B.M.S. COLLEGE OF ENGINEERING
Bengaluru-560019.

Autonomous College, affiliated to
Visvesvaraya Technological University, Belgaum

Project Work-I
REPORT

ON

"FPGA-BASED IMPLEMENTATION OF ENHANCED ERROR DETECTION CORRECTION ENCODER AND DECODER FOR SINGLE-BIT ERROR DETECTION AND CORRECTION"

Submitted in partial fulfilment of the requirement for completion of
PROJECT WORK –I [22EC5PWPJ1]

Submitted by

Manaswini S M Gowda	**1BM21EC075**
Naipunya Kiran	**1BM21EC081**
Nandan P Kashyap	**1BM21EC082**
Rachana Krishna Kulkarni	**1BM21EC119**

Under the guidance of

Dr. Archana H R

Assistant Professor

Academic Year

2023 – 2024

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

BMS College of Engineering
Bull Temple Road, Basavanagudi, Bangalore-560019

BMS COLLEGE OF ENGINEERING

Autonomous college, affiliated to VTU

Bull Temple Road, Bengaluru – 560 019

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

CERTIFICATE

This is to certified that the Project work -I entitled **“FPGA-Based Implementation of Enhanced Error Detection Correction Encoder and Decoder for Single-Bit Error Detection and Correction”** is a bonafide work carried out by **Manaswini S M Gowda (1BM21EC075), Naipunya Kiran (1BM21EC081), Nandan P Kashyap (1BM21EC082) and Rachana Krishna Kulkarni (1BM21EC119)** submitted in partial fulfilment of the requirement for completion of PROJECT WORK - I [22EC5PWPJ1] of Bachelor of Engineering in Electronics and Communication during the academic year 2023-24. The Project Work - I report has been approved as it satisfies the academic requirements.

Guide	**Head of Department**
Dr. Archana H R	**Dr. Siddappaji**
Assistant Professor	Professor and Head
Department of ECE	Department of ECE
BMS College of Engineering	BMS College of Engineering

Dr. Muralidhara S
Principal
BMS College of Engineering

External Viva

Name of the Examiners	Signature with Date
1.	
2.	

DECLARATION

We, **Manaswini S M Gowda (1BM21EC075), Naipunya Kiran (1BM21EC081),Nandan P Kashyap (1BM21EC082) and Rachana Krishna Kulkarni (1BM21EC119)** hereby declare that the Project Work -I entitled **"FPGA-Based Implementation of Enhanced Error Detection Correction Encoder and Decoder for Single-Bit Error Detection and Correction"** is a bonafide work and has been carried out by us under the guidance of **Dr. Archana H R,** Assistor Professor**,** Department of Electronics and Communication Engineering, BMS College of Engineering, Bengaluru submitted in partial fulfilment of the requirement for completion of PROJECT WORK - I [22EC5PWPJ1] of Bachelor of Engineering in Electronics and Communication during the academic year 2023-24. The Project Work -I report has been approved as it satisfies the academic requirements in Electronics and Communication engineering, Visvesvaraya Technological University, Belagavi, during the academic year 2023-24.

We further declare that, to the best of our knowledge and belief, this Project Work -I has not been submitted either in part or in full to any other university.

Place: Bengaluru

Data:05/03/2024

Manaswini S M Gowda: 1BM21EC075
Naipunya Kiran: 1BM21EC081
Nandan P Kashyap: 1BM21EC085
Rachana Krishna Kulkarni: 1BM21EC119

ACKNOWLEDGEMENTS

We take this opportunity to express our profound gratitude to the respected principal Dr. S. Muralidhara, BMS College of Engineering for providing a congenial environment to work in. Our sincere gratitude to Dr. Siddappaji, Head of the Department, Electronics and Communication Engineering for encouraging and providing opportunity to carry Project Work -I in the department.

We heartfully thank our guide Dr. Archana H R for the guidance and constant encouragement throughout the course of this Project Work -I without which this project would not be successful.

A number of personalities, in their own capacities have helped us in carrying out this Project Work -I. We would like to take this opportunity to thank them all.

Last but not the least, we thank our friends and family for their encouragement and help in accomplishing the objective of the Project Work -I work.

1) Manaswini S M Gowda: 1BM21EC075
2) Naipunya Kiran: 1BM21EC081
3) Nandan P Kashyap: 1BM21EC085
4) Rachana Krishna Kulkarni: 1BM21EC085

ABSTRACT

In packet-switched communication systems, ensuring error-free transmission is paramount for maintaining data integrity. Additional check bits, appended to input data as headers or payloads, aid in error detection. These bits, generated from the input message using a deterministic algorithm post-processing, are crucial for identifying transmission errors. The receiver system implements this algorithm to detect errors, while the transmitter uses it to gauge transmission reliability. When errors are detected, the receiver corrects and recovers corrupted bits, ensuring message integrity. To enhance error detection and correction, an advanced error detection correction code implementation was proposed. Digital encoding and decoding techniques are fundamental components of modern communication systems, enabling efficient and reliable data transmission. Encoding transforms data into a format suitable for transmission, while decoding reconstructs the original data at the receiver end. Various encoding methods, such as Manchester and NRZ encoding, offer distinct advantages tailored to specific communication scenarios. Similarly, decoding techniques, including error detection and correction algorithms, ensure data integrity during transmission.

Both encoding and decoding implementations were synthesized on the Xilinx Basys 3 platform, affirming the practical feasibility of the proposed algorithm for robust error detection and correction in packet-switched communication system. The logic implemented in the paper encodes input data bits using an algorithm that combines elements of Hamming and CRC techniques. Moreover, the paper presents a decoder code capable of detecting and rectifying a single error within the code word.

TABLE of CONTENTS

List of figures

List of abbreviations

Abbreviation	Full form
EEDC	Enhanced Error Detection and Correction Codes
FPGA	Field-Programmable Gate Array
CRC	Cyclic Redundancy Check
SED	Single Error Correction
DED	Double Error Detection
LDPC	Low-Density Parity-Check

Chapter 1: Introduction

Error detection and correction play a crucial role in ensuring the reliability and integrity of digital communication systems. In today's interconnected world, where data transmission occurs across various channels and platforms, the presence of errors can lead to significant disruptions and compromises in the accuracy of transmitted information. Enhanced Error Detection and Correction Codes (EEDC) represent a sophisticated approach to mitigating errors in data transmission, offering advanced mechanisms for identifying and correcting errors that occur during the communication process. At the heart of EEDC lies the imperative need for robust error detection and correction algorithms capable of handling various types of errors, including single-bit errors, which are among the most common in digital communication systems. Single-bit errors occur when only one bit in a data stream is corrupted during transmission, leading to potential inaccuracies or data loss if left undetected and uncorrected.

The implementation of a decoder algorithm tailored specifically for single-bit error detection and correction within EEDC is of paramount importance in ensuring the reliability and efficiency of digital communication systems. This decoder algorithm serves as a critical component in the error correction process, enabling the identification and rectification of single-bit errors to maintain data integrity and minimize the impact of transmission errors on system performance. In this context, this project focuses on the FPGA implementation of such a decoder algorithm, leveraging the principles of Hamming codes to enhance error detection and correction capabilities within EEDC. By developing and implementing a novel decoder algorithm, this project aims to address the challenges associated with single-bit error detection and correction in digital communication systems, thereby contributing to the advancement of error correction techniques and improving the overall reliability of data transmission processes.

Error detection and correction techniques are fundamental components of modern digital communication systems, ensuring the integrity and reliability of transmitted data. Traditional methods such as parity checking have long been employed to detect errors in data streams by appending an additional parity bit to the transmitted data. While effective in detecting errors, parity checking has limitations, particularly in its ability to correct errors or handle multiple-bit errors. To address the shortcomings of traditional methods, more advanced approaches have been developed, including Hamming codes and Cyclic Redundancy Check (CRC) codes. Hamming codes, named after Richard W. Hamming, are a class of linear error-correcting codes that introduce redundancy into the transmitted data through the addition of check bits. These check bits enable the detection and correction of single-bit errors and the detection of certain types of multiple-bit errors, thereby enhancing the reliability of data transmission. CRC codes, on the other hand, utilize polynomial division to generate a checksum, which is appended to the transmitted data. The receiver performs a similar polynomial division using the received data and compares the remainder with the checksum to detect errors. CRC codes are widely used in various communication protocols and storage systems due to their efficiency and robust error detection capabilities.

Despite the advancements offered by Hamming codes and CRC codes, existing error detection and correction methods still have limitations. Traditional methods like parity checking are unable to correct errors, while more advanced techniques like Hamming codes and CRC codes may struggle to handle certain types of errors or require significant computational resources for decoding. This highlights the need for enhanced error detection and correction codes that can address the limitations of existing methods and provide more robust error correction capabilities. Enhanced Error Detection and Correction Codes (EEDC) aim to fulfil this need by integrating advanced error correction techniques with efficient encoding and decoding algorithms.

The implementation of EEDC involves the development of customized encoder and decoder algorithms tailored to the specific requirements of the communication system. By leveraging the principles of Hamming codes, CRC codes, and other error correction techniques, EEDC offers improved error detection and correction capabilities, thereby enhancing the reliability and efficiency of digital communication systems.

While traditional error detection and correction methods have laid the foundation for reliable data transmission, the evolution of communication technologies necessitates the development of more advanced and efficient error correction techniques. Enhanced Error Detection and Correction Codes represent a promising approach to addressing the limitations of existing methods and ensuring the integrity of data transmission in modern communication systems.

