Proton Testing of nVidia Jetson TX1

E.J. Wyrwas[1]

NASA Goddard Space Flight Center
Code 561.4, Radiation Effects and Analysis Group
8800 Greenbelt RD
Greenbelt, MD 20771

Test Date: 16 October 2016
Report Date: 31 October 2016

[1]LENTECH Inc, Greenbelt, MD USA

To be published on nepp.nasa.gov.

1. Acronyms

BGA	Ball Grid Array
Cat5e	Category 5e (enhanced) specification
CPU	Central Processing Unit
CUDA	Compute Unified Device Architecture
CUFFT	CUDA Fast Fourier Transform library
DHCP	Dynamic Host Configuration Protocol
DRAM	Dynamic random-access memory
DUT	Device Under Test
EGL	Embedded-System Graphics Library
ES	Embedded Systems
GPU	Graphical Processing Unit
GUI	Graphical User Interface
HDMI	High-Definition Multimedia Interface
IPv6	Internet Protocol version
MGH	Massachusetts General Hospital
OpenGL	Open Graphics Library
OpenCL	Open Computing Language
RAM	Random Access Memory
RJ45	Registered Jack #45
SDK	Software Development Kit
SEE	Single Event Effects
SKU	Stock Keeping Unit
SNTP	Simple Network Time Protocol
SOC	System on Chip
SOM	System on Module
SRAM	Static Random Access Memory

2. Introduction and Summary of Test Results

Single-Event Effects (SEE) testing was conducted on the nVidia Jetson TX1 System on Chip (SOC); herein referred to as device under test (DUT). Testing was conducted at Massachusetts General Hospital's (MGH) Francis H. Burr Proton Therapy Center on October 16th, 2016 using 200MeV protons. This testing trip was purposed to provide a baseline assessment of the radiation susceptibility of the DUT as no previous testing had been conducted on this component. While not all radiation-induced errors are critical, the effects on the application need to be considered. More so, failure of the device and an inability to reset itself should be considered detrimental to the application. Radiation effects on electronic components are a significant reliability issue for systems intended for space.

The testing that has been conducted should be considered a very partial test vector. While it was possible to induce upsets in the nVidia Jetson TX1, we lack insight into which element within the device experienced the upset. Further testing should include memory mapping vectors in SRAM and DRAM, and a bus-level test as these were the majority of the upset events recorded. Because the device was recoverable upon a power cycle, its use in a radiative environment may be possible given a hardware or software watchdog routine to detect an error and reset the device.

3. Device Tested

The nVidia Jetson TX1 System on Chip (SOC) is provided on a modular printed circuit board connected to a main carrier board by a 400-pin connector. The SOC, itself is the device under test (DUT).

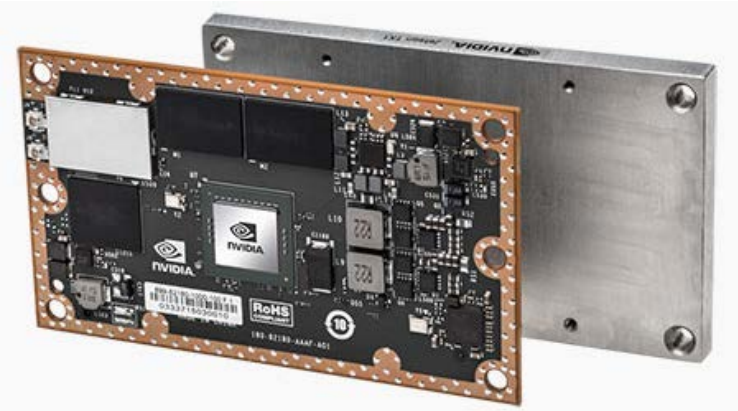


Figure 1: Nvidia Jetson TX1 System on Module

Table 1: Part Identification Information

Quantity	1
Part Model	Jetson TX1
Part Number	READ 16-038
Manufacturer	nVidia
Technology	20nm
Packaging	Flip Chip, BGA
Module SKU	699-82180-1000-100 U
Module Serial	0321516059688
Carrier Board SKU	699-82597-0000-302 E
Carrier Board Serial	0321716023673

The version of Ubuntu Linux that is pre-installed on the DUT's flash memory should be considered feature-lite as it is an embedded version based on the ARM 64-bit platform compiled specifically for the Tegra SOC. Very few sensors are accessible within Ubuntu as the hardware does not enumerate with compatible drivers in the Ubuntu tools. However, the temperature of the GPU and CPU are both accessible. The GPU temperature is used in this study which is designated as thermal zone #2 in the virtual devices in Ubuntu.

