Available online at www.elixirpublishers.com (Elixir International Journal)

Electrical Engineering

Elixir Elec. Engg. 68A (2014) 22515-22517

Elixir ISSN: 2229-712X

Implementation of CRC on DSP-TMS320VC5416

S. V. Viraktamath, Rashmi Kulkarni, Divya Suresh Moger and G. V. Attimarad
Department of ECE, SDMCET, Karnataka, India.

ARTICLE INFO

Article history:
Received: 20 September 2013;
Received in revised form:
22 February 2014;
Accepted: 14 March 2014;

Keywords
CRC,
Error Detection,
DSP,
Encoder,
Decoder.

ABSTRACT

All real systems that work with digitally represented data require error detecting codes because all real channels are noisy to some extent. The basic goal is to detect errors in data transmission over unreliable or noisy communication channels. Encoding and decoding techniques play a major role in digital communication as the received bit stream usually contains a number of errors. Cyclic Redundancy Codes (CRCs) provide a first line of defense against data corruption in many networks. The basic goal is to control errors in data transmission over unreliable or noisy communication channels. CRC code provides a simple, yet powerful, method for the detection of burst errors during digital data transmission and storage. In this paper simulation is shown and implementation of CRC-32 is done on TMS320VC5416.

Introduction

Cyclical Redundancy Check (CRC) is a kind of important linear block code, which has the advantages of easy coding and decoding as well as strong abilities of checking errors and correcting errors. Therefore, it was widely used in the field of communications and industrial measurement and control system whose industrial environment was even worse. The evolving world of telecommunications requires increasing reliability and speed in communications. Reliability in information storage and transmission is provided by coding techniques. Information is usually coded in bit streams and transmitted over the communication medium, channel. The communication media is prone to errors due to noise present in the analog portion of the channel. Therefore errors have to be detected and corrected while decoding. CRC is an error-detecting code designed to detect accidental changes to raw computer data, and is commonly used in digital networks and storage devices such as hard disk drives. Blocks of data entering these systems get a short check value attached, derived from 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. The CRC was invented by W. Wesley Peterson in 1961. CRC is an error detecting code that is widely used to detect corruption in blocks of data that have been transmitted or stored.

Principal of CRC

Are specifically designed to protect against common types of errors on communication channels, where they can provide quick and reasonable assurance of the integrity of messages delivered. However, they are not suitable for protecting against intentional alteration of data. The selection of generator polynomial is the most important part of implementing the CRC algorithm. The polynomial must be chosen to maximize the error-detecting capabilities while minimizing overall collision probabilities.

CRC is divided into the following types: Code CRC-12, code CRC-16, code CRC-CCITT, and code CRC-32. Code CRC-12 is usually used to send 6-bit string. Code CRC-16 and code CRCCCITT is used to send 8-bit string, and code CRC-16 is mainly used in America, however, code CRC-CCITT is often used in European countries. Code CRC-32 is often used in a kind of synchronous transfer which is called Point to-Point transfer [1].

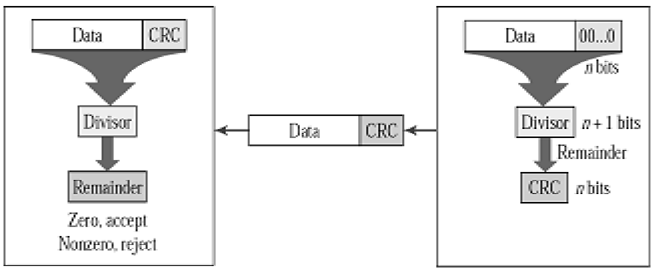


Fig. 1 Block diagram of CRC

A polynomial called generator polynomial must be chosen before the user computes the CRC of a transmitted message. The standard generator polynomials are shown in table -1. The generator polynomial must have the following features:

• The generator polynomial must have a degree greater than zero and a non-zero coefficient in the MSB and LSB positions.

• An attribute of the generator polynomial is that its length is equal to the degree +1. For example, in CRC-8, the degree is 8 and the length is 9Top = 19mm (0.75")

• The degree of the generator polynomial determines the length of the CRC code. For example, if the degree of the generator polynomial is 16, then the length of the CRC code is 16.