Chapter 2: Literature survey

The proposed system uses an algorithm for both transmitter and receiver to enhance information reliability by identifying and correcting error bits. Introducing Enhanced Error Detection Correction (EEDC) codes, derived from modified Hamming codes, improves error detection and correction beyond traditional cyclic redundancy checking (CRC) and Hamming codes. The algorithm focuses on minimizing redundancy bits ('r') in the polynomial generator for CRC and reducing overhead in Hamming code. The EEDC method exhibits a faster transmission rate and superior random error detection compared to CRC and Hamming codes. For each valid codeword of C bits comprises valid input data bits Di, allowing for the generation of C + 1 possible codewords conforming to a valid data entity. [1]

The paper presents a VLSI design for error detection and correction in wireless communication using parity check codes and Hamming codes. The proposed method aims to ensure error-free data transmission in cognitive radio technology. Various encoding and decoding methods are implemented, and simulation results are provided. The paper discusses the different parity check code methods, such as horizontal, vertical, and diagonal parity, as well as the HVDD parity check code. The results show the memory usage, timing, code rate, bit overhead, and error correction capabilities of each method. The proposed method offers an efficient and reliable solution for error detection and correction in wireless communication systems. [2]

The paper discusses the implementation of error correction techniques in memory applications. Soft errors in SRAM memories, caused by radiation, can lead to single cell and multiple cell upsets. The paper introduces the (7,4) Hamming code and the (8,4) SEC-DED code as error correction codes. However, these codes have limitations in terms of error detection and correction. To address these limitations, the paper proposes the (14,8) SEC-DED-DAEC code, which can detect and correct errors adjacently up to 2 bits. The encoding and decoding processes for these codes are studied and verified through simulations. The proposed code offers improved error coverage and reliability in memory systems. [3]

The paper explores fault-tolerant systems, emphasizing Modular Redundancy (MR) for reliability and timing constraints in real-time applications. N-modular redundancy (NMR) and duplex pairs enhance fault tolerance, with NMR systems utilizing parallel N-modules and duplex pairs employing duplicated modules. Triple Modular Redundancy (TMR) is widely used, but traditional TMR has drawbacks, particularly in handling TMR failures. The paper introduces Fault Recoverable Triple Modular Redundancy (FRTMR), using a roll-forward approach to avoid re-computation and address TMR's shortcomings. FRTMR locates and removes latent faults using TMR modules, recovering the system from multiple faults by reusing scan chains. [4]

The paper extends Fujiwara's algorithm to compute the minimum distance of shortened cyclic codes, particularly focusing on cyclic redundancy-check codes (CRC codes). Using a specialized processor, the study explores various classes of CRC codes with 24 and 32 parity bits, identifying codes with maximum minimum distance (dmin, l) within specific block length ranges. Comparative analyses include dmin profiles of proposed codes by Merkey and Posner, along with the widely used IEEE-802 standard. Emphasis is placed on optimizing the dmin profile, as it significantly impacts the probability of undetectable errors on low-noise binary symmetric channels. The flexibility in factoring the generator polynomial is a notable feature of the extended algorithm. Results present dmin profiles and undetected error probabilities for the identified optimal CRC codes, offering insights into their performance on both low-noise and fairly noisy binary symmetric channels. The paper concludes with reflections on the practical utility of the proposed CRC codes. [5]

The paper discusses the implementation of the Hamming code in the Modbus industrial protocol for error detection and correction in a 3-DOF manipulator robot, utilizing XBee-based wireless communication. It highlights the significance of fault tolerance in industrial automation due to the high reliance on information transmission for robotic systems. The hardware setup involves a PC as the master, a pcDuino 3 as the slave, and XBee S2 modules for wireless communication. The Hamming encoder and decoder are implemented using Python, Modbus-tk, and Blender. The communication tests demonstrate error correction capabilities, ensuring the robot follows the proposed trajectory. The integration of Hamming code provides forward error correction without the need for automatic repeat requests or CRC protocols. Overall, the system offers a robust solution for reliable and effective communication in industrial robotics. [6]

The paper emphasizes the significance of Low-Density Parity-Check (LDPC) codes in error correction, particularly their efficiency near the Shannon limit. It explores challenges in LDPC implementation, specifically memory requirements, and introduces an FPGA-based decoding architecture with a hierarchical quasi-cyclic matrix. Various components, including Variable Node Decoder Memory and Check Node Decoder Memory, are detailed, highlighting a goal to enhance error correction in communication systems via Altera Cyclone II FPGA. [7]

The paper offers a comprehensive literature survey on redundancy encoding techniques. Beginning with repetition codes, the paper explores their use of bit repetitions to enhance error detection and correction, balancing reliability against information rate reduction. It then delves into single parity check codes, where one check digit represents the modulo-2 sum of K information digits, adept at detecting single bit errors but lacking intelligent error correction. Hamming codes, exemplified by the 7,4,1 code, capable of correcting single-bit errors, are examined alongside a truncated version, like the 6,3,1 code, with limitations in handling double errors. Extended Hamming codes, such as the 8,4,1 code, are introduced, incorporating an extra overall parity check equation to detect all double bit errors, providing error correction for single bit errors and detection for double errors. The hardware implementation for encoding Hamming codes using shift registers, exclusive OR gates, or read-only memory (ROM), and decoding strategies employing feedback shift registers and real-time correction mechanisms are discussed. Berlekamp's method is highlighted for its applicability to codes of any length, providing a linear increase in hardware. The paper comprehensively explores the mathematical foundations, practical implementations, and capabilities of redundancy encoding techniques in error detection and correction based on minimum Hamming distance. [8]

Error correction codes play a pivotal role in data transmission, ensuring accuracy by detecting and rectifying errors introduced during the process. Among these codes, block codes and convolution codes stand out, both incorporating redundancy through parity symbols. Cyclic redundancy check (CRC) codes, a subset of cyclic codes and linear block codes, provide a simple yet effective means of error detection. However, traditional serial CRC circuits, based on linear feedback shift registers (LFSRs), encounter latency issues, especially with bit-serial streams. To overcome this limitation, parallel CRC implementations have emerged as a viable solution, particularly in high-speed data networking scenarios. The paper presents a novel approach to parallel CRC implementation, leveraging unfolding, pipelining, and retiming algorithms to enhance performance. It addresses the inefficiencies of serial implementations for high-speed data transmission and proposes parallel processing as a solution. By introducing pipelining methods to reduce iteration bounds, the proposed design significantly enhances processing speed while managing or reducing hardware costs. Unfolding algorithms are then utilized to parallelize CRC architectures, increasing throughput by reducing iteration bounds and addressing latency concerns associated with high-speed data transmission.[9]

In the realm of error correction coding, particularly in the domain of low-density parity-check (LDPC) codes, researchers have explored various strategies to enhance error correction performance, especially for short block lengths and low code rates. Traditional LDPC code designs often encounter challenges in achieving low error rates, particularly in scenarios with limited block lengths. One approach that has gained attention is the integration of Hamming codes into LDPC code structures, known as "Hamming code doping." This technique aims to improve minimum distances and mitigate error-rate floors by incorporating more complex component codes within the LDPC graph. By leveraging the strengths of both LDPC and Hamming codes, researchers have demonstrated remarkable improvements in error correction performance, as evidenced by numerical simulations on the additive white Gaussian noise (AWGN) channel.

Recent literature has also delved into iterative decoding algorithms tailored to graphs with "Hamming nodes," offering insights into optimizing decoding processes for enhanced performance. These algorithms, adapted for the binary-input AWGN channel, play a crucial role in efficiently decoding doped LDPC codes by computing soft outputs based on received soft inputs. Additionally, density evolution analysis based on Gaussian approximation has emerged as a valuable tool for understanding decoding convergence properties and guiding the design of Hamming-doped LDPC codes. By tracking probability density functions of exchanged messages along the LDPC graph, researchers gain insights into optimal node distributions that lead to improved decoding thresholds and error rate performance. Overall, the literature survey highlights the promising advancements in error correction coding achieved through the integration of Hamming codes into LDPC structures, paving the way for more resilient and efficient communication systems.[10]

The paper titled "Enhanced Hamming Codes: Reducing Redundant Bit for Efficient Error Detection and Correction" presents a novel approach called Enhanced Hamming Codes (EHC). This method aims to minimize redundant bits while maintaining error detection and correction capabilities. By optimizing encoding techniques and error detection algorithms, the proposed approach achieves enhanced storage usage for data transmission. The research demonstrates practical implications in telecommunications, data storage systems, and network protocols, offering an efficient solution for reliable data transmission and storage.[11]

Recent advancements in portable devices for data communication demand low-power and high-speed digital circuits. Designing VLSI circuits in deep sub-micron technology to achieve better performance metrics introduces challenges in power consumption. Various design styles such as Pass Transistor Logic (PTL), Domino Logic (DL), and Transmission Gate Logic (TGL) have emerged to enhance digital circuit performance. Pass transistor logic is highlighted for minimizing power consumption, while other techniques focus on improving circuit performance in terms of low power, speed, and area. Low-power techniques like Gate Diffusion Input (GDI) are gaining traction, especially for applications requiring a high Signal-to-Noise Ratio (SNR). In digital data communication, the emphasis is on transmitting error-free data by minimizing interference and incorporating error control coding for error detection and correction. Hamming code is one such efficient error control code, and this paper presents a low-power Hamming code encoder and decoder design using GDI logic. The GDI technique offers advantages in reducing power consumption, chip area, and improving circuit speed compared to traditional CMOS logic. Simulation results demonstrate the efficacy of GDI logic in achieving low-power VLSI designs, making it a preferred choice for circuit designers.[12]

Chapter 3: Problem Analysis & Solution

3.1 Problem Definition

In modern data communication and storage systems, ensuring the integrity of transmitted and stored data is of paramount importance. However, conventional error detection and correction techniques may not always suffice, particularly in environments susceptible to single-bit errors. The problem addressed by this project lies in the need for robust error detection and correction mechanisms capable of efficiently identifying and rectifying single-bit errors in data transmissions or storage systems. Existing solutions may lack the speed, efficiency, or adaptability required for real-time applications or hardware-constrained environments.

The project aims to develop and implement an FPGA-based solution utilizing EEDC (Error Detection and Correction) encoding and decoding techniques. Specifically, the project seeks to overcome the following challenges:

- Efficient Hardware Implementation: Designing an FPGA-based solution that optimally utilizes hardware resources while achieving high-speed error detection and correction.
- Single-Bit Error Detection and Correction: Developing encoding and decoding algorithms capable of accurately detecting and correcting single-bit errors in transmitted or stored data.
- Real-Time Processing: Ensuring that the implemented solution can operate in real-time, making it suitable for applications with stringent latency requirements.
- Adaptability and Scalability: Creating a solution that can be easily adapted to different FPGA platforms and scaled to accommodate varying data throughput rates and error correction requirements.

By addressing these challenges, the project aims to contribute to the advancement of error detection and correction techniques, particularly in scenarios where single-bit errors pose significant risks to data integrity.

3.2 Proposed Solution

The proposed FPGA-based implementation aims to develop an efficient encoder and decoder for Enhanced Error Detection Correction (EEDC) codes, which represent an enhanced version of traditional CRC and Hamming codes. The algorithm underlying EEDC codes is designed to improve error detection and correction capabilities while minimizing redundancy, making it a promising solution for single-bit error detection and correction in data transmission systems.

