NASA/TM—20230013162

Single-Event Effects Test Report Linux Operating System Configurations on TUL PYNQ-Z2

Seth S. Roffe

September 2023

NASA/TM—20230013162

Single-Event Effects Test Report Linux Operating System Configurations on TUL PYNQ-Z2

Seth S. Roffe
Goddard Space Flight Center, Greenbelt, MD

Test Date: 8/19/2023
Report Date: 9/8/2023

National Aeronautics and
Space Administration

Goddard Space Flight Center
Greenbelt, MD 20771

September 2023

Acknowledgments (optional)

This work was sponsored by the NASA Electronic Parts and Packaging (NEPP) Program.

Available from

NASA STI Program	National Technical Information Service
Mail Stop 148	5285 Port Royal Road
NASA's Langley Research Center	Springfield, VA 22161
Hampton, VA 23681-2199	703-605-6000

This report is available in electronic form at

https://nepp.nasa.gov/

1. Introduction and Purpose

This study was undertaken to determine the single-event functional interrupt (SEFI) susceptibility on different Linux operating system configurations. The device-under-test (DUT) was the Xilinx Zynq-7020 SoC on the TUL PYNQ-Z2 board. The device was monitored for kernel panics or hangs, classified as SEFIs, to observe any differences in the SEFI cross sections between operating system configurations. The primary purpose of this experiment was to observe if the number of drivers installed in a Linux system affects its overall execution reliability.

The operating system configurations included all possible iterations of four binary variables: whether the L2 cache (with parity detection) is on or off, whether the system was performing matrix multiplication (MM) or sitting idle, whether there were a large number of drivers installed in the kernel or a small number of drivers installed, and whether those drivers were loaded into the current shell or not loaded into the current shell. These four variables created 16 different configurations, shown in Table 1.

Table 1: Operating System Configurations

L2 Cache	Operation	Num. Drivers	Loaded State
ON	MM	LARGE	LOADED
ON	MM	LARGE	UNLOADED
ON	MM	SMALL	LOADED
ON	MM	SMALL	UNLOADED
ON	IDLE	LARGE	LOADED
ON	IDLE	LARGE	UNLOADED
ON	IDLE	SMALL	LOADED
ON	IDLE	SMALL	UNLOADED
OFF	MM	LARGE	LOADED
OFF	MM	LARGE	UNLOADED
OFF	MM	SMALL	LOADED
OFF	MM	SMALL	UNLOADED
OFF	IDLE	LARGE	LOADED
OFF	IDLE	LARGE	UNLOADED
OFF	IDLE	SMALL	LOADED
OFF	IDLE	SMALL	UNLOADED

2. Test Result Summary

On such as complex system such as the Zynq-7020 SoC, SEFIs and system crashes were prevalent, though no destructive single-event effects were observed during 200 MeV proton testing. This experiment primarily looked at any significant differences in the cross sections and the mean fluence to failure (MFTF) between any of the major OS configurations. Vulnerability tended to be higher when the L2 cache was on, as expected due to the larger memory space under irradiation. Additionally, when the L2 cache was on, the differences in the other configurations were more apparent. Namely, the IDLE configuration tended to be significantly more vulnerable to SEFIs than when matrix multiplication was happening. Similarly, the system was more vulnerable to SEFIs when the drivers were loaded than when they were installed but not loaded into the kernel. No significant differences were observed between the LARGE and SMALL configurations.

3. Device Description

The TUL PYNQ-Z2 board is a development board with a Xilinx Zynq-7020 SoC. The Zynq-7020 contains an Artix-7 Field-Programmable Gate Array (FPGA) fabric as programmable logic (PL), as well as a dual-core ARM Cortex-A9 processing system (PS). For the purposes of this experiment, only the ARM A9 processing core was used. Therefore, this test was only looking at the Zynq-7020 from a fixed-logic processor perspective.

