



KLE Technological
University
Creating Value
Leveraging Knowledge

**School of
Electronics and Communication Engineering**

Mini Project Report

on

**NB-IoT CRC Calculation: A Comparison
Between Series And Parallel Architecture**

By:

- | | |
|----------------------|--------------------|
| 1. Manjunath Inamati | USN : 01FE21BEC356 |
| 2. Goutami Naragund | USN : 01FE21BEC177 |
| 3. Chetan Paranatti | USN : 01FE21BEC163 |
| 4. Eshaan Shetty | USN : 01FE21BEC097 |
| 5. Chhavi Hiremath | USN : 01FE22BEC402 |

Semester: V, 2023-2024

Under the Guidance of

Saroja V Siddamal

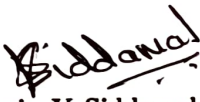
K.L.E SOCIETY'S
KLE Technological University,
HUBBALLI-580031
2023-2024




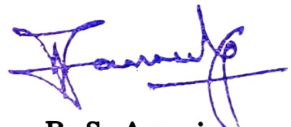
SCHOOL OF ELECTRONICS AND COMMUNICATION
ENGINEERING

CERTIFICATE

This is to certify that project entitled “ NB-IoT CRC Calculation: A Comparison Between Series And Parallel Architecture” is a bonafide work carried out by the student team of ”Manjunath Inamati(01FE21BEC356), Goutami Naragund(01FE21BEC177), Chetan Paranatti(01FE21BEC163), Eshaan Shetty (01FE21BEC097), Chhavi Hiremath(01FE22BEC402)”. The project report has been approved as it satisfies the requirements with respect to the mini project work prescribed by the university curriculum for BE (V Semester) in School of Electronics and Communication Engineering of KLE Technological University for the academic year 2023-2024.


Saroja V Siddamal
Guide


Suneeta V Budihal
Head of School


B. S. Anami
Registrar

External Viva:

Name of Examiners

1. Dr. Tanuja Patel
2. Dr. Saroja V. S.

Signature with date


17/12/24

ACKNOWLEDGMENT

We express our gratitude to the faculty and management for their professional guidance throughout the project's completion. We would like to extend our thanks to Dr. Ashok Shettar, Vice-Chancellor at KLE Technological University, Hubballi, for their visionary leadership and unwavering support. Special thanks to Dr. Suneeta V Budihal, Professor and Head of SoECE, for providing us with direction and fostering our skills and academic growth.

We thank our guide, Dr. Saroja V Siddamal, for their constant guidance during interactions and reviews. We acknowledge the reviewers for their valuable suggestions and inputs. Additionally, we express our thanks to the Project Coordinator Prof. Rajeshwari M. and other faculty in-charges for their support throughout the completion of the project.

Manjunath Inamati
Goutami Naragund
Chetan Paranatti
Eshaan Shetty
Chhavi Hiremath

ABSTRACT

Message corruption is a common problem in communication systems. The received signal should always be verified to know whether the signal is corrupted, or the same is transmitted. In communication systems, the CRC(Cyclic Redundancy Check) is an essential block for ensuring data integrity and error detection. Particularly our work focuses on 16-bit CRC using both series and parallel architectures used in low-power and low-bandwidth IoT devices like NB-IoT(Narrowband Internet of Things) modules. The series architecture involves sequential processing of data bits, while the parallel architecture utilizes parallel processing to enhance throughput and reduce latency. The design considerations and implementation details of both architectures are discussed, along with performance evaluation in terms of processing speed, and resource utilization. The hardware implementation of both are carried out on Spartan-6 FPGA using the Xilinx EDA tool. The simulation and implementation results are verified.

Contents

| | | |
|----------|--|-----------|
| 1 | Introduction | 9 |
| 1.1 | Motivation | 9 |
| 1.2 | Objectives | 9 |
| 1.3 | Literature survey | 10 |
| 1.4 | Problem statement | 10 |
| 1.5 | Application in Societal Context | 10 |
| 1.6 | Project Planning | 11 |
| 1.7 | Organization of the report | 11 |
| 2 | System design | 12 |
| 2.1 | Generator Polynomial | 12 |
| 2.1.1 | Hamming Distance | 12 |
| 2.2 | Manual calculation of 16-bit CRC | 13 |
| 2.3 | Functional block diagram | 13 |
| 2.3.1 | Series CRC | 13 |
| 2.3.2 | Parallel CRC | 14 |
| 3 | Implementation details | 18 |
| 3.1 | Specifications and final system architecture | 18 |
| 3.2 | Algorithm | 18 |
| 3.2.1 | Series | 18 |
| 3.2.2 | Parallel | 19 |
| 3.3 | Flowchart | 20 |
| 4 | Results and discussions | 21 |
| 4.1 | Result Analysis | 21 |
| 4.1.1 | Series CRC | 21 |
| 4.1.2 | Parallel CRC | 24 |
| 4.1.3 | Comparison | 27 |
| 5 | Conclusions and future scope | 28 |
| 5.1 | Conclusion | 28 |
| 5.2 | Future scope | 28 |
| | References | 28 |

List of Tables

| | | |
|-----|------------------------------|----|
| 2.1 | After one shift | 15 |
| 2.2 | After two shifts | 16 |
| 2.3 | After three shifts | 16 |
| 2.4 | After four shifts | 16 |
| 2.5 | After five shifts | 16 |
| 2.6 | After six shifts | 16 |
| 2.7 | After seven shifts | 17 |
| 2.8 | After eight shifts | 17 |
| 4.1 | Comparison table | 27 |

List of Figures

| | | |
|------|---|----|
| 1.1 | CRC Block of NB-IoT. | 9 |
| 2.1 | “Best” polynomials for HD at given CRC size and data word length. | 12 |
| 2.2 | Manual CRC calculation | 13 |
| 2.3 | Series Implementation. | 14 |
| 2.4 | Visual representation of parallel processing. | 15 |
| 2.5 | Parallel Implementation. | 15 |
| 3.1 | Board | 18 |
| 3.2 | Flow chart. | 20 |
| 4.1 | Simulation. | 21 |
| 4.2 | Example 1. | 22 |
| 4.3 | Example 2. | 22 |
| 4.4 | Example 3. | 23 |
| 4.5 | Example 4. | 23 |
| 4.6 | Simulation. | 24 |
| 4.7 | Example 1. | 25 |
| 4.8 | Example 2. | 25 |
| 4.9 | Example 3. | 26 |
| 4.10 | Example 4. | 26 |

Chapter 1

Introduction

Cyclic Redundancy Check is a block used in the communication system to add redundancy bits to the given data to detect the errors that occurred during the transmission of data.



Figure 1.1: CRC Block of NB-IoT.

CRC block can be implemented using series and parallel architecture. This is based on specifications such as a number of data bits and number of CRC bits. If the resultant CRC should be n bits then the polynomial should be of $(n+1)$ bits.

Output $Y = K + R$

Where Y – Length of output data.

K – Length of input data.

R – Length of redundant bits.

1.1 Motivation

The job involves designing and implementing an efficient CRC block for an NB-IoT system with specifications of a 16-bit CRC for the 34-bit input data stream to generate an output of 50 bits. The performance of both series and parallel CRC configurations must be compared and verified. A hardware implementation using an appropriate FPGA is required to validate the results.

