2015 International Conference on Computational Intelligence and Communication Networks

Implementation of Cyclic Redundancy Check in Data Communication

SANG Sheng-ju
School of Information Science and Technology
Taishan University
Taian Shandong, China
e-mail:sang1108@163.com

Abstract— **A cyclic redundancy check (CRC) is an error-detecting code commonly used in data communication and storage devices to detect accidental changes to raw data. This paper provides an overview and principle of CRC. Emphasis is placed on the implementation of the CRC algorithm by method of hardware as well as software. It is proved that the implementation method present has high practical value.**

Keywords- Error detecting coding; Data communication; CRC check; Generated polynomial

I. Introduction

In data communication and storage, burst errors are so inescapable in many communication channels, including magnetic and optical storage devices that high reliability of information transmission is required, that is, competitive low bit error rate is required. In order to ensure the accuracy of data in data communication, error control coding is commonly used. The central idea is the sender encodes his message in a redundant way by using an error-correcting code. A cyclic redundancy check (CRC) used commonly in data transmission and storage devices to detect accidental changes to raw data is an error-detecting code, which is simple to implement in binary hardware, easy to analyze mathematically, and particularly good at detecting common errors caused by noise in transmission channels. Blocks of data entering these systems get a short check value attached, based on the remainder of a polynomial division of their contents, on retrieval the calculation is repeated, and corrective action can be taken against presumed data corruption if the check values do not match.

For the purpose of error detection in communication networks, W. Wesley Peterson invented the CRC code in 1961 [1]. It is so called because the check (data verification) value is a redundancy (it adds no information to the message) and the algorithm is based on cyclic codes. Because the check value has a fixed length, the function that generates it is occasionally used as a hash function.

CRC is not only simple to implement but have the benefit of being particularly well suited for the detection of burst errors, contiguous sequences of erroneous data symbols in messages[2,3]. An n-bit CRC, applied to a data block of arbitrary length, will detect any single error burst error not longer than n bits and will detect a fraction $1-2-n$ of all longer error bursts. Typically, 16-bit CRC standard can detect all the single bit errors, double-bit errors, odd number of bits errors and burst errors which are less than or equal to 16 bits. The detection error rate of burst errors above 17 bits is 99.9984% [4, 5]. Therefore, CRC check can be applied to communication of essential data, such as the detection of running state of slave computers, online reset of running mode or parameters and so on.

II. Principle of CRC

The basic principle of CRC is as follows:

Firstly, adding r-bit check code after k-bit information code result in the length of entire code called (n, k) code is n (k + r) bits.

Secondly, for a given (n, k) code, we can prove the existence of a polynomial g (x) whose maximum power is n-k = r.

Finally, check code of k-bit information can be generated according to g (x), and g (x) is called generated polynomial of the CRC code.

CRC code of transmitted information is generally placed at the end of information. The receiving end conducts CRC calculation of the received information and compares it with that transmitted. This is useful when clocking errors might insert 0-bits in front of a message, an alteration that would otherwise leave the check value unchanged [6].

Sometimes an implementation appends n 0-bits (n being the size of the CRC) to the bit-stream to be checked before the polynomial division occurs. This has the convenience that the remainder of the original bit-stream with the check value appended is exactly zero, so the CRC can be checked simply by performing the polynomial division on the received bit-stream and comparing the remainder with zero. Thus, determining whether the received information is right.

III. Designing CRC polynomials

The most important part of implementing the CRC algorithm is the selection of generator polynomial. This polynomial resembles the divisor in a polynomial long division, which takes the message as the divided, and in which the quotient is discarded and the remainder becomes the result, with the important distinction that the polynomial coefficients are calculated according to the carry-less arithmetic of a finite field. The length of the remainder is always less than the length of the generator polynomial, which therefore determines how long the result can be.

978-1-5090-0076-0/15 $31.00 © 2015 IEEE
DOI 10.1109/CICN.2015.108

To make it easier to study the CRC code with algebraic theory, establish one-to-one relationship between the code group (binary number) with length of n and polynomial of n-1 degree, that is, take each bit of binary number as a polynomial coefficient. The most important attribute of the polynomial is its length (largest exponent) +1 of any one term in the polynomial), because of its direct influence on the length of the computed check value. The following bits correspond respectively to each exponent of the polynomial, and corresponding 1 with this power, corresponding 0 without it.