Table 2: Device-Under-Test Description

SoC	ZYNQ XC7Z020-1CLG400C
Processor (PS)	ARM 650MHz dual-core Cortex-A9
FPGA (PL)	Artix-7
Manufacturer	TUL
Memory Controller	512 MB DDR3; 16-bit bus @ 1050 Mbps
SPI Flash	16MB
REAG ID	20-004

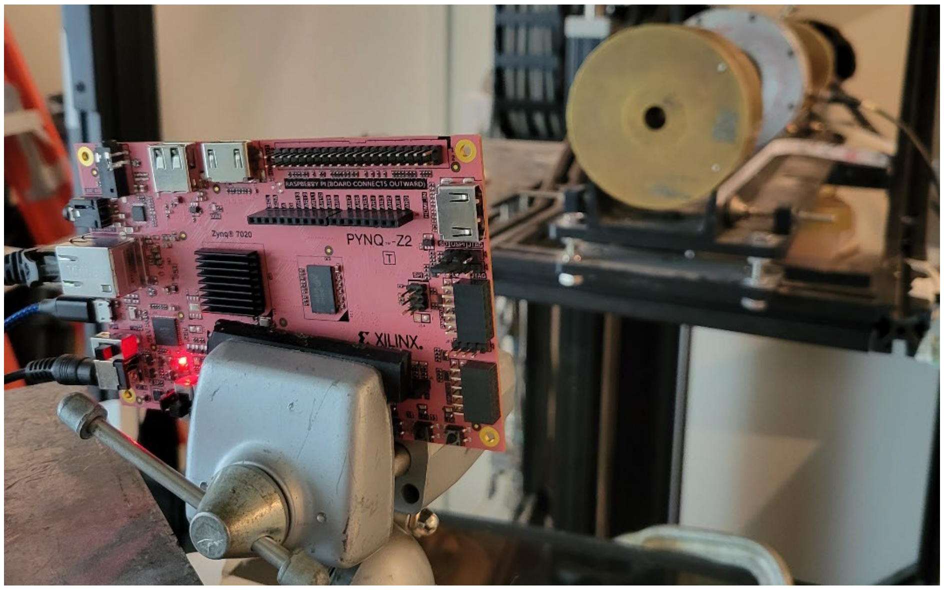

Figure 1: PYNQ-Z2 in the proton beam path

4. Test Setup

The experimental setup used USB connections for serial communication with the device-under-test (DUT) and ethernet for Trivial-File Transfer Protocol Boot (TFTPBoot) transfers. TFTPBoot was used to load each operating system configuration during boot-up of the DUT. Additionally, the current and voltage input to the DUT was monitored with a Phidget 30 Amp Current sensor and a Phidget Precision Voltage Sensor, respectively. To provide USB connections into the beam room, a USB-to-Ethernet extender was used, along with 100 ft ethernet cables connecting the transceiver and receivers. A web-powered power switch was used to remotely control the DUT and provide power cycles as needed. All commands were sent from and data was collected on a Laptop running Ubuntu 22.04 outside of the beam cave.