4. Test Facility

Facility: Massachusetts General Hospital's (MGH) Francis H. Burr Proton Therapy Center
Flux: $1.03 \times 10^6 - 2.83 \times 10^7$ protons/cm^2/sec
Fluence: $9.24 \times 10^6 - 3.77 \times 10^8$ protons/cm^2
Ion species: Proton, 200MeV

5. Test Setup

Because the DUT is a fully operable computer system, an in-situ testing environment is necessary to replicate a field application. This configuration consists of a DUT with an operating system, software applications and the underlying hardware. An external arbitration computer operating over a closed network is used to interrogate the device, execute remote commands and monitor the DUT health.

A. Software

Software was devised to remotely execute Linux shell commands that would otherwise be manually typed into the local terminal of the DUT. The purpose of this software was to instigate and record a set of commands sent to the DUT, the system's response in terms of system temperature and processor utilization, and any errors occurring during the testing. A screenshot is provided in the following figure.

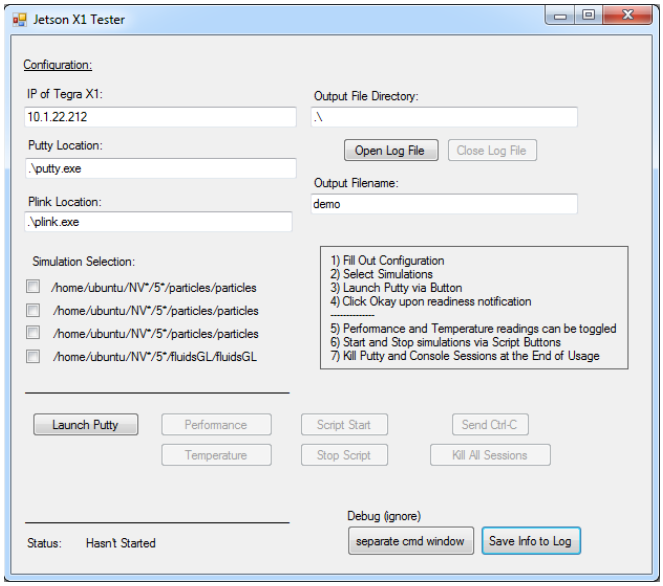


Figure 2: Screenshot of Jetson X1 tester software

The software was written in Visual Basic dot Net using the Visual Studio 2015 programming environment from Microsoft. This arrangement permitted use of both Windows and Linux commands while within a user friendly graphical interface and allowed portability of the application. Furthermore, it

allowed repeatability during test as any pausing of the scripts or intervals between commands were consistent.

The logical flow of software tasks was as follows:

1. User configurable DUT network address
2. User configurable record file location and filename
3. Local-sourced terminal client location (PuTTY[1] and Plink[2])
4. Simulation selection for workload deployment
5. Options to enable/disable record files and workload

The simulations that are deployed on the DUT are sample code runtimes from the nVidia Corporation's software development kit (SDK) for CUDA version 8. This code was used as part of this test protocol as it is highly optimized for the DUT by the vendor. Further, no claims can be made that custom code deployed on the DUT is responsible for errors experienced during testing. Future testing can compare the results of this baseline test with code that is vendor independent.

B. Test Vector

Two simulations were selected for this test: particles and fluidsGL. Particles uses CUDA to simulate and visualize a large set of particles and their physical interaction. FluidsGL is an example of fluid simulation using CUDA and CUFFT, with OpenGL rendering. A tertiary Ubuntu Linux environment was required to upgrade the DUT to the latest firmware (kernel version 3.10.96-tegra, L4T version r24.2) and monitor the DUT during lab bench evaluation prior to the test trip.

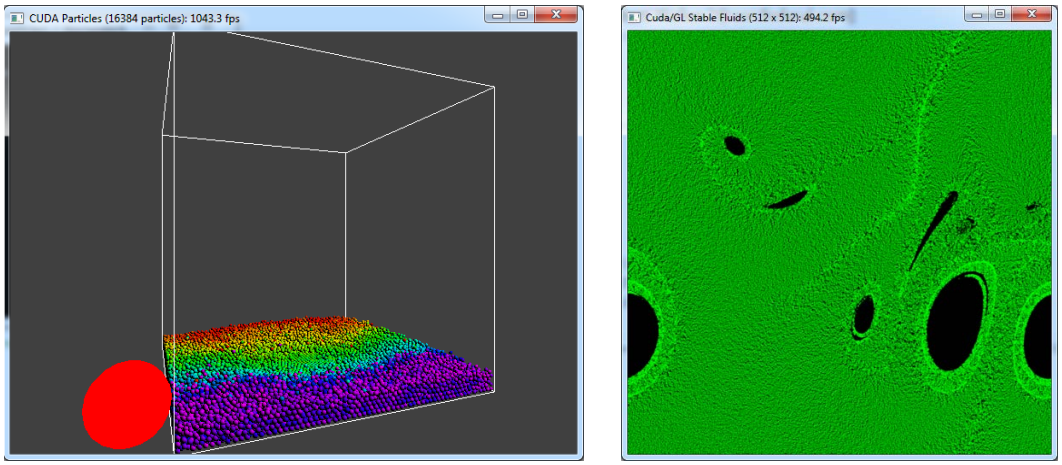


Figure 3: nVidia CUDA Samples (Particles and FluidsGL, left and right, respectively)

