Uluslararası İleri Doğa Bilimleri ve Mühendislik Araştırmaları Dergisi
Sayı 7, S. 86-90, 3, 2023
© Telif hakkı IJANSER'e aittir
Araştırma Makalesi

https://as-proceeding.com/index.php/ijanser
ISSN: 2980-0811

International Journal of Advanced Natural Sciences and Engineering Researches
Volume 7, pp. 86-90, 3, 2023
Copyright © 2023 IJANSER
Research Article

On Implementation of Reconfigurable Cyclic Redundancy Check for EVS Applications

Afaq Ahmad[1*], Fahad Mohamed Senan[2], Sayyid Samir Al-Busaidi[3] and Ahmed Ammari[4]

[1,2,3,4] *Department of Electrical and Computer Engineering, Sultan Qaboos University, PO Box 33, Zip-code 123, Muscat, Oman*

[*]*(afaq@squ.edu.om) Email of the corresponding author*

(Received: 1 March 2023, Accepted: 7 April 2023)

(2nd International Conference on Scientific and Academic Research ICSAR 2023, March 14-16, 2023)

ATIF/REFERENCE: Ahmad, A., Senan, F. M. & Al-Busaidi, S. S. (2023). On Implementation of Reconfigurable Cyclic Redundancy Check for EVS Applications. *International Journal of Advanced Natural Sciences and Engineering Researches*, 7(3), 86-90.

Abstract – The rising of Internet of Things (IoTs), Unmanned Aerial Vehicles (UAVs), 5G/6G technologies, and many other technologies, render the need for high-speed, low footprint implementation of Cyclic Redundancy Check (CRC) versions their Embedded Vision Systems (EVSs). However, the performance of a CRC architecture need to be mapped in terms of Hamming Distance (HD), area, memory, power dissipation and execution time. In addition, each of the terms depend on certain factors such as the data word size, sparse and dense nature of the data, the CRC length, the seed, the CRC characteristic polynomial and the operational frequency. In this paper, we investigate one specific factor that may potentially affect the execution time. This factor is the selection of a characteristic polynomial for a CRC implementation. The method used to investigate this aspect we developed an exhaustive code that iterates over all the chosen CRC types, finds all the primitive connection polynomials, calculate the CRC of different data word sizes against every determined characteristic polynomials and stamp the time. Then finally analyze the results and present the concluded findings. A thorough investigation was carried out on one of the performance factors i.e. the selection of CRC characteristic polynomial. Based on employed methodology and algorithms we found that the execution time is doubled whenever the data word size is doubled, the execution time increases whenever the CRC type increases, and most importantly, the execution time of any characteristic polynomial within the same CRC type will most probably have the same execution time.

Keywords – CRC, Hamming Distance, Characteristic Polynomial, Data Word Size, Execution Time

I. Introduction

A Cyclic Redundancy Check (CRC) is considered as the most common, reliable and dependable error checking protocol used to detect transmission errors. However, the CRC codes do not have a built-in error-correction capability. For the error-free transmission of data frames, CRCs operations are performed on each block of the data frames. When the system detects an error in the message, it acknowledges the sender to resend the message. In CRCs, the check bits are redundant as it increases the message devoid of additional bits of information. With implementation of CRC through the algorithmic centered approach their hardware implementation has grown popularity due to their ease of implementation and efficient analysis using well established comprehensive mathematical models [1] - [5].

A simple Shift Register (SR) circuit that performs the computations by handling the data one bit at a time (bit by bit) is used in the traditional hardware implementation, which is known as Linear Feedback Shift Register (LFSR) ([1], [3], [4], [5]). In software implementations, we can handle the data as bytes or words, which is more convenient and faster. CRC algorithm can be easily implemented using commonly used programming languages like Java, C++, C or using the Hardware Description Languages (HDL) like Verilog and VHDL or using the popular technical computing language like MATLAB ([3], [6]).

II. Construction of CRC

Cyclic Redundancy Check is a check code of n binary digits that is appended to a message of k binary digits for preserving the integrity of the message. This check code is the reminder resulted from dividing the message (the dividend) by a divisor of $n+1$ binary digits, in the $GF(2)$. If a receiver of this encoded message of $k+n$ binary digits, divides it by the same divisor, and resulted in a zero reminder, then the received message is error free, otherwise the received message is erroneous. In other words, an error is detected.

