intersection of creativity and engineering precision, propelling the field of digital design forward into new realms of possibility.

4.2.1 Algorithm

Start the ISE Software by clicking the XILINX ISE icon. Create a New Project and find the following properties displayed. If the design needs large number of LUTs there is a possibility to change the family ,device and package changes.

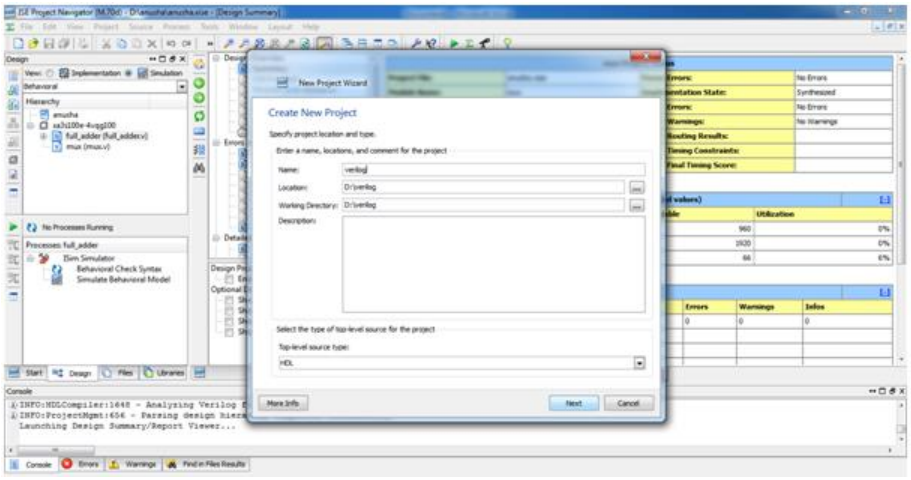


Fig 4.2.1.1 Create new folder for design

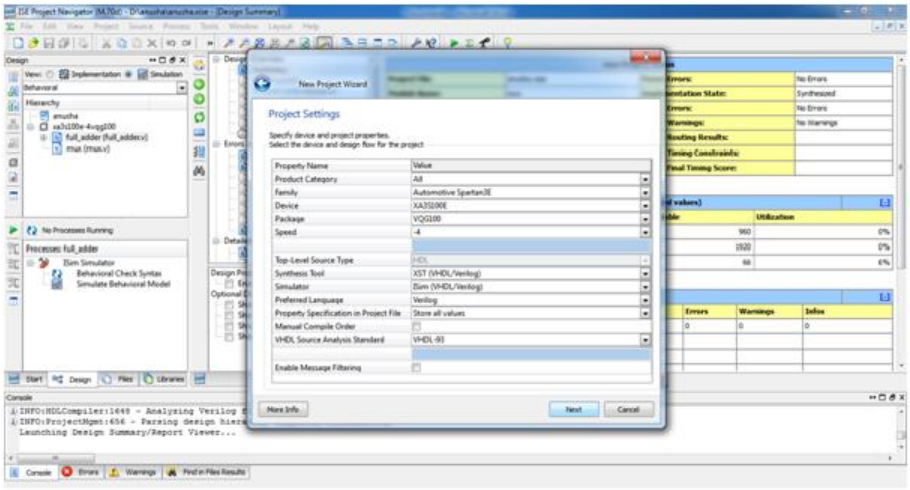


Fig 4.2.1.2. Set family and device before design a project

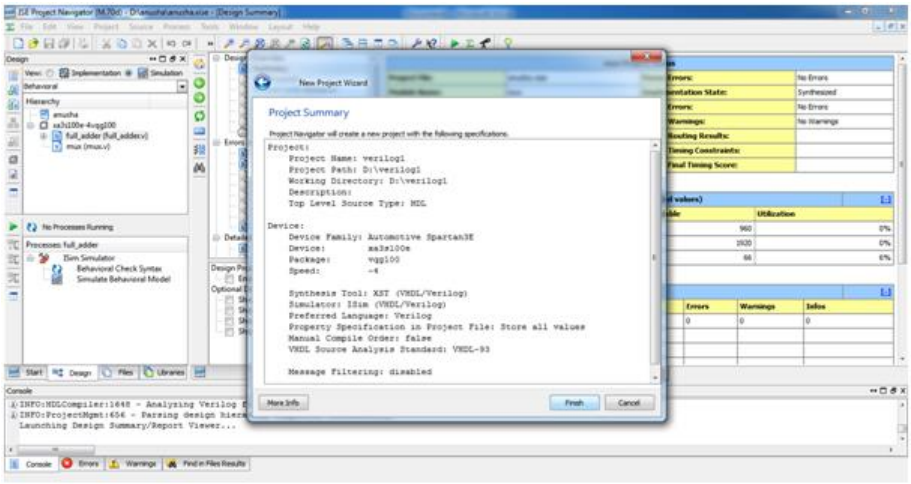


Fig 4.2.1.3. finishing new folder, Set family and device before design a project

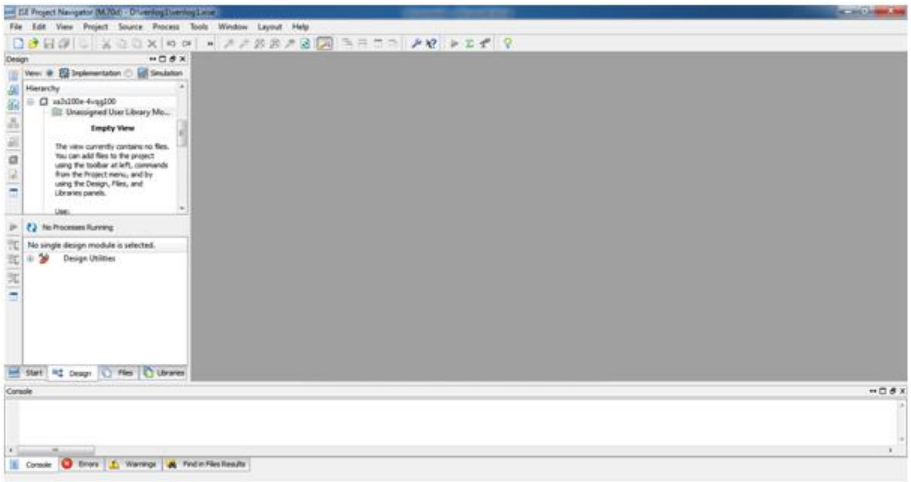


Fig 4.2.1.4. Ready to design a project

Create a HDL Source formatting all inputs, outputs and buffers if required. which provides a window to write the HDL code, to be synthesized.