Table I. Standard Generator Polynomials

CODE	GENERATOR POLYNOMIAL g(X)
CRC-4	$X^4 + X^3 + 1$
CRC-8	$X^8 + X^2 + X + 1$
CRC-10	$X^{10} + X^9 + X^5 + X^4 + X + 1$
CRC-12	$X^{12} + X^{11} + X^3 + X^2 + X + 1$
CRC-16	$X^{16} + X^{12} + X^5 + 1$
CRC-32	$X^{32} + X^{26} + X^{23} + X^{22} + X^{16} + X^{12} + X^{11} + X^{10} + X^8 + X^7 + X^5 + X^4 + X^2 + X + 1$

Tele:
E-mail addresses: svvmath@gmail.com

A scheme is proposed [2] in which CRC computation of a given whole message is done parallel by breaking the latter into segments which are then processed in parallel and, finally, their contributions are accumulated. A kind of common method of coding and decoding is introduced in [1]; which is based on the CRC arithmetic and implementation on DSP is also discussed. A skin detection algorithm for TV applications, and investigates the feasibility of its real-time implementation on a DSP and an FPGA platform is presented in [3]. The 8-bit parallel CRC-32 is proposed in [4] to meet the high throughput of USB3.0. Exhaustive survey of all CRC polynomials from 3 bits to 15 bits is presented in [5]. A set of 35 new polynomials in addition to 13 previously published polynomials are also described. The method that realizes the ability of multiple bits error correction using cyclic redundancy check codes is presented in [6]. The structures of 8-bit CRC are presented in [7]. The CRC structures are implemented using Linear Feedback Shift Registers (LFSRs) along with Exclusive-OR logic gates. A hardware efficient way of implementing CRC-I6 over 16 bits of data, multiple bit error detection and single bit error correction on FPGA device is presented in [8].

Encoder and Decoder

Division operation of CRC code is mod-2 operation. The divider circuit of CRC can be made up of shift registers and mod-2 adders. The calculation of CRC can is done by software to reduce the cost.

Encoder Algorithm

1. Read the message vector
2. Take the generator polynomial order ‘k’
3. Shift the message vector ‘k’ times and store it order as‘n’; Compute(n-k).
4. Shift generator polynomial (n-k) times and store the result ‘h’.
5. XOR the generator polynomial and message vector and store the result in ‘x’
6. Determine the highest polynomial index position of ‘x’ Where 1 is occurred and take it as a ‘n’
7. If n>=’k’ go to step 4
8. Concatenate the check bits with the message bits.

Decoder Algorithm

1. Read the received data and take its order as ‘n’ and store it in ‘h’
2. Take the order of generator polynomial as ‘k’
3. Compute (n-k)
4. Shift the generator polynomial left (n-k) times and store the result in ‘e’
5. XOR the generator polynomial and received vector and store the results in ‘x’
6. Determine the highest index of ‘x’ where 1 is occurred and take it as a ‘n’
7. If n>= ‘k’ go to step 3
8. Store ‘x’ in rem
9. If rem =0 then the received data is error free else data contains error.

Processor TMS 320VC5416

The TMS320VC5416 fixed-point, digital signal processor (DSP) is based on an advanced modified Harvard architecture that has one program memory bus and three data memory buses. This processor provides an arithmetic logic unit (ALU) with a high degree of parallelism, application-specific hardware logic, on-chip memory, and additional on-chip peripherals. The basis of the operational flexibility and speed of this DSP is a highly specialized instruction set.

Separate program and data spaces allow simultaneous access to program instructions and data, providing a high degree of parallelism. Two read operations and one write operation can be performed in a single cycle. Instructions with parallel store and application-specific instructions can fully utilize this architecture. In addition, data can be transferred between data and program spaces. Such parallelism supports a powerful set of arithmetic, logic, and bit-manipulation operations that can all be performed in a single machine cycle. The device also includes the control mechanisms to manage interrupts, repeated operations, and function calls. The TMS320VC5416 DSP starter kit (DSK) is a low-cost development platform designed to speed the development of power-efficient applications based on TI's TMS320VC54x DSPs. The kit, which provides new performance-enhancing features such as USB communications and true plug-and-play functionality, gives both experienced and novice designers an easy way to get started immediately with innovative product designs. The C5416 DSK offers the ability to detect, diagnose and correct DSK communications issues, download and step through code faster and get a higher throughput with Real Time Data Exchange (RTDX™).

The full contents of the kit include: C5416 DSK Code Composer Studio™ v2.1 IDE, Quick Start Guide, Technical Reference, Customer Support Guide, USB Cable, Universal Power Supply and AC Power Cord(s). The Fig 2 shows the full contents of TMS320VC5416 kit.