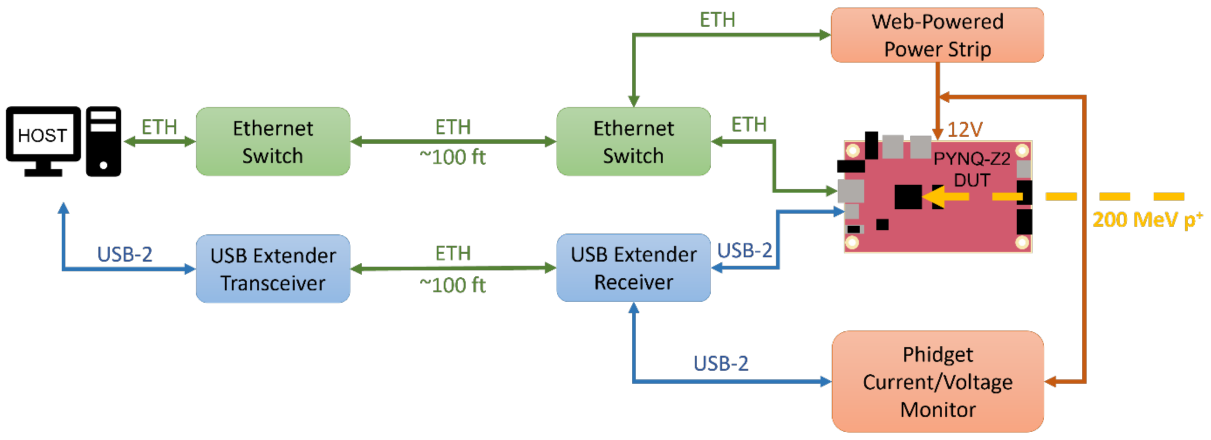

Figure 2: Experimental Setup Diagram

5. Test Facility

Facility:	Massachusetts General Hospital, Proton Therapy Clinic
Type of Radiation:	Proton
Facility Configuration:	200 MeV tune
Flux:	$\sim 10^8 \frac{p^+}{cm^2 s}$
Fluence:	Testing was conducted to the first SEFI for each run

6. Test Conditions

In-Air or Vacuum:	In-air
Supply Voltages:	12V

7. Test Methods and Procedures

This section describes the methodology used in the experiment. The procedure for a run is defined and how data was collected is specified. The configuration setup is also discussed herein.

7.1. Linux Kernel Setup

Each configuration is composed of a Linux kernel image that was generated using Buildroot and a device tree defining the hardware interfaces. The L2-Cache ON configuration uses the default device tree for the PYNQ-Z2. For the Cache OFF configuration, the L2 Cache was disabled by removing the memory interface for the cache in the device tree. When this modified device tree was used to boot the DUT, finding the interface on boot would fail, thus disabling the L2 cache, which could additionally be verified via the difference in execution time of any matrix operations between the OFF and ON configurations.

The MM configuration's main process was a 500×500 matrix multiplication kernel. This application kernel was chosen as it is a common application used in machine-learning and image-processing applications. After each matrix multiplication, the DUT would send a heartbeat signal to the host to acknowledge completion of proper operations, and then it would restart the application. The IDLE configuration would simply idle the system using the sleep command for 10 seconds, send the heartbeat signal, and then return to idling for another 10 seconds. Both of these configurations were set to run indefinitely on boot via initialization scripts.

The LARGE configuration was created by selecting several random built-in drivers from Buildroot to compile into the kernel, even if the supporting hardware for those drivers was not available. This configuration had a uImage size of 8.3 MB. Conversely, the SMALL configuration only used the minimum number of drivers necessary to compile Linux for the PYNQ-Z2 with a uImage size of 5.1 MB. Overall, the LARGE configuration contained 256 modules that were built-in to the kernel, and 170 modules that could be loaded in externally via the insmod command. Meanwhile, the SMALL configuration had 126 built-in kernel modules, and 10 modules that could be loaded externally. These equate to 66.4% and 7.9% of loaded drivers for each configuration, respectively.

The LOADED configuration found all possibly loaded drivers via modprobe, and loaded those drivers with insmod on boot to make them active, prior to running the process defined for the current MM/IDLE configuration. Due to differences in hardware and the lack of specific device tree entries for certain drivers, not all drivers found with modprobe could be loaded into the kernel. Overall, the LARGE configuration had 160 drivers loaded into the kernel to make the, and the SMALL configuration had 7 loaded (94.1% and 70% of the total possible drivers to externally load in each configuration, respectively). The UNLOADED configuration simply skipped loading any drivers with insmod and thus had zero drivers loaded. In this configuration, each driver was installed, but not actively loaded into the kernel.

7.2. Proton Test

To select a configuration, a pre-compiled Linux uImage was selected and place in a TFTP server directory to be pulled for the boot process. Once power was enabled to the DUT using the web-powered power switch, the system would pull the current configuration's uImage and respective device tree.

The DUT was booted and put into a stable state performing the configurations operations, whether that be matrix multiplication (MM) or idling via the sleep command (IDLE). Once the operations began, as noted by the test operator, the beam was turned on. The beam consisted of 200 MeV protons tuned to a flux of approximately $10^8 \frac{p^+}{cm^2 s}$. Once a SEFI was encountered, either by a kernel panic or a system hang event with a timeout set at 1 minute, the beam was turned off, the fluence was recorded, and the cross section (akin to mean fluence to failure in this study) was calculated. This process was then repeated for several runs.