1.2 Objectives

1. Design and implementation of 16-bit Series CRC for 34-bit data.
2. Design and implementation of 16-bit Parallel CRC for 34-bit data.
3. Performance comparison of series and parallel CRC.

1.3 Literature survey

1. Kanj, Matthieu, Vincent Savaux, and Mathieu Le Guen. "A tutorial on NB-IoT physical layer design." *IEEE Communications Surveys & Tutorials* 22, no. 4 (2020): 2408-2446 [2]. In this work, the authors aim to provide complete information on the physical layer of the NB-IoT system including the general overview and its features. The authors have provided a detailed description of the overall transmission chain of the NB-IoT evolved node base and user equipment. The operation modes, transmission modes, framing system, and transmission options in uplink and downlink are described in the paper. The transmission chain of the NB-IoT eNB consists of various blocks such as CRC, channel encoding, scrambler, rate matching, etc. The application in which we are focusing is on the transmission chain of the NB-IoT eNB, particularly on the CRC block.
2. Koopman, Philip, and Tridib Chakravarty. "Cyclic redundancy code (CRC) polynomial selection for embedded networks." In *International Conference on Dependable Systems and Networks*, 2004, pp. 145-154. IEEE, 2004 [3]. This paper presents a comprehensive analysis of the error detection performance of cyclic redundancy codes (CRCs) for embedded networks and other applications. The authors propose a set of standards for choosing "good" CRC polynomials that take into account various factors and maximize the Hamming distance for the longest data word lengths. In addition, they offer a thorough analysis of every CRC polynomial between 3 and 15 bits, contrasting them with previously published and standardized polynomials. The research presents better alternatives and shows that many widely used CRC polynomials perform poorly or are inappropriate for particular data word lengths. The paper also shows the potential of improved CRCs for error detection and examines various case studies of existing methods that use CRCs. The paper aims to provide quantitative information and engineering guidelines for CRC selection for embedded network designers.
3. Parallel Cyclic Redundancy Check (CRC) for Hotlink [1]. The capacity of CRC codes to detect faults and the creation of parallel implementation for serial data are discussed by the writers in this study. It gives details on how to calculate 16-bit CRC for each eight-bit chunk. The 32-bit CRC polynomial has also been calculated, one 16-bit chunk at a time.

1.4 Problem statement

Hardware implementation of 16-bit series and parallel CRC for 34-bit input data in an NB-IoT system.

1.5 Application in Societal Context

The CRC Block finds applications in various societal contexts contributing to the reliability and integrity of data transmission. It is widely used in wireless network protocols ensuring errorless data transmission. It is also used in file transfer protocol and data storage systems which help in detecting errors during file downloads ensuring that the received files are identical to the original files.

1.6 Project Planning

1. Understanding CRC block specifications for NB-IoT systems.
2. Implementing both series and parallel CRC architectures to compare the performance of both and coming to the conclusion where exactly these architectures have to be implemented.
3. Make the output of our block ready to be fed as input to the next block.

1.7 Organization of the report

- Chapter 2 consists of manual CRC calculation, Polynomial selection, functional block diagrams of series and parallel architecture.
- Chapter 3 consists of specifications, system architecture and flowchart.
- Chapter 4 has the implementation results of both series and parallel CRC along with their comparison.
- Chapter 5 consists of conclusion and future scope.

Chapter 2

System design

System design elaborates the functional block diagram of all the architectures.

2.1 Generator Polynomial

2.1.1 Hamming Distance

- The Hamming distance of a generator polynomial is the minimum number of bit changes required to transform it into another valid generator polynomial.
- Lesser the hamming distance better the performance of the crc.

| Max length at HD Polynomial | CRC Size (bits) | | | | | | | | | | | | | | |
|--------------------------------|---------------------|---------------------|----------------------|----------------------|----------------------|----------------------|-----------------------|-----------------------|-----------------------|----------------------|-----------------------|-----------------------|-----------------------|-----------------------|--|
| | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | |
| HD=2 | 2048+ <u>0x5</u> | 2048+ <u>0x9</u> | 2048+ <u>0x12</u> | 2048+ <u>0x21</u> | 2048+ <u>0x48</u> | 2048+ <u>0xA6</u> | 2048+ <u>0x167</u> | 2048+ <u>0x327</u> | 2048+ <u>0x64D</u> | – | – | – | – | – | |
| HD=3 | | 11 <u>0x9</u> | 26 <u>0x12</u> | 57 <u>0x21</u> | 120 <u>0x48</u> | 247 <u>0xA6</u> | 502 <u>0x167</u> | 1013 <u>0x327</u> | 2036 <u>0x64D</u> | 2048 <u>0xB75</u> | – | – | – | – | |
| HD=4 | | | 10 <u>0x15</u> | 25 <u>0x2C</u> | 56 <u>0x5B</u> | 119 <u>0x97</u> | 246 <u>0x14B</u> | 501 <u>0x319</u> | 1012 <u>0x583</u> | 2035 <u>0xC07</u> | 2048 <u>0x102A</u> | 2048 <u>0x21E8</u> | 2048 <u>0x4976</u> | 2048 <u>0xBAAD</u> | |
| HD=5 | | | | | | 9 <u>0x9C</u> | 13 <u>0x185</u> | 21 <u>0x2B9</u> | 26 <u>0x5D7</u> | 53 <u>0x8F8</u> | none | 113 <u>0x212D</u> | 136 <u>0x6A8D</u> | 241 <u>0xAC9A</u> | |
| HD=6 | | | | | | | 8 <u>0x13C</u> | 12 <u>0x28E</u> | 22 <u>0x532</u> | 27 <u>0xB41</u> | 52 <u>0x1909</u> | 57 <u>0x372B</u> | 114 <u>0x573A</u> | 135 <u>0xC86C</u> | |
| HD=7 | | | | | | | | | 12 <u>0x571</u> | none | 12 <u>0x12A5</u> | 13 <u>0x28A9</u> | 16 <u>0x5BD5</u> | 19 <u>0x968B</u> | |
| HD=8 | | | | | | | | | | 11 <u>0xA4F</u> | 11 <u>0x10B7</u> | 11 <u>0x2371</u> | 12 <u>0x630B</u> | 15 <u>0x8FDB</u> | |

Figure 2.1: “Best” polynomials for HD at given CRC size and data word length.

- In NB-IoT a 16-bit CRC should be appended to 34-bit input data. So, the generator polynomial 0xBAAD is selected which has the least hamming distance (HD = 4) among all the 16-bit polynomial mentioned in the table of reference [3].

Polynomial - $x^{16} + x^{15} + x^{13} + x^{12} + x^{11} + x^9 + x^7 + x^5 + x^3 + x^2 + 1$

Polynomial in HEX - 0xBAAD

Polynomial in Binary - 11011101010101101

2.2 Manual calculation of 16-bit CRC

The following figure gives the manual calculation of a 16-bit CRC specified for the selected generator polynomial (0xBAAD) with sample input data.