The goal of running these specific simulations was to assure low CPU usage (across all cores) and high GPU usage (as reported by grid usage) on the DUT. The selection of these simulations was based on GPU performance results as determined by the nVidia CUDA Profiler software (from nVidia Corporation) which was used to evaluate the consumption of system resources of the TX1. This software permitted evaluation of the simulation software threads in regards to core affinity, RAM usage, and CPU to GPU offloading.

[1] www.putty.org; PuTTY is an SSH and telnet client, developed originally by Simon Tatham for the Windows platform. PuTTY is open source software that is available with source code and is developed and supported by a group of volunteers.

[2] a command-line interface to the PuTTY back ends

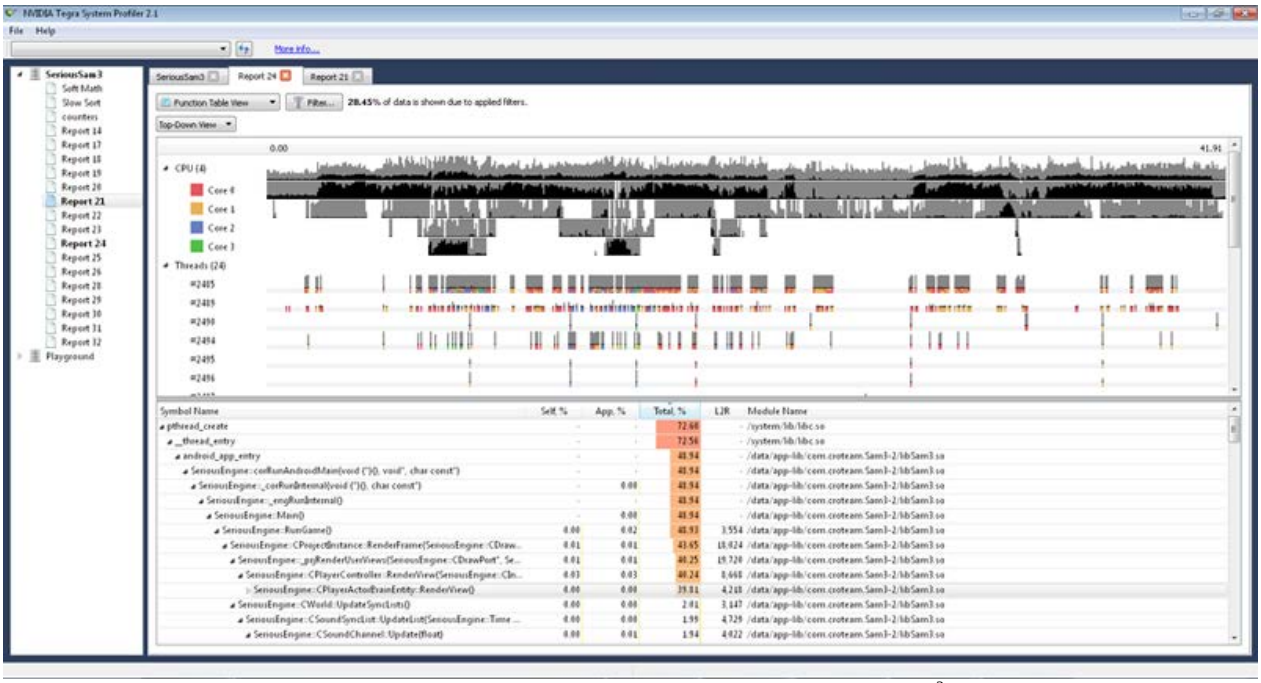


Figure 4: nVidia Tegra System Profiler, Representative Example[3]

C. Hardware

The DUT is found in the center of a metal core module that is used to sink the heat from the components. Due to the orientation of the module on the system board, the beam was aligned to the backside of the system board, through the thermal sink and onto the SOC. A secondary fan-sink is located on the component side of the module. There was sufficient clearance around the SOC and no components were present on the secondary side of the system board within the z-axis of the SOC. This was advantageous as it allowed some radiation mitigation to other system components such as the power management and flash memory components.