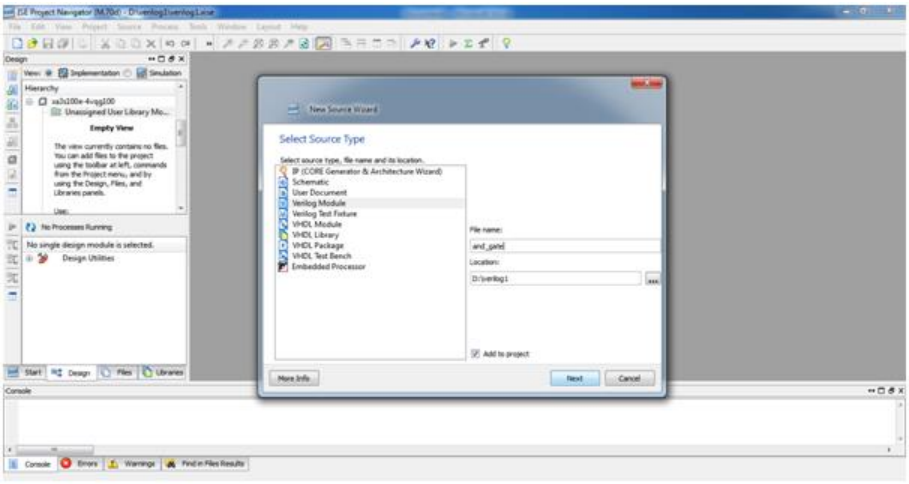


Fig 4.2.1.5. create module name(.v files names) for design

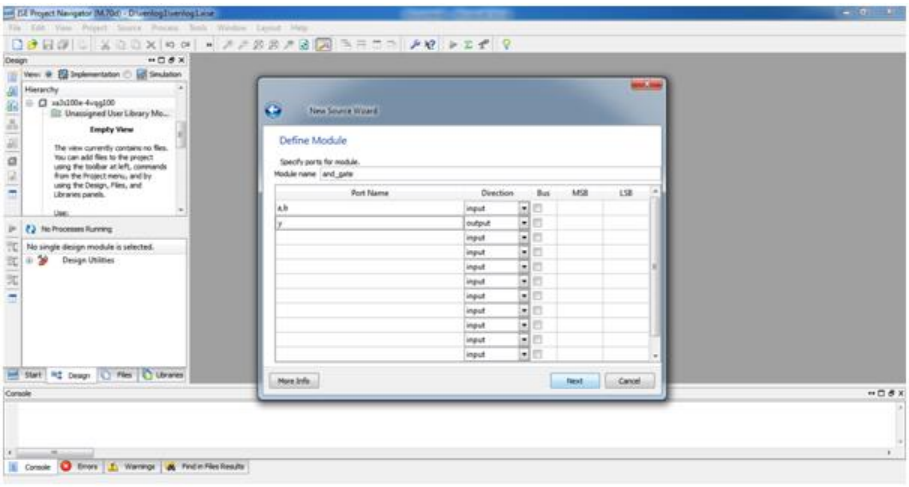


Fig4.2.1.6. Declaration of input and output ports with their bit lengths

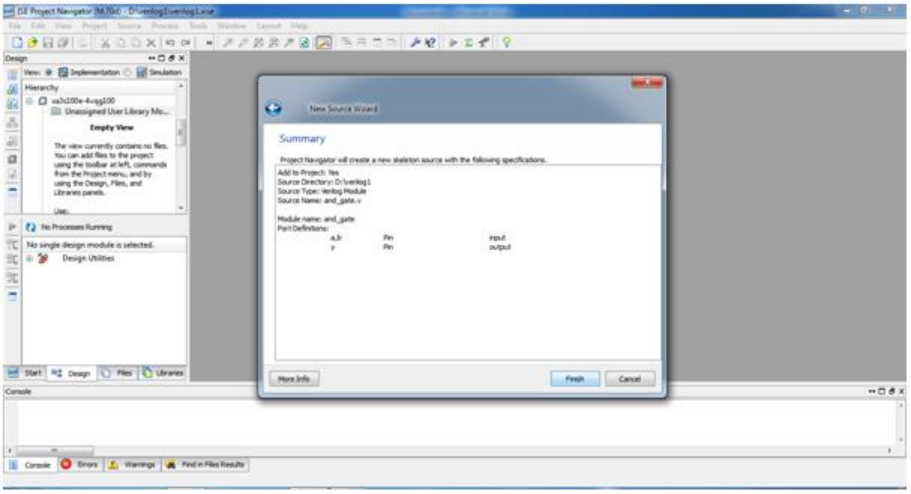


Fig 4.2.1.7. The schematic was created by its ports

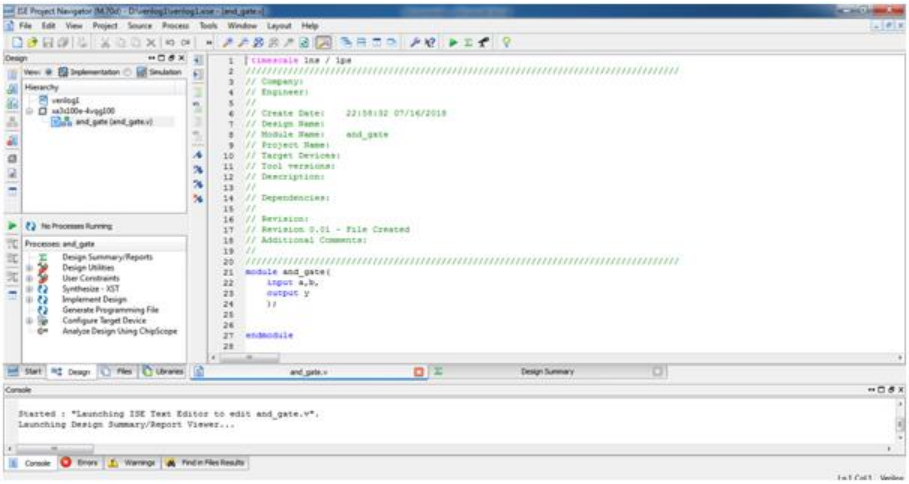


Fig 4.2.1.8: Ready to write the code for design

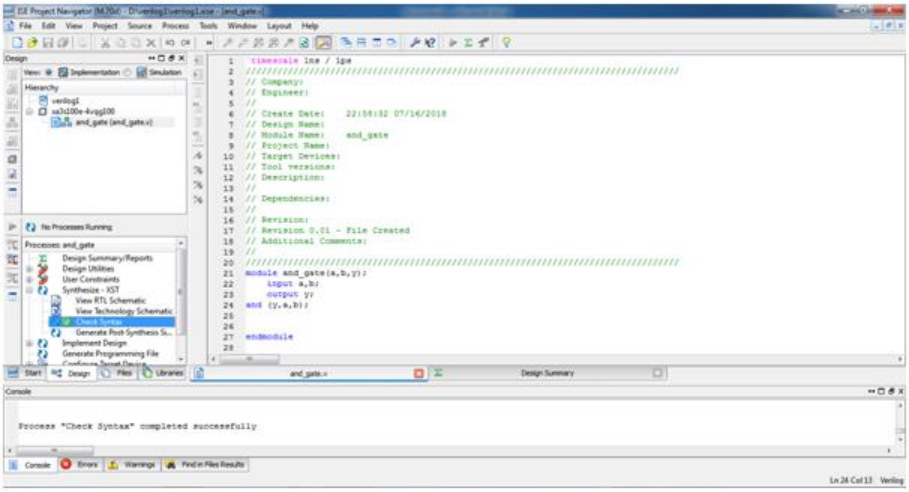


Fig 4.2.1.9: Check syntax for the Design

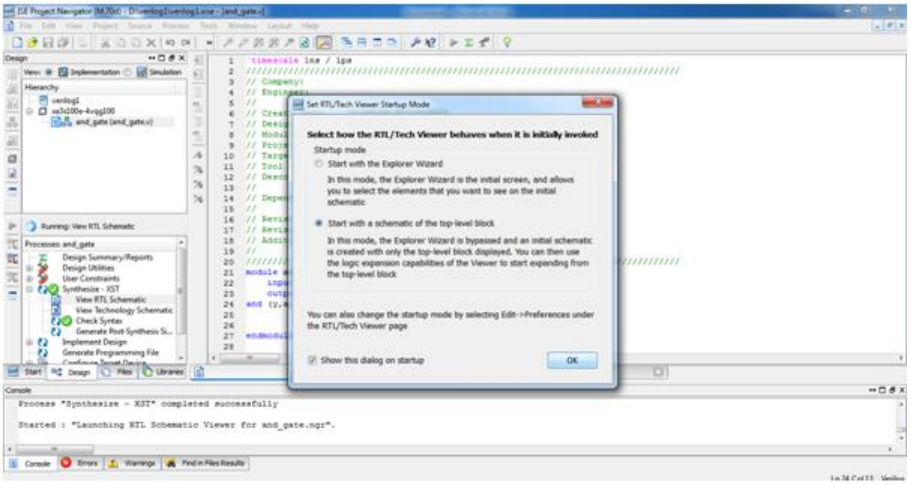


Fig 4.2.1.10: check for RTL schematic view

For RTL (register transfer logic) view, which is also known as designer view because it is look like what the designer thinks in his mind.

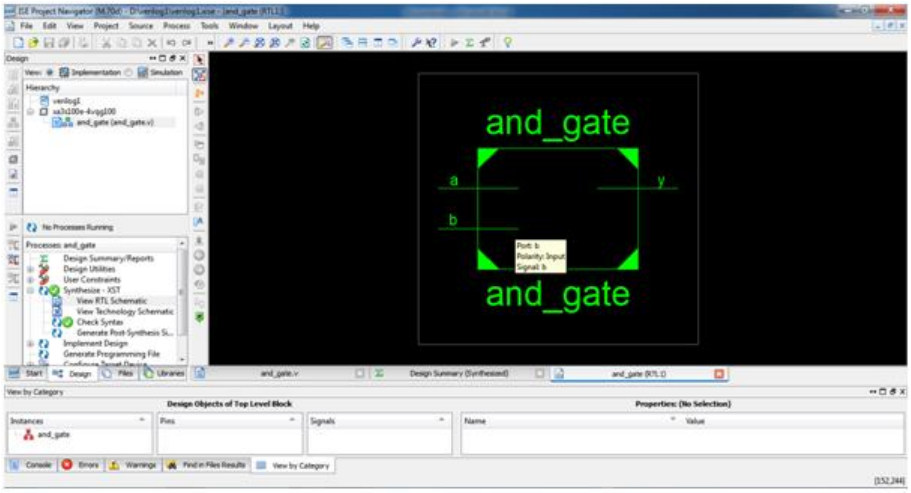


Fig 4.2.1.11.RTL schematic view of design (and gate)

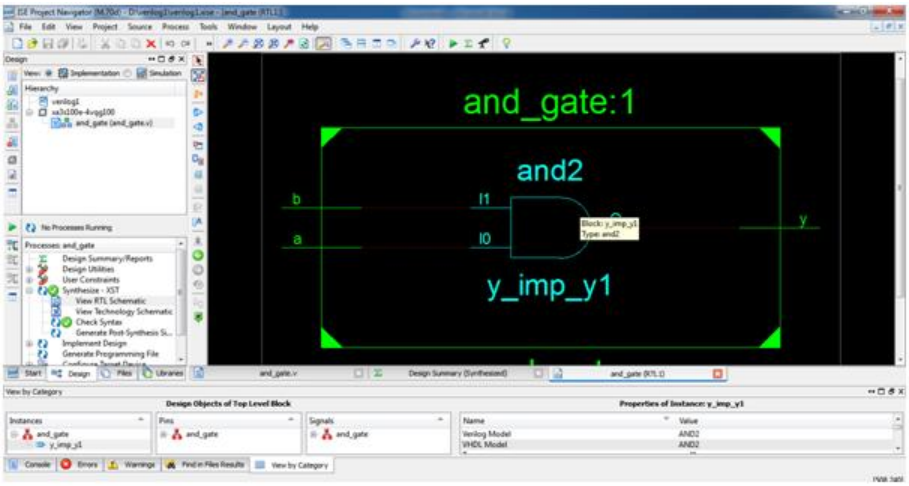


Fig 4.2.1.12: Internal structure of RTL schematic view of design(andgate)

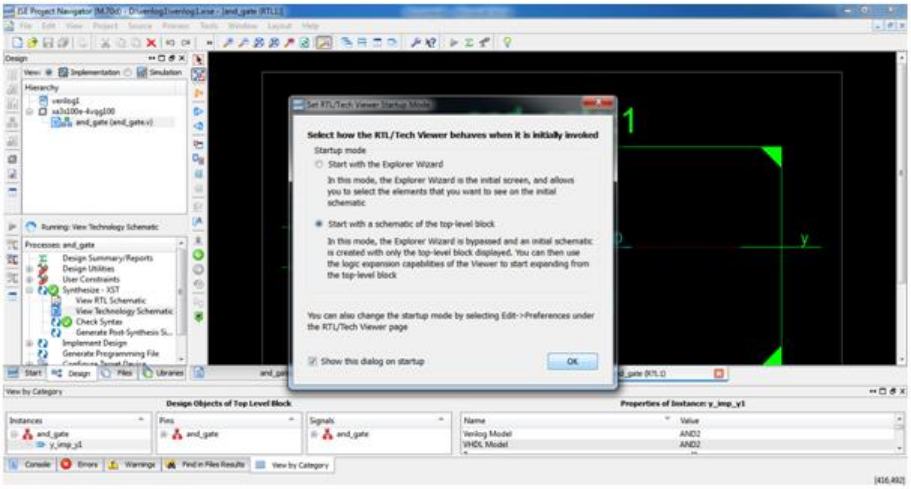


Fig 4.2.1.13. Check for view technology schematic view of the project

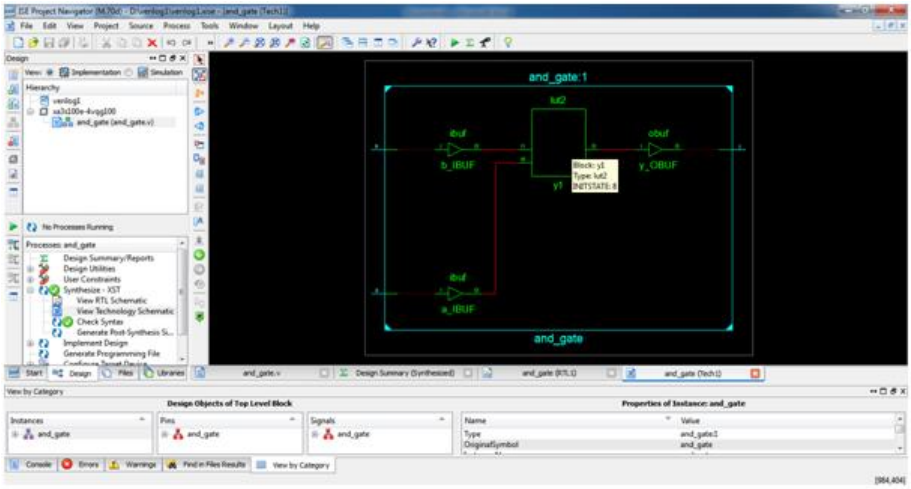


Fig 4.2.1.14: Internal structure of view technology schematic view

View technology schematic of design (and gate), here LUTs are displayed, luts are considered as area of the design.

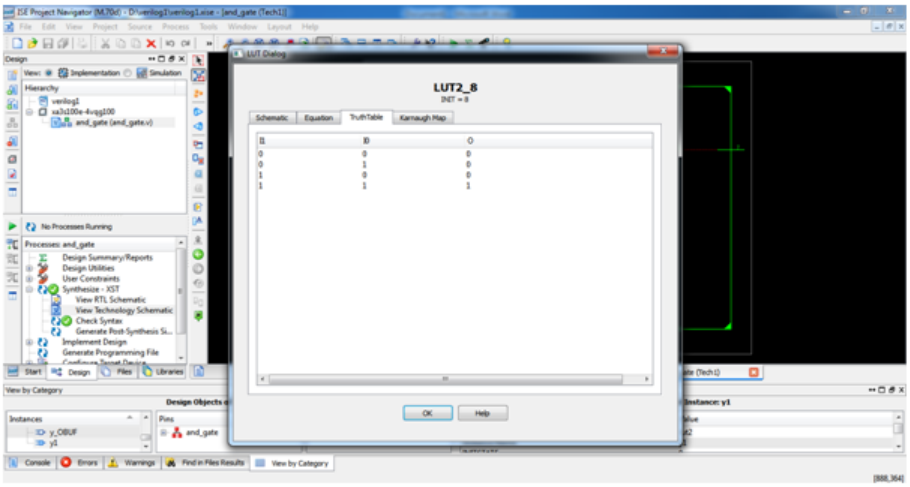


Fig4.2.1.15: The truth table, schematic of design, Boolean equation and k-map of design.

In Xilinx tool there is a availability to get truth table, schematic of design, Boolean equation and k-map

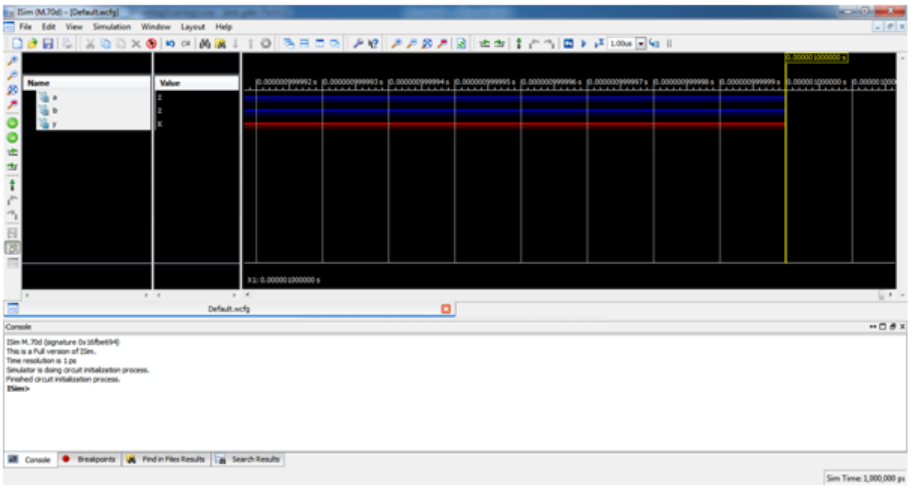


Fig4.2.1.16: Simulation of design to verifying the logics of design.