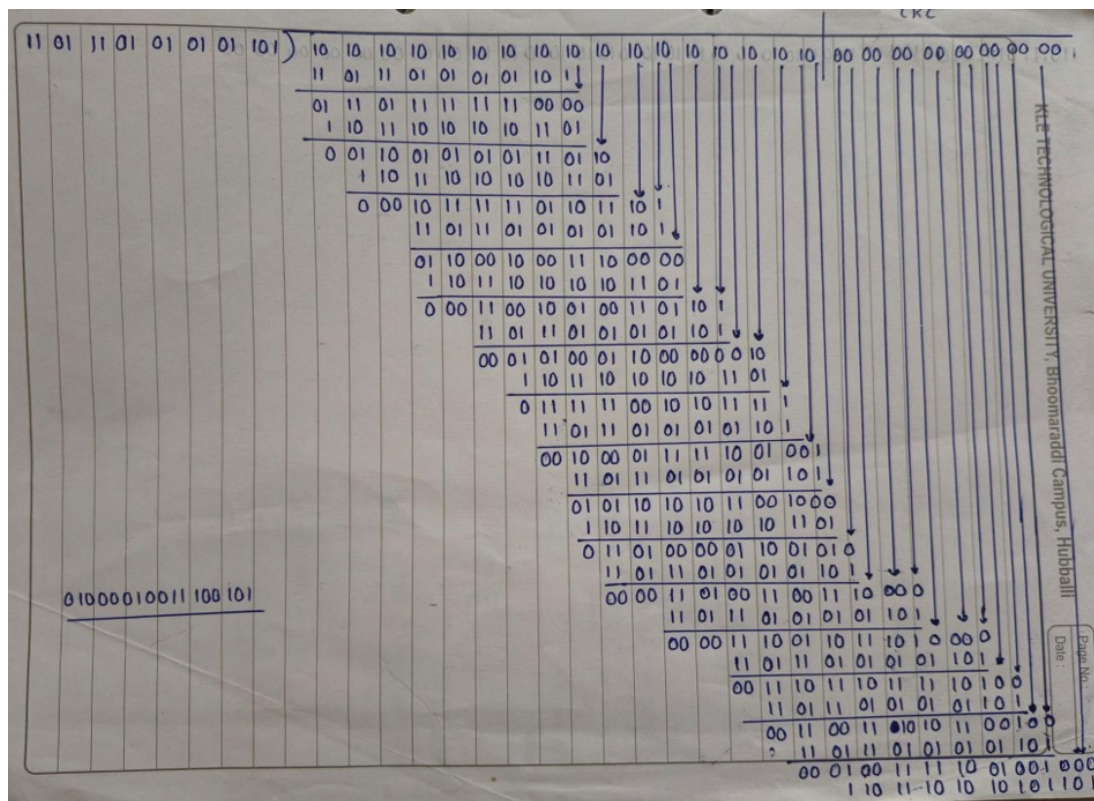


Figure 2.2: Manual CRC calculation

2.3 Functional block diagram

2.3.1 Series CRC

Carrying out the manual calculation, We got to know that implementing the series CRC requires flip-flops for the data storing and XOR gates for binary subtraction. The architecture is based on the binary division carried out by shifting the polynomial bits with respect to the data bits. The data bits are fed to the Linear Feedback Shift Registers (LFSR) from D33 (MSB) to D0 (LSB). The XOR gates are placed according to the coefficients of the polynomial selected for the calculation. The following figure gives the series architecture CRC implementation. The final bits in the registers give the 16-bit CRC which has to be appended to the data.

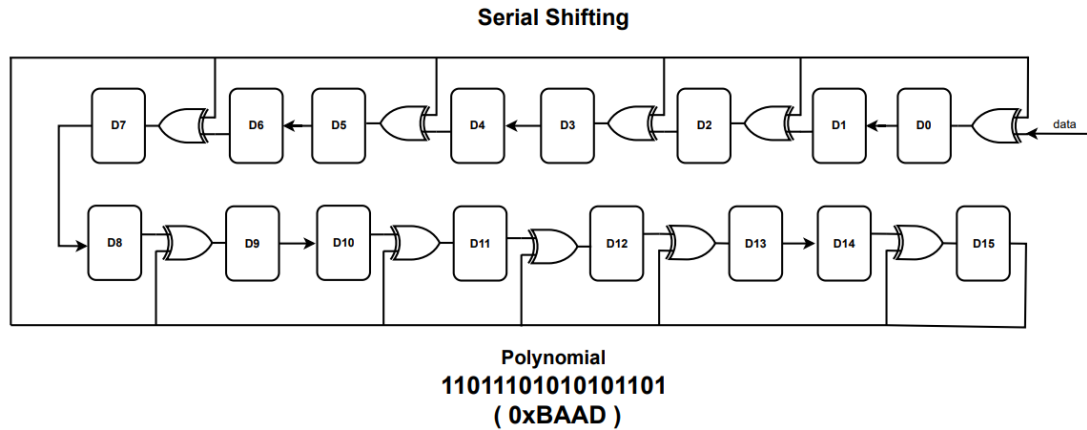


Figure 2.3: Series Implementation.

2.3.2 Parallel CRC

Parallel CRC is a different method of approach to calculate the redundancy bits for a given data. There are different techniques for parallel CRC generation given as follows.

1. A Table-Based Algorithm for Pipelined CRC Calculation.
2. Fast CRC Update.
3. F-matrix-based parallel CRC generation.
4. Unfolding, Retiming and Pipelining Algorithm.

We have implemented the parallel architecture using the "Table-Based Algorithm for pipelined CRC calculation" method. Table-based method is constructing the tables for the given bits of data and CRC. The following are the notes and properties to be taken care of while constructing the table.

NOTE

1. $(SR)_n$ is the n th bit of the CRC shift register.
2. C_n is the content of n -th bit of the initial CRC shift register before any shifts have taken place.
3. SR_0 is the LSB.
4. The entries under each CRC shift register bit indicate the value to be XORed together to generate the content of that bit in the CRC shift register.
5. D_n is the data input, with MSB input first.
6. D_7 is the MSB of the input byte, and D_0 is the LSB.

PROPERTIES

1. **Commutative Property** ($X \text{ XOR } Y = Y \text{ XOR } X$).
2. **Associative Property** ($X \text{ XOR } Y \text{ XOR } Z = X \text{ XOR } Z \text{ XOR } Y$).

3. Involution Property ($X \text{ XOR } X = 0$).

Based on the generator polynomial selected for our application the following tables have been constructed for 8-bit data and 16-bit CRC. The number of data bits corresponds to the number of tables to be constructed. As the complexity of the table increases after every shift, the table for 8-bit data has been constructed and the calculation of CRC for the whole 34-bit has been carried out by dividing the data into 8-bit chunks. The 34-bit data is rounded off to 40-bit data by appending the zeros at MSB to divide data equally into 8-bit chunks. The following figure shows the calculation of CRC for 34-bit data in steps.

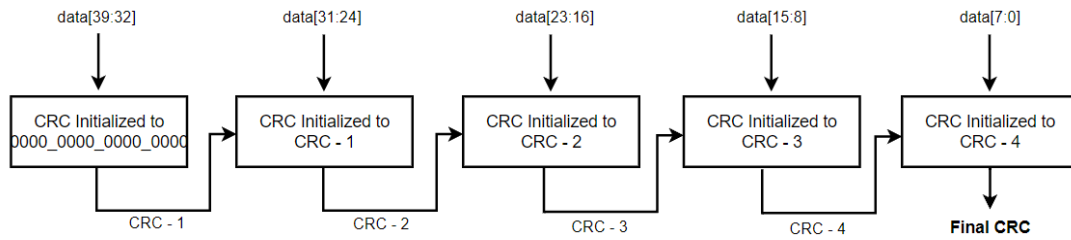


Figure 2.4: Visual representation of parallel processing.

