RISKVI Case Study

Jonathan Wang
Electrical and Computer Engineering
University of Utah
Salt Lake City, Utah
u1306458@umail.utah.edu

***Abstract*—RISK-V FPGAs are attracting chip developers worldwide due to their open source and ease of configuration. These FPGAs can be used in space applications. Implementing fast and reliable hardware on nanosatellites had to be tested under extreme circumstances. This report evaluates the impact of the Linux operating system on the reliability of RISC-V based FPGAs, against bare metal mode with memory upsets.**

***Index Terms*—RISC-V, FPGAs, nanosatellites, configuration memory upsets**

I. Introduction

Field-programmable gate arrays(FPGAs) are becoming attractive for nanosats due to the improvements in performance and in-field reconfigurability of new generations of SRAM-based FPGAs. A nanosat or nanosatellite(Fig. 1) is anything that weighs between 1 and 10 kilograms. They are becoming attractive for space travel due to improvements in performance and in-field reconfigurability compared to traditional computer chips. The software running on the FPGAs is called Embedded Operating System. An Embedded OS is a specialized operating system designed to perform a specific task for a device that is not a computer. The main job of an embedded OS is to run the code that allows the device to do its job, and it makes software development easier. Current operating systems that are qualified for space missions are generally very costly. Consequently, general-purpose OS-powered nanosat missions, such as Linux systems, have been deployed in the past. Leppinen [1] stated that choosing Linux limits the choice of hardware. Another major drawback is that Linux is not designed to be a real-time operating system. A real-time operating system is an OS that guarantees real-time applications a certain capability within a specified deadline. In some cases, design changes can reduce the number of hard real-time applications. The remaining constraints require another dedicated controller to handle the real-time problem.

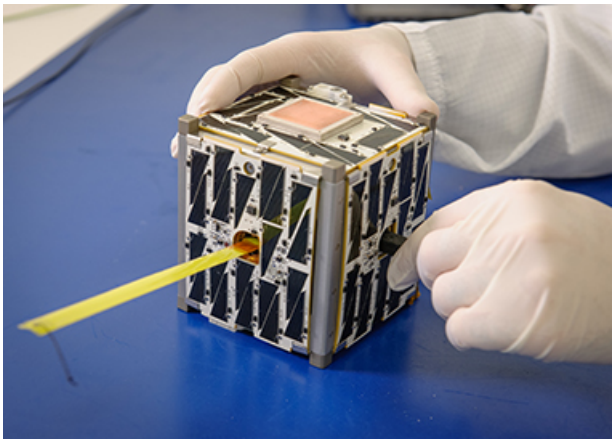

Fig. 1. A real life nanosat.

II. Background

The reliability of these OSs for a given hardware platform needs to be evaluated before their actual deployment. Wali et al. [2] assessed the effects of the Linux OS on the fault tolerance of applications running on a RISC-V SoC(System on Chip) implemented in a Xilinx FPGA. To address the evaluation of the effects, we need to define the background. Although previous studies analyzed the fault tolerance of OS in embedded platforms, the following report will focus on radiation-induced configuration memory upsets on the reliability of applications running on a Linux RISC-V FPGA. Radiation-induced configuration memory upsets are a type of computer hardware failure caused by exposure to ionizing radiation from space. The radiation can create a charge that alters the state of a memory cell, causing a bit flip when the chip is exposed. It can result in errors in data storage or retrieval. In some cases, it can crash the system.

A. FPGA Fault Injection

The method employed is called FPGA fault injection. It is defined as the validation technique of fault-tolerant systems where the observation of the system's behavior in the presence of faults is done explicitly by injecting faults into the system. Reference [2] pointed out that FPGAs are an ideal fault-emulation platform due to their high logic density and reconfigurability. The FPGA consists of two layers, the user layer, and the configuration layer. Partially Modifying the configuration layer can start the injection process. Tawfeek et al. categorized the fault injection technique into three categories in [4]. In our case, we will be using software-based injection. The objective is to inject errors at the software level. The injected error simulates those that may occur due to hardware faults.

III. Fault Injection Setup and Overview

The hardware the experiment uses includes an Artix-7 series FPGA from Xilinx Inc. and a host computer.

A. Hardware Setup

Figure 2 shows the block diagram of the hardware setup. The host computer is responsible for the fault injections and the debug software, which communicates with the FPGA's SoC through a debug interface. The interface controls the executions of workload programs, collection of data, logging the errors, and resets the program for the following execution. The workloads were tested both in bare-metal mode and with the Linux OS. Bare-metal mode runs directly on the hardware of a computer without any operating system or other software layers in between.

