10.48047/jocaaa.2024.33.08.50

FSM DESIGN FOR ENSURING CORRECT BLOCK LENGTHS IN SERIAL COMMUNICATION

Mrs.N.SWARUPA RANI [(1)] , Mr.M.RAMANA REDDY [(2)] , Mr.K.Ch.MALLA REDDY [(3),] MUPPURI NAVYA [(4),] GUNDAMCHARLA AMMANI [(5),] EPPALA CHARULATHA [(6)]
[1,2,3] Faculty ECE Department, Krishna Chaitanya Institute of Technology & Sciences-Markapur, AP, India.
[4,5,6,]Student , ECE Department, Krishna Chaitanya Institute of Technology & Sciences,Markapur, AP, India.

Abstract. In serial communication, data is transmitted in blocks of 1s and 0s, and the validity of these blocks is determined by their lengths. Specifically, blocks of 1s must have an odd length, while blocks of 0s must have an even length. This paper presents the design of a Mealy state machine (FSM) for detecting errors in such serial data transmission. The FSM monitors the data stream and asserts an error signal whenever it encounters a block of 1s with an even length or a block of 0s with an odd length, thus ensuring the integrity of the transmitted data. The state machine employs a simple yet effective method of counting the lengths of consecutive 1s and 0s, and it switches between states based on the current data value. A testbench is also provided to validate the design by simulating various valid and invalid transmission scenarios. The proposed solution can be applied to systems where maintaining specific data block lengths is critical, offering a robust mechanism for error detection in real-time communication systems.

I.INTRODUCTION

Serial communication is a widely used method for transmitting data between electronic devices, where data is sent sequentially, bit by bit, over a communication channel. However, ensuring that data blocks are transmitted with the correct length is essential for maintaining synchronization, preventing data loss, and ensuring accurate message interpretation.

This project focuses on designing and implementing a system to ensure correct block lengths in serial communication. It aims to detect and correct errors related to improper block sizes, misalignment, or missing data. The project will incorporate data framing techniques, error detection mechanisms (such as checksums or parity bits), and flow control strategies to enhance reliability and efficiency.

By implementing these techniques, the project will help improve data integrity, minimize transmission errors, and ensure smooth communication between devices in serial protocols such as UART, SPI, I2C, and CAN. The outcome of this project will contribute to more reliable serial communication systems used in industrial automation, embedded systems, and networking applications.

In serial communication, the block length refers to the number of bits or bytes that are transmitted together as a single unit. Choosing the correct block length is crucial to ensure reliable and efficient data transmission.

1.1 Importance of Block Length

1. Error Detection and Correction: A longer block length can provide better error detection and correction capabilities, but it also increases the overhead of error-checking bits.
2. Data Throughput: A shorter block length can increase data throughput, but it may also increase the overhead of transmission control bits.

10.48047/jocaaa.2024.33.08.50

3. Transmission Efficiency: The block length affects the transmission efficiency, as a longer block length can reduce the number of transmission control bits needed.

1.2 Factors Affecting Block Length Choice

1. Data Rate: Higher data rates typically require shorter block lengths to maintain transmission efficiency.
2. Error Rate: Higher error rates require longer block lengths to ensure reliable data transmission.
3. Latency: Applications requiring low latency may benefit from shorter block lengths.
4. Protocol Overhead: The block length choice should consider the protocol overhead, such as transmission control bits and error-checking bit.