3.2.1 FPGA-Based Implementation: The proposed solution will be implemented on FPGA hardware to take advantage of its flexibility, reconfigurability, and parallel processing capabilities. FPGA platforms offer the computational resources required to execute the algorithm in real time, making them well-suited for applications requiring low latency and high throughput rates.

3.2.2 Expected Benefits:

- Enhanced Error Detection and Correction: By leveraging EEDC codes, the proposed solution offers improved error detection and correction capabilities compared to traditional CRC and Hamming codes.
- Reduced Redundancy: The algorithm minimizes the number of redundancy bits required, optimizing resource utilization and reducing overhead in data transmission systems.
- Real-Time Performance: The FPGA-based implementation enables real-time processing of data, ensuring timely error detection and correction in high-speed communication channels.
- Scalability and Adaptability: The proposed solution can be easily adapted to different FPGA platforms and scaled to accommodate varying data throughput rates and error correction requirements, making it versatile for various applications. In summary, the FPGA-based implementation of EEDC encoder and decoder modules promises to provide efficient single-bit error detection and correction capabilities, enhancing the reliability of data transmission systems in diverse environments.

Chapter 4: Methodology & Implementation

4.1 Methodology

4.1.1 Algorithm Overview:

Encoder:

Codeword Generation: Each valid codeword of C bits consists of valid input data bits Di. The algorithm ensures that an invalid codeword is generated only when there are C bits that can be altered within any valid Di entity, resulting in a total of C + 1 possible codewords conforming to a valid data entity.

Redundancy Calculation: The algorithm calculates the required number of redundancy bits (r) based on the specified inequality, ensuring that the overall number of valid and invalid codes remains within predetermined limits.

Polynomial Transformation: Both the input data Di and the calculated redundancy bits r are transformed into polynomial forms with a degree of n-1. The degree of the polynomial D(x) is then multiplied by the nth value of r to finalize the structure of the EEDC codes.

Decoder:

Syndrome Calculation: Upon receiving the encoded input, the decoder calculates the syndrome, which is a crucial step in error detection. The syndrome is computed by XORing specific bits of the encoded input according to predefined rules.

Error Correction: If the calculated syndrome indicates the presence of an error, the decoder identifies the position of the error by comparing the syndrome with predefined patterns. Based on the syndrome pattern, the decoder determines which bit needs to be inverted to correct the error.

Bit Correction: Once the erroneous bit is identified, the decoder corrects it by flipping its value.
Output Generation: After error correction, the decoder generates the decoded output by extracting the corrected data bits from the encoded input.

Encoder and Decoder Implementation: The finalized EEDC codes are implemented in FPGA hardware to develop efficient encoder and decoder modules. The encoder module processes input data and adds redundancy bits to generate codewords, while the decoder module detects and corrects single-bit errors in received codewords.

4.1.2 Flow-Chart

a) Encoder

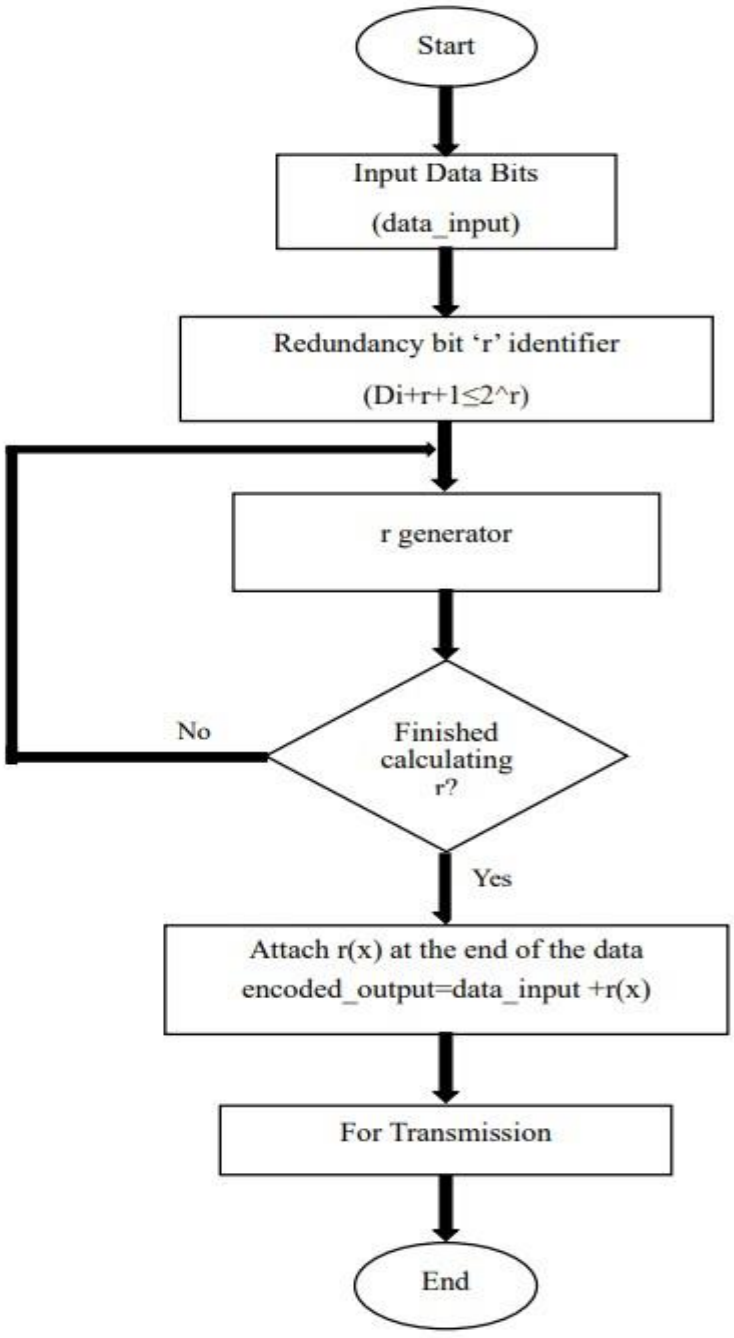


Fig 1. EEDC encoder Flow-Chart

b) Decoder

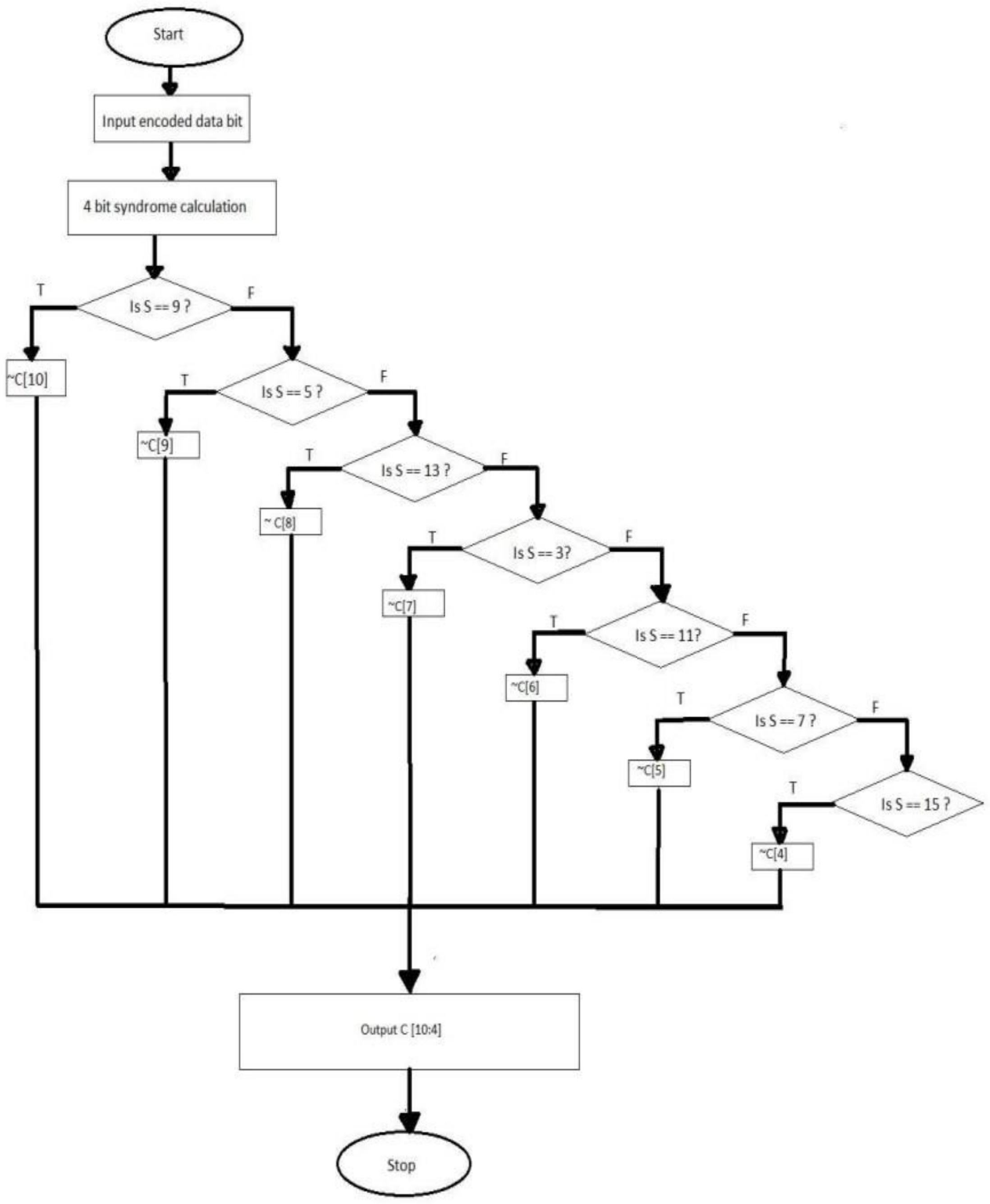


Fig 2. EEDC decoder Flow-Chart

4.2 Implementation

4.2.1 EEDC Encoder: 7 bit data encoding

a) Code:

```
module EEDC_encoder(
    input clk,
    input [6:0] data_input,
    output reg [10:0] encoded_output
);



wire r3, r2, r1, r0;
assign r3 = data_input[6] ^ data_input[4] ^ data_input[2] ^ data_input[0];
assign r2 = data_input[5] ^ data_input[4] ^ data_input[1] ^ data_input[0];
assign r1 = data_input[3] ^ data_input[2] ^ data_input[1] ^ data_input[0];
assign r0 = r3 ^ r2 ^ r1;

always @(posedge clk)
begin
    encoded_output[10]= data_input[6];
    encoded_output[9] = data_input[5];
    encoded_output[8] = data_input[4];
    encoded_output[7] = data_input[3];
    encoded_output[6] = data_input[2];
    encoded_output[5] = data_input[1];
    encoded_output[4] = data_input[0];
    encoded_output[3:0] = {r3, r2, r1, r0};
end

endmodule
```