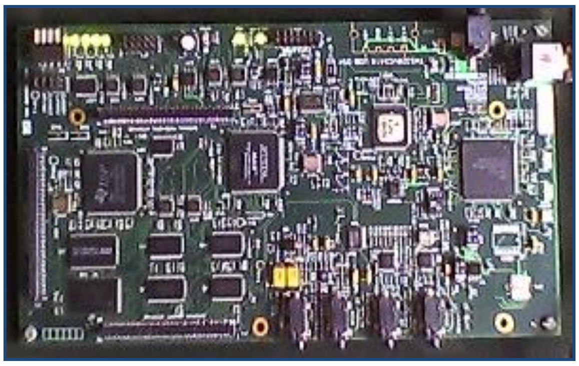
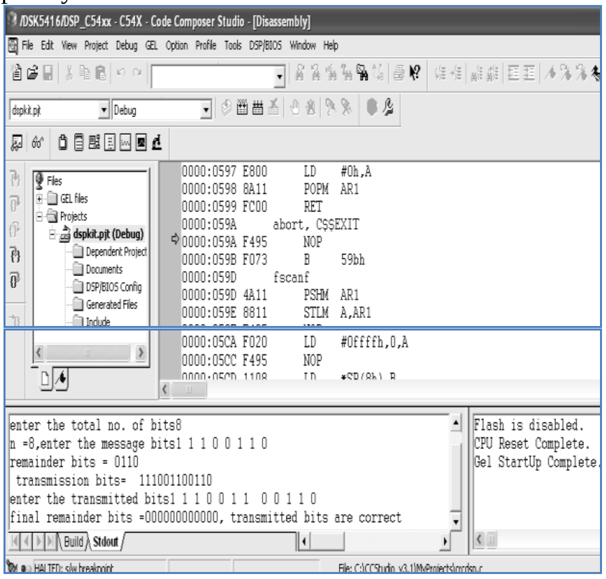


Fig. 2 The TMS320VC5416 kit

Results and Discussions

In this section, simulation results are presented. Different types of CRC are simulated from CRC-8 to CRC-32 is presented in the paper. Fig 3 shows the simulation environment completely.


/DSK5416/DSP_C54xx - C54X - Code Composer Studio - [Disassembly]
File Edit View Project Debug GEL Option Profile Tools DSP/BIOS Window Help
dspkit.pjt
Debug
Files
GEL files
Projects
dspkit.pjt (Debug)
Dependent Project
Documents
DSP/BIOS Config
Generated Files
Include
Build
Stdout
HALTED: s/w breakpoint
File: C:\CCStudio_v3.1\MyProjects\crcdsp.c

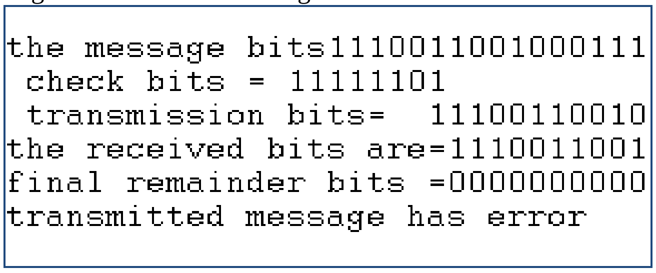
```
0000:0597 E800      LD    #0h,A
0000:0598 8A11      POPM  AR1
0000:0599 FC00      RET
0000:059A      abort, C$$EXIT
0000:059A F495      NOP
0000:059B F073      B     59bh
0000:059D      fscanf
0000:059D 4A11      PSHM  AR1
0000:059E 8811      STLM  A,AR1

0000:05CA F020      LD    #0ffffh,0,A
0000:05CC F495      NOP
```

```
enter the total no. of bits8
n =8,enter the message bits1 1 1 0 0 1 1 0
remainder bits = 0110
 transmission bits=  111001100110
enter the transmitted bits1 1 1 0 0 1 1  0 0 1 1 0
final remainder bits =000000000000, transmitted bits are correct
```

```
Flash is disabled.
CPU Reset Complete.
Gel StartUp Complete.
```

Fig. 3 Simulation Environment of DSP kit

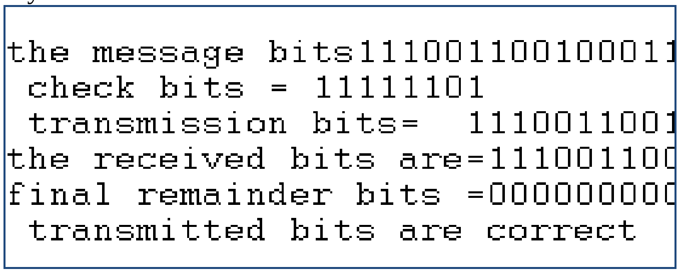
The bottom left window called ‘stdout’ shows the input and output of the program. The simulation result of CRC 8 is shown in Fig 4 when received message has an error.



```
the message bits1110011001000111
 check bits = 11111101
 transmission bits=  11100110010
the received bits are=1110011001
final remainder bits =0000000000
transmitted message has error
```