The check code using CRC process is calculated by dividing the message polynomial by the CRC characteristic polynomial after shifting left the characteristic polynomial n times where n is the number of bits in the CRC to be calculated.

Given below is the algorithm for the modulo-2 division process.

- Let the width of the CRC be n
- Assign the uppermost n+1 bits of the message as the remainder
- Starting with the Most Significant Bit (MSB) in the original message and for each bit position that follows, look at the n+1 bit remainder
    - If the MSB of the remainder is a one, the divisor will be divided into it and we should indicate a successful division in the appropriate bit position in the quotient and then compute the new remainder. So if the MSB is one
        - Set the appropriate bit in the quotient to a one, and
        - XOR the remainder with the divisor and store the result back to the remainder
    - If the MSB is not a one:
        - Set the appropriate bit in the quotient to a zero, and
    - Left-shift the remainder, shifting in the next bit of the message. The bit that's shifted out will always be a zero, so no information is lost.
- The final value that is getting stored in of the remainder is the CRC bits of the given message.

As the information contained in the quotient is not important in the calculation of CRC, no need to store the quotient. As the result of each XOR operation with the characteristic polynomial is a reminder that has zero in its MSB, We hardly lose any information when the next message frame (M) bit is shifted into the remainder. To ease to readership we present below the examples to elaborate the algorithm.

Most CRC specifications tend to drop the MSB from the characteristic polynomial (P) in binary representation. For example if$P = 01011$, which can be rewritten in hexadecimal as$0xB$. The examples shown in Figures 1 and 2 describe the binary and polynomial divisions' processes of CRC implementations respectively. In example, the data M = 10110011 = $0\ xB3 = x^7 + x^5 + x^4 + x^1 + x^0$. The value of P = 101011 $= 0x2B = \ x^5 + x^3 + x + 1$. Thereby, k = 8 and n = 5.

```
          10011000
        _______________
101011 | 1011001100000
_______
         101011
             111110
             101011
              101010
              101011
        _______________
                  1000
```

Fig. 1 Binary division process

III. CRC Implementation using LFSR

A CRC circuit is implemented using an SIPO (Serial Input Parallel Output) Galois LFSR. Various structures and controlling parameters of LFSRs are discussed in research work ([7] - [9]).

Figure 3 demonstrates the LFSR based implementation of CRC described through Figures 2 and 3. In Figure 3, it is to be noted that I = M.

$$
\begin{array}{r|l}
 & x^7+x^4+x^3 \\
\hline
x^5+x^3+x+1 & x^{12}+x^{10}+x^9+x^6+x^5 \\
 & \underline{x^{12}+x^{10}+x^8+x^7} \\
 & x^9+x^8+x^7+x^6+x^5 \\
 & \underline{x^9+x^7+x^5+x^4} \\
 & x^8+x^6+x^4 \\
 & \underline{x^8+x^6+x^4+x^3} \\
 & x^3
\end{array}
$$

Fig. 2 Polynomial division process

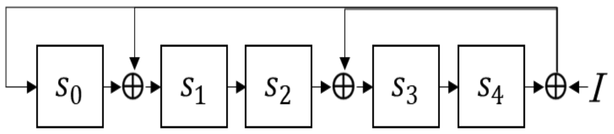

Fig. 3 Implementation of CRC using LFSR

The state equations and the table (see Table 1) of the CRC circuit shown in Figure 3 can be deduced from the figure, are:

- $s_0(t+1) = s_4(t) \oplus I(t)$
- $s_1(t+1) = s_0(t) \oplus s_4(t) \oplus I(t)$
- $s_2(t+1) = s_1(t)$
- $s_3(t+1) = s_2(t) \oplus s_4(t) \oplus I(t)$
- $s_4(t+1) = s_3(t)$

Table 1. CRC Execution Using LFSR<5,0xB> and I=0xB3

t	I	$s_4s_3s_2s_1s_0$	t	I	$s_4s_3s_2s_1s_0$
0	1	00000	6	1	11010
1	0	01011	7	1	10100
2	1	10110	8		01000
3	1	01100			
4	0	10011			
5	0	01101			

