Efficient CRC error detection using hybrid bitwise operations

Aadarsh Lakshmi Narasiman (211CS201), Sanchit Sudhakar Sakahre(211CS251),
Rishi Diwaker (211CS243)
Department of Computer Science and Engineering
National Institute of Technology Karnataka
Surathkal, Mangalore, India
+91 9611795232, +91 8248838100, +91 9968965073
aadarshln.211cs201@nitk.edu.in, sanchitsudhakarsakhare.211cs251@nitk.edu.in,
rishidiwaker.211cs243@nitk.edu.in

May 28, 2023

1 Abstract

1.1 Background of the problem statement

[1–3]CRC which stands for Cyclic Redundancy Check is an error detection algorithm. By this method, the errors in data and information transmitted over a network is detected.It is performed by applying binary operations over the transmitted data at the sender's end and verifying the same along the receiver's end. This algorithm is derived from CHECKSUM error detection using MODULO division. Here MODULO division is the binary operation used. [4]It is much more efficient than the CHECKSUM method."

1.2 Challenges and issues of the problem statement

[5–8] this method, conventionally we use long division to calculate the CRC. Long division involves the usage of multiple divisions and subtractions.This can be computationally expensive, especially for long messages and also unsuitable for resource constrained systems.

1.3 Existing approaches or methods and their issues

The current CRC method uses polynomial division this is indeed effective with respect to detecting errors introduced during transmission but suffers from [9]high computational complexity due to polynomial division.(Logarithmic Time complexity) As of now we can speed up CRC computations on the processors to handle longer messages. However increasing speed will in turn increase power consumption and costs which is undesirable. There is a need for a more efficient CRC algorithm that can be implemented in resource-constrained systems i.e. an algorithm that is less complex with respect to computation.

1.4 Your problem statement

[10–12]In our approach, we propose a new CRC algorithm that uses a hybrid approach of bitwise operations to achieve error detection while ensuring low computational complexity. This way the CRC error detection is made efficient with respect to computation time and speed using bitwise operations. From a logarithmic complexity, in the case of long division we reduce to linear complexity using bitwise operations.

1.5 Objectives of the proposed work

1. Design an algorithm to carry out the CRC using bitwise operations.The main operation used is XOR.
2. To try and implement the above algorithm in C++ using programming paradigms and analyze its efficiency using graphs in Desmos.

2 Literature Review

2.1 CRC Algorithm

The CRC algorithm is a commonly used method of error detection during data transmission. Ideally in a transmission system there are two components - the sender and the receiver. The sender or the transmitter in a CRC system transmits the data bits followed by the parity bits. The parity bit is the remainder obtained by the coefficient of the remainder polynomial when data polynomial is divided by the CRC Polynomial. CRC Polynomial is a fixed n degree polynomial. The presence of noise in the channel may cause the transmitted bits to not be received correctly. In this case the receiver detects the error in the channel by treating received data bits as coefficients of receiver polynomial and CRC bits are obtained by dividing the data polynomial by CRC polynomial. If the computed CRC isn't the same as the receiver CRC then there is an error in the transmission. In case of error there is a re-transmission of data.

2.2 Existing Optimization Techniques

Here are some of the existing methods of optimization of CRC algorithm:

2.2.1 Hardware Implementation

Ideally while CRC algorithms can be implemented in software by using code, Hardware implementations using shift registers make more practical sense. The reason being that it involves faster speed and lower power consumption.

Advantages

- It increases the speed of processing and further there is no CPU load.
- Hardware implementations allow customizations to the existing model (if needed) and thus is flexible.

Disadvantages

- The cost of implementation can be more than software implementations .
- Hardware design is more complex especially for larger CRC polynomials.
- The maintenance of the hardware components and circuitry is another challenge too.

2.2.2 Parallel processing

Currently with parallel processing trends, it makes more sense to carry out CRC calculations in a parallelized fashion. This means that multiple CRC calculations should be allowed to be run simultaneously thus reducing running time for larger data.

Advantages

- Given that in practicality speed is required for a large data, using parallel processing paradigms makes it more efficient.
- With the growth of parallel processing, it makes it the most apt method of application.

Disadvantages

- The implementation of this is very complex.

2.2.3 Effective CRC Polynomial Selection

The choice of CRC polynomial is essential in the case of performance and has been viewed to improve the performance. This is why standardized polynomials have been set.

Advantages:

- When the polynomial chosen is properly, it can improve the capability of CRC algorithm.

- When the polynomial is effective , it also enhances the computational ability of the algorithm. Thus it makes it faster.