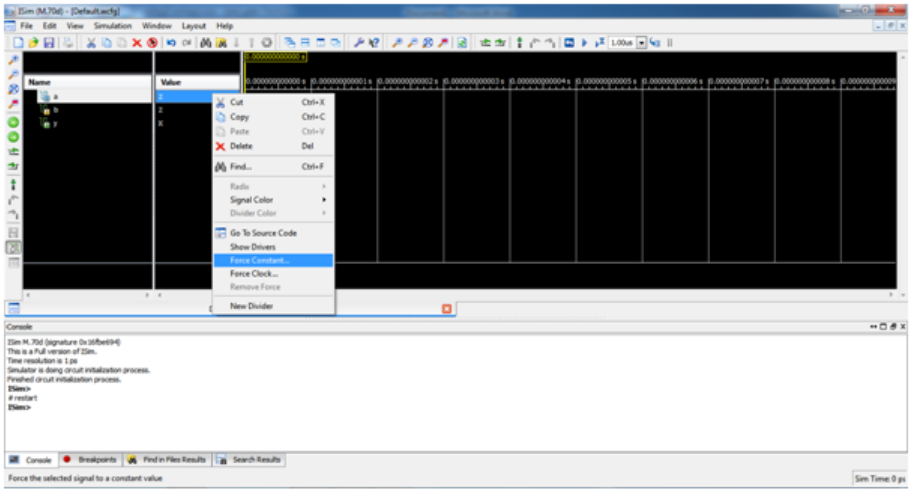


Fig 4.2.1.17: Apply inputs through force constant or force clock for input signals

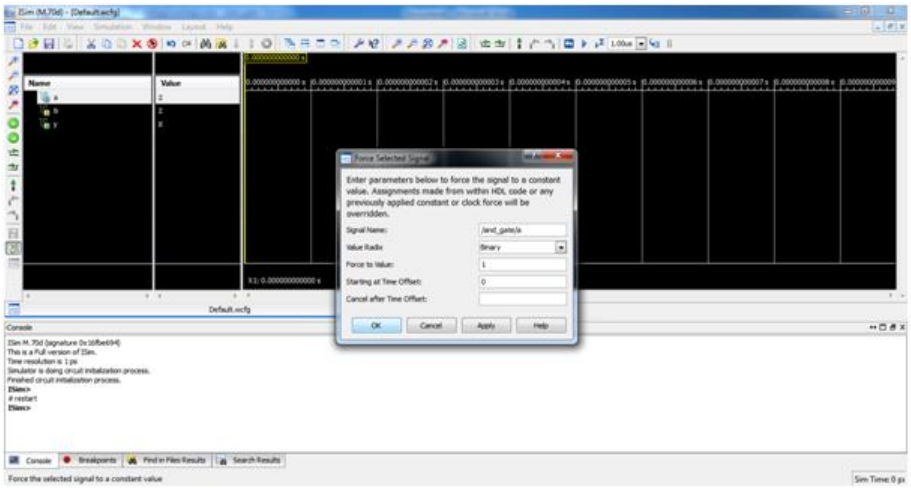


Fig4.2.1.18: Apply force to value

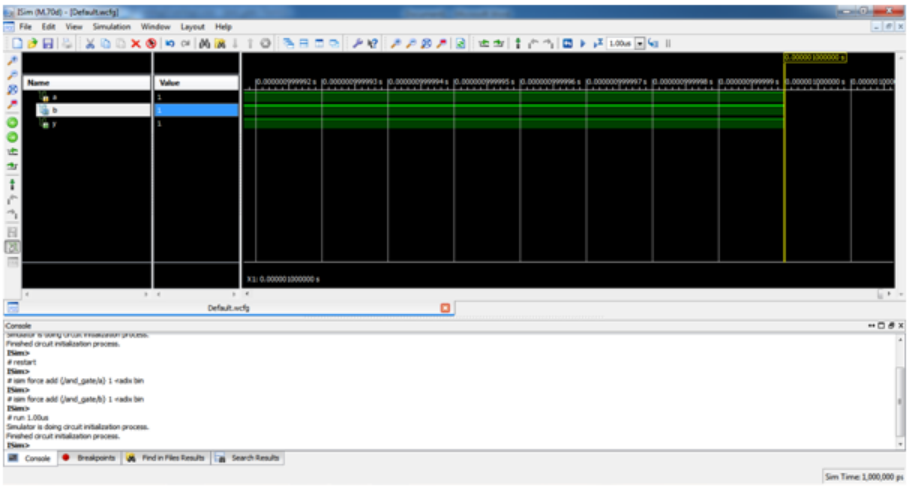


Fig 4.2.1.19: Run the design after applying inputs

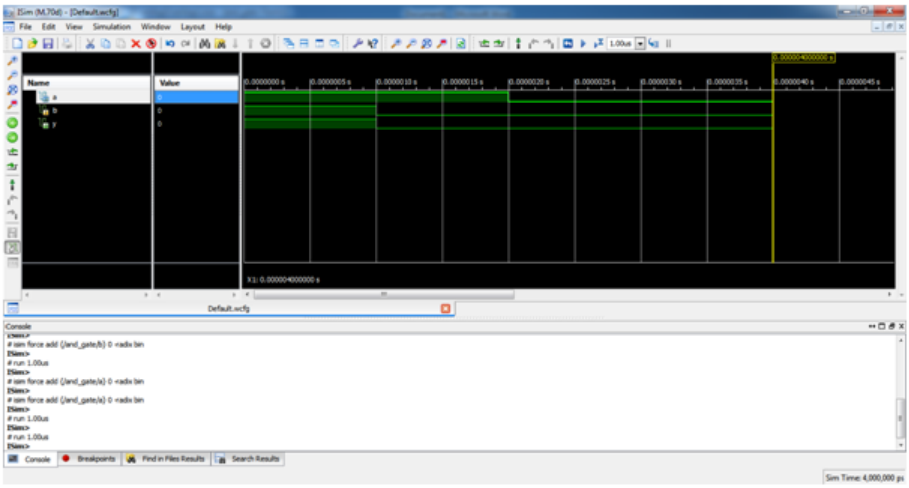


Fig 4.2.1.20: Show all values (zoom to full view) for the design

CHAPTER 5

RESULTS

5.1 Existed design results

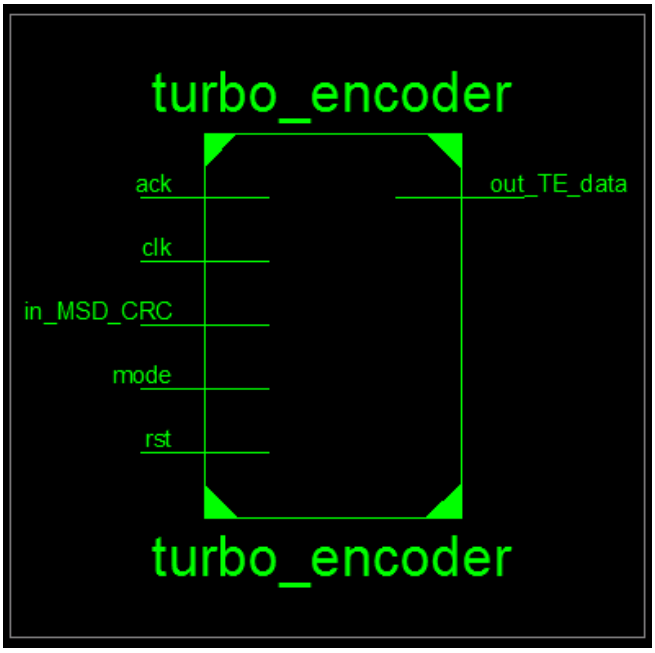


Fig 5.1.1: RTL Schematic of turbo encoder

RTL Schematic: RTL schematic is described as register transfer logic that means the logic is transferred to registers it is also known as designer view because of it is looking like what is the intension of designer.

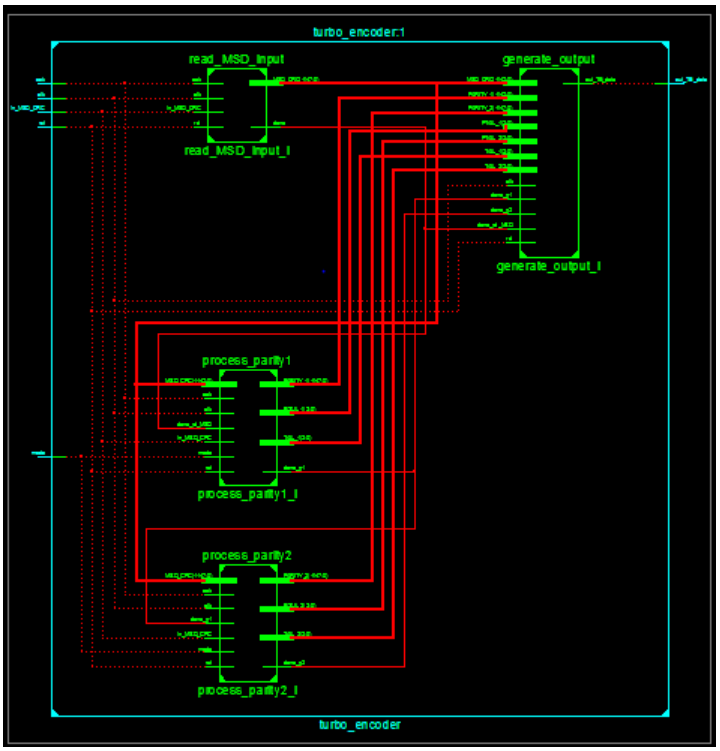


Fig 5.1.2: Internal structure of RTL schematic of turbo encoder

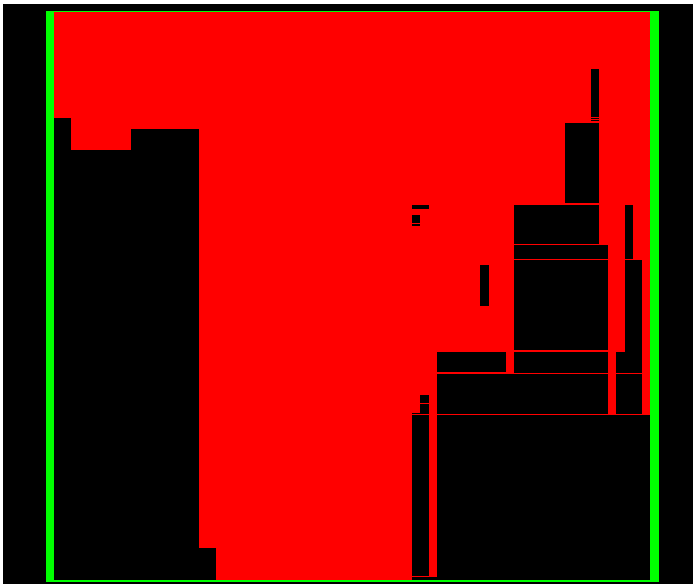


Fig 5.1.3: view technology schematic of turbo encoder

TECHNOLOGY SCHEMATIC:-

The technology schematic makes the representation of the architecture in the LUT format ,where the LUT is consider as the parameter of area that is used in VLSI to estimate the architecture design .the LUT is consider as an square unit the memory allocation of the code is represented in there LUT s in FPGA.