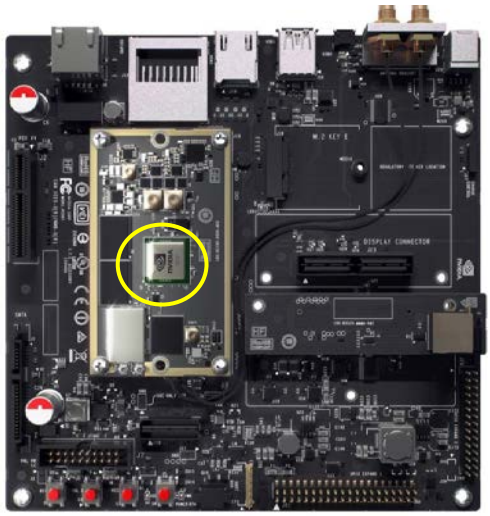


Figure 5: nVidia Image of Jetson TX1 System, with Tegra System on Chip marked[4]

[3] Image courtesy of https://developer.nvidia.com/tegra-system-profiler

[4] Image courtesy of http://www.anandtech.com/show/9779/nvidia-announces-jetson-tx1-tegra-x1-module-development-kit

Lucite bricks were used to shield the power supply of the DUT from scattered neutrons which are a result of proton collisions within materials in the beam's path. A photograph of the board under test is shown in the figure. An acrylic test fixture was used to mount the DUT. While seen in the image, an HDMI connection was not made with the DUT during testing as signal integrity was not sufficient enough to obtain an image from the test bench (~140FT away). The power cable is located on the upper right-hand side of the system board. The Ethernet cable plugs into an RJ45 jack on the upper-left hand side of the system board.

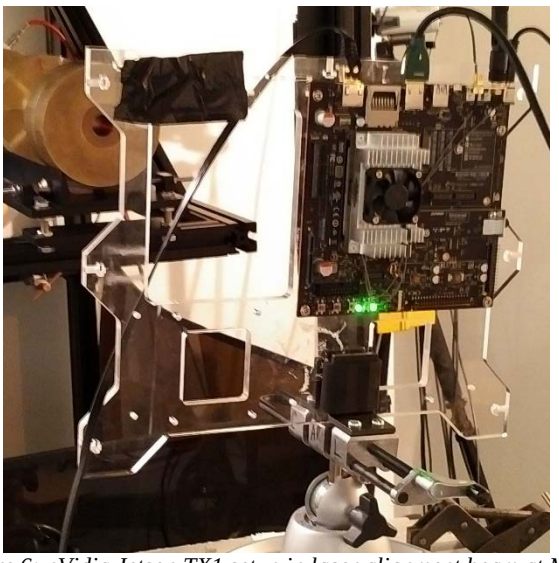


Figure 6: nVidia Jetson TX1 setup in laser alignment beam at MGH

The computer which runs the Jetson X1 tester software is considered the arbiter system. A secondary Ethernet dongle was connected to this laptop via USB to provide a DHCP service on a closed network. The DHCP service was Tftpd64[5]. A TrendNet network switch was utilized to provide an isolated medium between the arbiter and the DUT. Network communication was performed over a CAT5e cable from the DUT to the switch and the arbiter to the switch.

6. Test Procedure and Results

It is worth noting that the Date-Time of the DUT was incorrect at the start of the testing but remained consistent during all test runs. The maximum temperature recorded during each run was also tabulated. Room temperature was approximately 25°C. The DUT experienced a thermal loading of approximately 25°C during full load, 8°C during idle and 20°C during partial load, all above room temperature.

Upsets occurred during all sixteen runs. The occurrence and types of upset events are tabulated below. A screen shot is provided to show the visual record of each type. The network loss was recorded as the terminal client losing connection to the DUT during the run. For these instances, the beam was disengaged and an attempt was made to reestablish the network connection, though the latter was performed without success each time.

[5] Tftpd32.jounin.net; a free open-source IPv6 ready application which includes DHCP, TFTP, DNS, SNTP and Syslog servers as well as a TFTP client

Table 2: Upset Type Occurrence

Event	Occurrence
input address fault	1
level 3 cache	1
kernel paging request	2
level 2 cache	2
undefined instruction	2
bus failure	3
network loss	5

The first two runs had a higher proton flux than the others. It was noted during run #2 that the system crashed almost instantaneously as the beam was engaged. To allow for a longer evaluation period, the flux was decreased for all subsequent runs.