II.EXISTING SYSTEM

A. 5GL

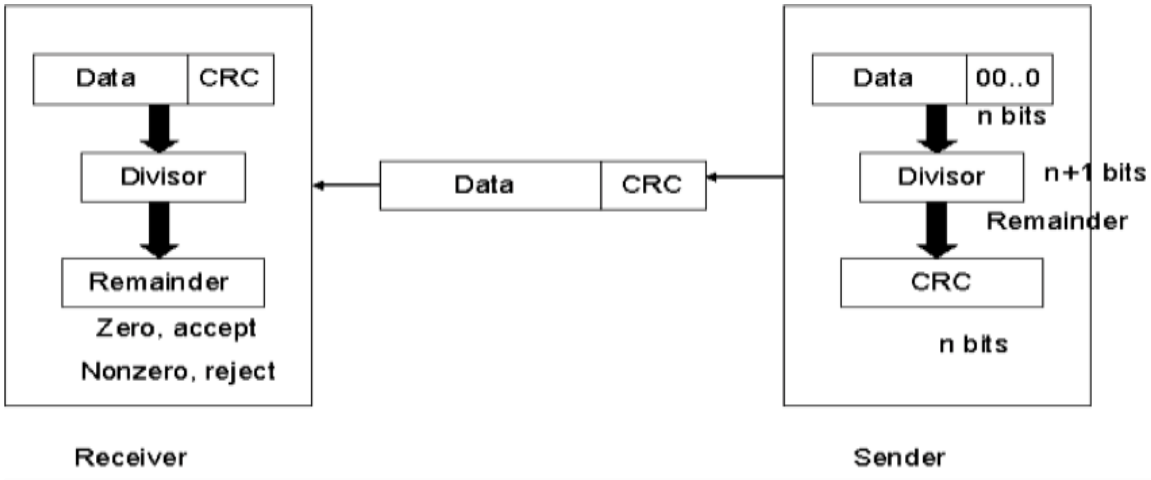


Fig 1:Block diagram of serial communication with crc

Cyclic Redundancy Check (CRC) is an error-detecting technique commonly used in digital communication systems to ensure data integrity during transmission. The primary goal of CRC is to detect accidental changes to raw data during transmission. The process of CRC involves both the sender and the receiver, each playing a crucial role in error detection.On the sender side, the process begins with the original data that needs to be transmitted. To prepare the data for error checking, the sender appends a series of zero bits (usually referred to as padding) to the end of the data. The number of zero bits added corresponds to the degree of the divisor polynomial used in the CRC process. This step is essential to ensure that the division can be performed correctly.Next, the sender performs a division operation, specifically modulo-2 division, between the padded data and a predetermined binary divisor (also known as a generator polynomial). The result of this division is a remainder, which represents the CRC value. This remainder, which is typically a fixed number of bits, is then appended to the original data, forming a complete message that includes both the data and the CRC value. The combined message is then transmitted over the communication channel to the receiver.At the receiver side, the received message, which contains both the original data and the appended CRC, undergoes the same division operation using the same divisor polynomial that was used by the sender. The objective of this division is to verify the correctness of the transmitted data. If the result of the division (remainder) is zero, it indicates that no errors have occurred during transmission, and the data is accepted as accurate. However, if the remainder is nonzero, it signifies that the data has been corrupted or altered during transmission, leading the receiver

10.48047/jocaaa.2024.33.08.50

to reject the data.In summary, CRC is an efficient and widely used technique to detect errors in transmitted data. By performing polynomial division both at the sender and receiver sides and comparing remainders, CRC helps ensure that data integrity is maintained throughout the communication process.

