Microprocessors and Microsystems 38 (2014) 1072–1081

Contents lists available at ScienceDirect

Microprocessors and Microsystems

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

A novel self-checking carry lookahead adder with multiple error detection/correction

Mojtaba Valinataj *

School of Electrical and Computer Engineering, Babol University of Technology, Shariati Street, 47148 Babol, P.B. 484, Iran

ARTICLE INFO

Article history:
Available online 30 October 2014

Keywords:
Carry lookahead adder
Analytical assessment
Fault tolerance
Soft errors
Parity prediction

ABSTRACT

Evolving processing units in complex computing systems are dealing with smaller gates and devices which are seriously influenced by external effects such as electromagnetic noises and single event transient (SET) effects. Due to this increasing sensitivity, several transient faults might appear at the same time in a processing unit which can lead to multiple simultaneous errors. In this paper, a novel carry lookahead adder design is presented with multiple error detection/correction capability. The proposed self-checking architecture utilizes a modified parity prediction scheme combined with a partial triple modular redundancy distributed in all of the bit slices. This scheme fits carry lookahead adder and its arithmetic logic unit (ALU) counterpart in which the carry logic occupies a large part of the circuit. Furthermore, two alternative methods are proposed to reduce the overall hardware cost in the main proposed adder. The efficiency of the proposed architecture in multiple error detection/correction is analytically shown by probabilistic assessments and verified by simulations. However, the hardware implementation reveals that this architecture incurs much lower hardware overheads relative to traditional architectures while it provides higher error detection/correction efficiency.

1. Introduction

Nowadays, the complexity of evolving integrated circuits is increasing while the reliability of processing components against environmental effects is decreasing due to employing smaller gates and transistors. Technology scaling results in smaller devices in integrated circuits and one of its impacts is the more sensitivity to transient faults or soft errors caused by the electromagnetic noises, cosmic rays, crosstalk, and the power supply noise. Thus, the probability of multiple concurrent soft errors occurrence is increased even in the presence of a single event.

Arithmetic operations are the essential part of a processing component. Until now, different self-checking or fault-tolerant techniques have been used in the design of arithmetic operators and ALUs. The first self-checking adders have been designed using arithmetic residue codes to detect single errors [1,2]. A recent work that describes error detection in ALUs using multi-residue codes is presented in [3]. However, arithmetic residue codes have some important drawbacks such as requiring complex checker circuits and incompatibility with self-checking memory systems [4]. A general method for concurrent error detection (CED) is duplication with comparison or double modular redundancy (DMR). A method incorporating a modified DMR to obtain a self-checking ALU is introduced in [5]. The DMR method can detect single errors but it requires more than 100% hardware overhead. Concurrent error detection is also used for other arithmetic operations such as [6,7]. Among self-checking adder/ALU designs, the parity prediction-based schemes require a lower hardware overhead. The first Parity prediction-based adders/ALUs have been proposed in [8,9], and the next designs have been presented in [4,10,11]. These designs are compatible with parity checked datapaths and parity encoded self-checking memory modules.

The introduced self-checking methods target to achieve single error detection. However, the parity prediction scheme can detect the odd number of errors. To obtain a fault-tolerant adder, a conventional method namely triple modular redundancy (TMR) can be used to mask a single error and produce the correct result. This way, an adder with concurrent error correction is attained but with a high amount of hardware cost due to the triplication of the primary adder and usage of a majority voter. In [12] an error correction approach is proposed for arithmetic operations. In this approach, first, error detection is performed by parity prediction and then, the corrected output is reached by inversing the inputs and obtaining the reversed output. This method which is suitable to tackle single stuck-at faults (a type of hard faults) has a large

* Tel.: +98 1132339214.
E-mail address: m.valinataj@nit.ac.ir

http://dx.doi.org/10.1016/j.micpro.2014.10.002
0141-9331/

hardware and delay overheads. However, its cost is lower than a TMR-based system. Another fault-tolerant method [13] in order to handle the hard faults in adders uses a parity-based technique to detect a fault and then, faulty digit localization is performed using a re-computation with shifted operands, and at last, error correction is made. A similar fault-tolerant method and its extension are presented in [14,15], respectively. The former acts by re-computing with shifted operands, and the latter by embedding some extra ripple carry adders (RCA) inside the sparse carry tree. However, both methods require a large amount of hardware.

Among arithmetic operations, addition, subtraction and a portion of some multiplier designs can be performed by carry lookahead adder, one of the fast adder designs. Error detection techniques proposed for carry lookahead adders, usually utilize the parity prediction scheme or DMR method along with a two-rail checker to achieve an efficient self-checking design [4,16]. In [4] a new self-checking architecture for carry lookahead adder, RCA and ALU, is proposed in which two-rail checkers are used to detect errors in the internal carries and the parity prediction is used to detect errors in the output. In [16] the parity prediction is used to detect errors in input operands while two-rail checkers are used for the output error detection in which output sum bits are duplicated based on the DMR method. These methods need less hardware overhead in comparison with DMR. However, both are designed to detect single errors. Another method [17] is designed to manage hard faults in a 32-bit ALU including carry lookahead adder. This method is capable of detecting and isolating two erroneous 8-bit sub-blocks, online, and it can be also classified as a fault-tolerant method because the main ALU can still work after deactivating two sub-blocks. This method is only useful for hard faults. In [18] a fault-tolerant carry lookahead adder is proposed to correct single soft errors. However, it can correct more soft errors if it uses more hardware redundancy. This method uses a TMR approach with unprotected voters in all the logics except the carry generation block in which smaller redundancies are used. This way, it reaches a lower hardware overhead compared to the basic TMR.