Disadvantage

- The method of chosing polynomials is to be done carefully by testing out different polynomials and measuring performance. This can be time consuming.
- If the polynomial chosen isn't very efficient, it will make the system vulnerable to external attacks.

2.2.4 CRC Algorithm using Table Lookup

[13]A method of implementing the CRC is using the Table lookup method. Here we create something called a lookup table that contains precomputed CRC values for all possible input sequences.

Algorithm 1 Table Lookup CRC algorithm

1: Initialize CRC to all 1's
2: **while** byte in message **do**
3: XOR(byte,higherCRCbyte)
4: Set CRC to lookup value in table with XOR result.
5: Left shift CRC bits by 8 bits.
6: **end while**
7: XOR(CRC,111...)
8: Transmit the original message with calculated CRC appended.
9: On the receiver end, the receiver performs the same CRC calculation with the lookup table and compares calculated CRC with the transmitted one.
10: **if** transmittedCRC = calculatedCRC **then**
11: Message is error free.
12: **else**
13: Message is not error free.
14: **end if**

Algorithm 2 Lookup table calculation

1: Initialize a lookup table with 256 entries of zeroes.
2: Initialize a temporary variable to the input byte value, shifted left by bit length of the CRC polynomial.
3: For each bit position in the input byte (begining from the leftmost bit) if the most significant bit is 1 ,then XOR the temporary variable with the CRC polynomial. Shift the temporary variable left by 1 bit.
4: Store the resulting value in the corresponding entry of the table.

Advantages of CRC using Table Lookup:

- It divides the data into smaller data elements and carries out the operation in this way it reduces the time complexity of the current CRC algorithm and increases the computation speed.
- The additional bits added to the message for error detection is small compared to the size of the original message. This reduces the overhead.

Disadvantages of CRC using Table Lookup:

- With larger CRC polynomials, the table size can go on increasing thus more memory will be needed to store.
- It is a reliable but not a foolproof method of error detection as it may not detect burst errors (this is when 2 or more bits in the data unit have got complemented).

2.3 Comparison between the existing techniques

- The first two methods discuss upon improving algorithmic efficiency by using different existing available hardware and software technologies and paradigm. However the algorithm was retained as same.
- The last two methods focus on improving the efficiency of algorithm by making changes to the existing algorithm. Using various programming techniques.
- The first two set of techniques were more design oriented approaches whereas the last two techniques were logically oriented.

3 Proposed Methodology

The following methodology is proposed by us to further enhance the style of implementing the CRC algorithm. Here we use only XOR operations -

Algorithm 3 Hybrid CRC algorithm

Require: A generator polynomial to obtain generator polynomial bits.
Ensure: Select same number of CRC bits as the number of generator polynomial bits

1: Initialize all CRC bits to zero of length 8.
2: **while** number of bits in the data **do**
3: Perform left shift of the data bits.
4: **if** leftmost bit = 0 **then** continue
5: **else** XOR(current data bit,generator polynomial bits)
6: **end if**
7: **end while**
8: The final value calculate is CRC.
9: This appended to the data and sent as received data. The algorithm is run again. If both CRCs are same, then noise is devoid. Else it is incorrect.

