A New Design of High Speed Parallel CRC Generator

Pyoungwoo Min, Hyunbean Yi, Sungju Park, *Kiman Jeon

minpw@mslab.hanyang.ac.kr bean@mslab.hanyang.ac.kr
parksj@mslab.hanyang.ac.kr kmjeon@keti.re.kr

Dept. of Computer Science & Engineering, Hanyang University
*Ubiquitous Computing Research Center, Korea Electronics Technology Institute

Abstract

This paper presents an optimization algorithm and technique for designing parallel Cyclic Redundancy Check (CRC) circuit, which is most widely adopted for error detection. A new heuristic algorithm is developed to find as many shared terms as possible, thus eventually to minimize the number and level of the exclusive-or logic blocks in parallel CRC circuits. 16-bit and 32-bit CRC generators are designed with different types of Programmable Logic Devices, and it has been found that our new algorithm and architecture significantly reduce the delay

I. INTRODUCTION

Cyclic Redundancy Check (CRC) is an error-detection scheme used in digital communication systems. Some systems often use "Parity" which can detect single bit error or "Hamming Code" which uses 3~4 bits a byte for error-detection [1] For packet transmission on high-speed medium and long distance links, however, not only most of the errors occurred tend to be multi-bit bursts, but overhead bits for error-detection are a huge burden Accordingly, the CRC, which can detect burst errors only with the same number of bits as degree of polynomial, no matter how big packet sizes are, is the best choice [1,2]. As requirements regarding the speed of network and communication systems increase, most of the recent I/O interconnect technologies go for over 10 Gbps rather than 1 Gbps. To keep up with these trends, design techniques, which make the CRC circuits run at high-speed, are needed.

CRC circuits are basically designed by using an LFSR (Linear Feedback Shift Register). They have to use a bit-clock corresponding to the speed of in/out serial data stream or use buffers to compensate code generation processing delays. A number of studies have been performed to speed up the parallel CRC circuits [3,4,5,6,7,8]. Two typical architectures are widely adopted for parallel CRC circuits, one is based on look-up table and the other is XOR (exclusive-or) cells. The main deficiency of the table look-up architecture is doubling the table size upon single increase of the polynomial degree, and accordingly the table access time becomes longer. Instead of table look-up approach, our paper is focused on developing a new algorithm to maximize the sharing terms and thus to minimize the logic level of the critical path in XOR cell architecture. This paper is organized as follows. Parallel CRC implementations are reviewed in Section 2, and our new heuristic algorithm is described in Section 3. The implementation detail of our parallel CRC algorithm is described in Section 4. Experimental results show the superiority of our approach in Section 5 followed by conclusion and future work in section 6.

II. Parallel CRC Algorithm and Related works

CRC circuit is widely used to detect errors in a large message. A division calculation by the divisor uniquely determined by different communication system is used to generate CRC values by taking the input data in order. These values generated are sent out along with the message. At the receiving end the CRC values are recalculated and compared to the expected values to check any error occurred during transmission. Since the CRC code is the remainder of the division, it must be equal to or smaller than the specific divisor regardless of the size of message [2].

Many network and communication systems usually make use of several standardized polynomials as divisors, and perform the modulo-2 binary division instead of the common division calculation. The modulo-2 binary division can be calculated by XOR (exclusive-or) operation and serial CRC generators are designed by using LFSR and XOR gates. Fig. 1 shows a primitive polynomial and its hardware implementation.

Since such a serial CRC circuit has been a stumbling block in improving the speed of the whole system. In order to remove this barrier, studies have bean made on generating CRC codes for parallel input data at a system clock and further raising the clock rate. A basic method of designing the parallel CRC generator is explained in detail in [1]. 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.

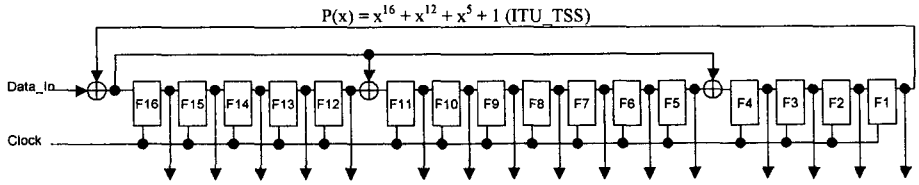

Fig. 1. Serial CRC Generator