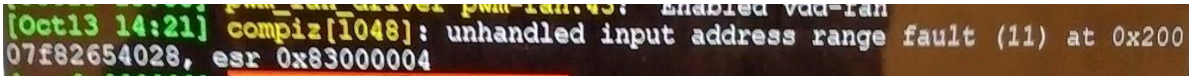


Figure 7: Error type – Input Address Fault

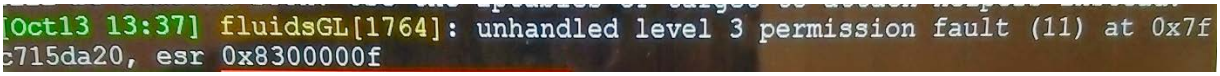


Figure 8: Error Type – Level 3 Cache

Figure 9: Error Type – Kernel Paging Request

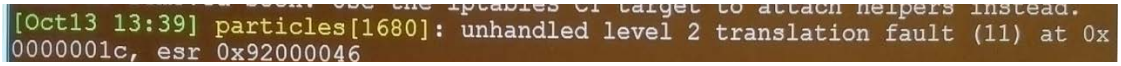


Figure 10: Error Type – Level 2 Cache

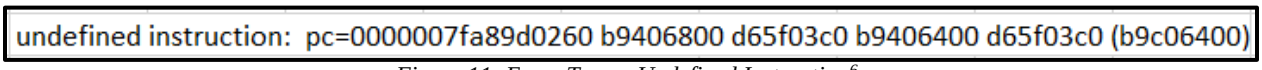


Figure 11: Error Type – Undefined Instruction[6]

Figure 12: Error Type – Bus Failure

Each run was conducted using a fresh session of the tester software and was performed after a power cycle of the DUT. A "kill all" command is contained in the tester software to ensure no remnant processes (ones that are utilized in the test sequence) are running prior to testing at both the arbiter machine and the DUT. The beam was engaged after the simulation software was running and at least one set of performance and temperature measurements were saved to the run's log file. Examples of both of these files is provided.

[6] No screenshot was taken during this run. Text extracted from SSH log file.

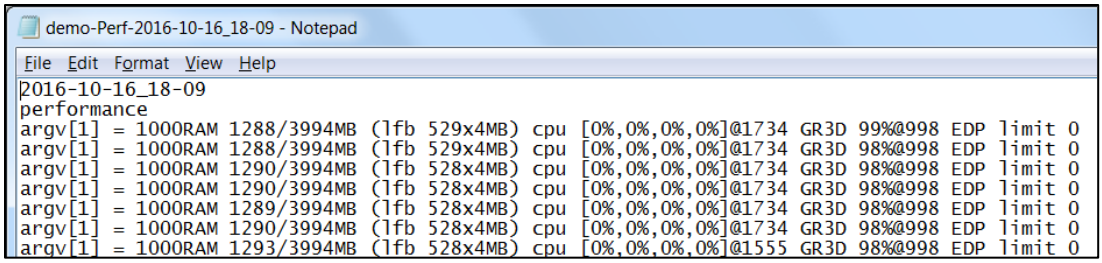


Figure 13: Example Performance Log (GR3D is the clock usage percentage of the GPU clock)

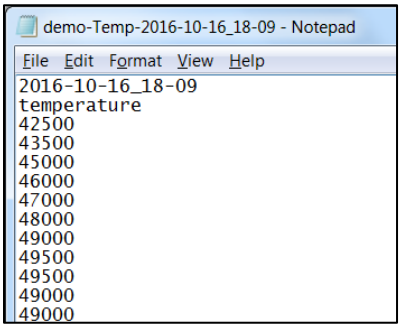


Figure 14: Example Temperature Log (Temperature in °C x 1000)

A tabulation of the crash conditions and beam characteristics is provided in Table 3. An in-depth assessment of the memory dumps was performed to confirm upset type using memory stack data. However some of the errors indicate a failure of a dongle which relates to the /drivers/net/wireless/bcmdhd device. This is listed as an "Android Wi-Fi" device which is not used during the proton testing. These events caused a kernel panic condition which subsequently crashed the firmware. Online research shows that this may be an issue with the kernel in this specific version of Linux for Tegra (L4T). It would seem that the Ethernet device (cable connected) and the Wi-Fi device are connected via software though typically these are independent devices.

It is also worth noting that upon power cycling the device, the device behaved normally. Further, no drift in temperature was noted other than an increase due to computational loading.

Table 3: Testing Results

run	fluence	flux (protons/sec)	load condition	T (°C)	crash condition
1	1.19E+08	2.83E+07	max	49.50	Invalid Instruction
2	1.20E+07	2.00E+07	max	49.50	Undefined Instruction
3	3.32E+08	1.09E+06	max	46.00	network failure, no errors recorded
4	1.07E+08	7.12E+06	max	48.50	Level 3 permission fault (11)
5	1.10E+08	1.23E+06	max	50.00	Crash of Wi-Fi driver
6	3.03E+08	1.08E+06	max	50.00	Crash of Wi-Fi driver
7	3.77E+08	1.19E+06	idle	42.50	network failure, no errors recorded
8	1.11E+08	1.21E+06	idle	34.00	Crash of Wi-Fi driver
9	5.72E+07	1.24E+06	idle	33.00	Kernel paging request Error
10	8.84E+07	1.04E+06	idle	33.00	network failure, no errors recorded
11	1.58E+07	1.06E+06	idle	31.00	Kernel paging request Error
12	1.35E+08	1.04E+06	idle	32.50	network failure, no errors recorded
13	9.24E+06	1.03E+06	minimal	46.50	Level 2 translation fault (11)
14	6.12E+07	1.32E+06	minimal	46.00	Level 2 translation fault (11)
15	2.16E+08	1.04E+06	minimal	49.50	Undefined Instruction
16	7.74E+07	1.18E+06	minimal	46.50	network failure, no errors recorded