IV. Study

In order to carry out the investigation and somehow end up with a relative comparable and examinable results, all other factors that affect the execution time must be fixed. So, in our study the factors were fixed as follows:

- An arbitrary data word has been chosen as an input, and its sizes were fixed to the following: 128b, 256b, 512b and 1Kb.
- The CRC types (a.k.a. CRC lengths, or polynomial degrees) were limited to the following: {1, 2, 3, 4, 5, 7, 8, 16, 32}.
- The seed is set to zero.
- The implementation was designed as a software. It means, it gets to be executed by a microprocessor, not by a dedicated hardware accelerator. In additions, only two algorithms were implemented: {Bit-Shift, State-Space}.
- The microprocessor that was used is "Intel® Core™ i5-9400T CPU @ 1.80GHz, 1800 MHz, 6 Core(s), 6 Logical Processor(s)". Hence, the frequency is fixed.

A. *Procedure*

High Level Procedure

The proposed method employs the following high-level procedure:

Inputs:

- Data Word
- Data Word Sizes
- CRC Types
- Algorithm
- Run Times

Outputs:

- Total Execution Time
- For every primitive connection polynomial, there is:
  - CRC Type
  - Connection Polynomial (in Binary, Hexadecimal, and Decimal)
  - CRC Execution Time
  - Function Execution Time
  - CRC

```
1: Iterate Manually by Data Word Sizes {
2:   Iterate by CRC Types {
3:      find all primitive connection polynomials
```

```
 4:     Iterate by primitive connection polynomials
{
 5:        calculate CRC (data word,
 6:                      primitive connection polynomial,
 7:                      run times,
 8:                      algorithm)
 9:     }
10:   }
11: }
```

B. *Output*

At the end, the results (the outputs) are tabulated and saved as .mat files. So that they can be easily plotted or retrieved later.

C. *Finding All Primitive Connection Polynomials*

For an LFSR of length$n$, there are 2^{n-1} applicable connection polynomials, and out of those, there are $\emptyset(2^n-1)/n$ polynomials that can generate an n-sequence, where $\emptyset$ is the Euler's totient function. In other words, out of the applicable connection polynomials, there are at least $\emptyset(2^n-1)/n$ primitive connection polynomials.

V. Results

The overall execution time is recorded and depicted in Figure 4. The shows the overall execution time for the 128b, 256b and 512b data word.

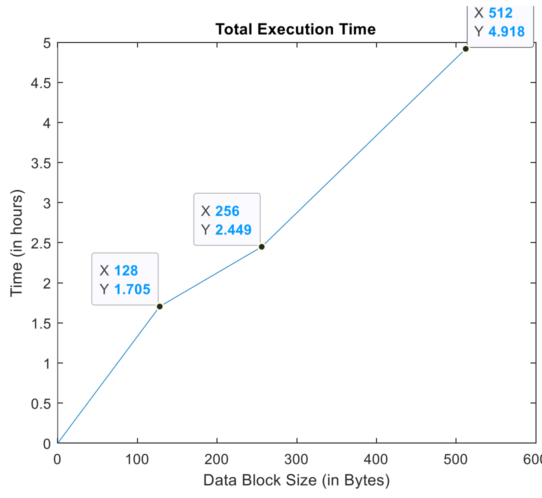

Fig. 4 Overall Execution Time for 128b, 256b and 512b Data Word

VI. DISCUSSION

This should explore the significance of the results of the work, not repeat them. The results should be drawn together. The results need to be compared with prior work and/or theory and interpreted to present a clear step forward in scientific understanding of CRC. Combined Results and Discussion sections comprising a list of results and individual interpretations in isolation are particularly discouraged. Also, in semiconductor devices testing is a bottleneck [10].

VII. CONCLUSION

Through this work, we comprehensively summarized the science and the mathematics behind CRC, in a clear and very concise way, including the state-space equations and state table with their matrices description. The report also visited some of the main conclusions of the study should be summarized in a short in this section.

Latest software and hardware CRC algorithms and implementations. Although, there seems to be an obvious progress in these algorithms and implementations, the researchers are consistently enhancing them so they can keep coping with the high requirements imposed by the rapid advancement of technology.