b) Testbench:

```
module EEDC_encoder_tb;
  reg clk;
  reg [6:0] data_input;
  wire [10:0] encoded_output;

  EEDC_encoder dut (
    .clk(clk),
    .data_input(data_input),
    .encoded_output(encoded_output)
  );

  always #50 clk <= ~clk;
  initial
  begin
    clk = 0;
    data_input = 7'b0000000;
    #100 data_input = 7'b0000001;
    #100 data_input = 7'b0000010;
    #100 data_input = 7'b0000011;
    #100 data_input = 7'b0000100;
    #100 data_input = 7'b0000101;
    #100 data_input = 7'b0000110;
    #100 data_input = 7'b1001001;
    #100 data_input = 7'b1011011;
    #100 data_input = 7'b1001101;
    #500 $finish;
  end

  always @(posedge clk) begin
    $display("Time=%0t, Input=%b, Encoded Output=%b", $time, data_input,
encoded_output);
  end

endmodule
```

c) Constraints:

```
## Clock signal
set_property -dict { PACKAGE_PIN W5   IOSTANDARD LVCMOS33 } [get_ports clk]
create_clock -add -name sys_clk_pin -period 10.00 -waveform {0 5} [get_ports clk]



## Switches
set_property -dict { PACKAGE_PIN V17    IOSTANDARD LVCMOS33 } [get_ports {data_input[0]}]
set_property -dict { PACKAGE_PIN V16    IOSTANDARD LVCMOS33 } [get_ports {data_input[1]}]
set_property -dict { PACKAGE_PIN W16    IOSTANDARD LVCMOS33 } [get_ports {data_input[2]}]
set_property -dict { PACKAGE_PIN W17    IOSTANDARD LVCMOS33 } [get_ports {data_input[3]}]
set_property -dict { PACKAGE_PIN W15    IOSTANDARD LVCMOS33 } [get_ports {data_input[4]}]
set_property -dict { PACKAGE_PIN V15    IOSTANDARD LVCMOS33 } [get_ports {data_input[5]}]
set_property -dict { PACKAGE_PIN W14    IOSTANDARD LVCMOS33 } [get_ports {data_input[6]}]

## LEDs
set_property -dict { PACKAGE_PIN U16    IOSTANDARD LVCMOS33 } [get_ports {encoded_output[0]}]
set_property -dict { PACKAGE_PIN E19    IOSTANDARD LVCMOS33 } [get_ports {encoded_output[1]}]
set_property -dict { PACKAGE_PIN U19    IOSTANDARD LVCMOS33 } [get_ports {encoded_output[2]}]
set_property -dict { PACKAGE_PIN V19    IOSTANDARD LVCMOS33 } [get_ports
{encoded_output[3]}]
set_property -dict { PACKAGE_PIN W18    IOSTANDARD LVCMOS33 } [get_ports
{encoded_output[4]}]
set_property -dict { PACKAGE_PIN U15    IOSTANDARD LVCMOS33 } [get_ports
{encoded_output[5]}]
set_property -dict { PACKAGE_PIN U14    IOSTANDARD LVCMOS33 } [get_ports
{encoded_output[6]}]
set_property -dict { PACKAGE_PIN V14    IOSTANDARD LVCMOS33 } [get_ports
{encoded_output[7]}]
set_property -dict { PACKAGE_PIN V13    IOSTANDARD LVCMOS33 } [get_ports
{encoded_output[8]}]
set_property -dict { PACKAGE_PIN V3     IOSTANDARD LVCMOS33 } [get_ports
{encoded_output[9]}]
set_property -dict { PACKAGE_PIN W3     IOSTANDARD LVCMOS33 } [get_ports
{encoded_output[10]}]
```

4.2.2 EEDC Decoder

a) Code:

```
module EEDC_decode(
    input clk,
    input [10:0] encoded_input,
    output reg [6:0] decoded_output
);

reg [10:0] c2;
reg [3:0] syndrome;

always @ (posedge clk)
begin
    syndrome[3] = encoded_input[3] ^ encoded_input[4] ^ encoded_input[6] ^ encoded_input[8] ^ encoded_input[10];
    syndrome[2] = encoded_input[2] ^ encoded_input[5] ^ encoded_input[4] ^ encoded_input[9] ^ encoded_input[8];
    syndrome[1] = encoded_input[1] ^ encoded_input[4] ^ encoded_input[5] ^ encoded_input[6] ^ encoded_input[7];
    syndrome[0] = encoded_input[1] ^ encoded_input[2] ^ encoded_input[3];
c2=encoded_input;

    if (syndrome != 4'b0000)
    begin
        if (syndrome == 4'b1001)
            c2[10] = ~encoded_input[10];
        else if (syndrome == 4'b0101)
            c2[9] = ~encoded_input[9];
        else if (syndrome == 4'b1101)
            c2[8] = ~encoded_input[8];
        else if (syndrome == 4'b0011)
            c2[7] = ~encoded_input[7];
        else if (syndrome == 4'b1011)
            c2[6] = ~encoded_input[6];
        else if (syndrome == 4'b0111)
            c2[5] = ~encoded_input[5];
        else if (syndrome == 4'b1111)
            c2[4] = ~encoded_input[4];
    end
    end
    always @(posedge clk)
    begin
        decoded_output[6] = c2[10];
        decoded_output[5] = c2[9];
        decoded_output[4] = c2[8];
        decoded_output[3] = c2[7];
        decoded_output[2] = c2[6];
        decoded_output[1] = c2[5];
        decoded_output[0] = c2[4];
    end
endmodule
```

b) Testbench:

```
module EEDC_decode_tb;
 reg clk;
 reg [10:0]encoded_input;
 wire [6:0] decoded_output;

 EEDC_decode dut (
 .clk(clk),
 .encoded_input(encoded_input),
 .decoded_output(decoded_output)
 );
```

```
always #50 clk<=~clk;
initial
begin
clk=0;
encoded_input=11'b00000011111;
#100 encoded_input=11'b00000001111;
#100 encoded_input=11'b00000101001;
#100 encoded_input=11'b00011010101;
#100 encoded_input=11'b00101010101;
#100 encoded_input=11'b00000100110;
#100 encoded_input=11'b10110111111;
#100 encoded_input=11'b10100111111;
#100 $finish;
end

always @(posedge clk)begin
$display("Time=%0t, Encoded Input=%b, Decoded Output=%b", $time, encoded_input, decoded_output);
end
endmodule
```

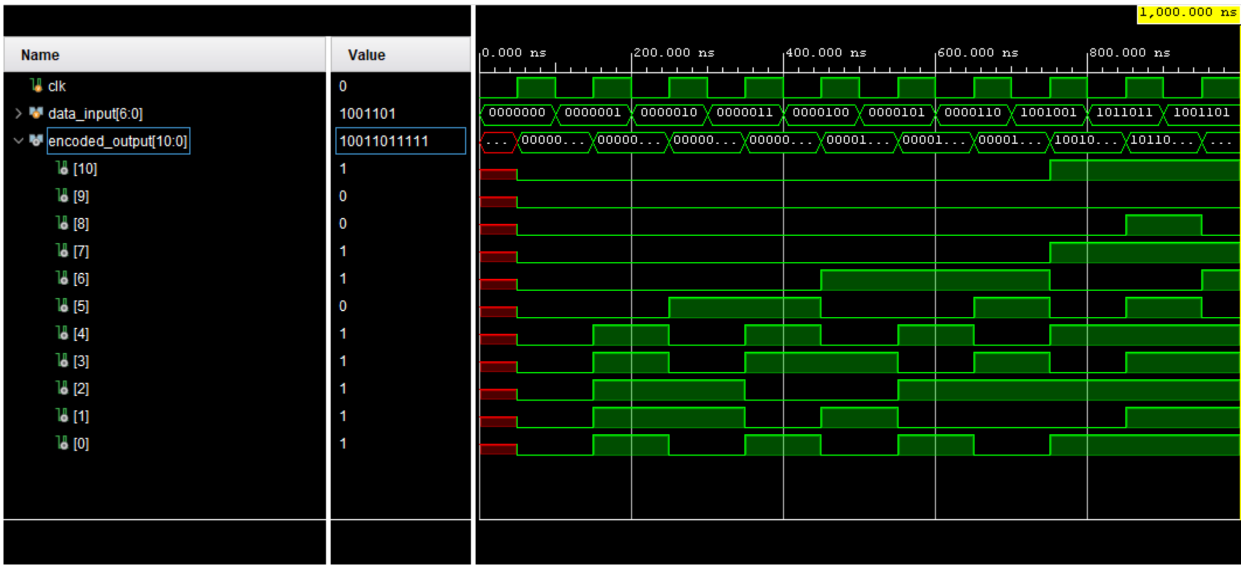
c) Constraints:

```
## Clock signal
set_property -dict { PACKAGE_PIN W5   IOSTANDARD LVCMOS33 } [get_ports clk]
create_clock -add -name sys_clk_pin -period 10.00 -waveform {0 5} [get_ports clk]

## Switches
set_property -dict { PACKAGE_PIN V17   IOSTANDARD LVCMOS33 } [get_ports {encoded_input[0]}]
set_property -dict { PACKAGE_PIN V16   IOSTANDARD LVCMOS33 } [get_ports {encoded_input[1]}]
set_property -dict { PACKAGE_PIN W16   IOSTANDARD LVCMOS33 } [get_ports
{encoded_input[2]}]

set_property -dict { PACKAGE_PIN W17   IOSTANDARD LVCMOS33 } [get_ports
{encoded_input[3]}]

set_property -dict { PACKAGE_PIN W15   IOSTANDARD LVCMOS33 } [get_ports
{encoded_input[4]}]

set_property -dict { PACKAGE_PIN V15   IOSTANDARD LVCMOS33 } [get_ports
{encoded_input[5]}]

set_property -dict { PACKAGE_PIN W14   IOSTANDARD LVCMOS33 } [get_ports
{encoded_input[6]}]

set_property -dict { PACKAGE_PIN W13   IOSTANDARD LVCMOS33 } [get_ports
{encoded_input[7]}]

set_property -dict { PACKAGE_PIN V2    IOSTANDARD LVCMOS33 } [get_ports
{encoded_input[8]}]

set_property -dict { PACKAGE_PIN T3    IOSTANDARD LVCMOS33 } [get_ports
{encoded_input[9]}]

set_property -dict { PACKAGE_PIN T2    IOSTANDARD LVCMOS33 } [get_ports
{encoded_input[10]}]


## LEDs

set_property -dict { PACKAGE_PIN U16   IOSTANDARD LVCMOS33 } [get_ports
{decoded_output[0]}]

set_property -dict { PACKAGE_PIN E19   IOSTANDARD LVCMOS33 } [get_ports
{decoded_output[1]}]

set_property -dict { PACKAGE_PIN U19   IOSTANDARD LVCMOS33 } [get_ports
{decoded_output[2]}]

set_property -dict { PACKAGE_PIN V19   IOSTANDARD LVCMOS33 } [get_ports
{decoded_output[3]}]

set_property -dict { PACKAGE_PIN W18   IOSTANDARD LVCMOS33 } [get_ports
{decoded_output[4]}]

set_property -dict { PACKAGE_PIN U15   IOSTANDARD LVCMOS33 } [get_ports
{decoded_output[5]}]

set_property -dict { PACKAGE_PIN U14   IOSTANDARD LVCMOS33 } [get_ports
{decoded_output[6]}]
```