Fig. 4 CRC-8 with error in the received message

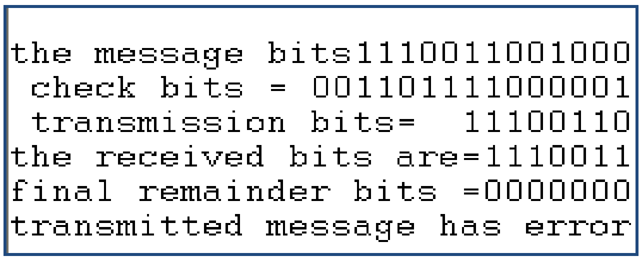
Fig 5 shows the CRC 8 without any errors. In all the simulation results the message vector, remainder bits, encoded output and remainder bits are shown. The standard generator polynomials have been chosen for the simulation.



```
the message bits111001100100011
 check bits = 11111101
 transmission bits=  1110011001
the received bits are=111001100
final remainder bits =000000000
 transmitted bits are correct
```

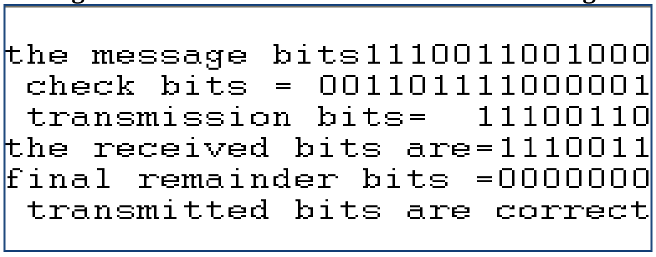
Fig. 5 CRC-8 with correct received message

The simulation for CRC 16 is also done. Fig 6 and Fig 7 shows the input and output of the CRC -32 with errors as well as without errors.



```
the message bits1110011001000
 check bits = 001101111000001
 transmission bits=  11100110
the received bits are=1110011
final remainder bits =0000000
transmitted message has error
```

Fig. 6 CRC-32 with error in the received message



```
the message bits1110011001000
 check bits = 001101111000001
 transmission bits=  11100110
the received bits are=1110011
final remainder bits =0000000
 transmitted bits are correct
```

Fig. 7 CRC-32 with correct received message

Conclusion

The procedure of calculation of remainder or redundant bits at the transmitter and checking the errors at the receiver is presented in the paper. The code has been verified for all possible lengths of message polynomial and generator polynomial. The CRC-8 to CRC-32 are successfully implemented on TMS320VC5416 fixed-point DSP. The simulation results are presented in the paper.

References

[1] Ma Youjie, Zhang Haitao, Zhou Xuesong, Qi Ming, Xu Lijin, “The Realization of the CRC Arithmetic which is based on DSP”, 2009

[2] International Forum on Computer Science-Technology and Applications. DOI 10.1109/IFCSTA.2009.125; 978-0-7695-3930-0/09 $26.00 © 2009 IEEE.

[3] Julian Satran, Dafna Sheinwald and Ilan Shimony “Brief Contributions- Out of Order Incremental CRC Computation”, IEEE Transactions on computers, VOL. 54, NO. 9, SEPTEMBER 2005

[4] Bahman Zafarifar, Tim van den Kerkhof, and Peter H. N, “Texture-adaptive Skin Detection for TV and its Real-time Implementation on DSP and FPGA”, IEEE Transactions on Consumer Electronics, Vol. 58, No. 1, February 2012.

[5] Ying Wu and Yuehong Qiu, “The 8-bit parallel CRC-32 research and Implementation in USB 3.0”, 2012 International Conference on Computer Science and Service System.

[6] Philip Koopman, Tridib Chakravarty, “Cyclic Redundancy Code (CRC) Polynomial Selection For Embedded Networks”, The International Conference on Dependable Systems and Networks, DSN-2004.

[7] Yanbin Zhang, Qi Yuan, “A Multiple Bits Error Correction Method Based on Cyclic Redundancy Check Codes”, ICSP2008 Proceedings, 978-1-4244-2179-4/08/$25.00 ©2008 IEEE.

[8] Ahmad, “On Design Of 8-Bit CRC Circuits Equipped With Primitive Characteristic Polynomials”, 2011 International Conference on Multimedia, Signal Processing and Communication Technologies.

[9] Sunil Shukla, Neil W. Bergmann, “Single bit error correction implementation in CRC-16 on FPGA”, ICFPT 2004, 0-7803-8652-3/04/$20.00 0 2004 IEEE.