



Karunya INSTITUTE OF TECHNOLOGY AND SCIENCES

(Declared as Deemed to be University under Sec.3 of the UGC Act, 1956)

MoE, UGC & AICTE Approved

NAAC A++ Accredited

**Division of Electronics and Communication Engineering
2023-2024 (EVEN SEM)**

III IA EVALUATION REPORT

for

DIGITAL SIGNAL PROCESSING-PROJECT BASED COURSE

*Title of the project: Error Detection and Correction in Digital Signal Processing using
Cyclic Redundancy Check (CRC) Codes in PYTHON*

A report submitted by

<i>Name of the Student</i>	<i>RACHEL M D</i>
<i>Register Number</i>	<i>URK22EC1044</i>
<i>Subject Name</i>	<i>DIGITAL SIGNAL PROCESSING</i>
<i>Subject Code</i>	<i>18EC2015</i>
<i>Date of Report submission</i>	<i>10/04/2024</i>

Project Rubrics for Evaluation

First Review: Project title selection - PPT should have four slides (Title page, Introduction, Circuit/Block Diagram, and Description of Project).

Second Review: PPT should have three slides (Description of Concept, implementation, outputs, results and discussion)

Rubrics for project (III IA - 40 Marks):

Content - 4 marks (based on Project)

Clarity - 3 marks (based on viva during presentation)

Feasibility - 3 marks (based on project)

Presentation - 10 marks

Project Report - 10 marks

On-time submission - 5 marks (before the due date)

Online submission-GCR - 5 marks

Total marks: _____ / 40 Marks

Signature of Faculty with date:



Karunya INSTITUTE OF TECHNOLOGY AND SCIENCES

(Declared as Deemed to be University under Sec.3 of the UGC Act, 1956)

MoE, UGC & AICTE Approved

NAAC A++ Accredited

TABLE OF CONTENTS

CHAPTER	TITLE	PAGE NO.
1.	INTRODUCTION	3
2.	DESCRIPTION OF THE PROJECT	4
3.	CONCEPT INVOLVED	5
4.	TOOLS	7
5.	IMPLEMENTATION	8
6.	RESULTS WITH GRAPH/SIMULATION	10
7.	INFERENCES	11
8.	CONCLUSION	12

CHAPTER 1

INTRODUCTION

In digital signal processing (DSP), the integrity and reliability of transmitted data are of utmost importance for successful communication. Errors that occur during transmission can result in significant data distortion or even complete loss, which can compromise the effectiveness of communication systems. To address this issue, error detection and correction techniques are essential. A widely used error detection method is the Cyclic Redundancy Check (CRC) code. CRC codes are known for their robustness, efficiency, and extensive use in various communication protocols, such as Ethernet, USB, and wireless networks. These codes work by appending a predetermined number of check bits to the original data, enabling the receiver to detect and, in some cases, correct errors that may have occurred during transmission. This report delves into the theoretical foundations of CRC codes, their application in error detection and correction, and the methodology employed in implementing CRC-based error detection and correction algorithms in Python. By conducting practical experiments and analysis, we aim to assess the effectiveness and performance of CRC-based error detection and correction in DSP applications. Furthermore, we explore potential enhancements and optimizations to improve the robustness and efficiency of the CRC-based system.

CHAPTER 2

DESCRIPTION OF THE PROJECT

This project focuses on the implementation of error detection and correction in digital signal processing using Cyclic Redundancy Check (CRC) codes in Python. The project aims to ensure the accuracy and reliability of transmitted data in digital communication systems by employing CRC codes, which are widely used due to their robustness and efficiency. CRC codes work by appending a predetermined number of check bits to the original data, allowing the receiver to detect and, in some cases, correct errors that may have occurred during transmission.

The CRC class encapsulates the CRC encoding and decoding logic, facilitating the encoding of input binary data and polynomial. Additionally, it enables the detection of errors in the encoded data by performing polynomial division. The application utilizes Tkinter, a Python library, to create a user-friendly interface comprising input fields for data and polynomial entry, buttons for encoding and error simulation, and output fields for displaying the encoded data and error detection results.

CHAPTER 3

CONCEPT INVOLVED

The project revolves around the concept of enhancing the reliability of digital signal transmission through the implementation of error detection and correction mechanisms using Cyclic Redundancy Check (CRC) codes. In digital signal processing (DSP), ensuring the integrity of transmitted data is vital, as errors can lead to distortions or loss of information. CRC codes offer an efficient solution by appending check bits to data, enabling receivers to detect and potentially correct errors.