Chapter 5: Results & Discussion

5.1 Output Waveforms

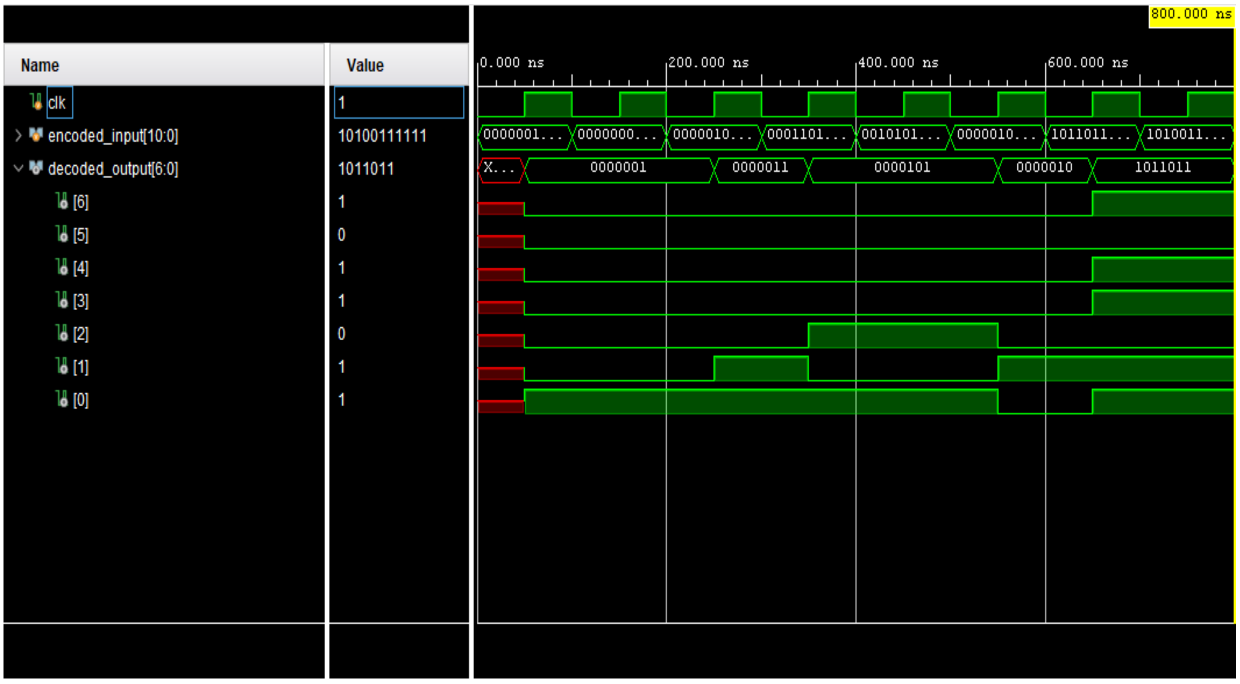
a) EEDC Encoder



```
Name                  Value
clk                   0
data_input[6:0]       1001101
encoded_output[10:0]  10011011111
  [10]                1
  [9]                 0
  [8]                 0
  [7]                 1
  [6]                 1
  [5]                 0
  [4]                 1
  [3]                 1
  [2]                 1
  [1]                 1
  [0]                 1
```

Fig 3. EEDC encoder simulation

b) EEDC Decoder



```
Name                  Value
clk                   1
encoded_input[10:0]   10100111111
decoded_output[6:0]   1011011
  [6]                 1
  [5]                 0
  [4]                 1
  [3]                 1
  [2]                 0
  [1]                 1
  [0]                 1
```

Fig 4. EEDC decoder simulation

RTL Design 1:

a) Encoder

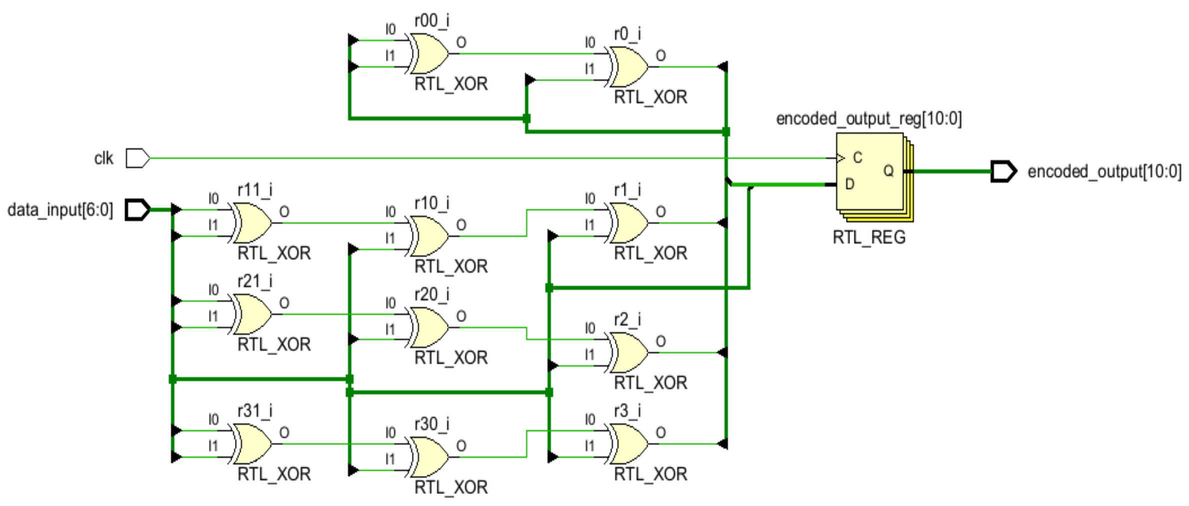


Fig 5. EEDC encoder RTL Design 1

b) Decoder

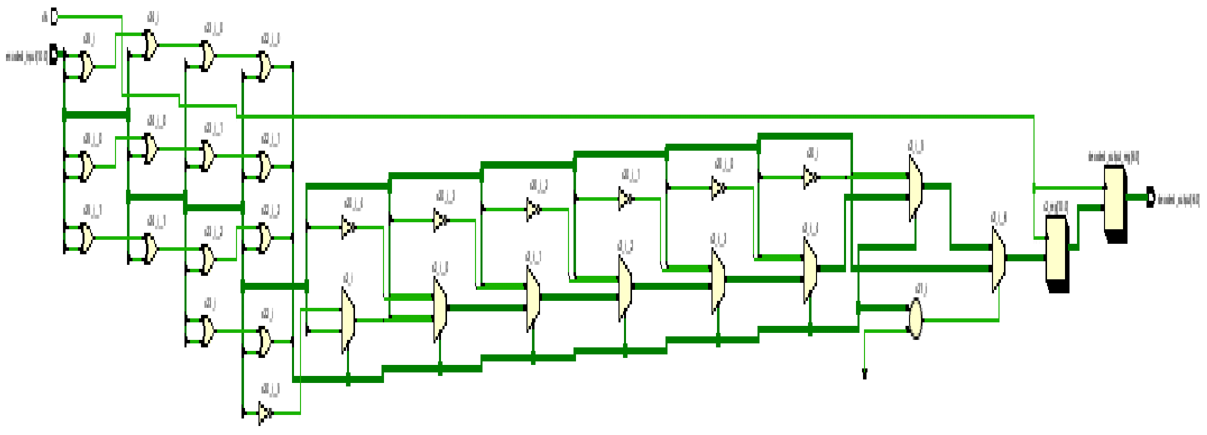


Fig 6. EEDC decoder RTL Design 1

RTL Design 2:

a) Encoder

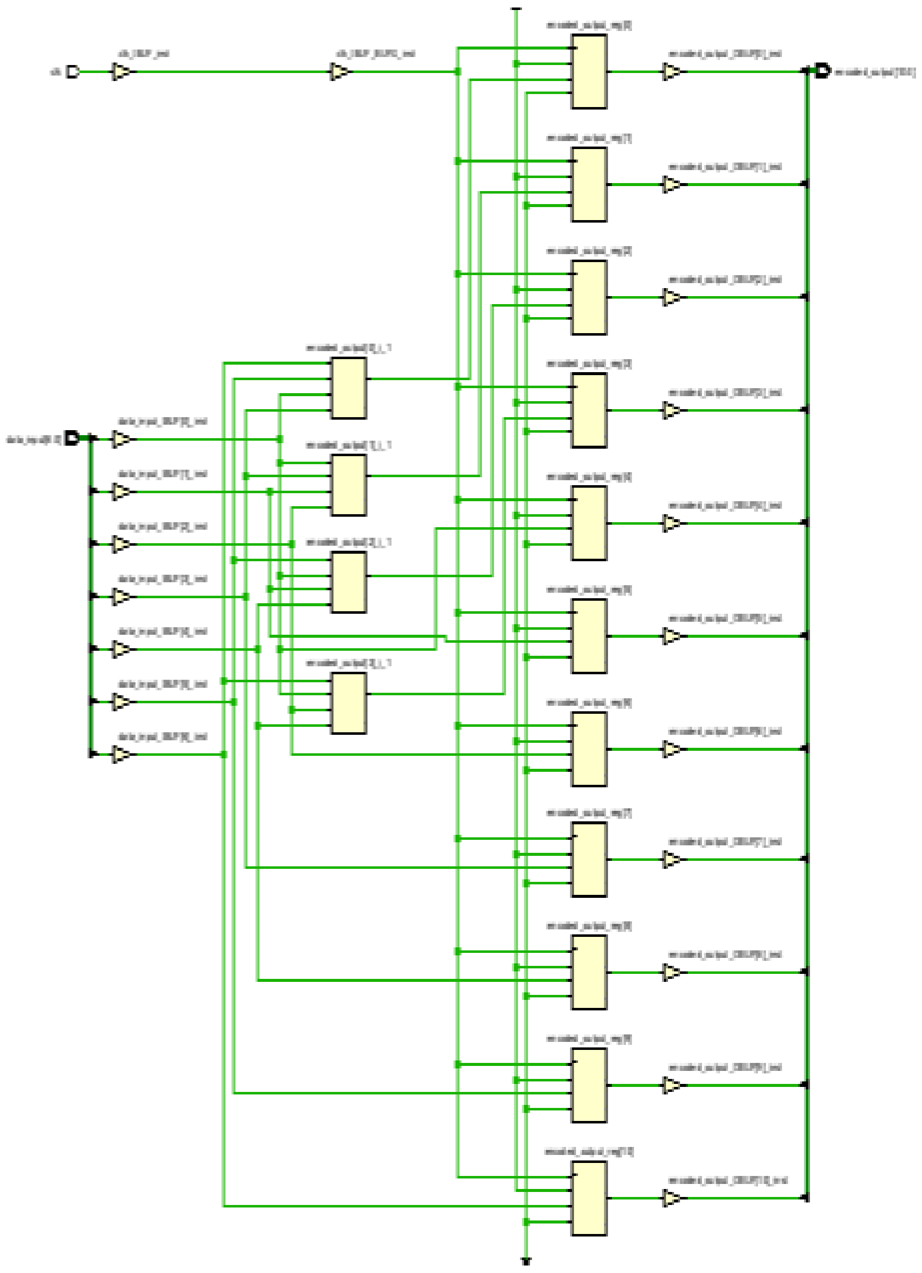


Fig 7. EEDC encoder RTL Design 2

b) Decoder

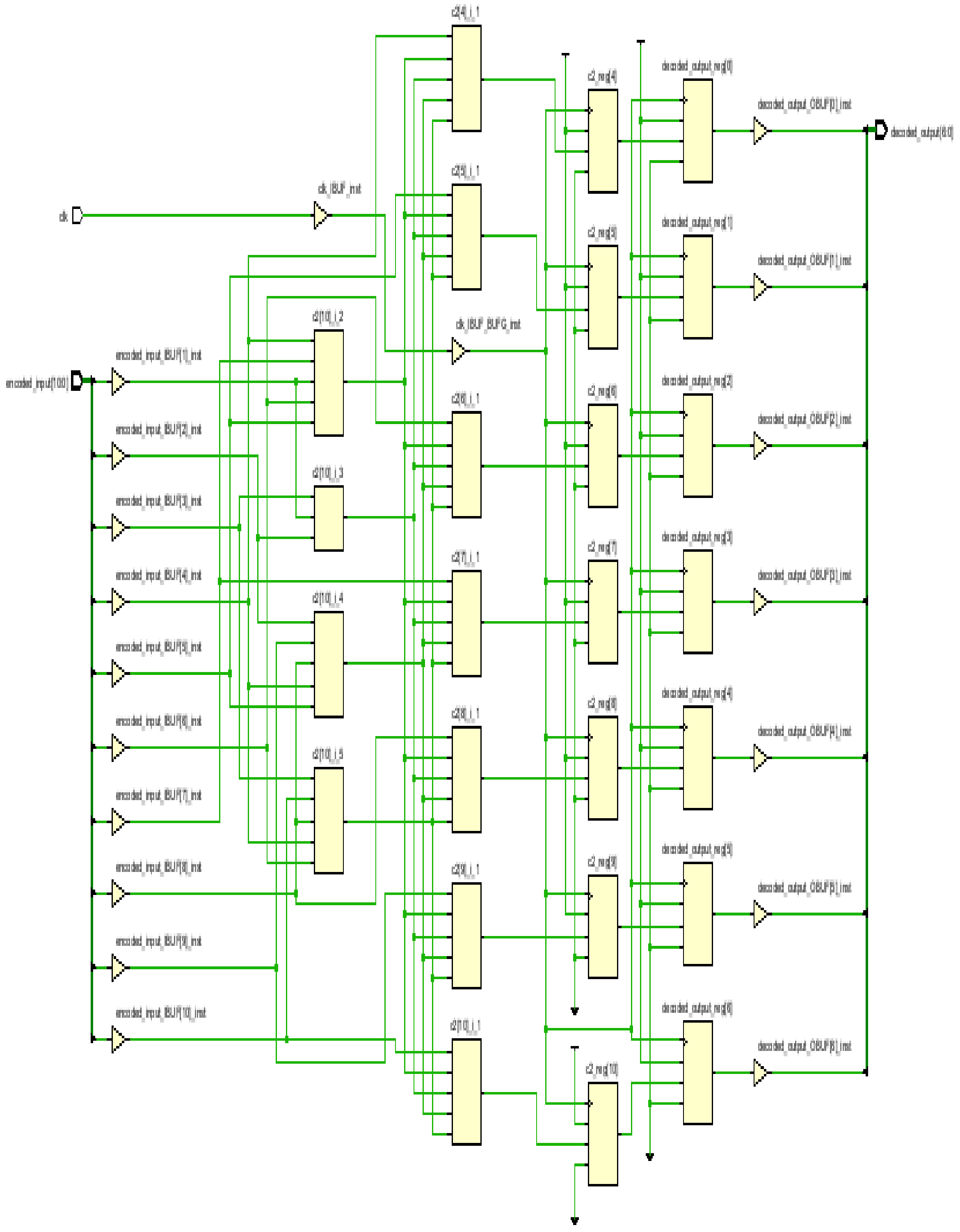

c2[4]_i_1
c2[5]_i_1
clk_IBUF_inst
clk
c2[10]_i_2
clk_IBUF_BUFG_inst
encoded_input_IBUF[1]_inst
encoded_input[10:0]
encoded_input_IBUF[2]_inst
c2[10]_i_3
encoded_input_IBUF[3]_inst
encoded_input_IBUF[4]_inst
c2[10]_i_4
encoded_input_IBUF[5]_inst
encoded_input_IBUF[6]_inst
c2[10]_i_5
encoded_input_IBUF[7]_inst
encoded_input_IBUF[8]_inst
encoded_input_IBUF[9]_inst
encoded_input_IBUF[10]_inst
c2[6]_i_1
c2[7]_i_1
c2[8]_i_1
c2[9]_i_1
c2[10]_i_1
c2_reg[4]
c2_reg[5]
c2_reg[6]
c2_reg[7]
c2_reg[8]
c2_reg[9]
c2_reg[10]
decoded_output_reg[0]
decoded_output_reg[1]
decoded_output_reg[2]
decoded_output_reg[3]
decoded_output_reg[4]
decoded_output_reg[5]
decoded_output_reg[6]
decoded_output_OBUF[0]_inst
decoded_output_OBUF[1]_inst
decoded_output_OBUF[2]_inst
decoded_output_OBUF[3]_inst
decoded_output_OBUF[4]_inst
decoded_output_OBUF[5]_inst
decoded_output_OBUF[6]_inst
decoded_output[6:0]

Fig 8. EEDC decoder RTL Design 2

5.2 FPGA Output

The identical output is validated on an FPGA board when subjected to the same inputs.

