INTERNATIONAL JOURNAL OF PURE AND APPLIED RESEARCH IN ENGINEERING AND TECHNOLOGY

A PATH FOR HORIZING YOUR INNOVATIVE WORK

A REVIEW PAPER ON PARALLEL CRC GENERATION FOR HIGH SPEED APPLICATION

PAYAL HAJARE[1], PROF. K. MANKAR[2]

1. M. Tech VLSI, GHRIETW, RTMN University, Nagpur, India
2. GHRIETW, RTMN University, Nagpur, India

Accepted Date: 05/03/2015; Published Date: 01/05/2015

Abstract: CRC is playing a main role in the networking environment to detect the errors. With challenging the speed of transmitting data to synchronize with speed, it is necessary to increase speed of CRC generation. Most engineers are familiar with the cyclic redundancy check (CRC). Many know that it is used in communication protocols to detect bit errors and that it is essentially a remainder of the modulo-2long division operation. As a vital method for dealing with data errors usually the hardware implementation of CRC computations is based on the linear feedback shift registers (LFSRs), which handle the data in a serial way. The serial calculation of the CRC codes cannot achieve a high throughput. In constant parallel CRC calculation can significantly increase the throughput of CRC computations. Types of CRCs are used in applications like CRC-16BISYNC protocols, CRC32 in Ethernet for error detection, CRC8 in ATM, CRC-CCITT in X-25 set of rule, disk storage, XMODEM and SDLC. This paper presents 64 bits parallel CRC architecture. The whole design is functionally verified using Xilinx ISE Simulator.

Keywords: Cyclic redundancy check, Parallel CRC calculation, Shift register, Error control coding.

Corresponding Author: MS. PAYAL HAJARE

Access Online On:

www.ijpret.com

How to Cite This Article:

Payal Hajare, IJPRET, 2015; Volume 3 (9): 892-897

PAPER-QR CODE

INTRODUCTION

In this modern era research is done for data acquisition in various fields such as industrial and Cyclic Redundancy Check (CRC) technology can be used to test the integrity of the process in data transmission or data compression. It has been widely used in the communication network and the data storage technology, etc. The receiver calculates the data of received from transmitter with the same CRC algorithm. If the value calculated equals to checking code in frame, it shows that the process of transmission or compression is successful. Classic CRC algorithm is realized through the structure of linear feedback registers. Because it is a serial coding method, it is intuitive and easy to realize. But the speed of calculate is very low, according to the classic CRC algorithm. It is not suitable to apply in high speed data transmission. Parallel CRC algorithm can meet the demand of the high-speed data transmission. The general method of the parallel CRC is realized by software or hardware. Because speed of the software calculation is slow, the algorithm has been used in much communication equipment by hardware for high speed According to the thoughts of traditional serial CRC algorithm, parallel CRC PIPELINE ALGORITHM based on MATRIX is referred for high-speed coding. The algorithm can reduce the delay time of circuit and improve the throughput rate of data.

CRC can generate in two ways: A. Serial CRC generation. B. Parallel CRC generation. Usually, the hardware implementation of CRC computations is based on the linear feedback shift registers (LFSRs), which handle the data in a serial way. Though, the serial calculation of the CRC codes cannot achieve a high throughput. So, to over that problem we are moving to Parallel CRC generation. In this we are discussing about 64-bit parallel CRC generation.32-bit parallel CRC generates gigabits per second, but it will not suited for high speed applications like Ethernet.

A. Serial CRC generation

In this CRC checking is done serially. The data input will be single (binary) and every clock pulse the data input will be one. There will some delay present between the consecutive data inputs and the output will be zero if the data will be encoded with same CRC value otherwise it shows non-zero value. By this form we will conclude where the data is accurate or corrupted. The data is serially processed and the polynomial is XORed with the data input, that will given to the D flip-flop (output same as the input) final CRC is generated as serially.

Shift Register:

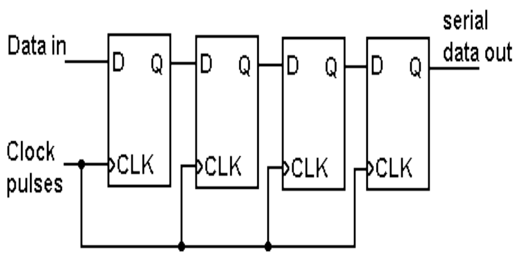

Fig 1: 4 Bit Shift Register

The diagram shows four flip-flops connected to form a input output shift register. At arrival of a clock pulse, data at the D input of each flip-flop is transferred to its Q output. First, the contents of the register can be set to zero by means of the CLEAR line. If 1 is applied to the input of the first flip-flop, then the arrival of the first clock pulse, this 1 is transferred to the output of flip-flop 1 (input of flip-flop 2). After 4 clock pulses this 1 will be at the output of flip-flop 4. In this way, a four bit number can be stored in the register. After 4 more clock pulses, this data will be shifted out of the register.

B. Parallel CRC generation

Every modern communication protocol uses one or more error-detection algorithms. CRC is most popular. The protocol specification usually defines CRC in hex or polynomial notation.