Assuming the polynomial g(x) is given as

$$g(x) = x^8 + x^2 + x + 1 \qquad (1)$$

Then, the corresponding binary code group is 1 0000 0111b

Generated polynomial of the receiving end is in agreement with the sending end, which remain unchanged throughout transmission process. The design of the polynomial depends on the maximum size in length of the block, the desired error protection features, and the type of methods for implementing the CRC as well as the desired performance. A common misconception is that the "best" CRC polynomials are derived from either an irreducible polynomial or an irreducible polynomial times the factor (1 + x), which adds to the code the ability to detect all errors affecting an odd number of bits.

The "best" CRC polynomials should satisfy the following conditions:

(1) The highest and the lowest exponent of generated polynomial should be one,

(2) With the generated polynomial divided by mode two, the remainder should not be zero, if any bit error happens,

(3) With error happening in different bits, the remainder should be different,

(4) With the reminder divided by mode two circularly, the remainder should be in loop.

The most commonly used d generated polynomials are:

$$CRC-8 = x^8 + x^2 + x + 1 \qquad (1)$$

$$CRC-12 = x^{12} + x^{11} + x^3 + x^2 + x + 1 \qquad (2)$$

$$CRC-16 = x^{16} + x^{15} + x^2 + 1 \qquad (3)$$

$$CCITT-16 = x^{16} + x^{12} + x^5 + 1 \qquad (4)$$

IV. Division by Mode 2

The main process of implementing the CRC algorithm is division by mode 2, which is similar to arithmetic division, but the results from division (subtraction) of each bit do not affect the results of other bits, that is, not borrow the higher bit. In fact, the XOR (exclusive OR) operator can implement the division above. The steps are as follows:

(1) Subtract the highest bits of dividend with divider by mode 2, without borrow-bit;

(2) Shift divider to the right bit, if the highest bit of remainder is 1, quotation is 1, and subtract the remainder by mode 2; If the highest bit of remainder is 0, quotation is 0;

(3) Until the bits of divider is less than the dividend, the remainder is the final.

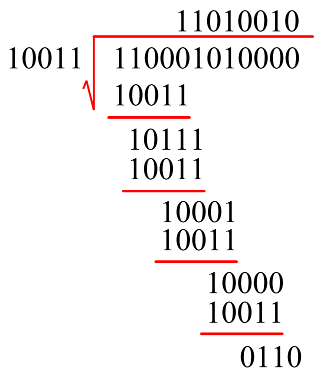

Figure 1. Procedure of division by mode 2.

Assume that the row information code is 11000101, the generated polynomial is 10011.

During to the exponent of generated polynomial above have five bits, the row information should be shifted left by four bits. Therefore, the entire code uncoded with CRC is comes into 110001010000. The procedure is shown as Fig.1.

From Fig.1, the remainder code is obtained, that is, the code of 0110 is remainder code. So, CRC code of the row information 11000101 is 1100010001010110.

V. Hardware Implementation

This CRC is typically implemented in hardware as a linear feedback shift register (LFSR) with a serial data input. Fig.2 shows the general circuit generating CRC code for given generated polynomial with R0, R1,…, Rk-1 as shift registers. The Division by mode 2 is implemented by the exclusive-OR gates (XOR gates) with ki (i=1, 2,…, k-1) switches ON or OFF. If the bit exponent of generated polynomial is 1, the corresponding switch should be ON, otherwise, it should be OFF.

According to Eq. (2), the bit exponent of generated polynomials of R1 and R2 are 1, the corresponding switches, namely k1 and k2, are closed as shown in Fig.3.

In many cases the serial LFSR implementation of the CRC is suboptimal for a given design, because of the serial data input only allows the CRC calculation of one data bit during every clock. If a design has an N-bit data-path—meaning that every clock CRC module has to calculate CRC on N bits of data—serial CRC will not work. To achieve higher throughput, the CRC's serial LFSR implementation must be converted into a parallel N-bit-wide circuit, where N is the design data-path width, so that N bits are processed in every clock. The implementation parallel N-bit-wide circuit will be discussed in the future research reports.

VI. Software Implementation

In order to improve coding efficiency of software implementation, the author proposes a calculation method based on MCS-51 assembly language and implementation program, which are briefly described in this section.