Error correcting codes (ECC) that are essentially efficient for protecting memory elements [19] can be used to correct multiple soft errors in combinational logic including arithmetic operations, but they require a high hardware cost [20,21]. As illustrated, a number of valuable previous works are only useful for hard faults, and other approaches only consider detection or correction of single soft errors or have a high hardware cost. In this paper, we combine and reinforce the parity prediction-based carry lookahead adder with a partial TMR distributed in all of the bit slices to achieve more reliability, error detection or error correction, against multiple simultaneous errors. This adder is designed in such a way that some parts of its combinational logic including the TMR voters are protected by parity prediction and the remaining logic is protected by partial TMR. As stated before, parity prediction is useful for single error detection while TMR is useful for single error correction. However, the proposed adder acts in such a way that makes it capable of multiple error detection or correction with a high probability in addition to being self-checking against single errors. This design is targeted to transient or soft errors. However, single or multiple error correction capability is also applicable to hard faults.

This paper includes the following contributions:

- With a graceful combination of different error detection/correction methods, the proposed carry lookahead adder architecture can detect or correct multiple simultaneous errors with a high probability. This characteristic will be analytically proved later.
- This architecture does not need two-rail checkers used in [4,11,16] to achieve the self-checking property. This reduces the overall hardware cost.
- Unlike TMR-based methods, the voters are protected by the parity prediction scheme with no extra hardware overhead. Thus, it is feasible to detect errors in the voters.
- Two alternative methods resulting new designs are proposed to reduce the overall hardware cost in the main architecture.

The rest of the paper is organized as follows. In Section 2, the background needed for the proposed architecture is discussed. In Section 3, the proposed scheme to address multiple simultaneous errors in carry lookahead adders and its analysis are described. In Section 4, two methods for cost reduction in order to obtain more efficient architectures are illustrated. The evaluation of the proposed architectures is discussed in Section 5. Finally, the conclusion remarks are drawn in Section 6.

2. Parity prediction scheme for addition

In a self-checking design, a circuit is said fault-secure if its output code detects all errors generated by each modeled fault. Thus, the fault-secure property ensures high quality concurrent error detection [22,23]. In addition, a circuit is said self-testing if it is guaranteed that for each modeled fault there is at least one input vector, occurring during the normal operation of the circuit that detects it [22]. A circuit is totally self-checking if it is both fault-secure and self-testing. Such a circuit ensures that under any modeled fault, the first erroneous output of the functional block is detected in its checker [4].

For a one-bit Full Adder (FA), the output bit S_i is equal to $a_i \oplus b_i \oplus c_{i-1}$ which means the logical XORing or modulo2 sum between two input operands and input carry. Therefore, by considering modulo2 sums, the output parity of an n-bit adder is given by Eq. (1):

$$P_S = \sum_{i=0}^{n-1} S_i = \sum_{i=0}^{n-1} (a_i \oplus b_i \oplus c_{i-1}) = \sum_{i=0}^{n-1} a_i \oplus \sum_{i=0}^{n-1} b_i \oplus \sum_{i=0}^{n-1} c_{i-1} = P_A \oplus P_B \oplus P_C \quad (1)$$

where a_i, b_i and S_i are the ith bits of the input operands A and B and the output S, respectively. Variable c_i is the input carry of ith slice, P_A, P_B are the parities of the input operands A and B, and P_C is the parity of the internal carries. The parity prediction scheme is also used to design other self-checking arithmetic operators [23–26]. However, an appropriate design modification has been applied to these designs to be fault-secure for single faults. This is due to the fact that the parity prediction scheme is not fault-secure for single faults in adders and as a result in many other arithmetic operators because common mode errors can affect both the predicted parity (P_S) and the output sum (S). This occurs when a carry bit is erroneous, because this bit produces at least one output bit error along with a toggle in P_C which makes the error undetectable. Thus, a duplicated carry logic is used and P_C is computed from the redundant carries instead of the normal carries to avoid this undesirable situation [4,23,26].

Fig. 1 shows two different FA designs with redundant carries [23]. The FA shown in Fig. 1a has a higher area and lower delay compared to the one shown in Fig. 1b. The redundant carries are produced as the complement of the normal carries since the normal and the redundant carries should be checked by a two-rail checker to achieve fault secureness. The fault-secure property of these full adders for single errors has been proved in [23]. These full adders are beneficial for the ripple-carry adder and the adder networks used in Wallace tree and array multipliers.

To predict the output parity (P_S) of an adder according to Eq. (1), only P_C has to be computed, since in a system using parity-encoded