Microelectronics Journal 54 (2016) 126–137

Contents lists available at ScienceDirect

Microelectronics Journal

journal homepage: www.elsevier.com/locate/mejo

Ternary cyclic redundancy check by a new hardware-friendly ternary operator

Mahya Sam Daliri [a], Reza Faghih Mirzaee [b], Keivan Navi [a,*], Nader Bagherzadeh [c]

[a] Faculty of Electrical and Computer Engineering, Shahid Beheshti University, G.C., Tehran, Iran
[b] Department of Computer Engineering, Shahr-e-Qods Branch, Islamic Azad University, Tehran, Iran
[c] Department of Electrical Engineering and Computer Science, University of California, Irvine, USA

ARTICLE INFO

Article history:
Received 7 October 2015
Received in revised form
6 March 2016
Accepted 29 April 2016

Keywords:
CRC
CNTFET
Error detection
MVL
Ternary logic
Ternary operator

ABSTRACT

This paper presents a new ternary operator for cyclic redundancy check in ternary logic with high hardware efficiency. It shows the essential properties of a ternary operator for calculating CRC. Gate-level implementation of CRC-16 is exemplified and comprehensively explained in this paper. Transistor-level realizations of the new and the previous operators are also given in full detail. This paper demonstrates that the employment of the proposed operator reduces hardware components such as the elimination of one entire operator in both transmitter and receiver sides in comparison with ternary *SUM*. In addition, simulation results by HSPICE and 32 nm CNTFET technology shows about 20.6% energy reduction for one of the implementation methods of the proposed operator with respect to the previously presented one (*CCW CYCLE*). Furthermore, five fewer transistors are needed to design the proposed operator in contrast with both operators *SUM* and *CCW*.

1. Introduction

Cyclic Redundancy Check (CRC) is a well-known error detection code for protecting transmitted data over a channel. It is also widely used in storage devices for discovering any accidental changes to raw data [1,2]. It is based on the use of redundancy checks and polynomial manipulations using modulo arithmetic. The CRC code is the remainder of dividing one polynomial ($x^n \times M(x)$) by another ($G(x)$), where n is the degree of the common generator polynomial $G(x)$ between transmitter and receiver.

The generator polynomial is chosen specifically to provide high error detection. There are several common generator polynomials. CRC generation in ternary and binary logics can be done with exactly the same polynomials, whose coefficients are either zero or one [3]. The typical CRC-16 with the representation of $G(x)=x^{16}+x^{15}+x^2+x^0$ is exemplified in this paper. It is widely used in modems and network protocols [4]. Fig. 1 shows how CRC computations are done by means of an LFSR.

In the transmitter side (Fig. 1a), each flip-flop is set to zero at the initial stage. The input data are sent to the receiver and shifted at the same time into the LFSR bit-by-bit per clock cycle. This continues until the last bit of the input data stream is processed. Meanwhile, the first and the second switches (SW1 and SW2) select the incoming data. Afterwards, the resulting values of the flip-flops (the CRC code) are also sent to the receiver (SW2 selects flip-flops) in the reverse direction (from A_{15} to A_0). While transmitting the CRC code serially, SW1 sends zero to the flip-flops consecutively so that they are reinitialized for the next round. There is not any extra time required for CRC calculation since it overlaps data transmission.

In the receiver side (Fig. 1b), the same procedure is simply repeated for the entire received codeword (data+CRC). The final value of each flip-flop has to be zero in case there are not any corrupted data [5]. The switches turn into the conducting state concurrently, and an *OR* gate plainly checks the error. The receiver asks for retransmission if error is detected.

Linear Feedback Shift Register (LFSR), comprising flip-flops and Exclusive-OR (XOR) gates, is a generic inexpensive approach for the hardware implementation of CRC [6]. It performs serial processing of one bit per clock cycle. Several parallelism techniques have been presented in the literature to expedite the entire process [7–9]. Regardless of what technique is used, Exclusive-OR is an essential logic gate in CRC computations, and hence its performance has a significant impact on the whole system.

The exact definition of the binary XOR is that it becomes true whenever an odd number of inputs are true. However, Exclusive-OR does not exist in Multiple-Valued Logic (MVL) with the same properties of the binary XOR [3]. Another common explanation is that it computes modulo-two sum of input variables (Eq. (1)) [10].

* Corresponding author.
E-mail addresses: m_sam@sbu.ac.ir (M. Sam Daliri), r.f.mirzaee@qodsiau.ac.ir (R. Faghih Mirzaee), navi@sbu.ac.ir (K. Navi), nader@uci.edu (N. Bagherzadeh).

http://dx.doi.org/10.1016/j.mejo.2016.04.018
0026-2692/© 2016 Published by Elsevier Ltd.