CYCLIC REDUNDANCY CHECK (CRC) CODE

Cyclic redundancy check (CRC) coding is an error-control coding technique for detecting errors that occur when transmitting a data frame. Unlike block or convolutional codes, CRC codes do not have a built-in error-correction capability. Instead, when a communications system detects a CRC coding error in a received codeword, the receiver requests the sender to retransmit the codeword. CRC is widely used in digital signal processing for error detection in data transmission. It is particularly useful in applications where a high level of reliability is required, such as in telecommunications, networking, and data storage.

It works by performing a polynomial division of the data to be transmitted by a divisor polynomial, and the remainder of this division is the CRC code. This CRC code is then transmitted along with the data. At the receiving end, the same polynomial division is performed again, and if the new remainder is zero, the data is assumed to be error-free.

Encoding Data with CRC:

Polynomial Selection: A CRC polynomial is chosen based on factors such as desired error detection capability and the characteristics of the communication channel.

Data Padding: The input data is padded with zeros to match the length of the CRC polynomial.

Polynomial Division: The padded data is divided by the CRC polynomial using binary polynomial division.

CRC Checksum: The remainder obtained after polynomial division, known as the CRC checksum, is appended to the original data to form the encoded data.

Transmitting and Receiving Data:

The encoded data, including the CRC checksum, is transmitted over the communication channel.

Upon receiving the data, the receiver performs the same polynomial division operation using the received data and the CRC polynomial.

Error Detection:

If the received data is error-free, the remainder obtained after polynomial division should be zero.

Any nonzero remainder indicates that errors have occurred during transmission.

The receiver detects errors by checking the remainder obtained after polynomial division. If the remainder is nonzero, errors are detected, and appropriate error handling mechanisms can be employed.

Checksum Verification:

To verify data integrity, the receiver recalculates the CRC checksum using the received data.

If the recalculated checksum matches the received CRC checksum, the data is considered error-free. Otherwise, errors are detected.

Optimization and Performance:

CRC algorithms are designed to have certain desirable properties, such as high error detection capability and efficient implementation.

Performance considerations, such as the choice of CRC polynomial and implementation optimizations, can influence the effectiveness and efficiency of CRC error detection.

CHAPTER 4

TOOLS

Python Programming Language:

Python was used as the primary programming language for implementing the CRC algorithm, GUI, and overall project structure. Python's simplicity, readability, and extensive standard library make it suitable for rapid development.

Tkinter Library:

Tkinter is Python's built-in GUI toolkit used for creating graphical user interfaces. It provides a simple and intuitive way to create windows, widgets, and handle user interactions.

Random Module:

The random module in Python was utilized to simulate errors in the encoded data. It provides functions for generating random numbers, essential for error simulation in testing error detection capabilities.

CRC Algorithm:

The CRC algorithm itself is a mathematical technique used for error detection in digital data transmission. While not a separate tool, it's a key component of the project and was implemented within the Python code.

CHAPTER 5

IMPLEMENTATION

1. Implementing the CRC Algorithm:

- Define a class named `CRC` that will contain methods for encoding and decoding data using CRC.
- Implement the encoding function (`encode`) that takes input data and CRC polynomial as parameters and returns the encoded data.
- Implement the decoding function (`decode`) that takes received data as input and checks for errors using CRC.
- Within the `CRC` class, implement a method (`polynomial_division`) to perform polynomial division, a key operation in CRC calculation.

2. Creating a Graphical User Interface (GUI):

- Use the Tkinter library to create a GUI window.
- Design the layout of the GUI with input fields for binary data and CRC polynomial, buttons for encoding data and simulating errors, and output fields for displaying encoded data and error detection results.
- Use Tkinter widgets such as `Label`, `Entry`, and `Button` to create these elements and organize them within the window.

3. Handling User Inputs and Interactions:

- Define functions to handle user inputs and interactions with the GUI elements.
- Implement functions to retrieve input data and CRC polynomial entered by the user.
- Write functions to encode the input data with CRC and display the encoded data.
- Implement a function to simulate errors in the encoded data and detect errors using CRC.

- Update the GUI elements to display the results of encoding and error detection.