Suppose that the five bytes which are going to be transmitted as Byte0, Byte1, Byte2, Byte3 and Byte1 respectively. The construction process of (48, 40) CRC-8 is as following:

$$\begin{aligned} CRCtmp &= CRC8\left(Byte0\right) XOR\ Byte1 \\ CRCtmp &= CRC8\left(CRC\right) XOR\ Byte2 \\ CRCtmp &= CRC8\left(CRC\right) XOR\ Byte3 \\ CRCtmp &= CRC8\left(CRC\right) XOR\ Byte4 \\ CRCOK &= CRC8\left(CRC\right) XOR\ \& HFF \end{aligned} \quad (5)$$

From Eq. (2) the corresponding generated polynomial can be expressed in hexadecimal (HEX) as 107H. For the highest bit of check codes are all 1 in all case, it can be removed for programming convenience. After removing the highest bit, the check code appears as 000000111 (07H). It can be found whether the highest bit is 1 or not by comparing with 80H successively.

When the highest bit of temporary storage unit turns into 1, which means 8-bit data have been moved into, then shift temporary storage unit left once more, performs XOR operation with 8-bit check code, which is equivalent to division by mode 2. Therefore, the final remainder in CRCOK (shown in Eq.5) is the CRC code needed. (Source code is omitted, please contact the author if necessary.)

VII. Conclusion

Although CRC check cannot guarantee 100% detection of error, it does save enormous expenses required by trying to obtain perfect detection. This paper, through application of CRC error detection code in the electric power dispatching system, analyzes the principle of CRC error detection code, gives the hardware and software implementation process and has high practical value. The further study will focus on the implementation of the parallel N-bit-wide circuit.

Acknowledgment

The Research was partially supported by the Natural Science Foundation of China (no. 61303022), the Natural Science Foundation of Shandong, China (no. 2009AA01105), the science and technology development fund of Tai'an (no.20133011), and the Foundation provided by Taishan University (no.Y-01-2013010). The author gratefully acknowledge Professor SHEN Ding for discussion and suggestions. The author also thank the associate editor and the anonymous reviewers for their valuable comments and kind suggestion to improve the quality of this paper.

References

[1] Peterson, W. W. and Brown, D. T. (). "Cyclic Codes for Error Detection". Proceedings of the IRE, January, 1961, 49: 228.

[2] T.Henriksson,Liu Dake.Implementation of fast CRC calculation.Proceedings of the Design Automation Conference(ASP-DAC 2003),Asia and South Pacific,Jan.2003,563-564.

[3] Class-1 Generation-2 UHF RFID Protocol. 1.2.0. EPCglobal. 23 October 2008. p. 35.

[4] Anachriz (30 April 1999). "CRC and how to Reverse it". http://www.woodmann.com/fravia /crctut1. htm. Retrieved 21 January 2010.

[5] Cook, Greg (17 October 2011). "Catalogue of parametrised CRC algorithms". http://regregex. bbcmicro.net /crc-catalogue.htm. Retrieved 17 October 2011.

[6] Gammel, B.M. (31 October 2005). "Crypto - Codes". http://users.physik.tu-muenchen.de /gammel /matpack /html/LibDoc/Crypto/MpCRC.html. Retrieved 10 February 2011.

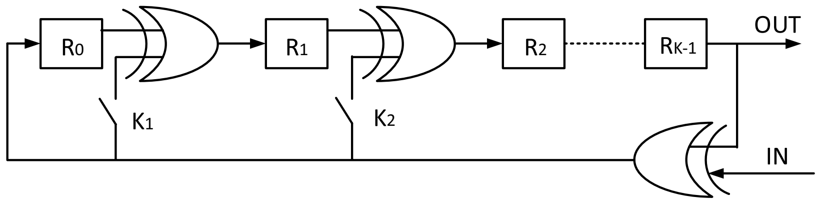

Figure 2. General Circuit of CRC code for given generated polynomial

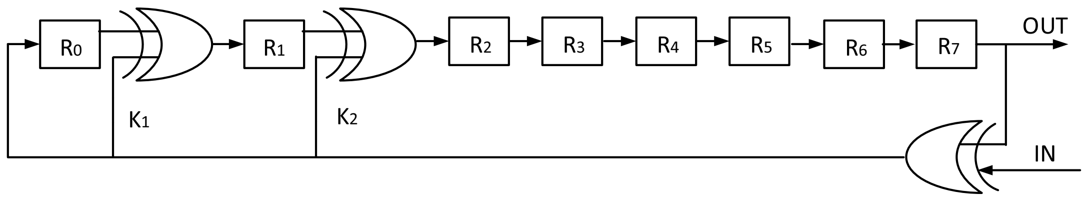

Figure 3. Circuit of CRC-8