Parallel CRC architecture for generator polynomial 0xBAAD gives the visual representation of the table entries after every shift of data input.

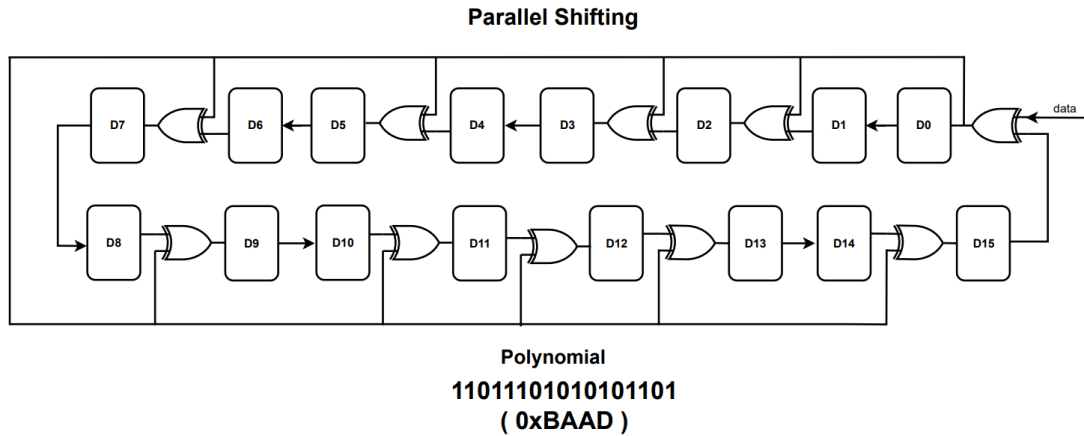


Figure 2.5: Parallel Implementation.

Based on the architecture the following tables are constructed after every shift. Initially, the contents of the CRC for the first chunk of the data are made zero. Later the initial contents of (n+1)-th chunk are initialised with CRC bits of n-th chunk. The final contents of the table after 8-shift have to be XORed to get the corresponding bits of CRC for given 8-bit data.

Table 2.1: After one shift

| SR15 | SR14 | SR13 | SR12 | SR11 | SR10 | SR9 | SR8 | SR7 | SR6 | SR5 | SR4 | SR3 | SR2 | SR1 | SR0 |
|------|------|------|------|------|------|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| D7 | C13 | D7 | D7 | D7 | C9 | D7 | C7 | D7 | C5 | D7 | C3 | D7 | D7 | C0 | D7 |
| C15 | | C15 | C15 | C15 | | C15 | | C15 | | C15 | | C15 | C15 | | C1 |
| C14 | | C12 | C11 | C10 | | C8 | | C6 | | C4 | | C2 | C1 | | |

Table 2.2: After two shifts

| SR15 | SR14 | SR13 | SR12 | SR11 | SR10 | SR9 | SR8 | SR7 | SR6 | SR5 | SR4 | SR3 | SR2 | SR1 | SR0 |
|------|------|------|------|------|------|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| D6 | D7 | D6 | D6 | D7 | D7 | D7 | D7 | D7 | C7 | D7 | D7 | D6 | D6 | D7 | D6 |
| D7 | C15 | C14 | C14 | D6 | C15 | D6 | C15 | D6 | C15 | D6 | C15 | C14 | C14 | C15 | C14 |
| C15 | C12 | C11 | C10 | C14 | C8 | C14 | C6 | C14 | C4 | C14 | C2 | C1 | D7 | | D7 |
| C14 | | | | C15 | | C15 | | C15 | | C15 | | | C15 | | C15 |
| C13 | | | | C9 | | C7 | | C5 | | C3 | | | C0 | | |

Table 2.3: After three shifts

| SR15 | SR14 | SR13 | SR12 | SR11 | SR10 | SR9 | SR8 | SR7 | SR6 | SR5 | SR4 | SR3 | SR2 | SR1 | SR0 |
|------|------|------|------|------|------|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| D5 | D6 | D5 | D5 | D5 | D6 | D5 | D6 | D5 | D6 | D5 | D6 | D5 | D5 | D6 | D5 |
| D6 | C14 | D7 | C13 | D6 | D7 | D6 | D7 | D6 | D7 | D6 | C14 | C13 | D6 | D7 | D6 |
| C13 | C11 | C13 | C9 | C13 | C14 | C13 | C14 | C13 | C14 | C13 | C1 | C0 | C13 | C14 | D7 |
| C14 | | C15 | | C14 | C15 | C14 | C15 | C14 | C15 | C14 | | | C14 | C15 | C13 |
| C12 | | C10 | | C8 | C7 | C6 | C5 | C4 | C3 | C2 | | | | | C14 |
| | | | | | | | | | | | | | | | C15 |

Table 2.4: After four shifts

| SR15 | SR14 | SR13 | SR12 | SR11 | SR10 | SR9 | SR8 | SR7 | SR6 | SR5 | SR4 | SR3 | SR2 | SR1 | SR0 |
|------|------|------|------|------|------|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| D4 | D5 | D4 | D4 | D4 | D5 | D4 | D5 | D4 | D5 | D4 | D5 | D4 | D4 | D5 | D4 |
| D5 | D7 | D6 | C12 | D5 | D6 | D5 | D6 | D5 | D6 | D5 | C13 | C12 | D5 | D6 | D5 |
| C12 | C13 | C12 | C8 | D7 | C13 | D7 | C13 | D7 | C13 | C12 | C0 | | D7 | D7 | D6 |
| C13 | C15 | C14 | | C12 | C14 | C12 | C14 | C12 | C14 | C13 | | | C12 | C13 | C12 |
| C11 | C10 | C9 | | C13 | C6 | C13 | C4 | C13 | C2 | C1 | | | C13 | C14 | C13 |
| | | | | C15 | | C15 | | C15 | | | | | C15 | C15 | C14 |
| | | | | C7 | | C5 | | C3 | | | | | | | |

Table 2.5: After five shifts

| SR15 | SR14 | SR13 | SR12 | SR11 | SR10 | SR9 | SR8 | SR7 | SR6 | SR5 | SR4 | SR3 | SR2 | SR1 | SR0 |
|------|------|------|------|------|------|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| D3 | D4 | D3 | D3 | D3 | D4 | D3 | D4 | D3 | D4 | D3 | D4 | D3 | D3 | D4 | D3 |
| D4 | D6 | D5 | D7 | D4 | D5 | D4 | D5 | D4 | D5 | D4 | C12 | D7 | D4 | D5 | D4 |
| D7 | C12 | C11 | C11 | D6 | D7 | D6 | D7 | D6 | C12 | C11 | | C11 | D6 | D6 | D5 |
| C11 | C14 | C13 | C15 | C11 | C12 | C11 | C12 | C11 | C13 | C12 | | C15 | D7 | C12 | C11 |
| C12 | C9 | C8 | C7 | C12 | C13 | C12 | C13 | C12 | C1 | C0 | | | C11 | C13 | C12 |
| C15 | | | | C14 | C15 | C14 | C15 | C14 | | | | | C12 | C14 | C13 |
| C10 | | | | C6 | C5 | C4 | C3 | C2 | | | | | C14 | | |
| | | | | | | | | | | | | | C15 | | |