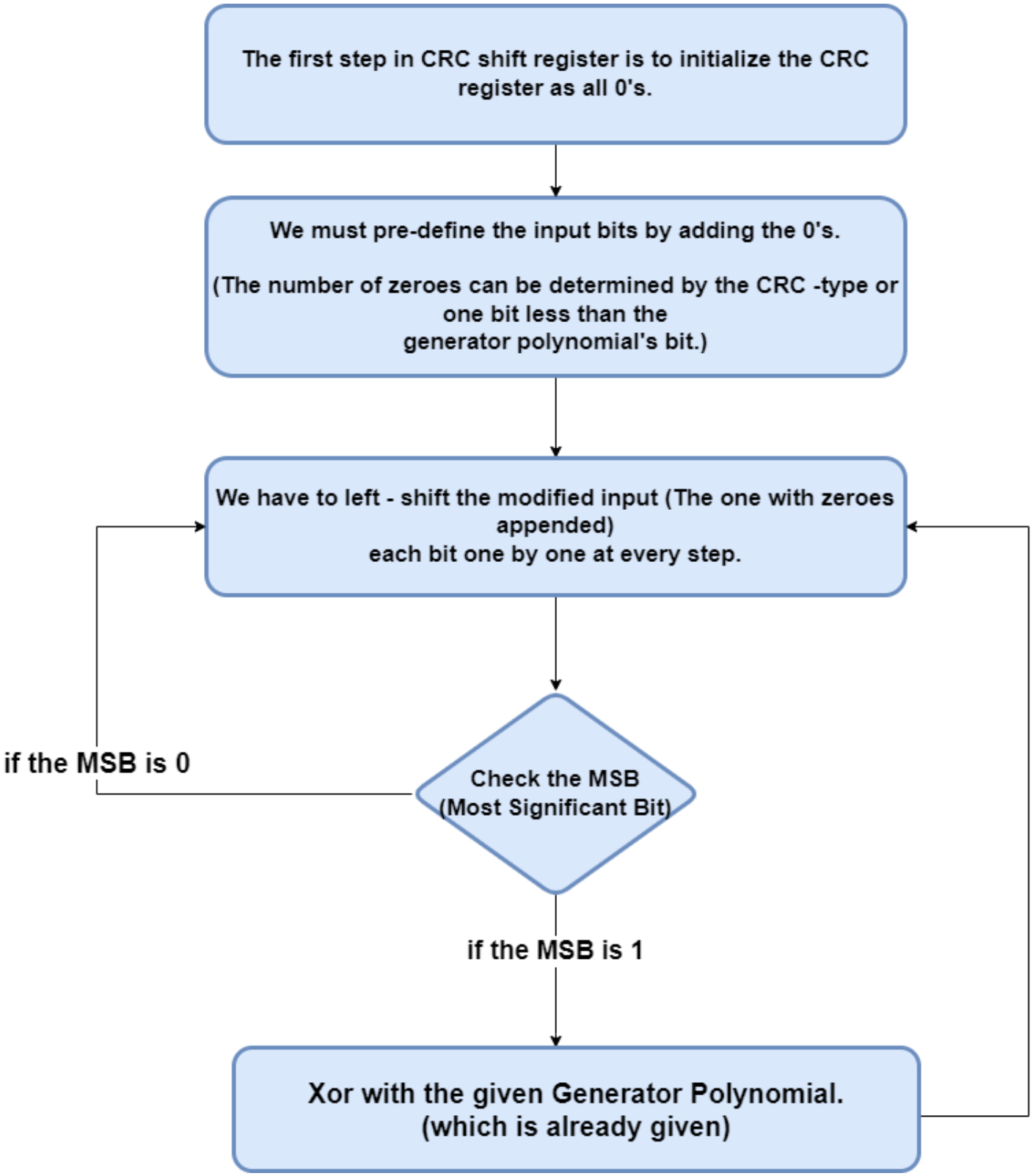

Figure 1: The following flowchart is a figurative description of the algorithm proposed above.