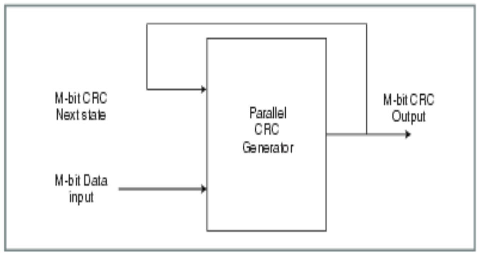

Fig 2: This is a parallel CRC block. The next state CRC output is a function of the current state CRC and the data.

CRCs are specifically designed to protect against common types of errors on communication channels.CRC is burst error detecting code designed to detect accidental changes to digital data in computer networks. CRC is characterized by specification called G(x), Generator Polynomial. Goal is to maximize the probability of detecting an error. In a scheme, the transmitter sends the original data, and then attaches a fixed number of check bits which are derived from the data bits by some deterministic algorithm.

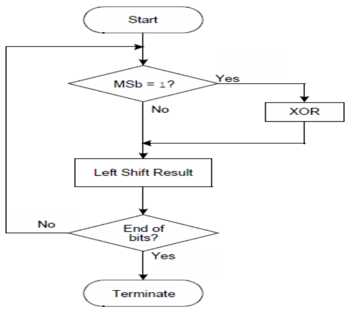

Fig 3: SOFTWARE FLOWCHART

PARALLEL CRC ALGORITHM AND RELATED WORKS

It pre-calculates code to be loaded in the shift register after a number of shifts and XOR operations in the serial CRC generator, and then generates the code at a system clock cycle. The CRC code is calculated with the data Di and the shift operation. The value in each flip-flop is updated by shift or/and XOR operation in every clock cycle. After these n times shift/XOR operations, the CRC code for n bits input data is generated. To minimize the area and critical path delays of the parallel CRC several researches have been performed. In pre-decoding logic and a binary tree optimization technique are used to reduce the routing overhead and propagation delay.

II. Literature Review

Hitesh H. Mathukiya, Naresh M. Patel[1], Proposed the parallel CRC generation deal with 64bit parallel processing based on built in F matrix with order of generator polynomial is 32. This gives CRC with half number of cycles. It drastically reduces computation time to 50% and same time increases the throughput. The no. of LUT get increased, so area also get increase.

Yan Sun; Min Sik Kim[2],Proposed a fast cyclic redundancy check (CRC) algorithm that performs CRC computation for an arbitrary length of message. This paper proposes a table-based hardware architecture for calculating CRCby taking advantage of CRC’s properties and pipelining the feedback loop. It achieves considerably higher throughput than existing serial or byte-wise lookup CRC algorithms.With delay increase in the critical path.

Weidong Lu and Stephan Wong[4],Proposed presented a novel method to update the CRC code when packets are passing through interconnecting devices. And focus on the CRC calculation that is performed during the routing of the Ethernet packets be encapsulating the packets into Ethernet frames, adding a frame header and adding a frame trailer. It calculates the intermediate results of the changed fields based on the parallel CRC calculation and performs a single step update afterwards. And the number of cycles is dramatically reduced.The fast CRC update only calculates the changed portion of a frame.

Campobello, G.; Patane, G.; Russo, M.; [3],Proposed a theoretical result in the context of realizing high speed hardware for parallel CRC checksums. The number of bits processed in parallel can be different from the degree of the polynomial generator. Presented Pre-calculated F-matrix based 32 bit parallel processing. Which is faster and more compact and is independent of the technology used in its realization. But it doesn't work if polynomial change.

III. Proposed Technique

CRCs are specifically designed to protect against common types of errors on communication channels. It is a burst error detecting code designed to detect accidental changes to digital data in networks. It is characterized by specification called G(x), Generator Polynomial. Goal is to maximize the probability of detecting an error. In this, the transmitter sends the original data, and attaches a fixed number of bits which are derived from the data bits by some algorithm. If error detection is required receiver can simply apply the same algorithm to the received data bits and compare its output with the received check bits. If the values do not match, an error occurred at same point during transmission. So by using 64-bit code, parallel crc code is generated for high speed application thus reducing no of clock cycles for better and high data transfer.

IV. CONCLUSION

Generally when high-speed data transmission is required serial implementation is not preferred because of slow throughput. So parallel implementation is preferred which takes minimum amount of time. CRC-32 requires 17 clock cycles to transmit 64bytes of data. But CRC-64 required 9 clock cycles to transmit the same data. So, it drastically reduces computation time to 50% and same time increases the throughput. Hence it is easy method for fast CRC generation.

V. REFERENCE

1. Hitesh H. Mathukiya, "A Novel Approach for parallel CRC generation for high speed application, "International conference on communication system and network technologies, no. 978-0-7695-4692-6/12-2012 IEEE.

2. Yan Sun; Min Sik Kim. "A Pipelined CRC Calculation Using Lookup Tables," Consumer Communications and Networking Conference (CCNC), 2010 7th IEEE, vol., no., pp.1-2, 9-12 Jan.2010

3. G. Campobello, G. Patane, and M. Russo, "Parallel CRC realization," IEEE Transactions on Computers, Oct. 2003.

4. Weidong Lu and Stephan Wong, "A Fast CRC Update Implementation", IEEE Workshop on High Performance Switching and Routing, Oct. 2003.