Table 2.6: After six shifts

| SR15 | SR14 | SR13 | SR12 | SR11 | SR10 | SR9 | SR8 | SR7 | SR6 | SR5 | SR4 | SR3 | SR2 | SR1 | SR0 |
|------|------|------|------|------|------|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| D2 | D3 | D2 | D2 | D2 | D3 | D2 | D3 | D2 | D3 | D2 | D3 | D2 | D2 | D3 | D2 |
| D3 | D5 | D4 | D6 | D3 | D4 | D3 | D4 | D3 | D4 | D3 | D7 | D6 | D3 | D4 | D3 |
| D6 | C11 | C10 | D7 | D5 | D6 | D5 | D6 | D5 | C11 | D7 | C11 | C10 | D5 | D5 | D4 |
| D7 | C13 | C12 | C10 | C10 | C11 | C10 | C11 | D7 | C12 | C10 | C15 | C14 | D6 | C11 | D7 |
| C10 | C8 | C7 | C14 | C11 | C12 | C11 | C12 | C1 | C0 | C11 | | | D7 | C12 | C10 |
| C11 | | | C15 | C13 | C14 | C13 | C14 | C10 | | C15 | | | C10 | C13 | C11 |
| C14 | | | C6 | C5 | C4 | C3 | C2 | C11 | | | | | C11 | | C12 |
| C9 | | | | | | | | C13 | | | | | C13 | | C15 |
| C15 | | | | | | | | C15 | | | | | C14 | | |
| | | | | | | | | | | | | | C15 | | |

Table 2.7: After seven shifts

| SR15 | SR14 | SR13 | SR12 | SR11 | SR10 | SR9 | SR8 | SR7 | SR6 | SR5 | SR4 | SR3 | SR2 | SR1 | SR0 |
|------|------|------|------|------|------|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| D1 | D2 | D1 | D1 | D1 | D2 | D1 | D2 | D1 | D2 | D1 | D2 | D1 | D1 | D2 | D1 |
| D2 | D4 | D3 | D5 | D2 | D3 | D2 | D3 | D2 | D3 | D2 | D6 | D5 | D2 | D3 | D2 |
| D5 | C10 | C9 | D6 | D4 | D5 | D4 | D5 | D4 | D7 | D6 | C10 | C9 | D4 | D4 | D3 |
| D6 | C12 | C11 | D7 | D7 | C10 | D7 | D7 | D6 | C10 | C9 | C14 | C13 | D5 | D7 | D6 |
| D7 | C7 | C6 | C9 | C9 | C11 | C9 | C1 | D7 | C11 | C10 | | | D6 | C10 | D7 |
| C9 | | | C13 | C10 | C13 | C10 | C10 | C9 | C15 | C14 | | | D7 | C11 | C9 |
| C10 | | | C14 | C12 | C3 | C12 | C11 | C12 | | | | | C9 | C12 | C10 |
| C13 | | | C15 | C4 | | C15 | C13 | C14 | | | | | C10 | C15 | C11 |
| C14 | | | C5 | C15 | | C2 | C15 | C15 | | | | | C12 | | C14 |
| C15 | | | | | | | | C10 | | | | | C13 | | C15 |
| C8 | | | | | | | | C0 | | | | | C14 | | |
| | | | | | | | | | | | | | C15 | | |

Table 2.8: After eight shifts

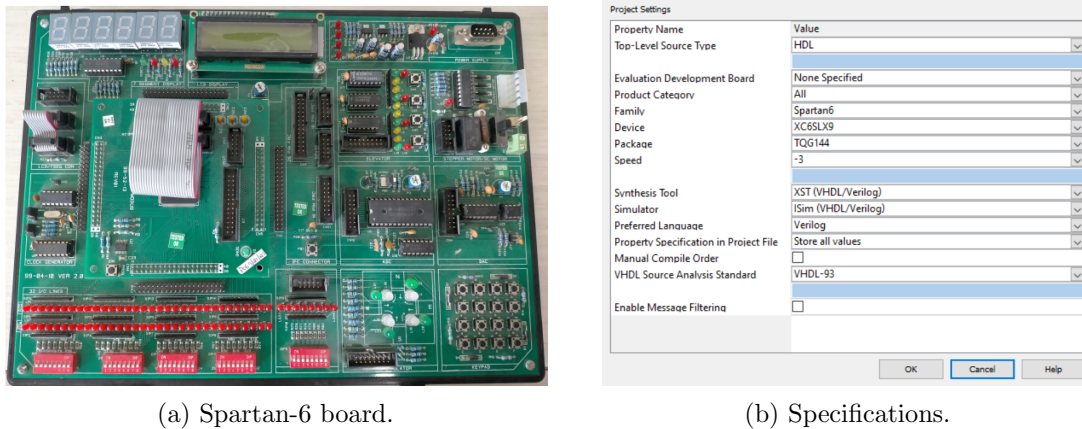
| SR15 | SR14 | SR13 | SR12 | SR11 | SR10 | SR9 | SR8 | SR7 | SR6 | SR5 | SR4 | SR3 | SR2 | SR1 | SR0 |
|------|------|------|------|------|------|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| D0 | D1 | D0 | D0 | D0 | D1 | D0 | D1 | D0 | D1 | D0 | D1 | D0 | D0 | D1 | D0 |
| D1 | D3 | D2 | D4 | D1 | D2 | D1 | D2 | D1 | D2 | D1 | D5 | D4 | D1 | D2 | D1 |
| D4 | C9 | C8 | D5 | D3 | D4 | D3 | D4 | D3 | D6 | D5 | C9 | C8 | D3 | D3 | D2 |
| D5 | C11 | C10 | D6 | D6 | D7 | D6 | D6 | D5 | C9 | D7 | C13 | C12 | D4 | D6 | D5 |
| D6 | C6 | C5 | C8 | D7 | C9 | C8 | D7 | D6 | C10 | C8 | | | D5 | D7 | D6 |
| D7 | | | C12 | C8 | C10 | C9 | C9 | C8 | C14 | C9 | | | D6 | C9 | D7 |
| C8 | | | C13 | C9 | C12 | C11 | C12 | C9 | | C13 | | | C8 | C10 | C8 |
| C9 | | | C14 | C11 | C15 | C14 | C14 | C11 | | C15 | | | C9 | C11 | C9 |
| C12 | | | C4 | C14 | C2 | C1 | C15 | C13 | | | | | C11 | C14 | C10 |
| C13 | | | | C15 | | | C10 | C14 | | | | | C12 | C15 | C13 |
| C14 | | | | C3 | | | C0 | | | | | | C13 | | C14 |
| C15 | | | | | | | | | | | | | C14 | | C15 |
| C7 | | | | | | | | | | | | | | | |

Chapter 3

Implementation details

3.1 Specifications and final system architecture

Fig 3.1 shows spartan-6 FPGA and its specifications.



(a) Spartan-6 board.

(b) Specifications.

Figure 3.1: Board

3.2 Algorithm

3.2.1 Series

- Sequential circuit to carry out the shifting of the bits.
- Data is fed from MSB as it is done in manual calculations.
- The values are XORed at the registers where the coefficients of the polynomial are high.
- Initially, The MSB has to be fed to the R0 register for the first cycle.
- The next bit has to be fed in the next cycle until all the bits of the data are completely fed into the circuit.
- The XORed bits have to be fed into the next registers to carry out the binary division.
- After all the bits are fed into the circuit the final XORed values give the 16-bit CRC for a given data.

```

module Series(clk,reset,data_in,data_out);
input [33:0]data_in;
input clk,reset;
output [49:0]data_out;

```

3.2.2 Parallel

The Structural type verilog coding is used to implement the parallel CRC. Calculating the CRC of each chunk of data is called instances by sending the required CRC initialization values, input data chunk, and clk for synchronization with the main module. The clk_initial instance appends 6 zeros at the MSB to round off the data to 40 bits.