8. Test Results

Each configuration was run 15 times during the allotted beam time. The histograms of cross sections for each configuration is shown in Figure 3. Each distribution is approximately a Poisson distribution, as was expected with collecting data across more than 10 runs per configuration. Each cross section was defined with an event-quantity of 1 since a SEFI would shut down the system and end the run.

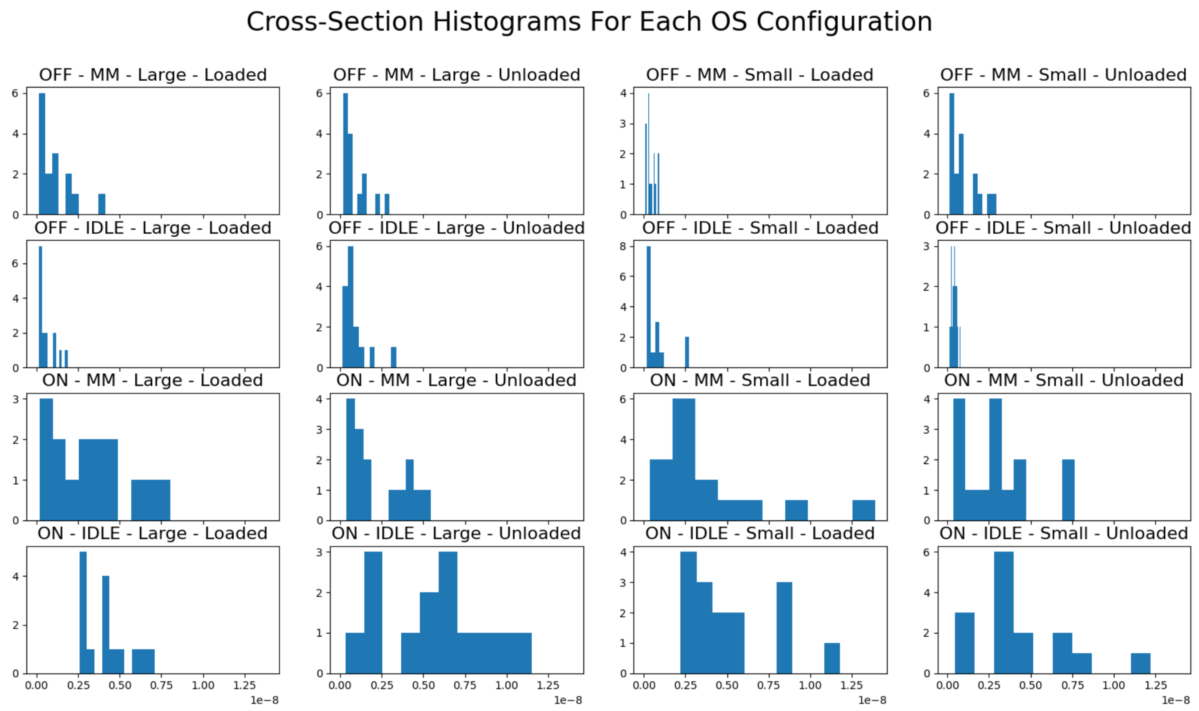

Figure 3: Cross Section Histograms for Each Configuration

However, it is easier to analyze a metric that is not bound between 0 and 1, therefore the mean fluence to failure (MFTF) was used. MFTF is defined, in this case, as the inverse of the cross section since the event-quantity was 1 for each run. The MFTFs for all the L2-Cache ON configurations can be seen in Figure 4.