III. PROPOSED SYSTEM

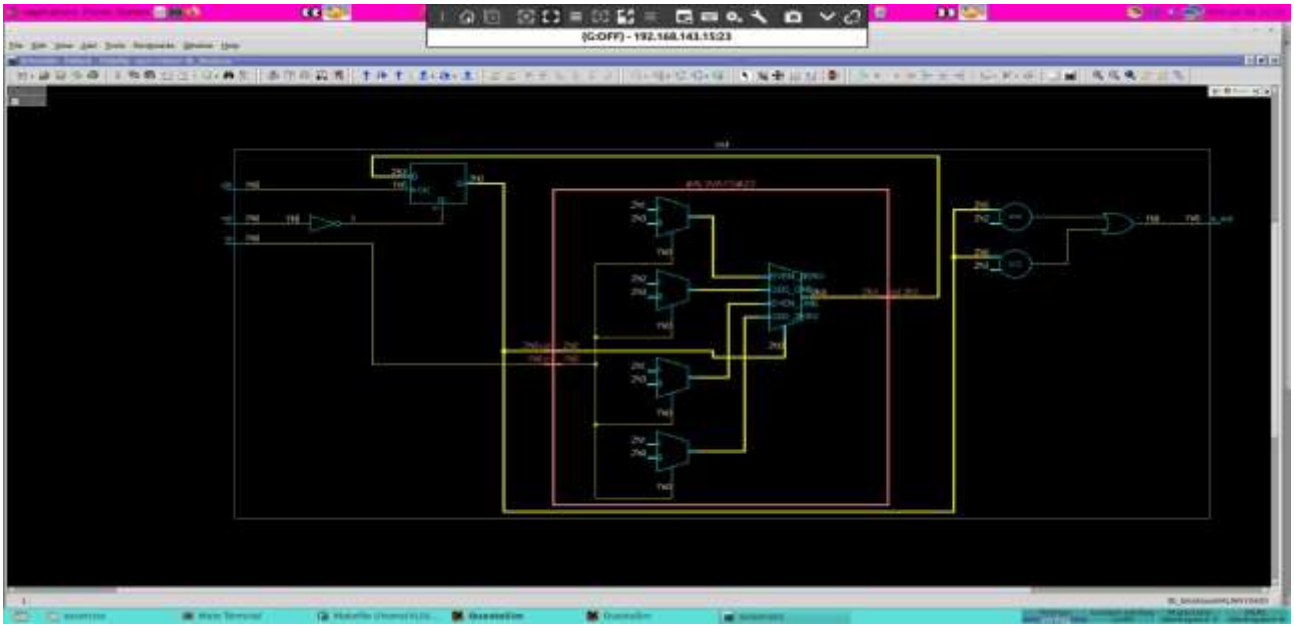


Fig 2: Architecture on ensuring correct block length

WORKING:

The project implements a reliable data transmission system using Reed-Solomon (RS) coding to ensure error-free communication. The process begins with system initialization, where the necessary hardware and software components are configured. Next, the raw data is segmented into smaller, manageable blocks, which are then passed through the RS encoding phase to generate parity symbols, forming codewords that enhance error detection and correction capabilities. The encoded data is transmitted over a communication channel, where it may encounter noise and interference, leading to potential errors. Upon reception, the system verifies the block length to check for inconsistencies, followed by block length correction if needed. Depending on whether the block length is even or odd, the

10.48047/jocaaa.2024.33.08.50

system processes the data accordingly, applying RS decoding and error correction algorithms to fix any detected errors. Once the errors are corrected, the system verifies the integrity of the corrected data and, if successful, forwards it to the communication protocol handler for further processing. The finite state machine (FSM) manages the transitions between states, ensuring smooth and efficient error handling. This structured approach ensures robust and accurate data transmission even in noisy environments.

The project presented here aims to establish a robust and reliable data transmission system using **Reed-Solomon (RS) coding**, an effective error-correction technique widely used in communication systems. The primary objective of the system is to ensure data integrity during transmission, even in the presence of noise and interference that can corrupt the transmitted data. The system workflow begins with the **initialization phase**, where hardware and software components are prepared for communication. Once initialized, the system proceeds to **data segmentation**, where the input data is divided into smaller, manageable blocks. This segmentation is crucial as it not only reduces complexity but also enhances the efficiency of error detection and correction. After segmentation, the data blocks are passed through the **RS encoding phase**, where parity symbols are appended to each block to create a codeword. These parity symbols are instrumental in detecting and correcting multiple errors in the received data. The encoded data is then sent through the **transmission channel**, which could be a wired or wireless medium. During transmission, the data is prone to noise, interference, and other disturbances that may introduce errors, affecting the accuracy of the received data.