a) EEDC Encoder

data_input = 1011011

encoded_output = 10110111111

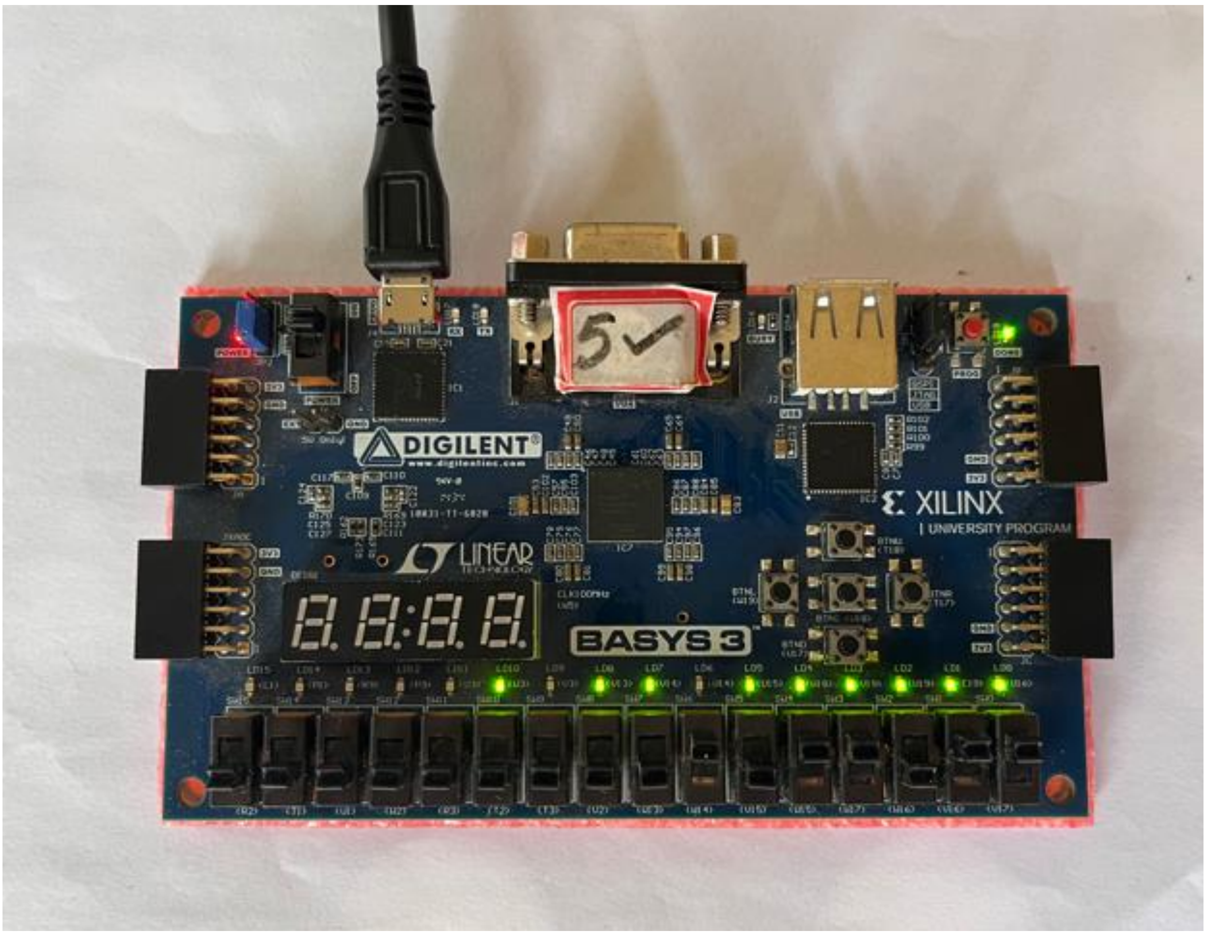


Fig 9. EEDC encoder FPGA execution

b) EEDC Decoder

i) Wrong input correction

encoded_input = 10100111111

decoded_output= 1011011

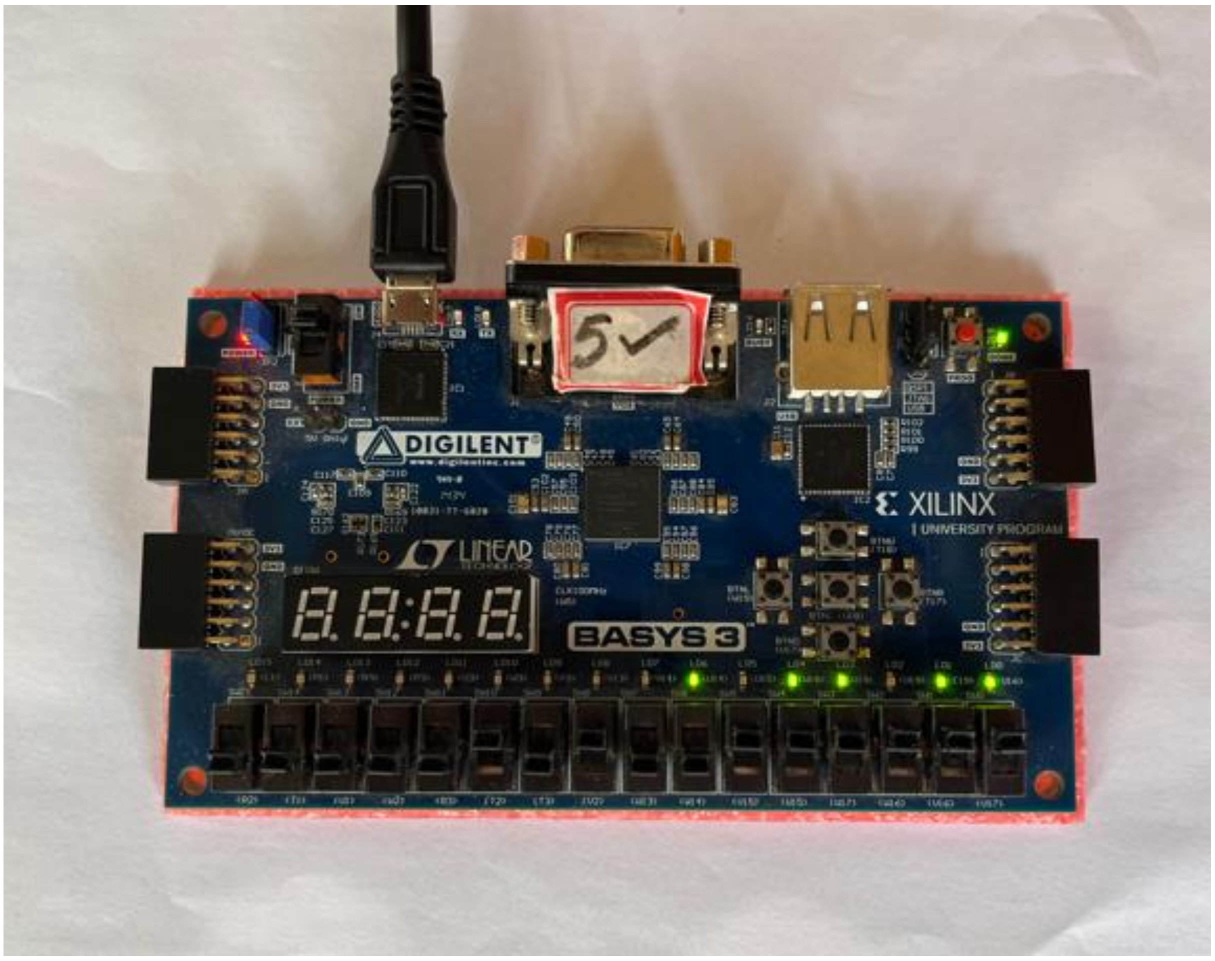


Fig 10. EEDC decoder FPGA execution (i)

ii) Correct Input

encoded_input = 10110111111

decoded_output= 1011011

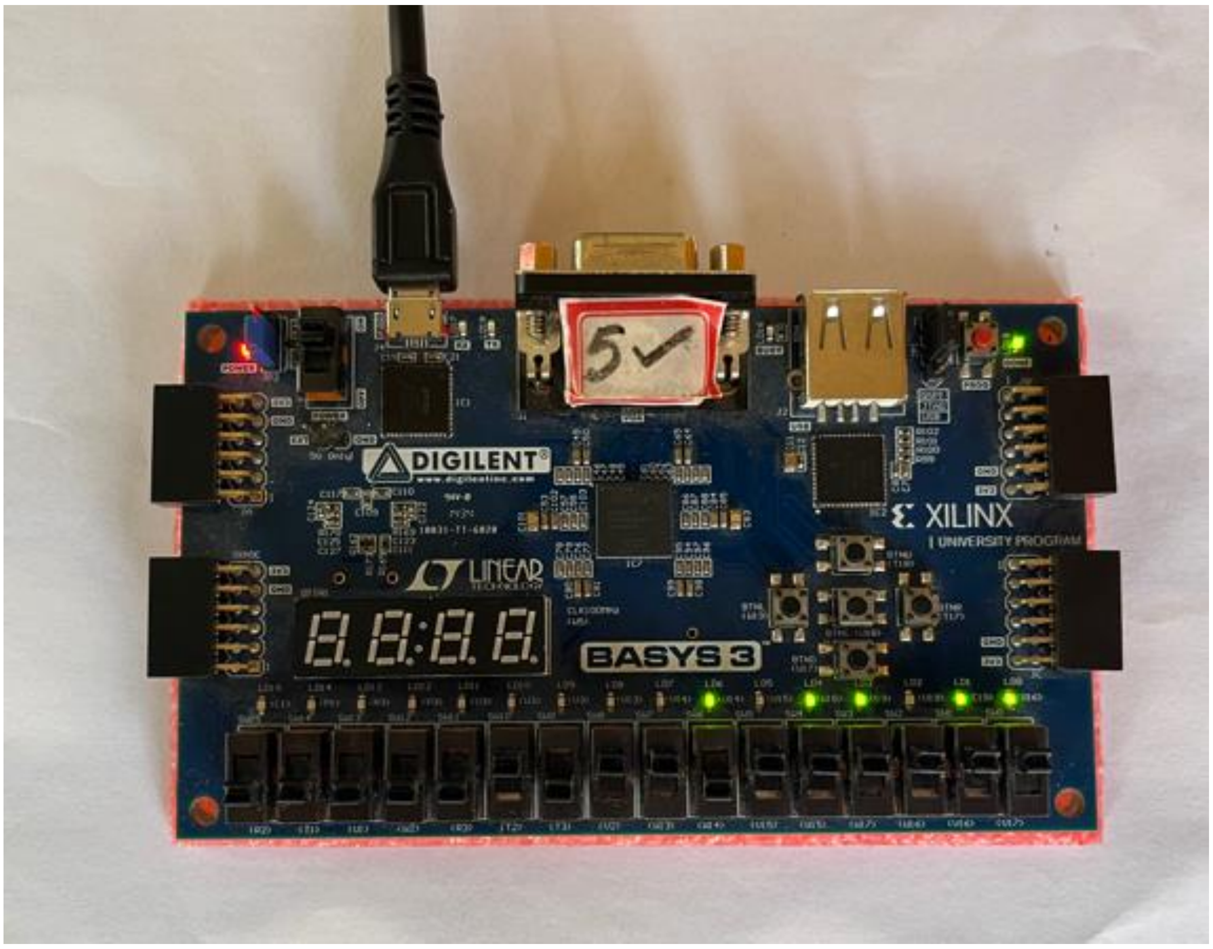


Fig 11. EEDC decoder FPGA execution (ii)

Chapter 6: Conclusion and Future Trends

6.1 Conclusion

The project addresses the critical need for robust error detection and correction mechanisms capable of efficiently identifying and rectifying single-bit errors in data transmissions or storage systems. Traditional error detection and correction techniques may not always suffice, particularly in environments where single-bit errors pose significant risks to data integrity. Therefore, the project proposes an FPGA-based solution utilizing Enhanced Error Detection and Correction (EEDC) encoding and decoding techniques, which represent an enhanced version of traditional CRC and Hamming codes.

The proposed solution aims to overcome several challenges, including efficient hardware implementation, single-bit error detection and correction, real-time processing, and adaptability and scalability. By addressing these challenges, the project contributes to the advancement of error detection and correction techniques, particularly in scenarios where single-bit errors are prevalent. The proposed algorithm leverages the concept of CRC and Hamming codes while introducing enhancements to address their limitations. EEDC codes are derived from modified and improved Hamming codes, offering superior error detection and correction performance. The encoder module generates codewords by ensuring the validity of input data bits and calculating the required redundancy bits based on predetermined limits. The decoder module calculates the syndrome upon receiving the encoded input and corrects errors by identifying and flipping erroneous bits.

The FPGA-based implementation of EEDC encoder and decoder modules promises several benefits, including enhanced error detection and correction, reduced redundancy, real-time performance, scalability, and adaptability. By leveraging FPGA hardware, the solution ensures efficient processing of data, making it suitable for applications with stringent latency requirements. In summary, the proposed FPGA-based solution offers efficient single-bit error detection and correction capabilities, enhancing the reliability of data transmission systems in diverse environments.