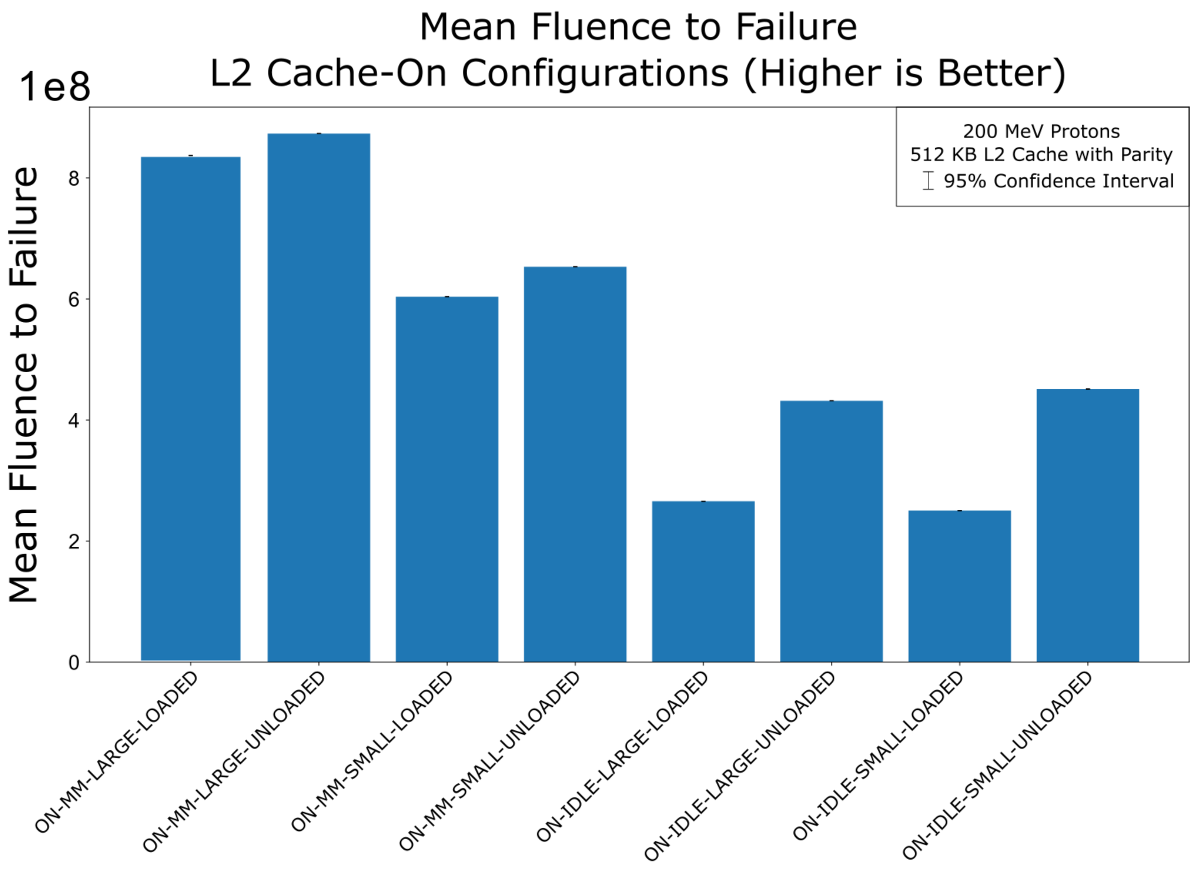

Figure 4: MFTF for L2-Cache ON Configuration

When the L2 Cache is ON, the MFTF is significantly worse for the IDLE configurations compared to the MM configurations. This is most likely due to the critical data retrieval processes of the DUT. When matrix multiplication is processing, critical kernel data is being flushed out of the cache to make room for the matrix data. If the system is idling, then that critical data never gets flushed out of the cache and thus has a higher chance of corruption, especially since the DDR3 SDRAM chip was not under irradiation. Additionally, the UNLOADED configurations were significantly better than the LOADED configurations. This implies that the drivers being loaded and active in the kernel do affect the reliability of the system. This is expected since the drivers have memory access to kernel space, even when they aren't actively being used. However, the number of drivers installed and loaded do not seem to have a significant effect on MFTF. We surmise that this could be because there wasn't enough of a significant difference in the number of drivers between the configurations, or because the drivers were not being actively used on real hardware devices.

The MFTF for each of the L2-Cache OFF configurations can be seen in Figure 5.

