Research Paper

Volume : 2 | Issue : 11 | November 2013 • ISSN No 2277 - 8179

Implementation of CRC on FPGA

Engineering

KEYWORDS : CRC, FPGA, VHDL

Prof. S. V. Viraktamath	Dept. of Electronics and Communication, SDM College of Engineering and Technology, Dhavalagiri, Dharwad-580002
Ms. Veena Joshi	Dept. of Electronics and Communication, SDM College of Engineering and Technology, Dhavalagiri, Dharwad-580002
Ms. Usha Nagesh Naik	Dept. of Electronics and Communication, SDM College of Engineering and Technology, Dhavalagiri, Dharwad-580002
Dr. Girirsh V. Attimarad	HOD Dept. of E&CE, Dayanand sagar college of Engineering, Bangalore, Karnataka.

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 defence 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 FPGA.

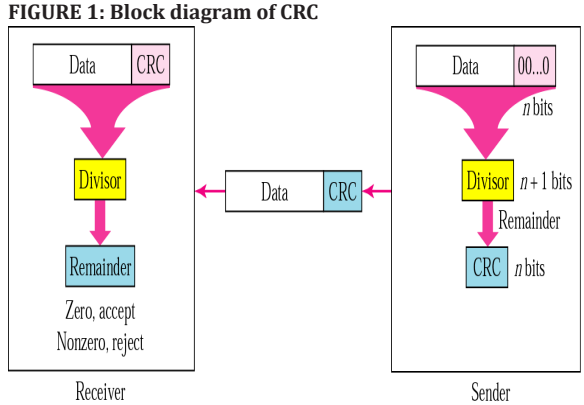
I. Introduction

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.*

II. Principle of crc

CRCs 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.

FIGURE 1: Block diagram of CRC



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].

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 9.

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 1
Standard generator polynomials

CODE	GENERATOR POLYNOMIAL g(X)
CRC-4	++1
CRC-8	++X+1
CRC-10	++++X+1
CRC-12	++++X+1
CRC-16	+++1
CRC-32	+++++ +++++ +++X+1

A scheme is proposed [2] in which CRC computation of a given whole message is done parallel by breaking the message 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 code 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].

III. 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 be done by the software to reduce the cost.

A. 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 'n'.
7. If n >=k go to step 4.
8. Concatenate the check bits with the message bits.

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

IV. simulator model

The SpartanTM-II 2.5V Field Programmable Gate Array family gives users' high performance, abundant logic resources and a rich feature set, all at an exceptionally low price. The six member family offers densities ranging from 15,000 to 200,000 system gates. System performance is supported up to 200MHz. Spartan-II devices deliver more gates, I/Os and features per dollar than other FPGAs by combining advanced process technology with streamlined vertex based architecture.

Features include block RAM (to 56K bits), distributed RAM (to 75,264 bits), 16 selectable I/O standards, and four DLLs. Fast, predictable interconnect means that successive design iterations continue to meet timing requirements. The Spartan-II family is superior alternative to mask-programmed ASICs [9]. The FPGA avoids initial cost, lengthy development cycles and inherent risk of ASICs. Also, FPGA programmability design upgrades in the field with no hardware replacement necessary (impossible with ASICs).

A field-programmable gate array (FPGA) is a semiconductor device that can be configured by the customer or designer after manufacturing hence the name "field-programmable". FPGAs are programmed using a logic circuit diagram or a source code in a hardware description language (HDL) to specify how the chip will work. They can be used to implement any logical functions. FPGAs contain programmable components called "logic blocks", and hierarchy of reconfigurable interconnects that allow the blocks to be "wired together". Fully fabricated FPGA chips containing thousands of logic gates or even more, with programmable interconnects, are available to users for their custom hardware programming to realise desired functionality. A typical FPGA chip consists of I/O buffers, an array of configurable logic blocks (CLBs), and programmable interconnect structures. The programming of interconnect structure is implemented by programming the gates of MOS pass transistors. The typical design flow of an FPGA chip starts with the behavioural description of its functionality, using a hardware description language such as VHDL.

V. SIMULATION RESULTS

In this section, simulation results are presented for message of length 100 bits. Message bits=1110110011100010100011011011100001111000001101011110010110001100010000111100111010101011001011010. Figure 2 shows the encoded bits i.e., message bits along with check bit (1000111011000010100010111001011). Figure 3 shows zero remainder for received bits without any error. Figure 4 shows non-zero remainder for received bits with error.