7. Appendix

Table 4: Error Identification using Memory Addresses and Memory Stack

Run	Fault Type	Program Counter	Page - Global	Page - Middle	Page - Entry
1	Invalid Instr	0020007f82654028	000000015d963003	000000016002b003	002000015b307fd3
2	Undef Instr	0000007fa89d0260			
3	Network error: Software caused connection abort				
4	L3 Permission	0000007ff903a660	00000001297ef003	000000012903b003	02e000012b0e5f53
5	wl_cfg80211_hang : In : chip crash eventing				
6	/dvs/git/dirty/git-master_linux/kernel/drivers/net/wireless/bcmdhd/dhd_sdio.c				
7	disconnected terminal client during message transmission				
8	/dvs/git/dirty/git-master_linux/kernel/drivers/net/wireless/bcmdhd/dhd_sdio.c				
9	Kernel	ffffffc000185ba4	0000000000000000	PGD NULL	PGD NULL
10	disconnected terminal client during message transmission				
11	Kernel	ffffffc00019d7e0	0000000000000000	PGD NULL	PGD NULL
12	disconnected terminal client during message transmission				
13	L2 Translation	000000000041bb28	0000000156ec6003	0000000000000000	PMD NULL
14	L2 Translation	0000000000000000	000000015f56a003	0000000000000000	PMD NULL
15	Undef Instr	ffffffc000b638b4			
16	disconnected terminal client during message transmission				

- Addresses shown are as they displayed on the terminal client. The hexadecimal digits would be displayed as reverse bits to obtain the actual bit-wise address in pairs of digits.
  i.e. AB123456 -> 563412AB
- For Program Counter (PC), the first two bits <1:0> need to be 00[7] in order for it to be aligned correctly in memory. A misalignment of the PC is a common indication of a serious error; for example, software corruption of an address.
- In the ARMv8 stack for 'Linux for Tegra', it seems as though a three level memory mapping schema is being employed. Here, the address pointers are Page Global Database (PGD), Page Middle Database (PMD) and Page Table Entry (PTE). These are considered synonymous to Level 1, Level 2 and Level 3, respectively, for the depth of the memory address lookup dictating where the fault occurs.
  - The PGD and PMD are used to *narrow down* the memory address during a table search (often referred to as table walking). Typically the left side of the *as shown* addresses is the same for the PGD and PMD. When it isn't, address corruption has occurred.
  - The PTE is considered the final page lookup (as indicated by the word entry) and should match the left hand side of PMD and part(s) of its right hand side. The validity of the actual address cannot be obtained with the information present.
- An address other than null (all zeros) must be present for the memory fetch to take place.

[7] ARM Architecture Reference Manual: ARMv8, for ARMv8-A architecture profile. Section D1.8.1: PC Alignment Checking. https://people.mozilla.org/~sstangl/arm/AArch64-Reference-Manual.pdf. Copyright © 2013 ARM Limited. All rights reserved. ARM DDI 0487A.a (ID090413)

Table 5: Bit Flip Assessment

	erroneous		expected		
Run	**hex**	**binary**	**hex**	**binary**	**# Flips**
1	2	0010	0	0000	1
2					1
3					1
4	2e	0010 1110	00	0000 0000	4
5					1
6					1
7					1
8					1
9	ff	1111 1111	00	0000 0000	8
10					1
11	ff	1111 1111	00	0000 0000	8
12					1
13	1	0001	0	0000	1
14	1	0001	0	0000	1
15	ff	1111 1111	00	0000 0000	8
16					1

- For each erroneous memory address found in the stack, a high level assessment was devised to determine the minimum number of bit flips that may have taken place. This table shows the hexadecimal and binary equivalents of each portion of the suspect memory addresses and the expected value, respectively.
    - It is worth noting that it is not possible to fully quantify the number of bit flips without having access to a complete memory mapping.
    - It is possible that a shift register became upset causing a 1-bit error to cascade through the memory address. Thus, a single bit flip is observed as multiple bit flips as in runs #9, 11 and 15.