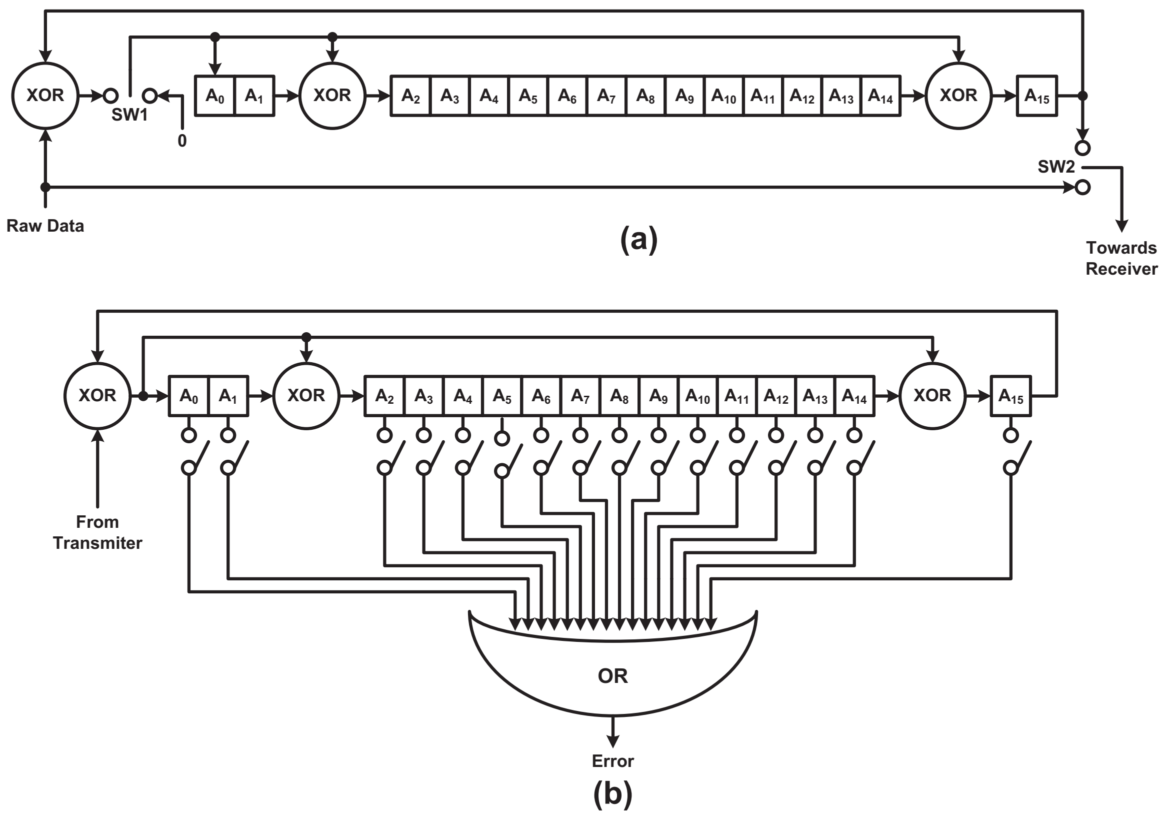
XOR
SW1
0
A_0 A_1
XOR
A_2 A_3 A_4 A_5 A_6 A_7 A_8 A_9 A_{10} A_{11} A_{12} A_{13} A_{14}
XOR
A_{15}
SW2
Raw Data
(a)
Towards Receiver
From Transmiter
OR
Error
(b)

Fig. 1. CRC circuits (the overall structure), (a) In the transmitter side, (b) In the receiver side.

The second description is usually considered in ternary logic as *SUM generator* (or *SUM*). Several ternary adders based on this interpretation have been recently proposed [11–14]. In spite of its practicality in designing arithmetic components, it is not functional for generating CRC codes. Another operator (Eq. (2)), called Counter-Clockwise Cycle (*CCW CYCLE*), has particularly been presented in [3] to perform MVL CRC computations. The efficiency degree of both operators will be fully investigated in this paper.

$$SUM(a,\ b) = SUM(b,\ a) = (a + b)\ mod\ Radix \tag{1}$$

$$CCW(a,\ b) = (a - b)\ mod\ Radix \tag{2}$$

In today's IC design industry, interconnections have become a serious challenge because they consume a lot of power [15,16], occupy a large portion of the chip area, and lead to restrictions in placement and routing of logic elements [17]. They impose parasitic effects by increasing the number of connections inside a chip. It has been reported that wire capacitance can contribute up to 70% of the total chip capacitance [16,18].

MVL can be considered as a promising practical solution to these problems. It reduces the amount of interconnections inside and outside a chip by transmitting more information over a single line than binary logic, where there are only two values ($\{0,\ 1\}_2$). MVL circuits will play an important role for the next generation electronic systems [19]. The current dominance of binary logic is because of widespread circuits and design techniques which have dominated the design process over the past decades. Exactly the same has to happen for MVL to be as successful as binary logic. Since there has been an encouraging trend towards multiple-valued systems, the design and implementation of high-performance and practical MVL logic gates and computational units are highly in demand. Amongst many MVL systems, ternary is the most efficient, economical, and advantageous logic than binary in the design of digital systems since it has the least complexity of system hardware [17].

Error detection and correction codes are very important subjects for all communication systems, including those that use MVL. Following the attempt of improving the practicalities of ternary logic, this paper presents a new ternary operator particularly and efficiently applicable in CRC generation. The proposed operator benefits from significant transistor-level optimization without any gate-level overhead with respect to the previous ones.

The rest of the paper is organized as follows: The proposed ternary operator for CRC computations is presented in Section 2. Gate-level and transistor-level implementations are discussed in this section. Section 3 contains simulation results. Finally, Section 4 concludes the paper.

2. The proposed ternary operator for CRC computations

In order to perform CRC computations in ternary logic, a bitwise operator is required. It manipulates ternary digits (Trits). It must not comply with the *idempotency* property (Eq. (3)). The binary XOR does not comply with it either, due to the fact that it returns ‘0’ when computing $a \oplus a$. Eq. (3) expresses that an idempotent operator removes the repetitive term and does not take it into account.

$$a\ \oplus\ a = a \quad (\textit{The idempotency property}) \tag{3}$$

For each logic value of the set $\{0,\ 1,\ 2\}_3$, the operator $\oplus$ must return ‘0’ if it takes the same elements (Eq. (4)). This is an