Upon reaching the receiver, the data undergoes the **reception phase**, where the received blocks are collected for further processing. At this stage, the system performs **block length verification** to ensure that the length of the received blocks matches the expected length. Any discrepancy in block length indicates the presence of errors, triggering the **block length correction** process. Depending on whether the block length is identified as even or odd, the data is categorized accordingly and processed through appropriate channels. The **Reed-Solomon decoder** then analyzes the received codewords and attempts to correct the errors using the parity symbols embedded during encoding. If successful, the corrected data proceeds to the **error correction phase**, where final adjustments are made to ensure data accuracy.

The system employs a **finite state machine (FSM)** to manage transitions between various states, including initialization, transmission, reception, error detection, and correction. This FSM ensures that the system operates smoothly and efficiently without any conflict between states. Additionally, **communication protocols** are integrated into the system to regulate data flow and maintain synchronization between the sender and receiver. These protocols also handle retransmissions in case of persistent errors or critical failures. Once the errors are corrected and the data integrity is verified, the system moves to the final stage where the corrected data is forwarded to the designated application or storage. The project's architecture is designed to maximize reliability and minimize error propagation, making it ideal for real-time communication systems where data accuracy is paramount. The combination of **Reed-Solomon coding, block length management, and FSM control**

10.48047/jocaaa.2024.33.08.50

ensures that the system can handle varying error conditions while maintaining robust data integrity and communication efficiency.

IV. RESULTS AND ANALYSIS DISCUSSION

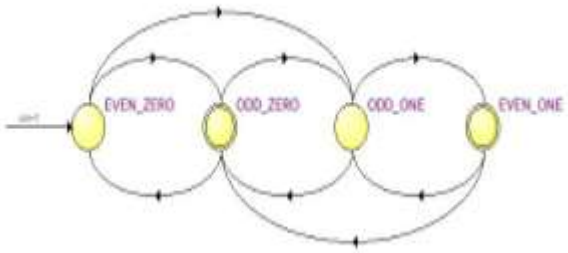
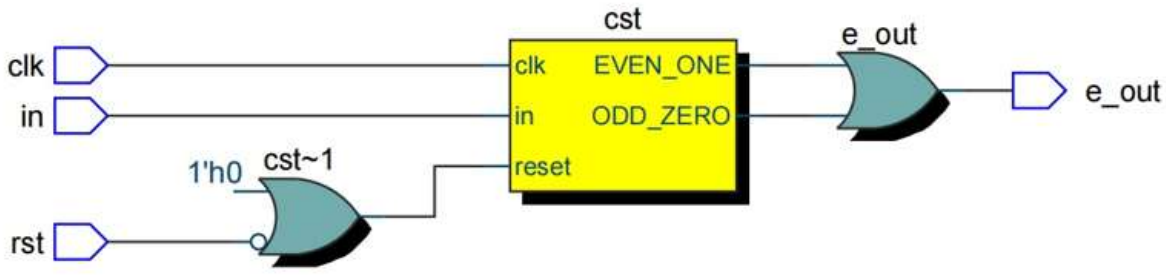


Fig3: RTL schematic diagram.