Table 6: Radiation Data per Run

run	const.	channel	fluence	time (dec min)	time (sec)	flux (protons /sec)	min. bit flips	cross section	load	max temp (°C)
1	1.20E+08	0.990	1.19E+08	0.07	4.2	2.83E+07	1.00	8.42E-09	max	49.50
2	1.20E+08	0.100	1.20E+07	0.01	0.6	2.00E+07	1.00	8.33E-08	max	49.50
3	1.20E+08	2.770	3.32E+08	5.10	306.0	1.09E+06	1.00	3.01E-09	max	46.00
4	1.20E+08	0.890	1.07E+08	0.25	15.0	7.12E+06	4.00	3.75E-08	max	48.50
5	1.20E+08	0.914	1.10E+08	1.49	89.4	1.23E+06	1.00	9.12E-09	max	50.00
6	1.20E+08	2.529	3.03E+08	4.67	280.2	1.08E+06	1.00	3.30E-09	max	50.00
7	1.20E+08	3.143	3.77E+08	5.27	316.2	1.19E+06	1.00	2.65E-09	idle	42.50
8	1.20E+08	0.922	1.11E+08	1.53	91.8	1.21E+06	1.00	9.04E-09	idle	34.00
9	1.20E+08	0.477	5.72E+07	0.77	46.2	1.24E+06	8.00	1.40E-07	idle	33.00
10	1.20E+08	0.737	8.84E+07	1.42	85.2	1.04E+06	1.00	1.13E-08	idle	33.00
11	1.20E+08	0.132	1.58E+07	0.25	15.0	1.06E+06	8.00	5.05E-07	idle	31.00
12	1.20E+08	1.122	1.35E+08	2.16	129.6	1.04E+06	1.00	7.43E-09	idle	32.50
13	1.20E+08	0.077	9.24E+06	0.15	9.0	1.03E+06	1.00	1.08E-07	minimal	46.50
14	1.20E+08	0.510	6.12E+07	0.77	46.2	1.32E+06	1.00	1.63E-08	minimal	46.00
15	1.20E+08	1.798	2.16E+08	3.45	207.0	1.04E+06	8.00	3.71E-08	minimal	49.50
16	1.20E+08	0.645	7.74E+07	1.09	65.4	1.18E+06	1.00	1.29E-08	minimal	46.50

- Runs #1 & #2 had a higher flux than the remaining runs. This parameter was decreased by approximately 20X to permit a relatively longer run duration.
- Run #2 failed too quickly to record an accurate channel and duration. The values highlighted in yellow are assumptions.
- The minimum number of bit flips that is provided for each run is based on the memory stack analysis in Table 4. In runs where no analysis was possible, a one (1) was recorded.
    - It is assumed that at least one bit had flipped either in memory data, memory addressing, a latch, a register or in control logic in order to fail the DUT.
    - Because of this assumption, it is unreasonable to provide statistical bounds across all runs. A censored table is provided as a supplement.

Table 7: Radiation Data per Run (Censored by Stack Failures)

run	const.	channel	fluence	time (dec min)	time (sec)	flux (protons /sec)	min. bit flips	cross section	load condition	max temp (°C)
1	1.20E+08	0.990	1.19E+08	0.07	4.2	2.83E+07	1.00	8.42E-09	max	49.50
4	1.20E+08	0.890	1.07E+08	0.25	15.0	7.12E+06	4.00	3.75E-08	max	48.50
9	1.20E+08	0.477	5.72E+07	0.77	46.2	1.24E+06	8.00	1.40E-07	idle	33.00
11	1.20E+08	0.132	1.58E+07	0.25	15.0	1.06E+06	8.00	5.05E-07	idle	31.00
13	1.20E+08	0.077	9.24E+06	0.15	9.0	1.03E+06	1.00	1.08E-07	minimal	46.50
14	1.20E+08	0.510	6.12E+07	0.77	46.2	1.32E+06	1.00	1.63E-08	minimal	46.00
15	1.20E+08	1.798	2.16E+08	3.45	207.0	1.04E+06	8.00	3.71E-08	minimal	49.50

- The minimum number of bit flips that is provided for each run is based on the memory stack analysis in Table 4. In runs where no analysis was possible, this data is excluded from the table.
- This data is displayed visually on the following charts.