5.2 Proposed design results

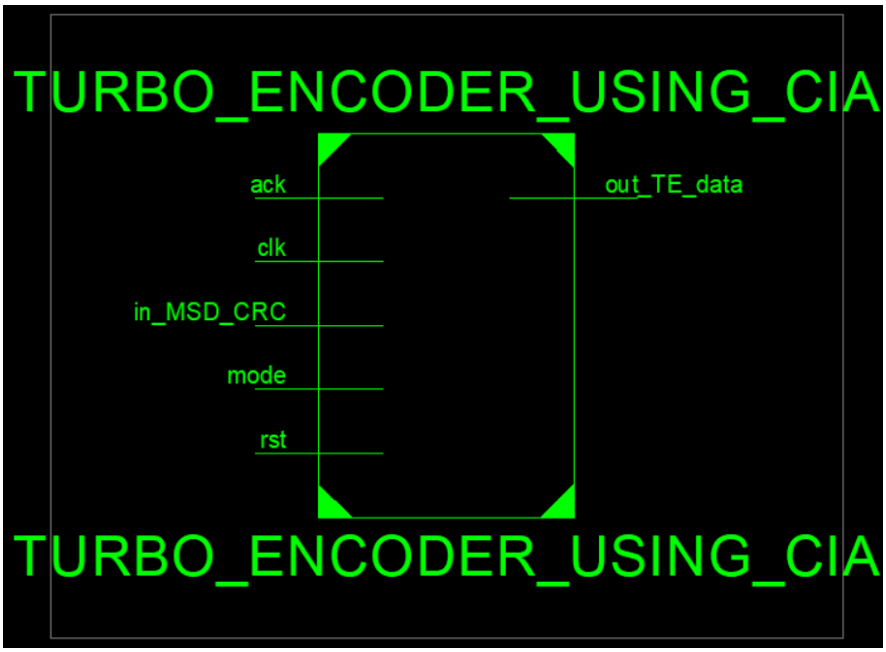


Fig 5.2.1: RTL Schematic of turbo encoder using CIA

RTL Schematic: RTL schematic is described as register transfer logic that means the logic is transferred to registers it is also known as designer view because of it is looking like what is the intension of designer.

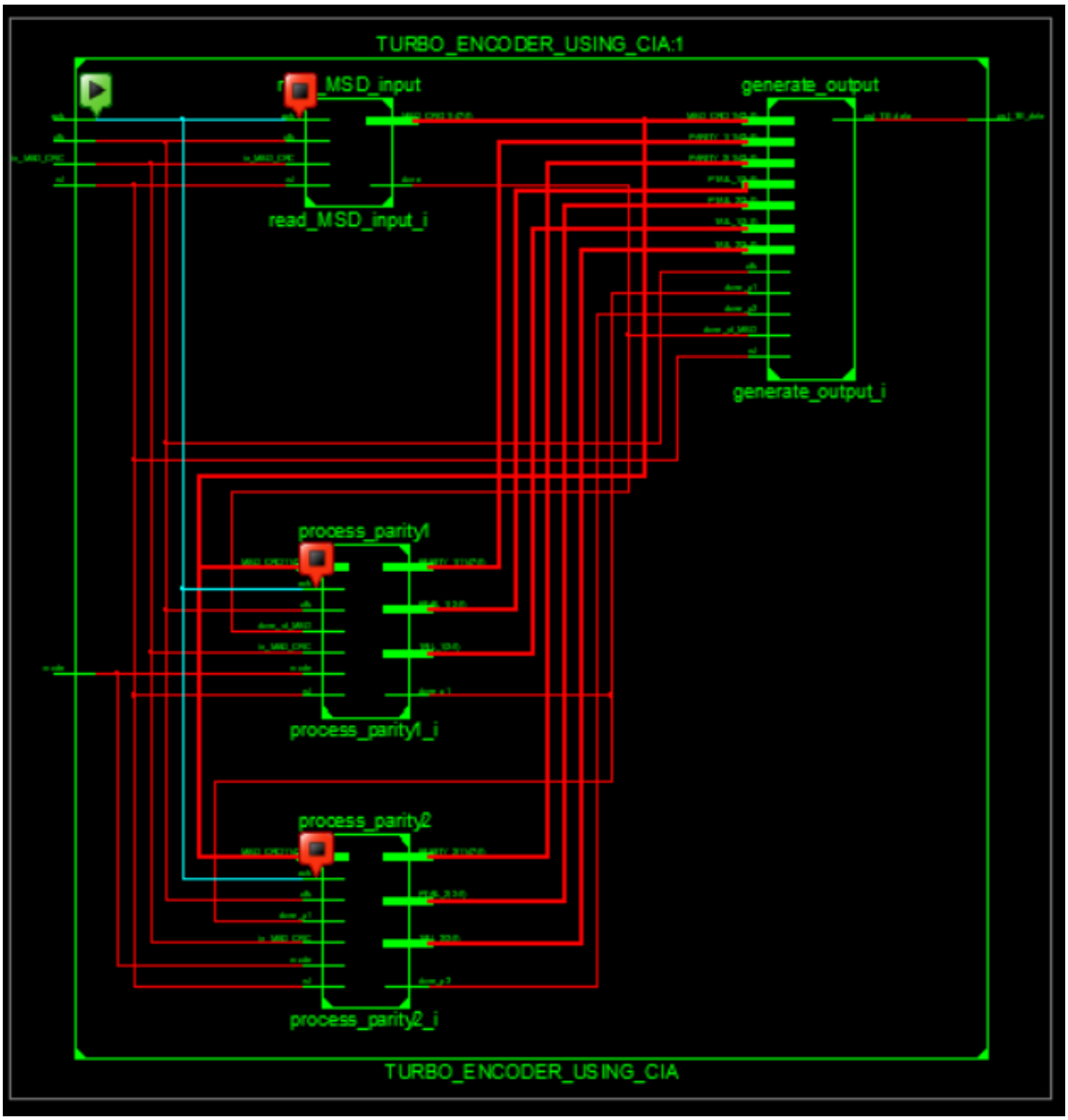


Fig 5.2.2: Internal structure of RTL schematic of turbo encoder using CIA

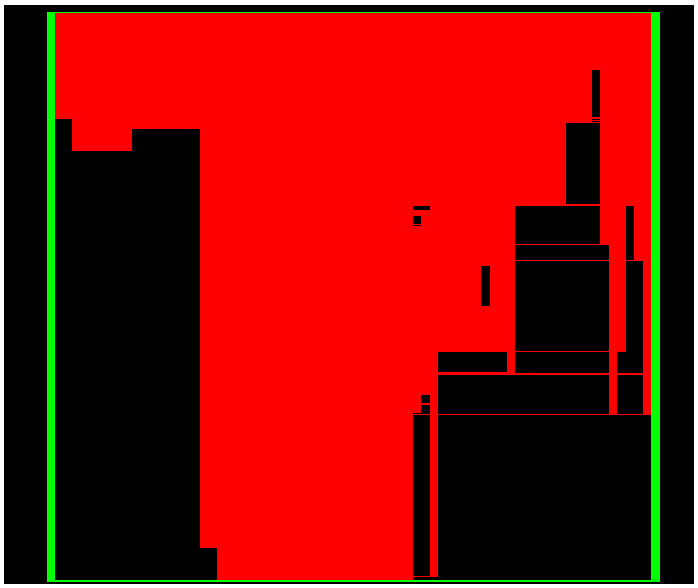


Fig 5.2.3: view technology schematic of turbo encoder using CIA

TECHNOLOGY SCHEMATIC:

The technology schematic makes the representation of the architecture in the LUT format, where the LUT is consider as the parameter of area that is used in VLSI to estimate the architecture design .the LUT is consider as an square unit the memory allocation of the code is represented in there LUT s in FPGA.

Simulation

The simulation is the process which is termed as the final verification in respect to its working where as the schematic is the verification of the connections and blocks. The simulation window is launched as shifting from implementation to the simulation on the home screen of the tool, and the simulation window confines the output in the form of wave forms output. Here it has the flexibility of providing the different radix number systems. In this project serial architecture and parallel methodologies are used.

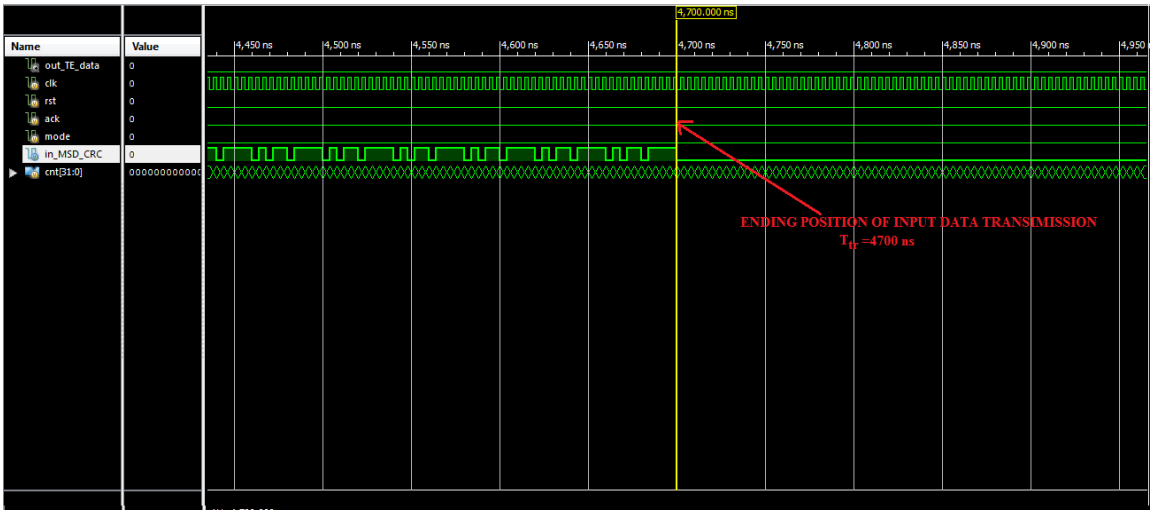


Fig 5.2.4: Simulated wave form of serial computation when mode is 0 at time 4700 ns

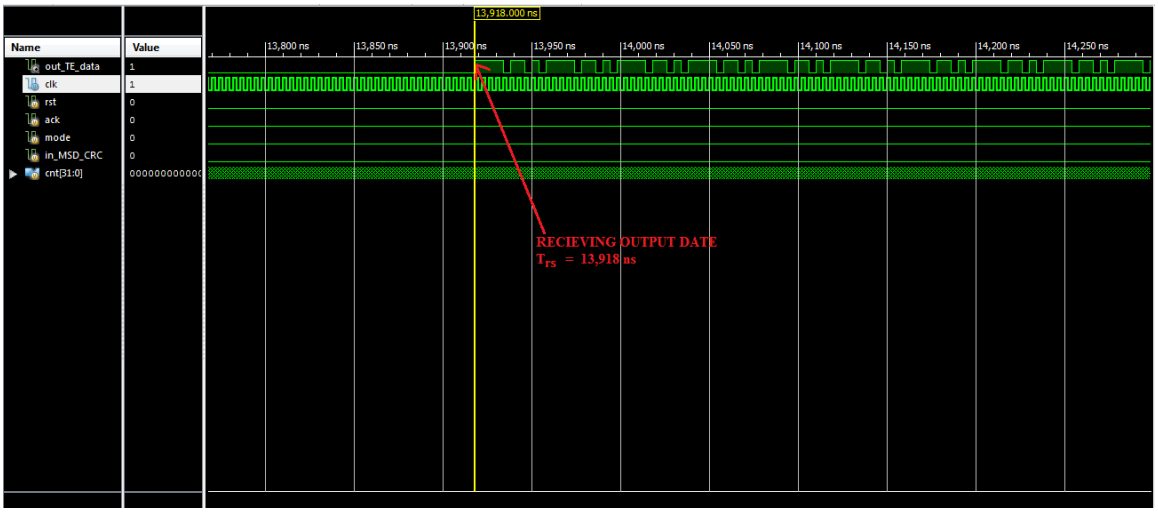


Fig 5.2.5: Simulated wave form of serial computation when mode is 0 at time 13918 ns