10.48047/jocaaa.2024.33.08.50

Fig4: Synthesis

Output wave form

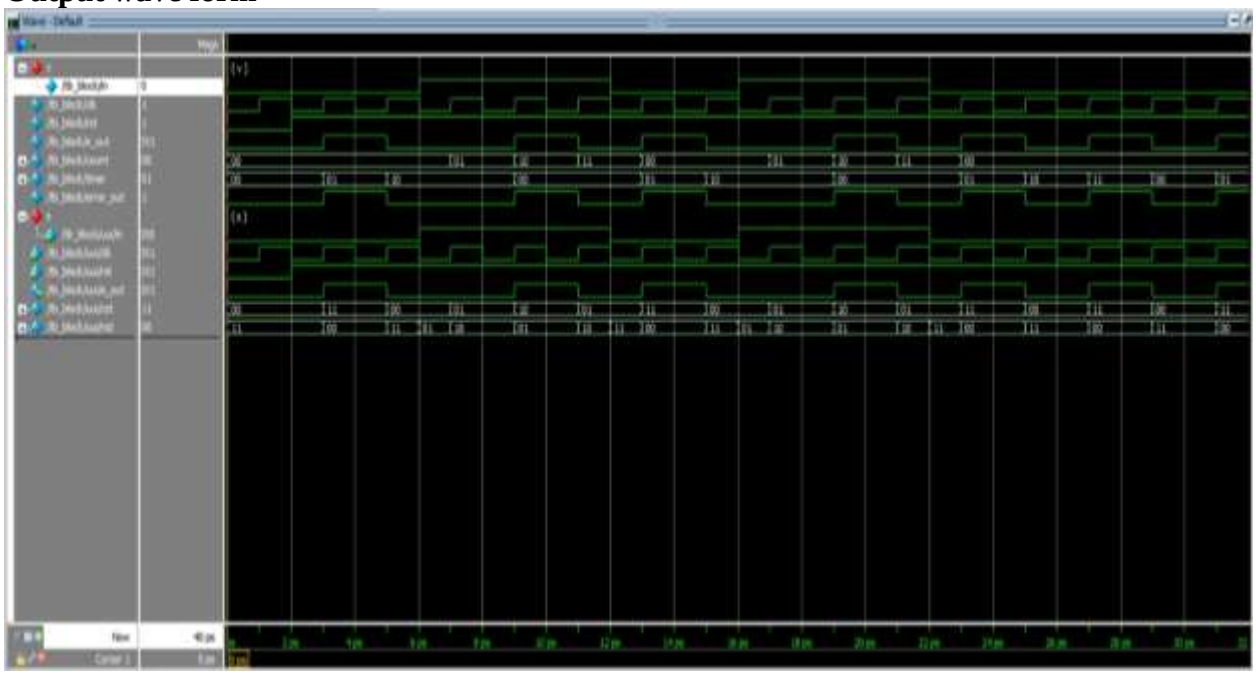


Fig5: Simulation wave form

V CONCLUSION

The design and implementation of Reed-Solomon codes for ensuring correct block length in serial communication provide a highly reliable method for error detection and correction. These codes effectively manage burst errors and improve data integrity in noisy transmission environments. By incorporating Reed-Solomon coding, communication systems achieve enhanced reliability and efficiency, making them suitable for critical applications such as telecommunications, data storage, and deep-space communication.

VI REFERENCES

1. Reed, I. S., & Solomon, G. (1960). Polynomial Codes Over Certain Finite Fields. Journal of the Society for Industrial and Applied Mathematics, 8(2), 300–304.

2. Blahut, R. E. (2003). Algebraic Codes for Data Transmission. Cambridge University Press.

3. Wicker, S. B., & Bhargava, V. K. (1999). Reed-Solomon Codes and Their Applications. Wiley-IEEE Press.

4. Lin, S., & Costello, D. J. (2004). Error Control Coding, Second Edition. Pearson.

5. Berlekamp, E. R. (1968). Algebraic Coding Theory. McGraw-Hill.

10.48047/jocaaa.2024.33.08.50

6. MacWilliams, F. J., & Sloane, N. J. A. (1977). The Theory of Error-Correcting Codes. North-Holland Mathematical Library.

7. Moon, T. K. (2005). Error Correction Coding: Mathematical Methods and Algorithms. Wiley-Interscience.

8. Peterson, W. W., & Weldon, E. J. (1972). Error-Correcting Codes. MIT Press.

9. Schlegel, C., & Perez, L. C. (2004). Trellis and Turbo Coding: Iterative and Graph-Based Error Control Coding. Wiley.

10. ITU-T Recommendation G.975.1 (2004). Forward Error Correction for High Bit-Rate DWDM Submarine Systems.