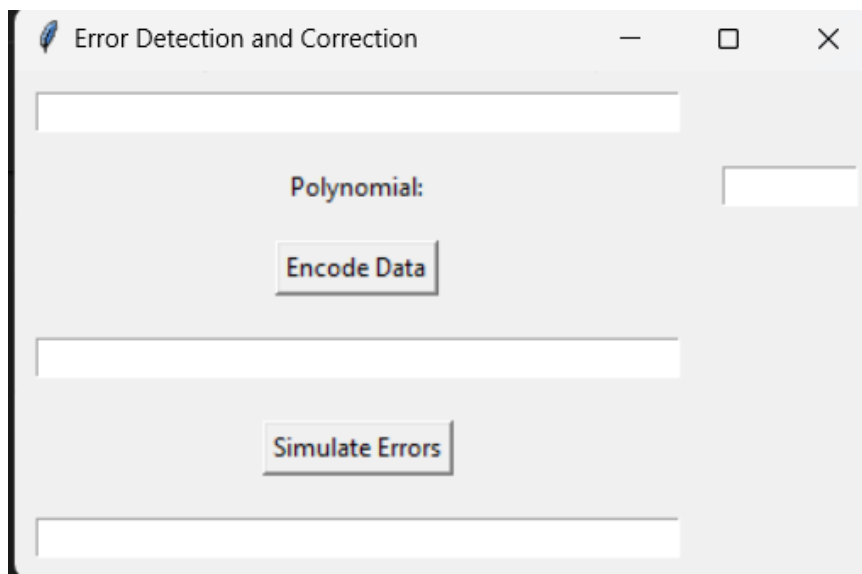
4. Thorough Testing:

- Test the implementation thoroughly to ensure it functions correctly under various scenarios.
- Test different input data and CRC polynomials to verify that encoding and decoding work as expected.
- Simulate errors in the encoded data and verify that the error detection mechanism works correctly.
- Test the GUI interface to ensure it responds appropriately to user interactions and displays the expected results.

CHAPTER 6

RESULTS

```
• [steevepp@fedora RACHEL]$ python3 test.py 1001
*****
* Original Data: [1, 0, 0, 1] *
*****
*****
* Encoded Data with CRC: [1, 0, 0, 1, 1, 1, 0] *
*****
*****
* No errors detected. *
*****
```



The screenshot shows a web application window titled "Error Detection and Correction". It features a light gray background and standard window controls (minimize, maximize, close) in the top right corner. The interface includes several input fields and two buttons:

- A large empty text input field at the top.
- A label "Polynomial:" followed by a smaller empty text input field.
- A button labeled "Encode Data" below the polynomial input.
- Another large empty text input field below the encode button.
- A button labeled "Simulate Errors" below the second input field.
- A final large empty text input field at the bottom of the form.

CHAPTER 7

REFERENCE

- <https://github.com/1044rachel/Error-Detection-and-Correction-using-Cyclic-Redundancy-Check-CRC-Codes-in-PYTHON.git>
- Sheong, S. (2021, September 1). *Explaining error detection and correction codes with Python*. Medium. <https://levelup.gitconnected.com/explaining-error-detection-and-correction-codes-with-python-be517596d42f>
- Yangyang, G. (2021, July 1). *What Is Cyclic Redundancy Check? How to Fix CRC Errors?* Huawei. <https://info.support.huawei.com/info-finder/encyclopedia/en/CRC.html>
- *What is Cyclic Redundancy Check (CRC) and How Does it Work?* | Lenovo US. (n.d.). Retrieved April 8, 2024, from <https://www.lenovo.com/us/en/glossary/cyclic-redundancy-check/>
- *What is CRC-4 (Cyclic Redundancy Check 4)?* (n.d.). Networking. Retrieved April 8, 2024, from <https://www.techtarget.com/searchnetworking/definition/CRC-4>
- *Python Program to Cyclic Redundancy Check*. (n.d.). Retrieved April 8, 2024, from <https://www.tutorialspoint.com/python-program-to-cyclic-redundancy-check>
- *Error Detection and Correction—MATLAB & Simulink*. (n.d.). Retrieved April 8, 2024, from <https://www.mathworks.com/help/comm/error-detection-and-correction.html>
- *Python Program to Cyclic Redundancy Check*. (n.d.). Retrieved April 8, 2024, from <https://www.tutorialspoint.com/python-program-to-cyclic-redundancy-check>

CHAPTER 8

CONCLUSION

In this project, we investigated the use of Cyclic Redundancy Check (CRC) codes for detecting and correcting errors in digital signal processing. Our implementation in Python included a graphical user interface (GUI) built using the Tkinter library, which allowed users to interactively demonstrate the functionality of CRC codes.

The GUI enabled users to input binary data, specify a CRC polynomial, encode the data with CRC, simulate errors, and detect errors using CRC. By employing CRC codes, we can detect errors in transmitted data, ensuring the integrity and accuracy of the information being transmitted.

The project highlighted the significance of error detection and correction techniques in digital communication systems, particularly in applications where data integrity is critical. By using CRC codes, we can improve the reliability of data transmission and prevent errors from compromising the transmitted information.

Overall, the project provided a comprehensive solution for detecting and correcting errors in digital signal processing using CRC codes. The implementation of CRC codes in Python and the development of a user-friendly GUI made it easy for users to understand and apply CRC codes in various digital communication systems.