Required time for serial computation = 13618 – 4700 = 9218 ns

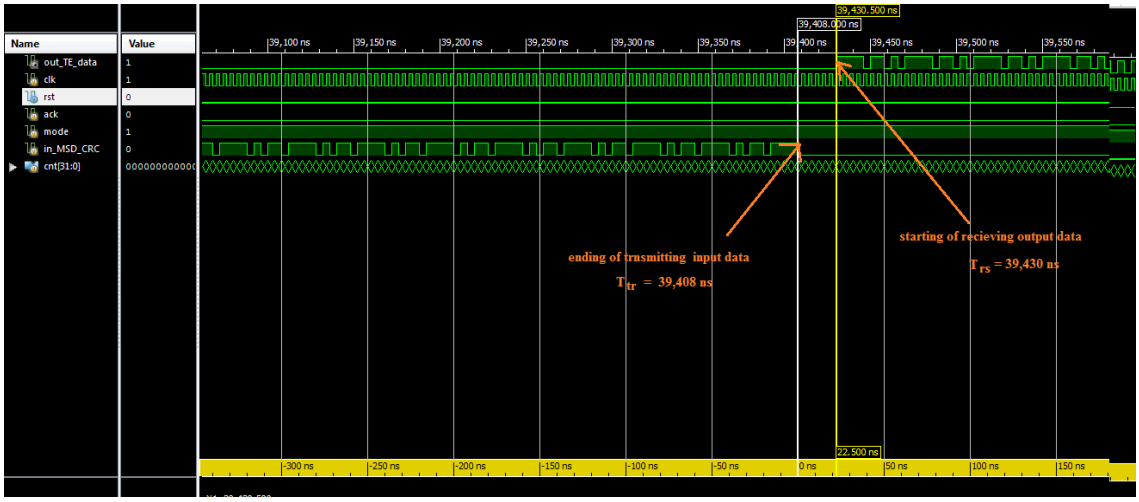


Fig 5.2.6: Simulated wave form of parallel l computation when mode is 1

Input data transmitting time Ttr = 39408 ns

Output data receiving time Trs = 39430 ns

Required time for parallel computation 39430 – 30408 = 22 ns

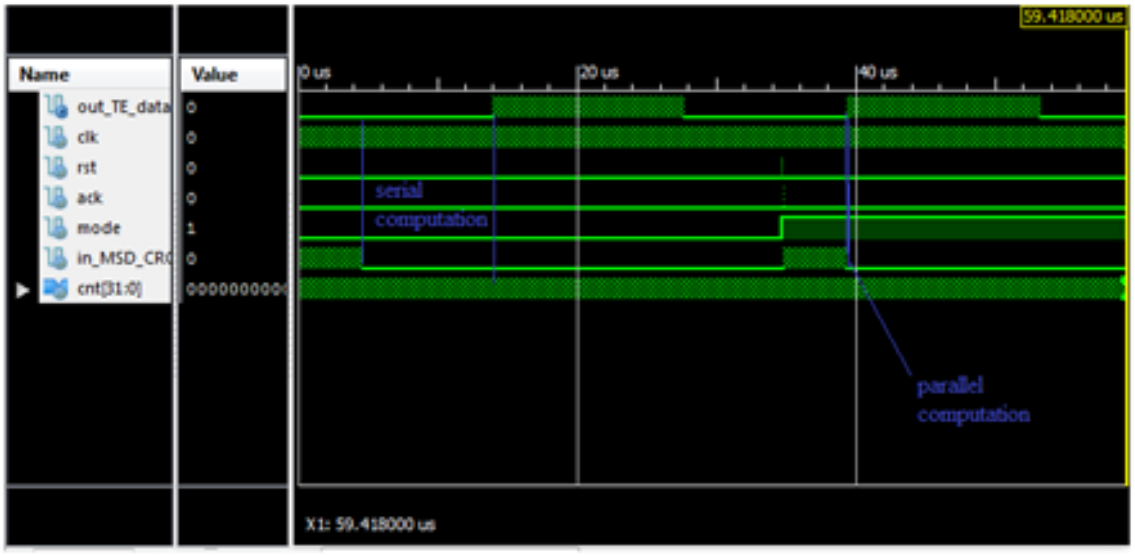


Fig 5.2.7: simulated wave form of turbo encoder

Table 5.2.1: Clock comparison table

Parameter	Serial computation	Parallel computation
Required time to receiving output (ns)	**9218**	**22**

From table1 the parallel computation method uses less number of clock cycles it will be causes to reduce the power consumption because of 50% of power of the design was utilized by clock pulse only.

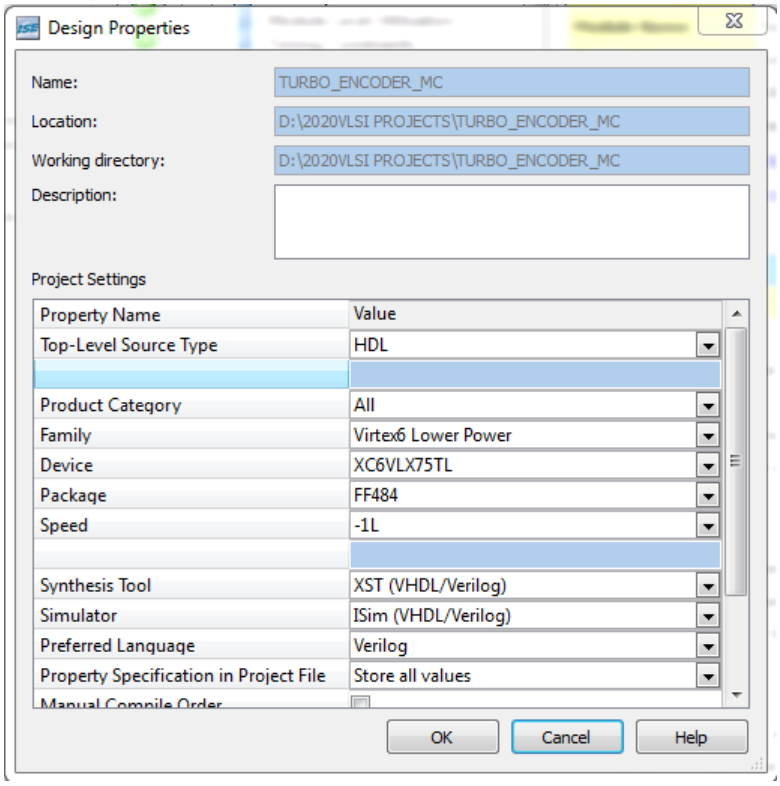
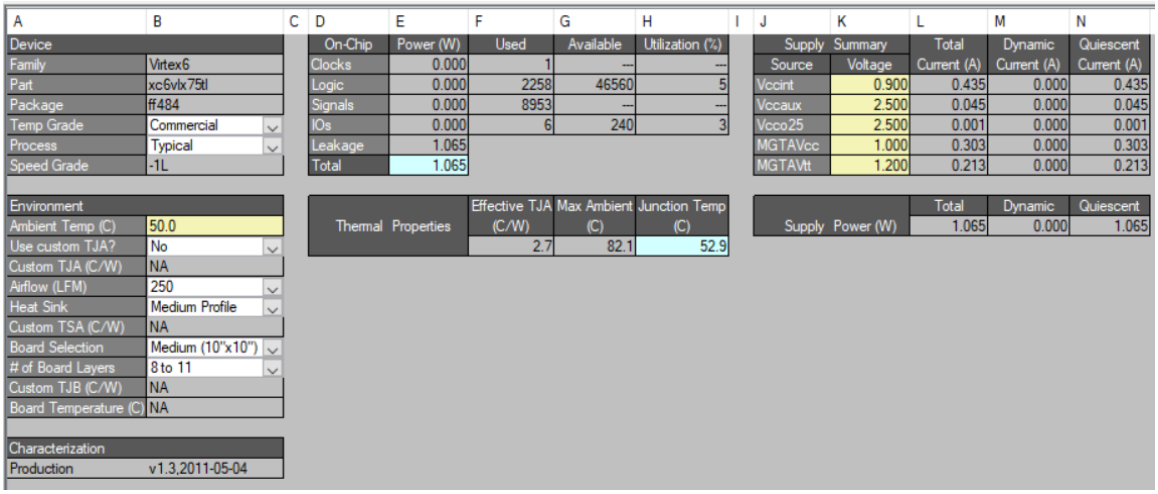


Fig 5.2.8: device preferred for simulation



Device	
Family	Virtex6
Part	xc6vlx75tl
Package	ff484
Temp Grade	Commercial
Process	Typical
Speed Grade	-1L

Environment	
Ambient Temp (C)	50.0
Use custom TJA?	No
Custom TJA (C/W)	NA
Airflow (LFM)	250
Heat Sink	Medium Profile
Custom TSA (C/W)	NA
Board Selection	Medium (10"x10")
# of Board Layers	8 to 11
Custom TJB (C/W)	NA
Board Temperature (C)	NA

Characterization	
Production	v1.3,2011-05-04

On-Chip	Power (W)	Used	Available	Utilization (%)
Clocks	0.000	1	---	---
Logic	0.000	2258	46560	5
Signals	0.000	8953	---	---
IOs	0.000	6	240	3
Leakage	1.065			
Total	1.065			

Thermal Properties	Effective TJA (C/W)	Max Ambient (C)	Junction Temp (C)
	2.7	82.1	52.9

Supply Summary		Total	Dynamic	Quiescent
Source	Voltage	Current (A)	Current (A)	Current (A)
Vccint	0.900	0.435	0.000	0.435
Vccaux	2.500	0.045	0.000	0.045
Vcco25	2.500	0.001	0.000	0.001
MGTAVcc	1.000	0.303	0.000	0.303
MGTAVtt	1.200	0.213	0.000	0.213

Supply Power (W)	Total	Dynamic	Quiescent
	1.065	0.000	1.065

Fig 5.2.9: power consumption for vertex low power

Table 5.2.2parameter comparison table

Parameter	Existed Turbo encoder	Proposed Turbo encoder
No of Lut's	2,320	2,258

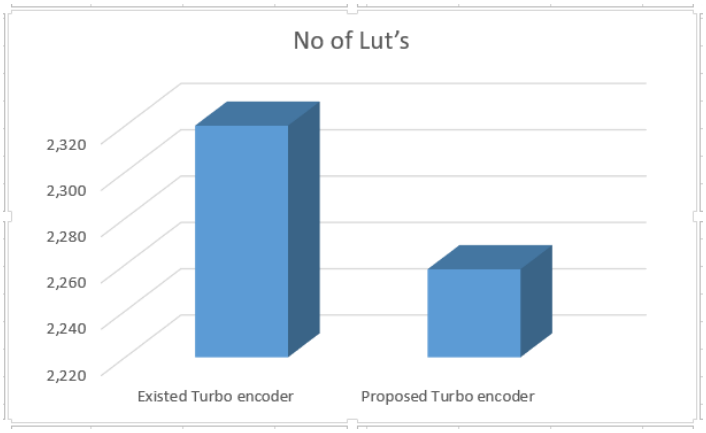


Fig 5.2.10: LUT comparison bar graph

Advantages And Disadvantages

Advantages

The proposed design turbo encoder with parallel computation using less number of clock pulses to getting output. Infact 50% of the power should be clock pulse only, so the parallel computation uses less power compared to serial computation because serial computation method uses more clock pulses compared to parallel computation and area also reduced by using carry increment adder for counter design in turbo encoder module.

Disadvantages

The Circuit complexity of proposed design turbo encoder is little bit more.

Applications

Turbo encoders are used in minimum and less error rate communication mediums. Turbo encoder has also opened up a new way of thinking in the construction of communication algorithms. The Turbo encoders are used in low latency designs

CONCLUSION

The Turbo encoder module using carry increment adder is designed and implemented to be an embedded module in the IVS modem. FPGA technologies are employed to develop the Turbo encoder module. Xilinx tools and Verilog HDL are employed to design and simulate the module. Both serial and parallel computation techniques are studied for the encoding process. It is shown that the parallel computation can improve the chip size and processing time of the module. Comparing with the serial computation technique, the parallel computation encoding, using only 22 clock pulses and it improves the processing time by and logic utilization by 9218 clock pulses. Additionally the proposed structure reduced area and power also. The processing time enhancement can be seen in both simulation and analyzing the chip processing.

FUTURESCOPE

Turbo encoders are error-correcting codes with performance close to the Shannon theoretical limit [SHA]. The encoder is formed by the parallel concatenation of two convolution codes separated by an interleaver or permuter.In this project, I modeled and implemented a turbo encoder using magnitude comparator scheme using serial and parallel computing algorithm. The advantage of turbo codes over existing coding schemes is that it attains a very low BER at low signal-to-noise ratios,so it will be used in used in communications . This makes it suitable for wireless applications where low transmission power is desired. However, the performance of turbo encodes on Rayleigh and Ricean fading channels remain an active subject of research. The portability of this code to Advanced Design System (ADS) would be very beneficial for code re-usability and also the synthesis capability of ADS would result in faster product development.