6.2 Future Trends

As the Internet of Things continues to grow, there will be a need for EEDC techniques tailored for low-power, resource-constrained devices at the edge of networks. Future trends might involve developing lightweight error detection and correction algorithms optimized for devices, ensuring reliable communication over wireless and unreliable networks. Future EEDC trends could involve integrating advanced coding schemes with new technologies to ensure robust communication in high-speed, low-latency environments. Error detection and correction will be critical in the communication systems of autonomous vehicles to ensure the reliability of sensor data and communication between vehicles and infrastructure.

Future trends might include the development of fault-tolerant communication protocols and EEDC techniques specifically designed for the automotive industry. In financial services and banking applications, secure and error-free transmission of transaction data is crucial to prevent fraud and ensure data integrity. Future EEDC trends could involve the implementation of advanced error detection and correction mechanisms in financial communication protocols, enhancing the security and reliability of electronic transactions. With the increasing frequency and sophistication of attacks, there will be a growing emphasis on security and data privacy in digital communication systems. Future trends in EEDC could involve the development of error detection and correction techniques that not only ensure data integrity but also mitigate security threats such as data breaches, and unauthorized access.Traditional communication protocols often operate in silos, with separate layers for error detection/correction, routing, and transport.

Future trends could involve cross-layer optimization approaches that integrate error detection and correction mechanisms with other protocol layers to achieve better overall system performance, throughput, and reliability. This may involve the utilization of specialized hardware accelerators such as Field-Programmable Gate Arrays (FPGAs) or Graphics Processing Units (GPUs) to parallelize and accelerate the encoding and decoding processes of Hamming codes. Research efforts could focus on developing efficient hardware architectures that exploit parallelism to significantly reduce latency without compromising error correction performance. Ongoing research may lead to the development of novel encoding and decoding algorithms optimized for low-latency operation in Hamming codes.

Future trends could involve the exploration of algorithmic optimizations, such as streamlined computations and reduced complexity, to minimize encoding and decoding latency while ensuring robust error detection and correction. Future EDC schemes may incorporate adaptive error correction strategies that dynamically adjust the error correction mechanism based on the characteristics of the communication channel and the observed error patterns. Trends in this direction could involve the integration of machine learning algorithms to analyze real-time feedback and optimize error correction parameters for latency-sensitive applications. Hybrid error correction techniques that combine the strengths of Hamming codes with other coding schemes, such as Reed-Solomon codes or Turbo codes, could emerge as future trends. Research efforts may focus on developing hybrid schemes that effectively balance latency, error correction capability, and computational complexity to meet the requirements of diverse communication scenarios.

Future trends may involve tighter integration of EDC schemes with network protocols to optimize latency performance. Research directions could explore ways to leverage the characteristics of specific network protocols (e.g., packet-based or streaming protocols) to design EDC schemes that minimize latency and overhead while maximizing error correction efficiency. Future EDC schemes may incorporate mechanisms for real-time feedback and adaptation, allowing the system to dynamically adjust encoding and decoding parameters based on changing network conditions and performance requirements. Trends in this direction could involve the development of adaptive algorithms that continuously monitor latency and error rates, triggering adjustments in the error correction strategy as needed to maintain optimal performance. As latency becomes an increasingly critical factor in various communication applications, future trends may involve standardization efforts to define low-latency EDC schemes and protocols.

Collaboration between industry stakeholders, standardization bodies, and research communities could drive the adoption of standardized low-latency EDC solutions in diverse domains, accelerating the realization of future trends in this direction.

REFERENCES

[1] Lean Karlo S. Tolentino, Maria Victoria C. Padilla, Ronnie O. Serfa Juan, *"FPGA-based redundancy bits reduction algorithm using the enhanced error detection correction code",* International Journal of Engineering & Technology, 7 (3) (2018) 1008-1013, 2018.

[2] Subhasri.G, Radha.N, "*VLSI design of Parity check Code with Hamming Code for Error Detection and Correction*", Proceedings of the International Conference on Intelligent Computing and Control Systems (ICICCS 2019) IEEE Xplore Part Number: CFP19K34-ART; ISBN: 978-1-5386-8113-8, 2019.

[3] Nandivada Sridevi, K. Jamal, Kiran Mannem, *"Implementation of Error Correction Techniques in Memory Applications",* Proceedings of the Fifth International Conference on Computing Methodologies and Communication (ICCMC 2021), 2021.

[4] Shubham C. Anjankar, Dr. Mahesh T. Kolte, Ajinkya Pund, Pratiksha Kolte, Ankita Kumar ,Pranav Mankar, Kunal Ambhore*, "FPGA Based Multiple Fault Tolerant and Recoverable Technique Using Triple Modular Redundancy (FRTMR)",* 7th International Conference on Communication, Computing and Virtualization, 2016.

[5] Guy Castagnoli, Stefan Brauer, Martin Herrmann*, "Optimization of cyclic redundancy-check codes with 24 and 32 parity bits"*, IEEE Transactions on Communications, Vol 41, No 6, June 1993.

[6] Hugo Marcelo Torres Salamea, Dalton Demetrio Toledo Torres, Pablo Dario Urgiles Cardenas, Claudia Urrea Onata, *"Implementation of the Hamming code for the detection and correction of errors in a telerobotic system using an industrial communication protocol"*, 978-1-7281-9365-6/20, 2020.

[7] H Praveena, K.Kalyani, *"FPGA implementation of parity check matrix based low-density parity check decoder"*, Proceedings of the second international conference on inventive systems and control (ICISC 2018),2018.

[8] Marwin W Williard*,"Introduction to redundancy coding",* IEEE transaction on vehicular technology, vol. vt-27,no.3, August, 1978.

[9] Sani, Usman & Shanono, Ibrahim, "*Design of (7, 4) Hamming Encoder and Decoder Using VHDL*",2015.

[10] Liva, Gianluigi & Ryan, William, "*Short Low-Error-Floor Tanner Codes with Hamming Nodes*". 208 - 213 Vol. 1. 10.1109/MILCOM.2005.1605687, 2005.

[11] Rahaman, Muhammad, "*Enhanced Hamming Codes: Reducing Redundant Bit for Efficient Error Detection and Correction*", 2023.

[12] B, Sravya & Sucharitha, D. & Raj, N Prudhvi & Sravya, Reddy & Venishetty, Sudheer. (2019). GDI Logic Based Design of Hamming-Code Encoder and Decoder for Error Free Data Communication. 1-5. 10.1109/ICCMC.2019.8819665.

[13] Shim, B. and Shanbhag, N. *"Reduced precision redundancy for low-power digital filtering"*, Conference Record of the Thirty-Fifth Asilomar Conference on Signals, Systems, and Computers, vol. 1, Pacific Grove.

[14] Sullivan, M. A. *"Reduced precision redundancy applied to arithmetic operations in field programmable gate arrays for satellite control and sensor systems."* Master's thesis, Dept. of Mechanical and Astronautical Engineering and Dept. of Electrical and Computer Engineering, Naval Postgraduate School, Monterey, CA, Dec. 2008.

[15] Anil Kumar Singh, "*Error detection and correction by hamming code*", 2016 International Conference on Global Trends in Signal Processing, Information Computing and Communication (ICGTSPICC), 22-24 December 2016.

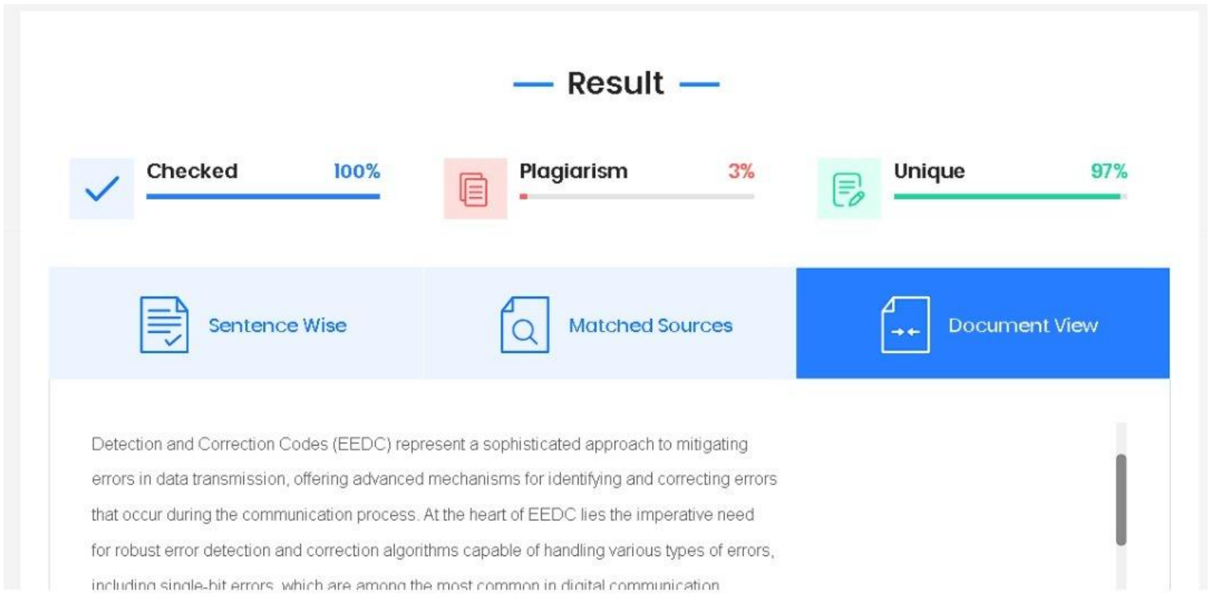
[16] U. K. Kumar; B. S. Umashankar, "*Improved Hamming Code for Error Detection and Correction*", 2007 2nd International Symposium on Wireless Pervasive Computing, 05-07 February 2007.

[17] Sang Sheng-Ju, “*Implementation of Cyclic Redundancy Check in Data Communication*”, 2015 International Conference on Computational Intelligence and Communication Networks (CICN), 12-14 December 2015.

[18] Mohamed S. Abdulnabi, Hisham Ahmed, “*Design of Efficient Cyclic Redundancy Check-32 using FPGA*”, 2018 International Conference on Computer, Control, Electrical, and Electronics Engineering (ICCCEEE), 12-14 August 2018.

APPENDIX A:

Plagiarism Report



Detection and Correction Codes (EEDC) represent a sophisticated approach to mitigating errors in data transmission, offering advanced mechanisms for identifying and correcting errors that occur during the communication process. At the heart of EEDC lies the imperative need for robust error detection and correction algorithms capable of handling various types of errors, including single-bit errors, which are among the most common in digital communication

https://searchenginereports.net/plagiarism-checker

APPENDIX C:

Research Publications