Furthermore, a thorough investigation was carried out on one of the performance factors. That's the selection of CRC connection polynomial based on its execution time and the potential possibility that different connection polynomials from the same CRC type will have different execution times. We can conclude, considering the employed methodology and algorithms, that the total execution time is doubled whenever the data word size is doubled, the execution time increases whenever the CRC type increases, and most importantly, the execution time of any connection polynomial within the same CRC type will most probably have the same execution time. So, it is safe to say that, based on the execution time, it does not matter which connection polynomial is selected, and such a possibility is ruled out.

There are many performance indicators that a CRC can be measured by. Perhaps, the most mature of them all, is the hamming distance (HD). However, such investigation should be carried out on each one of them, and if any of them were to influence the selection of CRC connection polynomials, then it must be approached mathematically and scientifically. Moreover, a founded mathematical tool must be developed, so it would aid a one on the selection. It is one of our aims for the future work.

Let alone, that our main aim is to design and implement a reconfigurable and optimized CRC architecture, and have those investigations guide us on achieving such an aim. We will continue to conduct experiments on All Programmable System-on-Chip (APSoC) FPGAs for the sake of developing a novelty algorithm or implementation that meets the requirements of the embedded network systems and vision systems. Further, investigations need to be carried out in context to automobiles, where FPGAs are extensively used.

ACKNOWLEDGMENT

The acknowledgements are due to authorities of Sultan Qaboos University (Sultanate of Oman) for providing generous research support grants and environments for carrying out the research works. This research work was supported by the internal grant of Sultan Qaboos University (number IG/ENG/ECED/20/01)

REFERENCES

[1] S. W. Golomb, *Shift Register Sequences*, Aegean Park Press, Leguna Hills - U.S.A., 1982.

[2] W. W. Peterson, and J. J. Weldon, *Error Correcting Codes*. MIT Press, Cambridge, London, 1972.

[3] A. Ahmad, “A Simulation experiment on a built-in self-test equipped with pseudorandom test pattern generator and multi-input shift register (MISR),” *International Journal of VLSI Design & Communication Systems(VLSICS),* vol. 1, no. 4, pp. 1-12, December 2010.
(8 Apr 2021) The SSRN website. [Online]. Available: https://ssrn.com/abstract=3811385.

[4] A. Ahmad, A Ahmad, D Al-Abri, S. S. Al-Busaidi, “Adding pseudo-random test sequence generator in the test simulator for DFT approach,” *Computer Technology and Application*, vol. 3, no. 7, pp. 463-470, July 2012.

[5] L.T. Wang and E.J. McCuskey, “Linear Feedback Shift Register Design Using Cyclic Codes,” *IEEE Transactions on Computers*, vol. C-37, no. 10, pp. 1302-1306, 1987.

[6] Development of digital logic design teaching tool using MATLAB & SIMULINK,” *IEEE Education Society Students Activities Committee (EdSocSAC)*, vol. 8, no. 1, pp. 7-12, March, 2013.

[7] A. Ahmad, N. K. Nanda, K. Garg, “The use of irreducible characteristic polynomials in an LFSR based testing of digital circuits,” in Proc. *Fourth IEEE Region 10 International Conference TENCON*, pp. 494-496. 1989.
(August 2002) Available: IEEE Xplore website [Online].

[8] N. K. Nanda, A. Ahmad and V. C. Gaindhar, “Brief Communication Shift Register Modification for Multipurpose use in Combinational Circuit Testing,” International Journal of Electronics, vol. 66, no. 6, pp. 875-878, 1989.
(26 April 2007) Available: Taylor & Francis website [Online].

[9] A. Ahmad, N. K. Nanda, “Effectiveness of Multiple Compressions of Multiple Signatures,” International Journal of Electronics, vol. 66, no. 5, pp. 775-787, 1989.
(6 April 2007) Available: Taylor & Francis website [Online].

[10] A. Ahmad, “Testing of complex integrated circuits (ics) – the bottlenecks and solutions,” *Asian Journal of Information Technology*, vol. 4, no. 9, pp. 816-822, 2005.

[11] A. Ahmad, "Automotive Semiconductor Industry - Trends, Safety and Security Challenges," in Proc. *8th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions) (ICRITO 2020)*, pp. 1373-1377, 2020.
(15 September 2020) Available: IEEE Xplore website [Online].