BIBBLIOGRAPHY

[1] “eCall data transfer; in-band modem solution; general description,” 3GPP,Tech. Rep. TS26.267.

[2] Eroupean Commission, “eCall: Time saved / lives saved,” Press Release,Brussels, August 21, 2015. website: http : //ec:europa:eu/digital –agenda/en/ecall - time - saved - lives - saved.

[3] A. Saleem et at. “Four-Dimensional Trellis Coded Modulation for Flexible Optical Communications,” IEEE Journal of Lightwave Technology., vol. 35, no. 2, pp. 151-158, Nov. 2017.

[4] C. Studer, C. Benkeser, S. Belfanti, and Q. Huang “Design and Implementationof a Parallel Turbo-Decoder ASIC for 3GPP-LTE,” IEEE Journal of Solid-state Circuits., vol. 46, no. 1, pp. 8-17, Jan. 2011.

[5] M. Nader and J. Liu, “Design and implementation of CRC module of eCall in-vehicle system on FPGA,” SAE Technical 2015-01-2844, 2015,doi:10.4271/2015-01-2844.

[6] M. Nader and J. Liu. “Developing modulator and demodulator for the EU eCall in-vehicle system in FPGAs” in IEEE, 2016 International Conference on Computing, Networking and Communications (ICNC),Hawaii, USA, Feb. 15-18, 2016, pp. 1-5.

[7] M. Nader and J. Liu. “FPGA Design and Implementation of Demodulator/Decoder Module for the EU eCall In-Vehicle System” in International Conference on Embedded Systems and Applications (ESA’15), Las Vegas,USA, July 27-30, 2015, pp. 3-9.

[8] “Technical Specification Group Radio Access Network; Multiplexing and channel coding (FDD),” 3GPP, Tech. Rep. TS22.212.

[9] D. Viktor, K. Michal, and D. Milan“Impact of trellis termination on performance of turbo codes,” in ELEKTRO, 2016, pp.48-51.

[10] B. Muralikrishna, G.L. Madhumati, H. Khan, K.G. Deepika, “Reconfigurable system-on-chip design using FPGA,” in 2nd International Conference on Devices, Circuits and Systems (ICDCS), 2014, pp.1-6.

[11] J. J. Rodriguez-Andina, M. J. Moure, and M. D. Valdes, “Features, design tools, and application domains of FPGAs,” IEEE Trans. Ind. Electron., vol. 54, no. 4, pp. 18101823, Aug. 2007.

DESIGN OF TURBO ENCODER FOR IN- VEHICLE SYSTEM

Ms. G. ANITHA CHOWDARY

Assistant Professor

Dept. OF ECE

Tkr college engineering and technology, Meerpet,Hyderabad

Miryala Siddartha, Muthineni Sai Jagadeesh, Nallagatla Nivas

Dept. OF ECE

Sidduchintu143@gmail.com, *saijagadeeshmuthineni@gmail.com*, *chinninallagatla2002@gmail.com*

ABSTRACT

This project study design of the Turbo encoder in the in-vehicle system (IVS). Developing the parallel computation method using carry increment adder, it is shown that both chip size and processing time are improve. The logic utilization is enhance by reduce area. The Turbo encoder module designing, simulating, and synthesing using Xilinx tools. Xilinx vertex low power is employ. The Turbo encoder module design to be a part of the IVS chip on a single programmable device.

The EU eCall system, mandated since March 2018, facilitates immediate communication between vehicles and emergency centers post-accident. Key components include the in-vehicle system (IVS), public safety answering point (PSAP), and cellular communication channel. The IVS, equipping with a Turbo encoder, automatically activates a data channel upon a collision, sending crucial data like GPS coordinates and VIN number to the nearest PSAP within 4 seconds. The Turbo encoder, employing a parallel concatenated convolutional code (PCCC), uses two constituent encoders with eight states and a 1/3 code rate to enhance digital communication reliability. The encoded data structure involves 1148 input bits, generating 3456 output bits with the influence of the Turbo encoder's thrills structure. The PCCC incorporates a 3GPP-designed interleaver technique for improved performance.

INTRODUCTION

The European eCall system, mandated for implementation by March 2018, serves as a crucial telematics solution aimed at enhancing emergency response in vehicle accidents. This governmental initiative ensures the establishment of an immediate voice and data channel between vehicles and

emergency centers post-accident. Comprising essential components such as the in-vehicle system (IVS),

Public safety answering point (PSAP), and cellular communication channel, the eCall system operates seamlessly to facilitate rapid response in critical situations. Upon detecting a vehicular accident, the IVS automatically triggers the data channel, swiftly collecting vital information known as the minimum set of data (MSD). This includes GPS coordinates, vehicle identification number (VIN), and other pertinent details necessary for prompt emergency assistance. Within a mere 4 seconds, the IVS transmits the MSD via cellular communication to the nearest PSAP. Subsequently, the PSAP coordinates the dispatch of emergency teams to the accident location, ensuring swift and effective response to mitigate potential harm.

Within the IVS modem, multiple modules are dedicated to processing the MSD signal. This intricate system, illustrated in Figure 1, is designed to optimize the handling of data transmissions. Notably, a Turbo encoder serves as a crucial component, functioning as a forward error correcting (FEC) mechanism. Leveraging advanced digital data encoding techniques, the Turbo encoder plays a pivotal role in enhancing the bit error rate (BER) within digital communications. Among the various modules integrated into the IVS, several key functions are encapsulated within the IVS chip itself. These include the cyclic redundancy check (CRC), modulator, and demodulator-decoder modules, all meticulously crafted to operate seamlessly within the IVS architecture. To ensure efficiency and reliability, these modules are meticulously projected and implemented on a dedicated FPGA device, offering robust embedded solutions for the IVS's signal processing requirements.

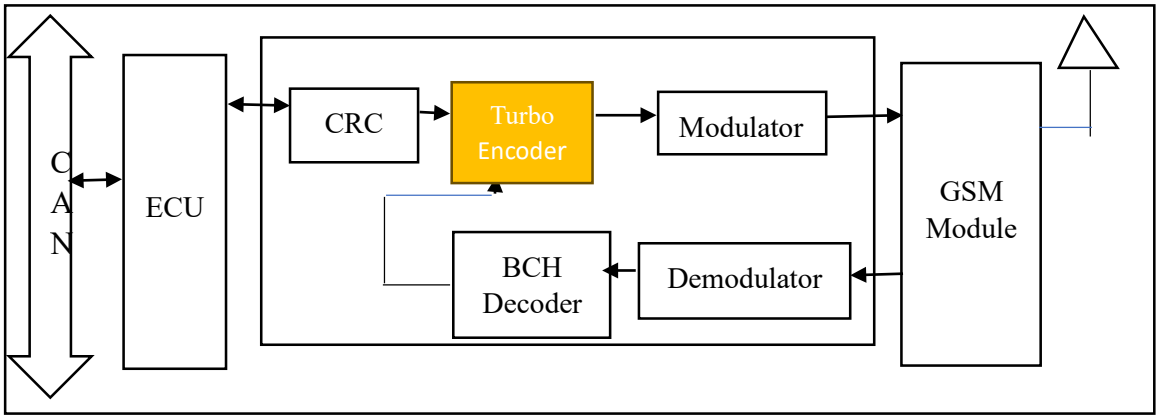


Fig: IVS Block Diagram

REVIEW OF LITERATURE SURVEY

Hardware Development of the In-vehicle system modules Emergency Call DOI: 10.12691/ajeee-6-1-1 Copyright © 2018 Science and Education Publishing. Hardware

Development of the In-Vehicle System Modules for the EU Emergency Call. *American Journal of Electrical and Electronic Engineering*. 2018; 6(1):1-10. doi: 10.12691/ajeee-6-1-1

Performance Enhancement of SOVA Based Decoder in SCCC and PCCC Schemes**.** Ahmed A. Hamad Department of Electrical Engineering, University of Babylon, Babel, This study proposes a simple scaling factor approach to improve the performance of parallel-concatenated convolutional code (PCCC) and serial concatenated convolutional code (SCCC) systems based on suboptimal soft-input soft-output (SISO) decoders Design and Implementation of a Parallel Turbo-Decoder ASIC for 3GPP-LTE Christoph Studer, Student Member, IEEE, Christian Benkeser, Member, IEEE, Sandro Belfanti, and Quiting Huang, Fellow, IEEE.Turbo-decoding for the 3GPP-LTE (Long TermEvo-lution) wireless commu- -nication standard is among the most chal-lenging tasks in terms of computational comp -lexity and power con-sumption of corresponding cellular devices

Design and Implementation of CRC Module of eCall In-Vehicle System on FPGA ISSN: 0148-7191, eISSN: 2688-3627 DOI: https://doi.org/10.4271/2015-01-2844 Published September 29, 2015 by SAE International in United States**.**

Impact of trellis termination on performance of turbo codes.Viktor Durcek, M. Kuba, M. Dado Published in ELEKTRO 16 May 2016 Computer Science, Engineering2016 ELEKTRO. TLDR Various types of trellis termination are described for improving performance of turbo codes by periodically adding tail bits into information sequence and performance of a turbo code without trellis termination is compared

Turbo Encoder Module of Existed design

The IVS employs a Turbo encoder module with 1/three code rate. The enter sign of the rapid encodes

is the MSD data. The block duration of the MSD statistics is 1148 bits. The output of the module is the MSD encode data in binary. Implementing the rapid coding method with 1/three coding rate, the duration of the output is 3456 bits. The Turbo encoder employs a parallel concatenated convolutional code (PCCC). The period of the enter data, parity1, and parity2 are 1148 bits. There are 12 bits of the tail bits, They are pushed from the shift sign in feedback. The tail bits are carried out for give up factors among the encoded information blocks.

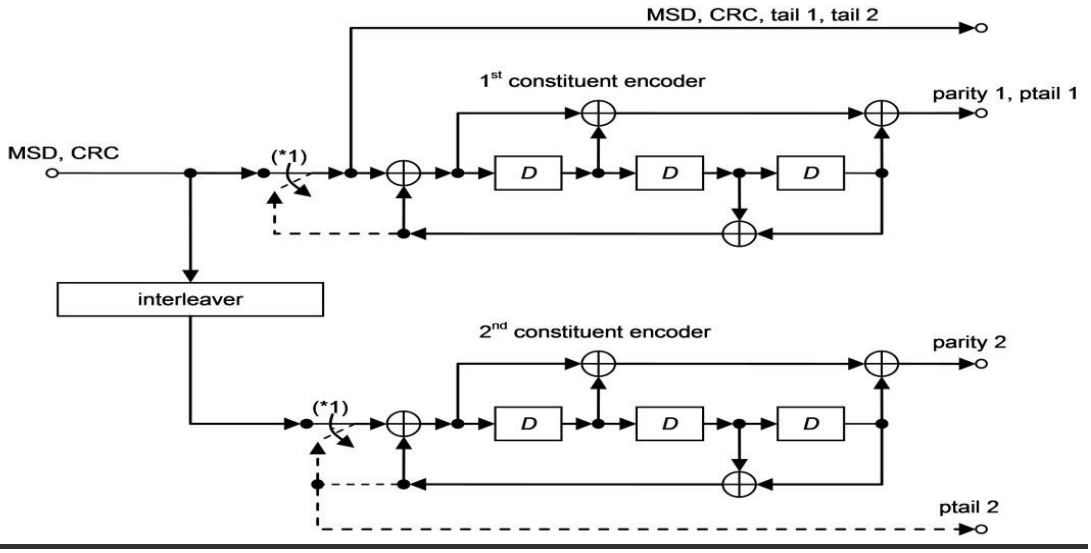


Fig: The structure of the Turbo encoder

The Turbo encoder outputs encoded data with a structure that includes tail bits, which are derived from feedback in the shift register. These tail bits consist of 12 bits and are applied at the endpoints between the encoded data blocks. In total, the input data, along with the two sets of parity bits (parity1 and parity2), spans 1148 bits.

Fig: The output buffer of the Turbo encoder

Turbo Encoder Module for Proposed Design:

In this module we are adding the carry increment adder to the existed module for reducing the LUT'S and Power consumption of the Turbo Encoder Module.

