Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 107 (2017) 313 – 318

Procedia Computer Science

International Congress of Information and Communication Technology (ICICT 2017)

Error Detection by Diverse Instructions and Loop Optimization in DSP

Zeming Shao[a,*], Qingping Tan[a], Jianjun Xu[a]， Xiankai Meng[a]

[a]*College of Computer, National University of Defence Technology, Changsha, China*
** Corresponding authors: szmlonely@163.com*

Abstract

For the digital signal processors (DSP), a new approach of detecting soft errors is proposed to overcome the transient fault, named EDIO. The goal of EDIO is to enhance the reliability of a DSP software system with reduced performance overhead, compared to former software fault tolerant techniques. EDIO employs diverse instructions for the sake of furthest exploring errors and increasing performance. The loop optimization delays the fault tolerance latency between errors detecting and errors handling, only in this way can EDIO schedule the software pipeline to reduce performance overhead significantly. We evaluate EDIO by the performance experiments and the ion irradiation experiments, which demonstrate exceptional fault coverage with a reasonable performance cost. Compared to the well-known software approach, EDIO demonstrates 6.4 times average speedup while remaining the ability to detect faults.

Keywords: DSP, transient fault, soft error, error detection, diverse instruction, loop optimization, linear assembly

1. Introduction

During the recent a few decades, processor designers employ the techniques of in-creasing clock rates, reducing noise margins, lowing voltage levels, and shrinking feature sizes, with the purpose of improving performance and reducing power. However, these techniques make processers more and more susceptible to transient faults which can jeopardize the correct software execution. These transient faults, incurred by the hitting of high energy particles from cosmic rays or packaging materials on the transistors, are also known as soft errors[1,2].

Soft errors occurring at the hardware-layer as bit flips may propagate and manifest at the software-layer, which may finally alter stored values or signal transfers and result in incorrect program execution. Unlike the permanent physical damage, transient faults do not occur consistently and they can be corrected by covering right values. To deal with these issues, comprehensive research has been conducted and can be categorized in two major classes, i.e. hardware-level techniques and software-level techniques.

doi:10.1016/j.procs.2017.03.111

1.1. Relation to prior work

Hardware-level techniques primarily employ architectural redundancy[3,4], design with reduced architectural vulnerability factor[5,6] and pipeline protection with shadow latches[7] to detect and correct the soft errors. For instance, Triple-modular redundancy (TMR)[8] is a widely used technology in hardware-level fault tolerance. TMR corrects the soft errors by the voting of three copies of the hardware component working at the same time. Error correcting codes (ECC) and parity bits are often used to detect and correct the fault in the storage structure, such as memory and caches. These techniques can achieve the significant improvement of reliability. However, they also introduce significant hardware costs in terms of area and power overhead, which is typically prohibitive for embedded systems.

To mitigate the hardware costs, also to complement with the hardware-level techniques, some software-level fault tolerance techniques have been proposed. Stanford proposed a software redundancy approach known as EDDI[9], in which all instructions are duplicated and checking instructions are inserted before the "store" and "conditional branches". ED4I[10] introduces data diversity based on EDDI to detects both permanent and temporary errors by executing two "different" programs (with the same functionality) and comparing their outputs. ED4I maps each number in the original program into a new number, and then transforms the program so that it operates on the new number so that the results can be mapped backwards for comparison with the results of the original program.

1.2. The defect of traditional algorithms applied to DSP

The high-performance digital signal processor (DSP) has a powerful digital signal processing capability, and has been widely used in electronic systems of the spacecraft. Therefore, effective hardening techniques should be applied to DSP software to protect it from cosmic rays.

However, DSP's software pipeline which is the critical factor of the execution performance is susceptible to the "branches instructions". That is if there are more than one branches instructions in a loop, the assembler can't schedule software pipeline of this loop. It is indispensable for almost every software-level hardening technique to introduce branches instructions. These traditional fault tolerance algorithms incur significant performance overhead in DSP, which is sometimes unacceptable in the real-time systems. To overcome these issues, this paper presents the EDIO approach.

1.3. Goal and Novel Contributions

The goal of this paper is to come up with a novel DSP soft error detection method with higher reliability and lower performance overhead. For this reason, this paper introduces EDIO, a DSP soft error detection method by diverse instructions and loop optimization.

EDIO transforms equivalent some specific instructions and duplicates other instructions to detect soft errors. Optimized checking instructions are inserted before the "store" and "conditional branches". What's more, loop optimization utilized in EDIO. Therefore, EDIO can significantly improve the efficiency of the code and broaden the difference between the two redundancy instructions.

The remainder of this paper is organized as follows. Section 2 details the EDIO technique as an impressive work found in the field of DSP fault tolerance. Section 3 presents and analyses the result of ion irradiation experiments and performance experiments. Section 4 concludes the paper.

2. The EDIO approach

EDIO enhances soft error detection capability and reduces performance overhead through three main diverse instructions transformation techniques: the equivalent transformation of instructions, the optimization of the detection instructions and loop optimization. This section will describe the three diverse instructions techniques of EDIO respectively.