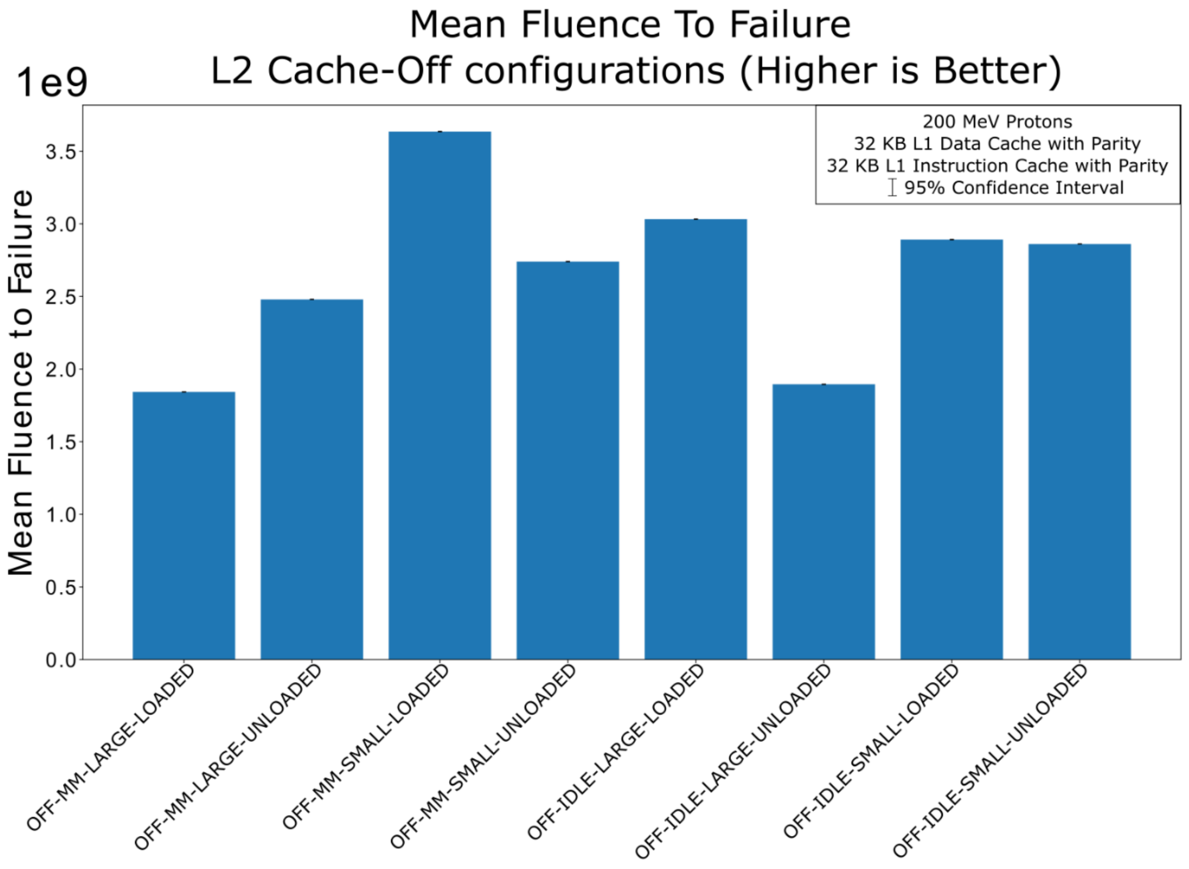

Figure 5: MFTF for Each L2-Cache OFF Configuration

For the L2-Cache OFF configurations, it is more difficult to discern any obvious patterns. This lack of crucial information could be because, without the L2 cache, there is not enough memory to hold much critical information in the 32 KB of both of the L1 data and instruction caches. Therefore, most of the critical data is held in the DDR3 SDRAM which was not under irradiation for this test.

9. Conclusion

This test was the first step in understanding how the configuration of the operating system affects the reliability of a system-on-chip (SoC) devices. Knowing which configurations drive worst-case and best-case radiation performance permits a more streamlined approach to characterizing these complex devices using well defined test corners. This understanding will lead to better testing methodologies on SoCs, and lead to a standardization of how these complex devices are evaluated. Finally, in knowing how different OS configurations affect the reliability of a system, mission designers can take proper precautions to ensure the highest availability of their software.