Interleaver

Denote the transfer function of the employed PCCC as:

$$G(D) = \left(1, \frac{g_1(D)}{g_0(D)}\right) \quad (1)$$

where

$$g_1(D) = 1 + D^2 + D^3$$

$$g_0(D) = 1 + D + D^3$$

CARRY INCREMENT ADDER:

An 8-bit increment adder employs a combination of two 4-bit Ripple Carry Adders (RCAs) to achieve its operation. The first RCA handles the addition of the initial 4-bit inputs, producing partitioned sum and carry outputs. The carry-out signal from this first RCA serves as the carry-in input for the subsequent conditional increment block.

In the conditional increment block, the carry-out signal from the first RCA determines whether an

increment operation is necessary. This block utilizes half adders to perform the increment operation based on the carry-out value from the first RCA. The second RCA, regardless of the output from the first RCA, then proceeds with the addition operation.

The input carry-in to the first RCA block is set to a low value consistently. The conditional increment block comprises half adders, which execute the increment operation based on the carry-out value from the first RCA. The output sum from the second RCA is obtained through the conditional increment block.

This design configuration ensures that the 8-bit increment adder efficiently handles addition operations while incorporating conditional increment logic based on the carry-out signal from the first RCA. The overall schematic of the Carry Increment Adder provides a comprehensive visualization of its operation and interconnection of components.

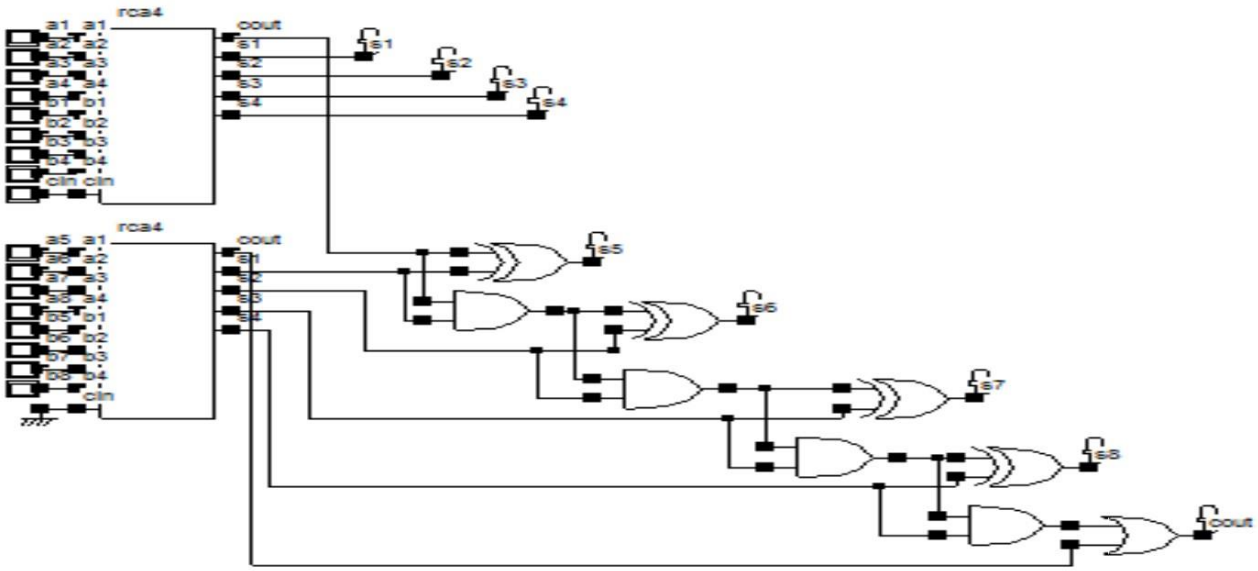


Fig: carry increment adder

RESULT

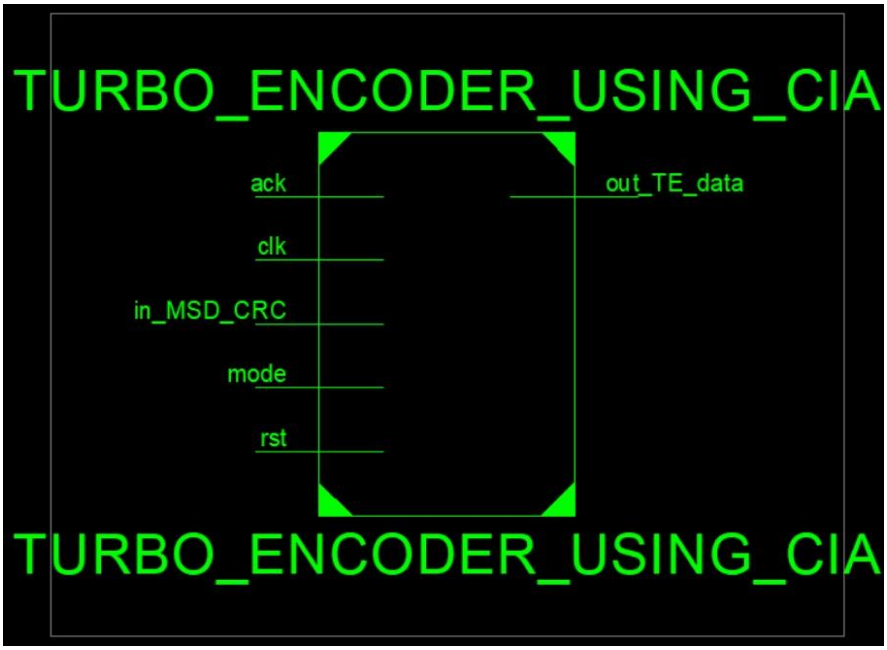


Fig: RTL Schematic of turbo encoder using CIA

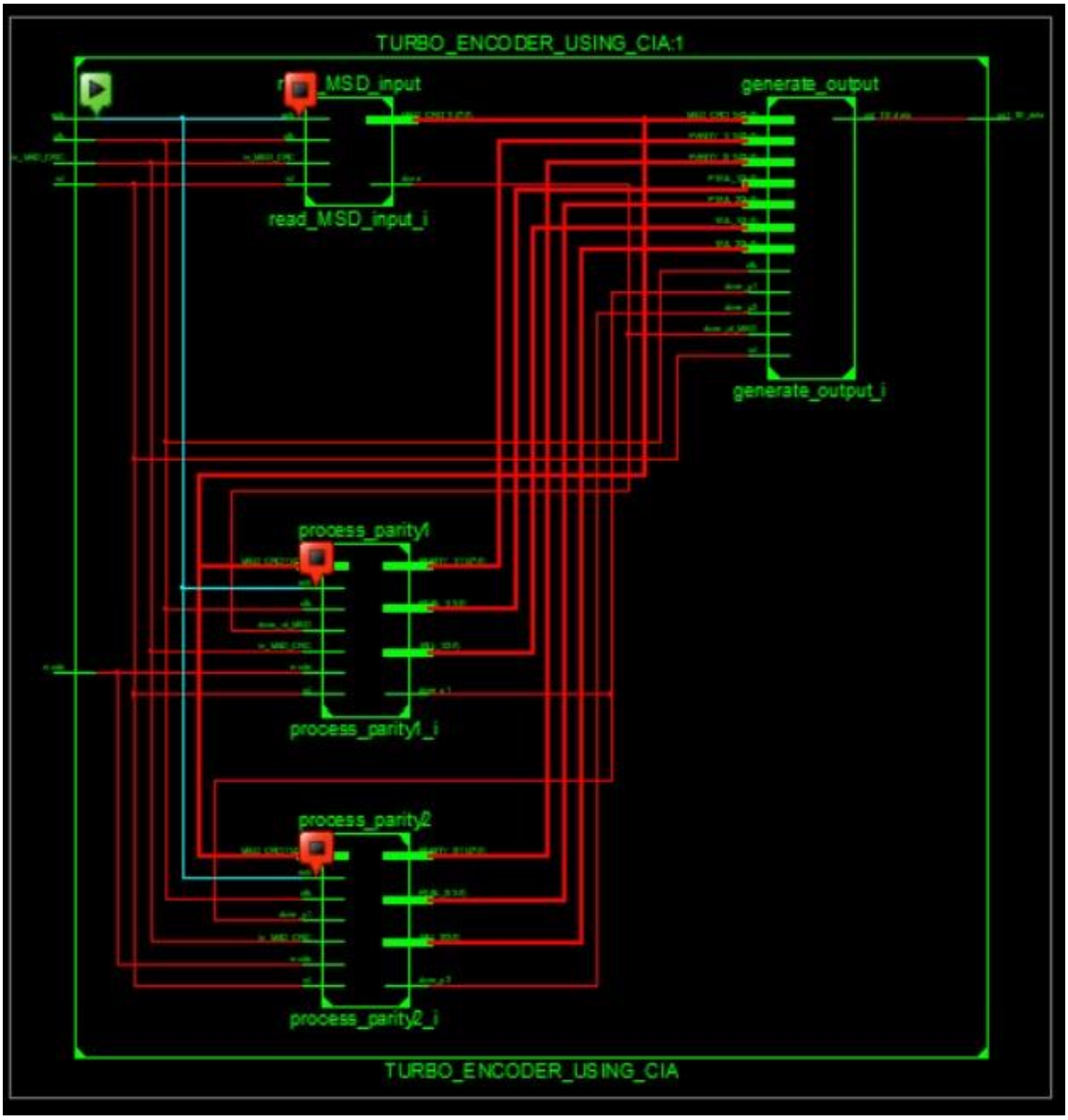


Fig: Internal structure of RTL schematic of turbo encoder using CIA

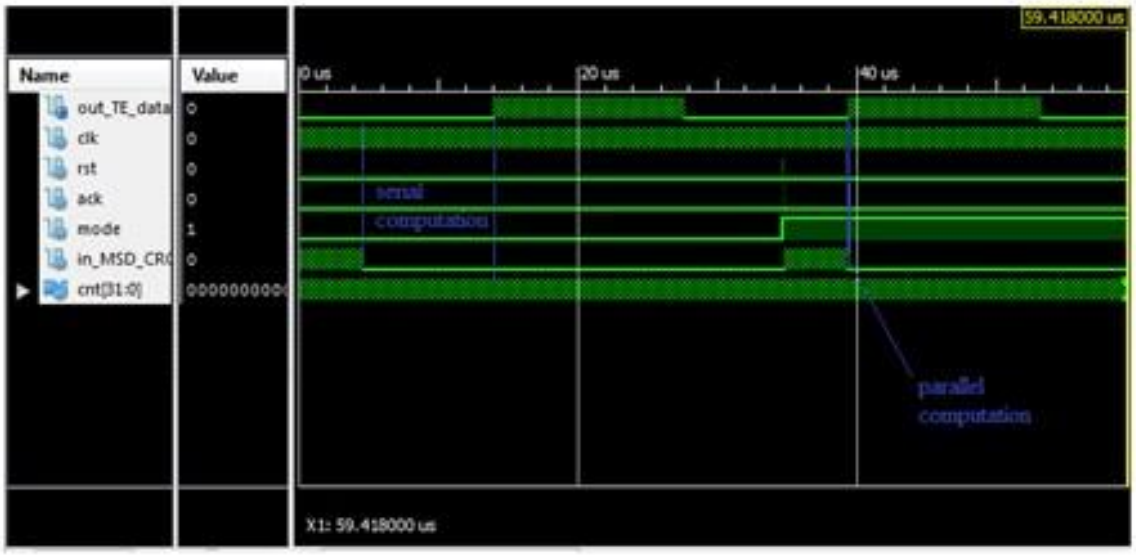
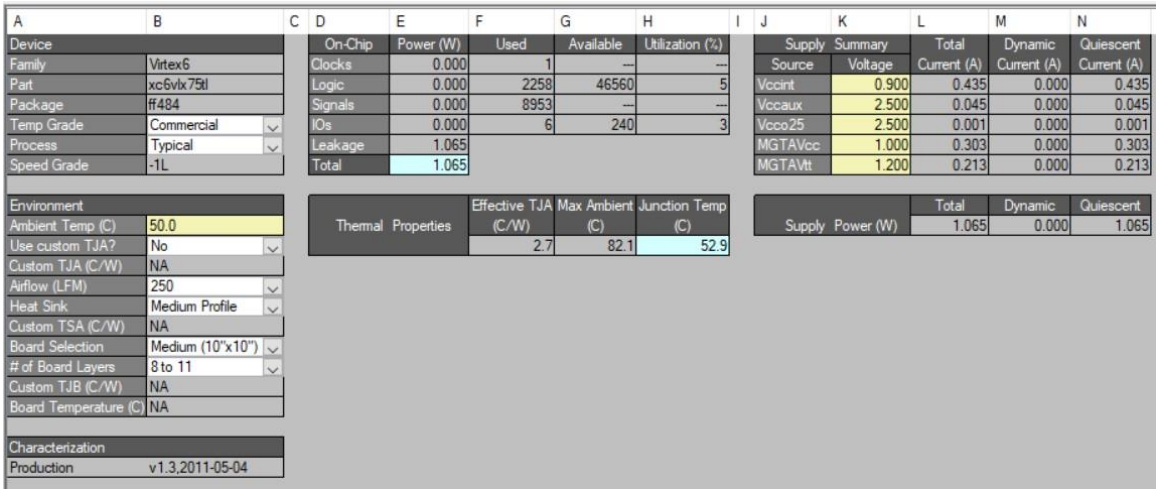