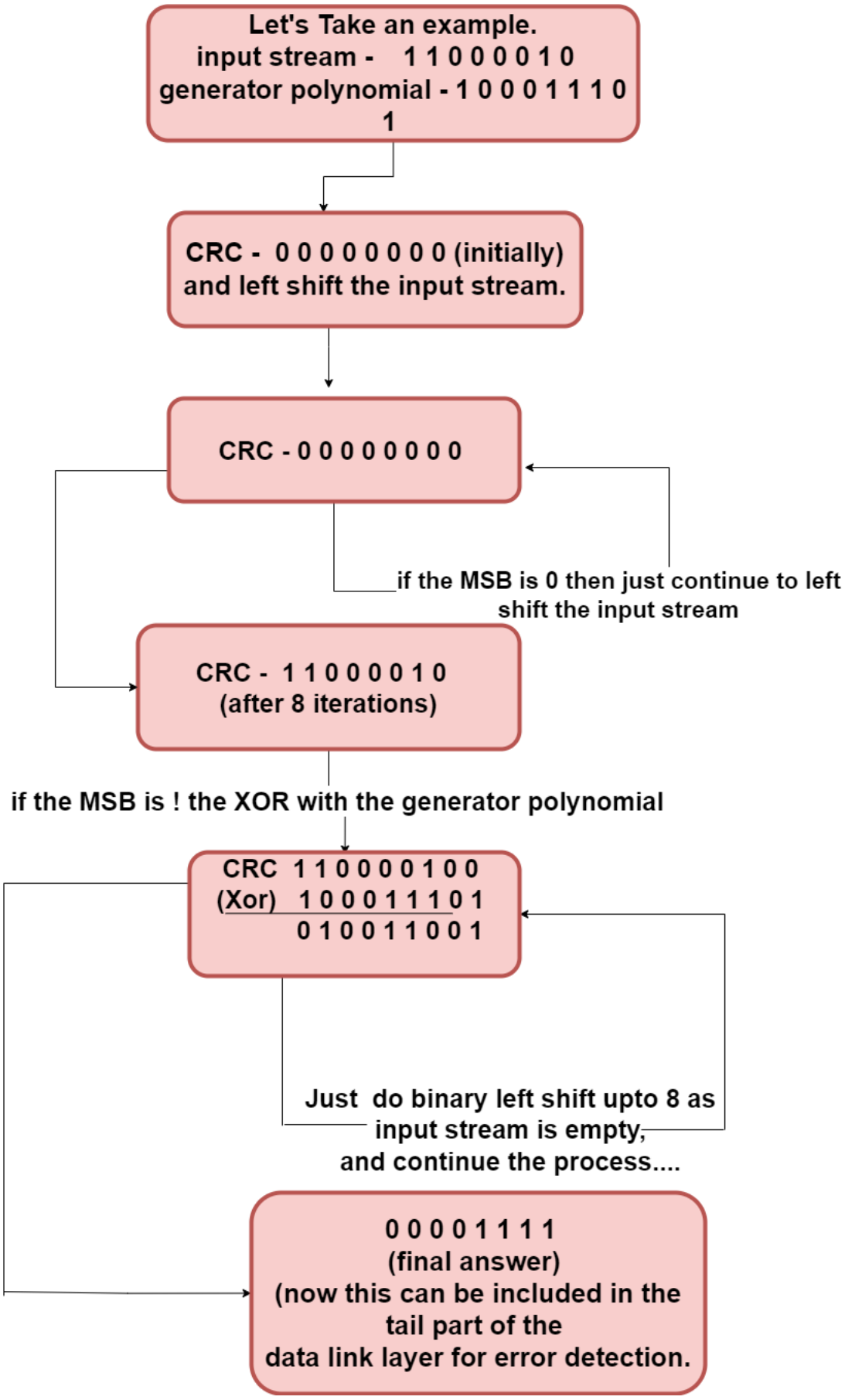

Figure 2: To justify our approach we have taken up the following example.This example has been explained through a flowchart for a CRC-8 example.

4 Implementation

- The implementation was carried out using C++.The calculation involves shifting the elements of the input array to the left, and if the MSB (Most Significant Bit) is 1, performing an XOR operation with the elements of the gen array. This process is repeated for each element of the input array.
- In order to compare and analyze the efficiency of our algorithm we drew a time of execution vs number of frames graph for both algorithms.
- The program measures the execution time for each calculation using the high_resolution_clock from the chrono library. The start time is recorded before the calculation loop, and the end time is recorded after the calculation loop. The duration is then calculated by subtracting the start time from the end time.
- While the conventional method uses bitsets and the calculations involve shifting the bits to the left using inbuilt left shift operator, performing XOR operations, and updating the bits based on the specific rules.
- In both the methods the execution times that were calculated were used as data points and a graph was drawn using Desmos.
- Here we have implemented for only CRC-8 polynomial. However for longer polynomials like CRC-16 and CRC-32 similar trends can be expected as the method would be to use CRC register of longer lengths.