- The data is received and zeros are appended at MSB.
- Calculating the CRC for the first chunk (i.e.,[39:32]data) as initial contents assigned to zero and getting the output as 16-bit out1.
- Calculating the CRC for second chunk (i.e.,[31:24]data) as initial contents assigned to out1 and getting the output as 16-bit out2.
- Calculating the CRC for third chunk (i.e.,[23:16]data) as initial contents assigned to out2 and getting the output as 16-bit out3.
- Calculating the CRC for fourth chunk (i.e.,[15:8]data) as initial contents assigned to out3 and getting the output as 16-bit out4.
- Calculating the CRC for fifth chunk (i.e.,[7:0]data) as initial contents assigned to out4 and getting the output as 16-bit out5.
- Appending 16-bit out5 to 34-bit input data and assigning it to 50-bit output.

```

module Parallel(clk,reset,data_in,data_out);
input [33:0]data_in;
input clk,reset;
output [49:0]data_out;

```

```

module clk_initial(clk,reset,data_in,data);
input clk,reset;
input [33:0]data_in;
output reg [39:0]data ;

```

```

module parallel_compute(clk, reset, in, crc, out);
input clk,reset;
input [7:0]in;
input [15:0]crc;
output reg [15:0]out;

```

3.3 Flowchart

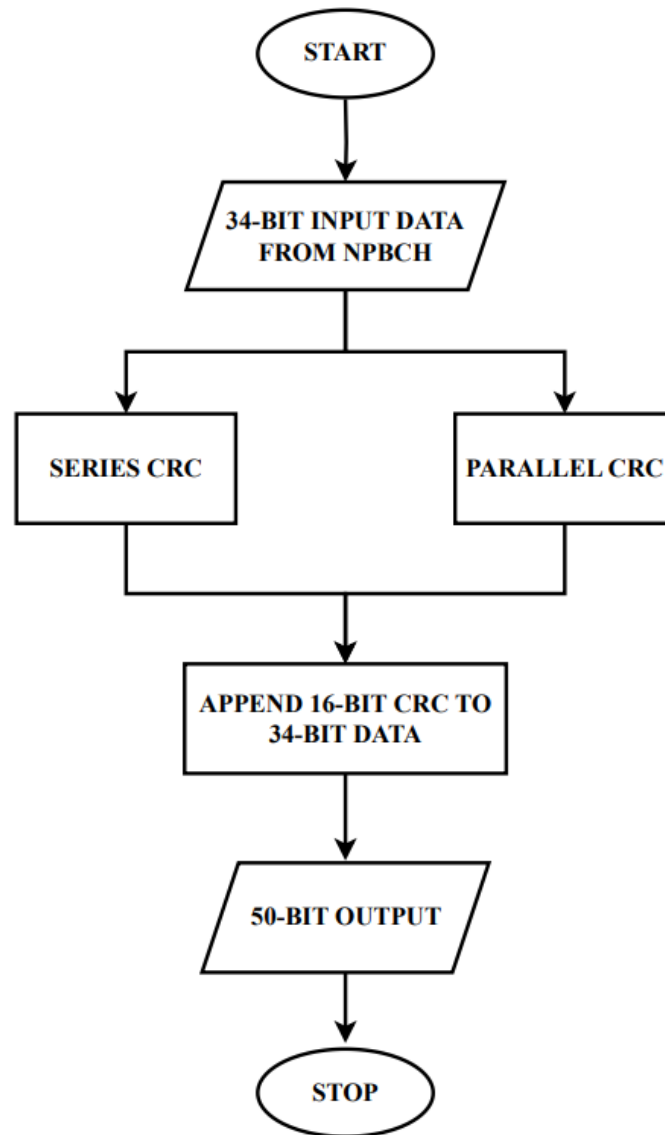


Figure 3.2: Flow chart.

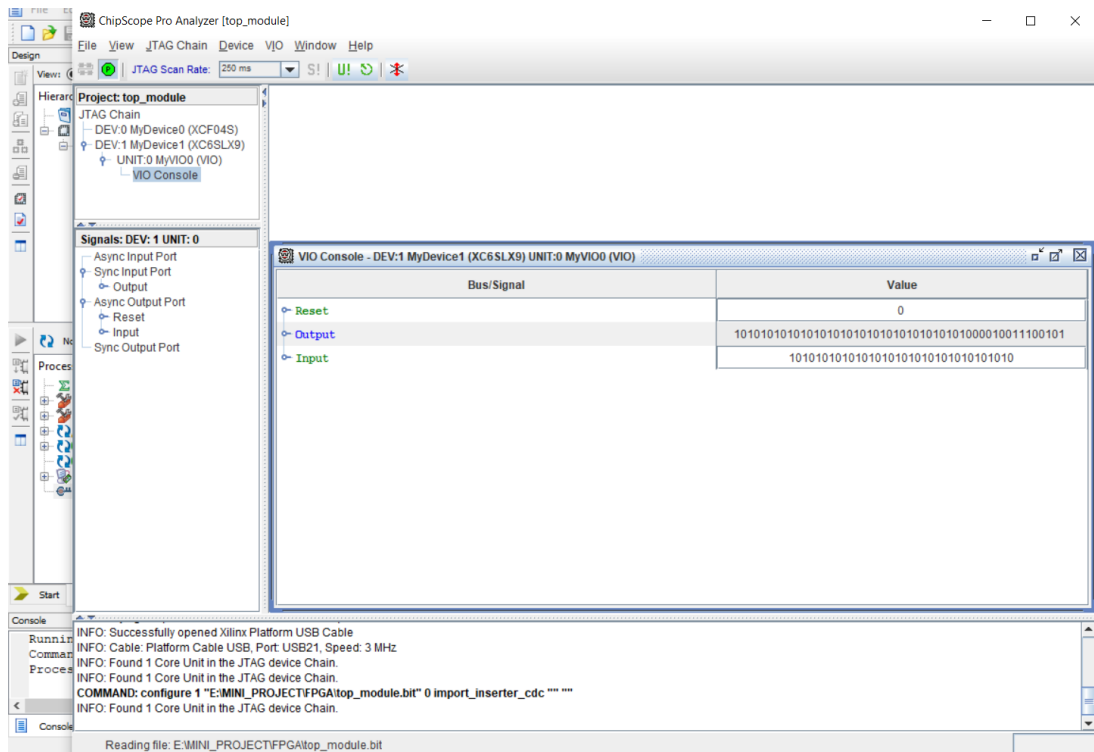


Figure 4.2: Example 1.