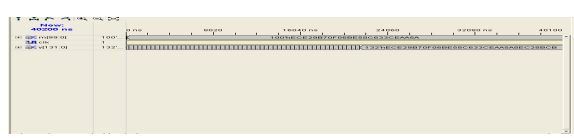


FIGURE 2: Encoded bits.

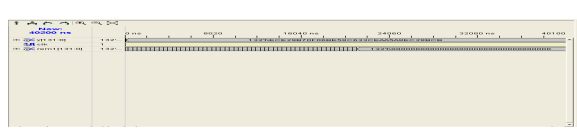


FIGURE 3: Message bits without error.

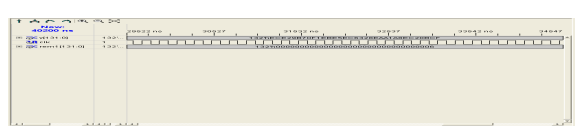


FIGURE 4: Message bits with error.

VI. SYNTHESIS REPORT

This section consists of synthesis report of encoder and decoder of CRC-32.

Volume : 2 | Issue : 11 | November 2013 • ISSN No 2277 - 8179

```
#      IBUF                       : 100
#      OBUF                       : 132
=========================================================================

Device utilization summary:
---------------------------

Selected Device : 3s400pq208-4

 Number of Slices:                     1101  out of   3584    30%
 Number of Slice Flip Flops:            373  out of   7168     5%
 Number of 4 input LUTs:               2101  out of   7168    29%
 Number of bonded IOBs:                 233  out of    141   165% (*)
 Number of GCLKs:                         1  out of      8    12%

WARNING:Xst:1336 -  (*) More than 100% of Device resources are used


=========================================================================
TIMING REPORT

NOTE: THESE TIMING NUMBERS ARE ONLY A SYNTHESIS ESTIMATE.
      FOR ACCURATE TIMING INFORMATION PLEASE REFER TO THE TRACE REPORT
      GENERATED AFTER PLACE-and-ROUTE.

Clock Information:
------------------
-----------------------------------+------------------------+-------+
Clock Signal                       | Clock buffer(FF name)  | Load  |
-----------------------------------+------------------------+-------+
clk                                | BUFGP                  | 373   |
-----------------------------------+------------------------+-------+
```

FIGURE 5: Synthesis report of Encoder

```
#      FD                         : 2
#      FDE                        : 105
#      FDS                        : 267
# Clock Buffers                   : 1
#      BUFGP                      : 1
# IO Buffers                      : 264
#      IBUF                       : 132
#      OBUF                       : 132
=========================================================================

Device utilization summary:
---------------------------

Selected Device : 3s400pq208-4

 Number of Slices:                     1125  out of   3584    31%
 Number of Slice Flip Flops:            374  out of   7168     5%
 Number of 4 input LUTs:               2124  out of   7168    29%
 Number of bonded IOBs:                 265  out of    141   187% (*)
 Number of GCLKs:                         1  out of      8    12%

WARNING:Xst:1336 -  (*) More than 100% of Device resources are used


=========================================================================
TIMING REPORT

NOTE: THESE TIMING NUMBERS ARE ONLY A SYNTHESIS ESTIMATE.
      FOR ACCURATE TIMING INFORMATION PLEASE REFER TO THE TRACE REPORT
      GENERATED AFTER PLACE-and-ROUTE.

Clock Information:
```

FIGURE 6: Synthesis Report of Decoder.

VII. Conclusions

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-4 to CRC-32 is successfully implemented on FPGA. The simulation results are presented in the paper.

REFERENCE

[1] Ma Youjie, Zhang Haitao, Zhou Xuesong, Qi Ming, Xu Lijin, "The Realization of the CRC Arithmetic which is based on DSP", 2009 International Forum on Computer Science-Technology and Applications. DOI 10.1109/IFCSTA.2009.125; 978-0-7695-3930-0/09 $26.00 © 2009 IEEE. | [2] 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 | [3] 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. | [4] 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. | [5] Philip Koopman, Tridib Chakravarty, "Cyclic Redundancy Code (CRC) Polynomial Selection For Embedded Networks", The International Conference on Dependable Systems and Networks, DSN-2004. | [6] 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. | [7] A. Ahmad, "On Design Of 8-Bit CRC Circuits Equipped With Primitive Characteristic Polynomials", 2011 International Conference on Multimedia, Signal Processing and Communication Technologies. | [8] 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. | [9] Metastable Recovery in Virtex-II FPGAs, Xilinx Application Note XAPP094. http://www.xilinx.com/support/documentation/application_notes/xapp094.pdf. |