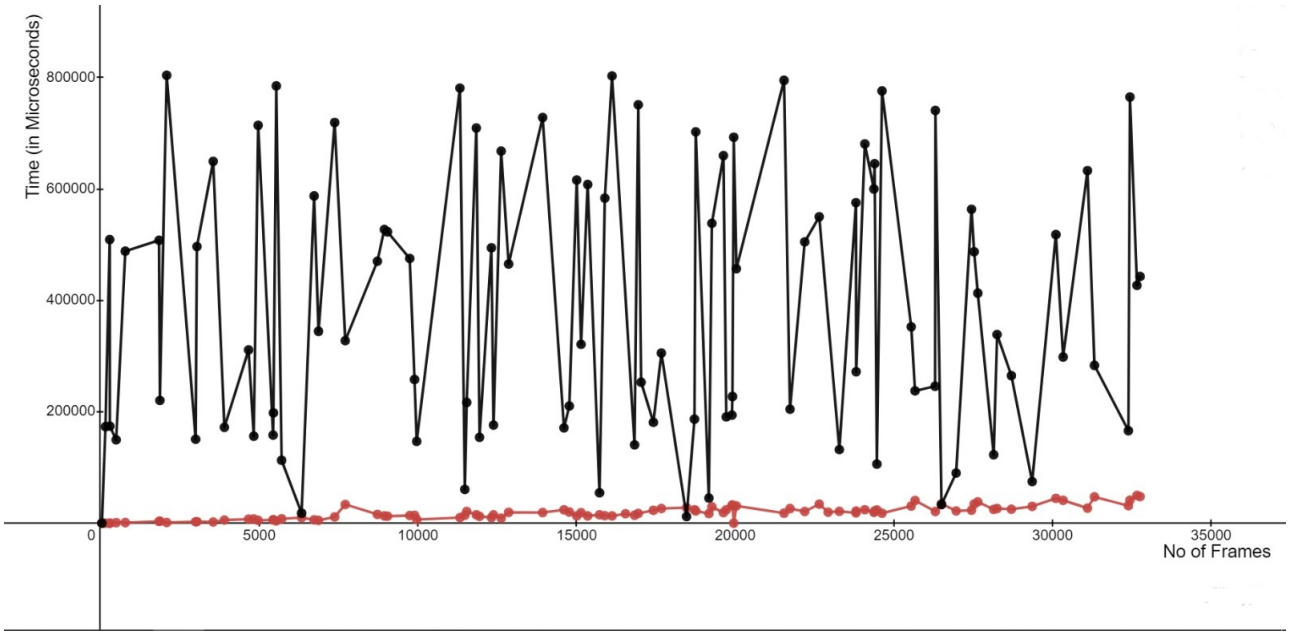

Figure 3: The following graph plots the execution time in microseconds vs number of frames for both the methods. The red plot shows the result of the proposed method and the black plot shows the result shown by the conventional method.

5 Result and Analysis

- The presented graph shows the comparison between time taken by conventional and the proposed method CRC bits calculation as the size of frames increase. The red graph is based on our proposed method and the black one is drawn by running the conventional method.
- It can be deduced from the graph that the proposed method is faster than the long division method to calculate CRC bits.
- The proposed method uses both XOR gates and shift registers. This result is obtained as we have used less number of XOR gate as compared to the conventional method and replaced the rest with shift registers. Shift registers are in general faster than XOR gates. Hence, the result is obtained.

6 Conclusion

In this project report, we have briefly explained Cyclic Redundancy Check(CRC) and its working. We have also looked into the challenges involved in the conventional method of CRC calculation and also currently existing methods on improving the calculation's performance like implementing using hardware , parallel processing paradigms and table lookup methods. Following this, our method of optimizing the CRC calculation was explained with an example for justification. Finally, our method was implemented on C++ with graphs drawn to compare the efficiency of our algorithm with the conventional algorithm based on which it can be concluded that our method has reduced the run-time by significant amounts.For further works, we can look into optimizing the number of XOR gates involved , this way further optimizing the calculation technique.

References

[1] P. W.W, B. D.T., "Cyclic Codes for Error Detection",Proceedings of the IRE, vol. 49, no. 1, pp. 228-235, https://ieeexplore.ieee.org/document/4066263, [Accessed: 07-03-2023] (1961).

[2] F. Paul, A Painless Guide to CRC Error Detection Algorithms, http://www.zlib.net/crc_v3.txt, [Accessed: 07-03-2023] (1993).

[3] W. Stallings, Error Detection and Correction Techniques," in Data and Computer Communications,10th edition (2013).

[4] C. P. J. J.Stone, M.Greenwald, Performance of checksums and crcs over real data, https://www.researchgate.net/publication/3334567_Performance_of_checksums_and_CRCs_over_real_data, [Accessed: 07-03-2023] (1998).

[5] P. H. Tanenbaum A.S, Computer Networks (1981).

[6] L. P. Bruce.S.Davie, Computer networks: A systems approach,5th edition (2012).

[7] B. A. Forouzan, Data Communications and Networking,4th editon (2007).

[8] W. Stallings, Data computer communications.

[9] M. Sudan, Algebra-Polynomial Division Algorithm, http://people.csail.mit.edu/madhu/ST12/scribe/lect06.pdf, [Accessed:07-03-2023].

[10] K. Y. Yordzhev, The Bitwise Operations in Relation to the Concept of Set, https://www.researchgate.net/publication/327989207_The_Bitwise_Operations_in_Relation_to_the_Concept_of_Set, [Accessed: 07-03-2023] (2018).

[11] A. A. M. H. Elham Yaghoubi, Leily A.Bakhtiar, All optical or/and/xor gates based on nonlinear directional coupler, https://www.researchgate.net/publication/272015389_All_optical_ORANDXOR_gates_based_on_nonlinear_directional_coupler, [Accessed: 07-03-2023] (2014).

[12] Berkley, Bit Shifting, https://inst.eecs.berkeley.edu/~cs61bl/r/cur/bits/bit-shifting.html?topic=lab28.topic&step=4&course=, [Accessed: 07-03-2023].

[13] D. V. Sarwate, Computation of Cyclic Redundancy Checks via Table Look-Up, https://dl.acm.org/doi/pdf/10.1145/63030.63037, [Accessed: 07-03-2023] (1988).

[1–7, 9–13]