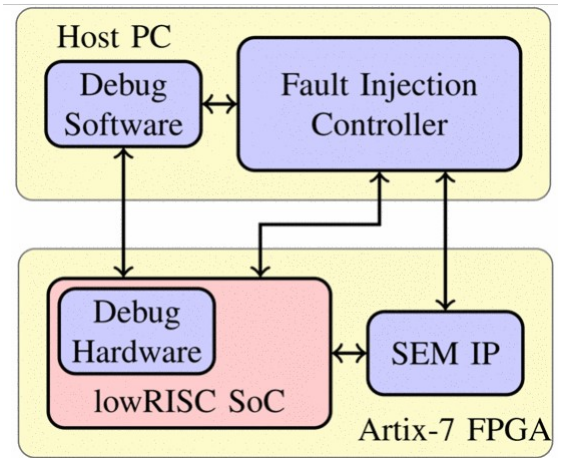

Fig. 2. Block diagram of the fault injection setup.

The fault injection controller is located inside the host computer and utilizes Soft Error Mitigation Intellectual Property (SEM IP) to introduce single event upsets (SEUs) into the Artix-7 FPGA's configuration memory. Maillard et al. from Xilinx performed a similar analysis [3] to simulate the impact of radiation on the FPGA, but they utilized a proton beam rather than actual radiation.

B. Benchmarks

The workloads for fault injection consist of five mathematical algorithms: Tower of Hanoi, Dijkstra's algorithm, Matrix multiplication, Quick sort, and Merge sort. It is essential to know the Big-O(runtime complexities) of these algorithms because they can affect the fault injection results. Runtime complexity is a measure of how much time or space a program or algorithm takes to run as the input size increases. An algorithm with high runtime complexity is more vulnerable due to its increased complexity. It is also more difficult to identify the root cause of any observed failures. Table 1 shows the Big-O of the algorithms.

C. Fault Models

There are a total of 10 fault injection campaigns in the experiment, and each campaign is carried out on 5 distinct workloads. Each fault injection campaign injects bit-flips in 10,000 randomly selected essential bits in the memory. Faults are only injected in the configuration memory because it accounts for most of the on-chip memory, which is large enough that it is very sensitive to SEUs. This fault model has been proven in detail by [5].

Algorithm	Complexity
Towers of Hanoi	$O(2^n - 1)$
Dijkstra	$O(\|V\|^2 + \|E\|)^\dagger$
Matrix Multiplication	$O(n^3)$
Quicksort	$O(n \log n)$
Merge sort	$O(n \log n)$

n is the number of steps in computational mode.
† V represents the number of Vertices and E represents the number of Edges.

Fig. 3. Workload's runtime complexity.

IV. Fault Injection Process

To begin the injection process, we first need to generate a faults-site list

References

[1] H. Leppinen, "Current use of linux in spacecraft flight software," in IEEE Aerospace and Electronic Systems Magazine, vol. 32, no. 10, pp. 4-13, October 2017, doi: 10.1109/MAES.2017.160182.

[2] I. Wali, A. Sánchez-Macián, A. Ramos and J. A. Maestro, "Analyzing the impact of the Operating System on the Reliability of a RISC-V FPGA Implementation," 2020 27th IEEE International Conference on Electronics, Circuits and Systems (ICECS), Glasgow, UK, 2020, pp. 1-4, doi: 10.1109/ICECS49266.2020.9294858.

[3] P. Maillard et al., "Single-Event Evaluation of Xilinx 16nm UltraScale+™ Single Event Mitigation IP," 2018 IEEE Radiation Effects Data Workshop (REDW), Waikoloa, HI, USA, 2018, pp. 1-5, doi: 10.1109/NSREC.2018.8584298.

[4] R. M. Tawfeek, M. G. Egila, Y. Alkabani and I. M. Hafez, "Fault injection for FPGA applications in the space," 2017 12th International Conference on Computer Engineering and Systems (ICCES), Cairo, Egypt, 2017, pp. 390-395, doi: 10.1109/ICCES.2017.8275338.

[5] L. Sterpone and M. Violante, "A New Partial Reconfiguration-Based Fault-Injection System to Evaluate SEU Effects in SRAM-Based FPGAs," in IEEE Transactions on Nuclear Science, vol. 54, no. 4, pp. 965-970, Aug. 2007, doi: 10.1109/TNS.2007.904080.