Fig: simulated wave form of turbo encoder

Parameter	Existed Turbo encoder	Proposed Turbo encoder
No of Lut's	2,320	2,258



Device	
Family	Virtex6
Part	xc6vlx75tl
Package	ff484
Temp Grade	Commercial
Process	Typical
Speed Grade	-1L

Environment	
Ambient Temp (C)	50.0
Use custom TJA?	No
Custom TJA (C/W)	NA
Airflow (LFM)	250
Heat Sink	Medium Profile
Custom TSA (C/W)	NA
Board Selection	Medium (10"x10")
# of Board Layers	8 to 11
Custom TJB (C/W)	NA
Board Temperature (C)	NA

Characterization	
Production	v1.3,2011-05-04

On-Chip	Power (W)	Used	Available	Utilization (%)
Clocks	0.000	1	---	---
Logic	0.000	2258	46560	5
Signals	0.000	8953	---	---
IOs	0.000	6	240	3
Leakage	1.065			
Total	1.065			

Thermal Properties	Effective TJA (C/W)	Max Ambient (C)	Junction Temp (C)
	2.7	82.1	52.9

Supply Summary Source	Voltage	Total Current (A)	Dynamic Current (A)	Quiescent Current (A)
Vccint	0.900	0.435	0.000	0.435
Vccaux	2.500	0.045	0.000	0.045
Vcco25	2.500	0.001	0.000	0.001
MGTAVcc	1.000	0.303	0.000	0.303
MGTAVtt	1.200	0.213	0.000	0.213

Supply Power (W)	Total	Dynamic	Quiescent
	1.065	0.000	1.065

Fig: power consumption for vertex low power

CONCLUSION

The Turbo encoder module using carry increment adder is designed and implemented to be an embedded module in the IVS modem. FPGA technologies are employed to develop the Turbo encoder module. Xilinx tools and Verilog HDL are employed to design and simulate the module. Both serial and parallel computation techniques are studied for the encoding process. It is shown that the parallel computation can improve the chip size and processing time of the module. Comparing with the serial computation technique, the parallel computation encoding, using only 22 clock pulses and it improves the processing time by and logic utilization by 9218 clock pulses. Additionally the proposed structure reduced area and power also. The processing time enhancement can be seen in both simulation and analyzing the chip processing

REFERENCES

[1] "eCall data transfer; in-band modem solution; general description," 3GPP,Tech. Rep. TS26.267.

[2] Eroupean Commission, "eCall: Time saved / lives saved," Press Release,Brussels, August 21, 2015. website: http : //ec:europa:eu/digital –agenda/en/ecall - time - saved - lives - saved.

[3] A. Saleem et at. "Four-Dimensional Trellis Coded Modulation for Flexible Optical Communications," IEEE Journal of Lightwave Technology., vol. 35, no. 2, pp. 151-158, Nov. 2017.

[4] C. Studer, C. Benkeser, S. Belfanti, and Q. Huang "Design and Implementationof a Parallel Turbo-Decoder ASIC for 3GPP-LTE," IEEE Journal of Solid-state Circuits., vol. 46, no. 1, pp. 8-17, Jan. 2011.

[5] M. Nader and J. Liu, "Design and implementation of CRC module of eCall in-vehicle system on FPGA," SAE Technical 2015-01-2844, 2015,doi:10.4271/2015-01-2844.

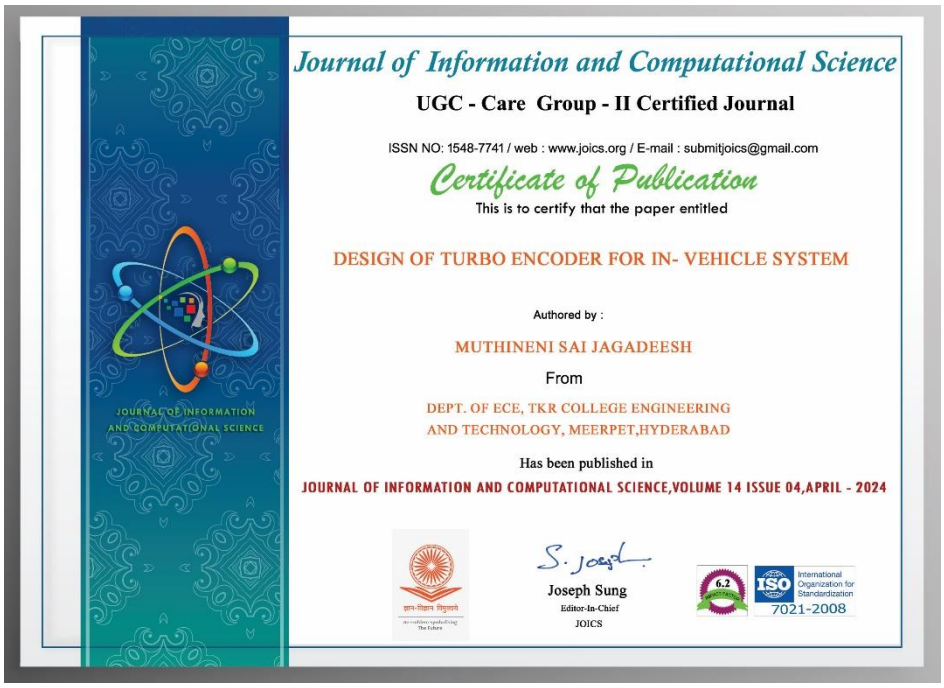
[6] M. Nader and J. Liu. "Developing modulator and demodulator for the EU eCall in-vehicle system in FPGAs" in IEEE, 2016 International Conference on Computing, Networking and Communications (ICNC),Hawaii, USA, Feb. 15-18, 2016, pp. 1-5.

[7] M. Nader and J. Liu. "FPGA Design and Implementation of Demodulator/Decoder Module for the EU eCall In-Vehicle System" in International Conference on Embedded Systems and Applications (ESA'15), Las Vegas,USA, July 27-30, 2015, pp. 3-9.

[8] "Technical Specification Group Radio Access Network; Multiplexing and channel coding (FDD)," 3GPP, Tech. Rep. TS22.212.

[9] D. Viktor, K. Michal, and D. Milan"Impact of trellis termination on performance of turbo codes," in ELEKTRO, 2016, pp.48-51.

[10] B. Muralikrishna, G.L. Madhumati, H. Khan, K.G. Deepika, "Reconfigurable system-on-chip design using FPGA," in 2nd International Conference on Devices, Circuits and Systems (ICDCS), 2014, pp.1-6. [11] J. J. Rodriguez-Andina, M. J. Moure, and M. D. Valdes, "Features, design tools, and application domains of FPGAs," IEEE Trans. Ind. Electron., vol. 54, no. 4, pp. 18101823, Aug. 2007.



Journal of Information and Computational Science

UGC - Care Group - II Certified Journal

ISSN NO: 1548-7741 / web : www.joics.org / E-mail : submitjoics@gmail.com

Certificate of Publication

This is to certify that the paper entitled

DESIGN OF TURBO ENCODER FOR IN- VEHICLE SYSTEM

Authored by :

MUTHINENI SAI JAGADEESH

From

DEPT. OF ECE, TKR COLLEGE ENGINEERING
AND TECHNOLOGY, MEERPET,HYDERABAD

Has been published in

JOURNAL OF INFORMATION AND COMPUTATIONAL SCIENCE,VOLUME 14 ISSUE 04,APRIL - 2024

Joseph Sung
Editor-In-Chief
JOICS

ISO International Organization for Standardization 7021-2008

Journal of Information and Computational Science

UGC - Care Group - II Certified Journal

ISSN NO: 1548-7741 / web : www.joics.org / E-mail : submitjoics@gmail.com

Certificate of Publication

This is to certify that the paper entitled

DESIGN OF TURBO ENCODER FOR IN- VEHICLE SYSTEM

Authored by :

MIRYALA SIDDARTHA

From

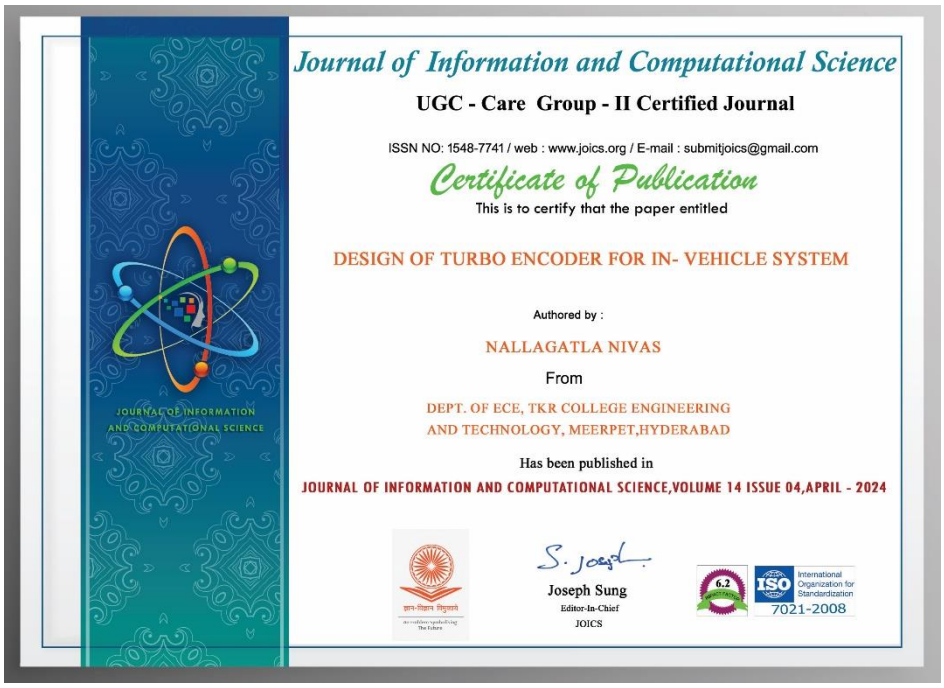
DEPT. OF ECE, TKR COLLEGE ENGINEERING
AND TECHNOLOGY, MEERPET,HYDERABAD

Has been published in

JOURNAL OF INFORMATION AND COMPUTATIONAL SCIENCE,VOLUME 14 ISSUE 04,APRIL - 2024

Joseph Sung
Editor-In-Chief
JOICS

International Organization for Standardization
7021-2008



Journal of Information and Computational Science

UGC - Care Group - II Certified Journal

ISSN NO: 1548-7741 / web : www.joics.org / E-mail : submitjoics@gmail.com

Certificate of Publication

This is to certify that the paper entitled

DESIGN OF TURBO ENCODER FOR IN- VEHICLE SYSTEM

Authored by :

NALLAGATLA NIVAS

From

DEPT. OF ECE, TKR COLLEGE ENGINEERING
AND TECHNOLOGY, MEERPET,HYDERABAD

Has been published in

JOURNAL OF INFORMATION AND COMPUTATIONAL SCIENCE,VOLUME 14 ISSUE 04,APRIL - 2024

Joseph Sung
Editor-In-Chief
JOICS

ISO International Organization for Standardization 7021-2008