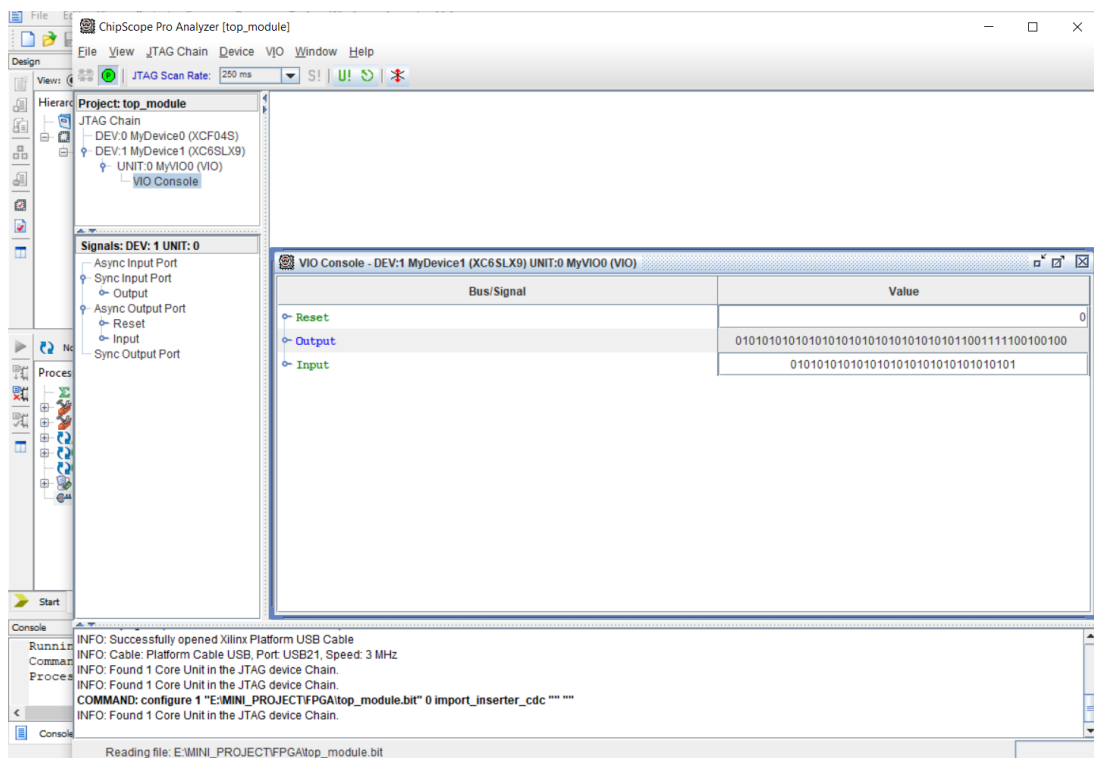


Figure 4.3: Example 2.

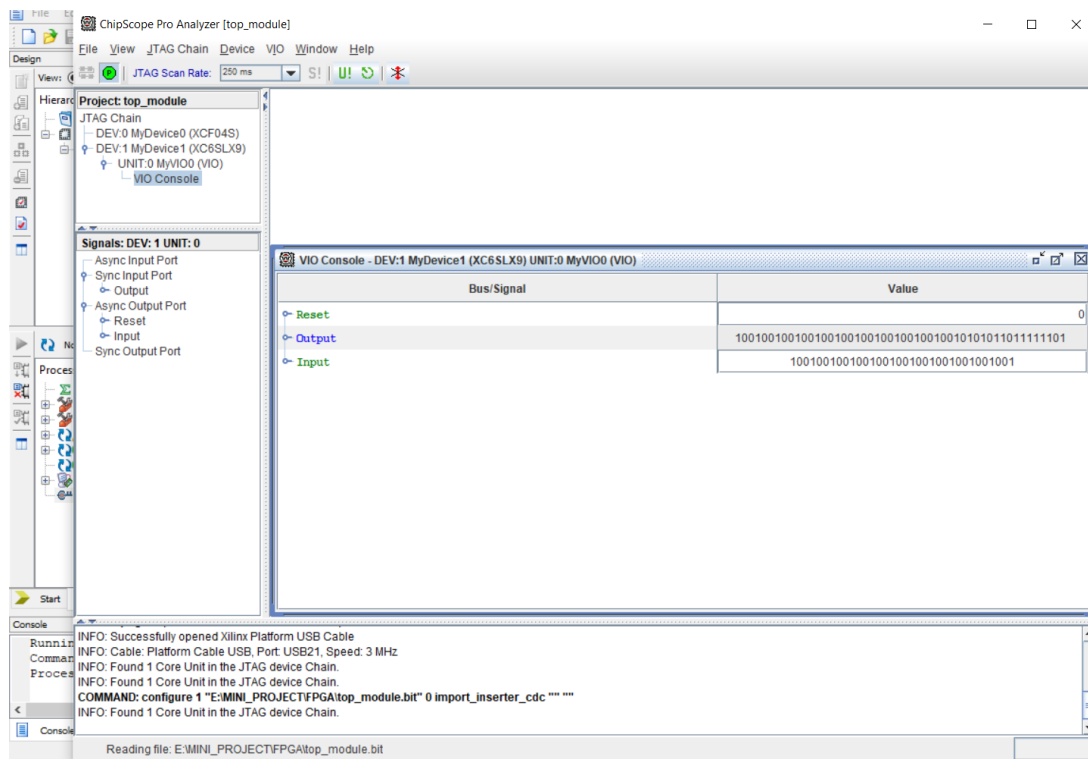


Figure 4.4: Example 3.

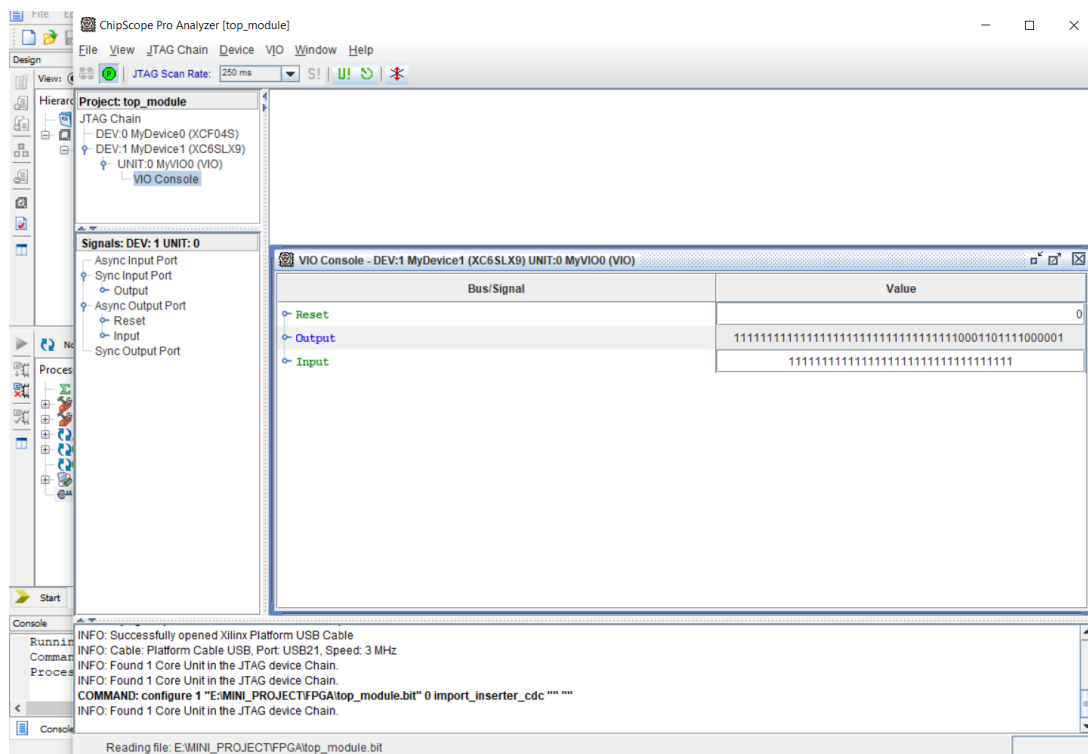


Figure 4.5: Example 4.