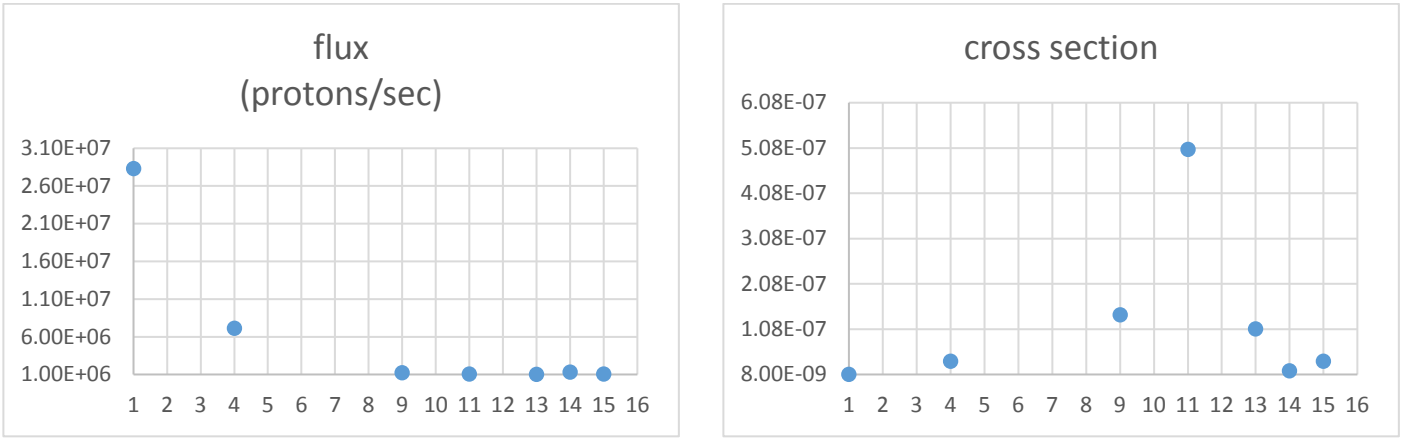
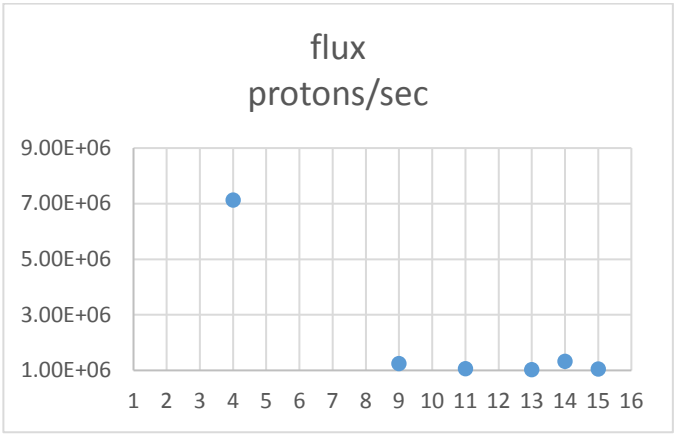


Figure 15: Flux and Cross Section of Runs with Bit Flips

Table 8: Radiation Data per Run (censored by Stack Failures and Flux Intensity)

run	const.	channel	fluence	time (dec min)	time (sec)	flux (protons /sec)	min. bit flips	Cross section	load condition	max temp (°C)
4	1.20E+08	0.890	1.07E+08	0.25	15.0	7.12E+06	4.00	3.75E-08	max	48.50
9	1.20E+08	0.477	5.72E+07	0.77	46.2	1.24E+06	8.00	1.40E-07	idle	33.00
11	1.20E+08	0.132	1.58E+07	0.25	15.0	1.06E+06	8.00	5.05E-07	idle	31.00
13	1.20E+08	0.077	9.24E+06	0.15	9.0	1.03E+06	1.00	1.08E-07	minimal	46.50
14	1.20E+08	0.510	6.12E+07	0.77	46.2	1.32E+06	1.00	1.63E-08	minimal	46.00
15	1.20E+08	1.798	2.16E+08	3.45	207.0	1.04E+06	8.00	3.71E-08	minimal	49.50

- The minimum number of bit flips that is provided for each run is based on the memory stack analysis in Table 4. In runs where no analysis was possible, this data is excluded from the table.
- Runs #1 & #2 had a higher flux than the remaining runs. Run #1 was censored from this data.
- This data is displayed visually on the following charts.



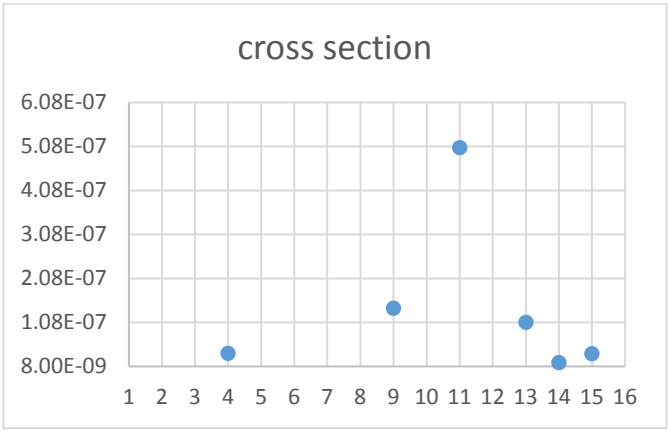


Figure 16: Flux and Cross Section of Runs with Bit Flips