4.1.2 Parallel CRC

The following figures show the simulation and hardware implementation of parallel CRC using Spartan-6 FPGA. The 16-bit CRC resultants arrive after 5 cycles when 34-bit data is initialized. The results are verified with the series implementation of CRC carried out earlier and manual calculations.

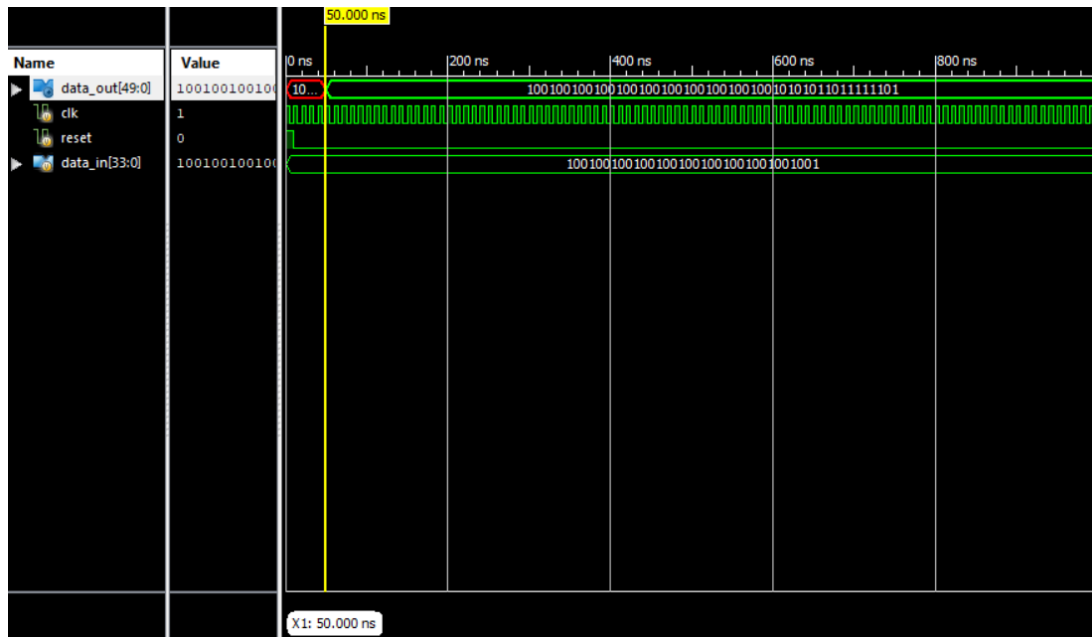
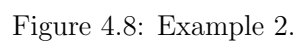
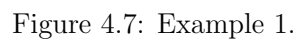


Figure 4.6: Simulation.



4.1.3 Comparison

Table 4.1: Comparison table

| Parameters | Series | Parallel |
|--------------------------|----------|----------|
| Area (μm^2) | 3195.856 | 6185.841 |
| Power (m Watt) | 1.032 | 3.742 |
| No. of clock cycles | 50 | 5 |

Chapter 5

Conclusions and future scope

5.1 Conclusion

Analyzing the parameters like area, power, delay of series and parallel architecture is performed. It is observed that parallel architecture is 90% faster than series architecture, which is the preferred architecture for applications that need faster outputs. Series architecture when compared to parallel is 48.33% lesser in area and 72.42% lesser power is observed.

5.2 Future scope

The CRC block has a major role in every communication system. Its Specifications can be different but the operation carried out in every CRC block is the same. Implemented parallel CRC for 8 shift table is the efficient CRC architecture for the NB-IoT application. The table can be made even more efficient by constricting it to 34 shifts by handling its huge complexity. The series architecture also has a place in the application where area and hardware are the constraints.

Bibliography

- [1] Parallel cyclic redundancy check (crc) for hotlink.
- [2] Matthieu Kanj, Vincent Savaux, and Mathieu Le Guen. A tutorial on nb-iot physical layer design. *IEEE Communications Surveys & Tutorials*, 22(4):2408–2446, 2020.
- [3] Philip Koopman and Tridib Chakravarty. Cyclic redundancy code (crc) polynomial selection for embedded networks. In *International Conference on Dependable Systems and Networks*, pages 145–154. IEEE, 2004.

Report

ORIGINALITY REPORT

18%

SIMILARITY INDEX

17%

INTERNET SOURCES

8%

PUBLICATIONS

%

STUDENT PAPERS

PRIMARY SOURCES

1

data.crazyengineers.com

Internet Source

4%

2

www-dev.cldnet.analog.com

Internet Source

3%

3

www.slideshare.net

Internet Source

2%

4

ijesc.org

Internet Source

1%

5

tudr.thapar.edu:8080

Internet Source

1%

6

Jingtao Li, Adnan Siraj Rakin, Zhezhi He, Deliang Fan, Chaitali Chakrabarti. "RADAR: Run-time Adversarial Weight Attack Detection and Accuracy Recovery", 2021 Design, Automation & Test in Europe Conference & Exhibition (DATE), 2021

Publication

1%

7

archive.org

Internet Source

1%

| | | |
|----|--|------|
| 8 | www.coursehero.com Internet Source | 1 % |
| 9 | hal.archives-ouvertes.fr Internet Source | <1 % |
| 10 | www.arasan.com Internet Source | <1 % |
| 11 | www.collectionscanada.ca Internet Source | <1 % |
| 12 | Vera Pawlowsky-Glahn, Juan José Egozcue, Raimon Tolosana-Delgado. "The Aitchison geometry", Wiley, 2015 Publication | <1 % |
| 13 | www.grin.com Internet Source | <1 % |
| 14 | Matthieu Kanj, Vincent Savaux, Mathieu Le Guen. "A Tutorial on NB-IoT Physical Layer Design", IEEE Communications Surveys & Tutorials, 2020 Publication | <1 % |
| 15 | www.e3s-conferences.org Internet Source | <1 % |
| 16 | Basavaraj Hungund, Chaitanya Habib, Vaibhav Hiregoudar, Sona Umloti, Saiprasad Wandkar, Gururaj Tennalli. "Chapter 5 Production and Characterization of | <1 % |

Hydrophobins from Fungal Source", Springer Nature, 2016

Publication

17

Theresa Maxino. "The Effectiveness of Checksums for Embedded Control Networks", IEEE Transactions on Dependable and Secure Computing, 2008

Publication

<1 %

18

www.freepatentsonline.com

Internet Source

<1 %

19

docshare.tips

Internet Source

<1 %

20

dokumen.pub

Internet Source

<1 %

21

olympias.lib.uoi.gr

Internet Source

<1 %

22

www.thepilot.com

Internet Source

<1 %

23

www.wikizero.com

Internet Source

<1 %

24

Mathukiya, Hitesh H., and Naresh M. Patel. "A Novel Approach for Parallel CRC Generation for High Speed Application", 2012 International Conference on Communication Systems and Network Technologies, 2012.

Publication

<1 %