DATAC Radiation Testing

Initial Meeting with Maris - 11/07/23

2 topics for meeting:

1. Info required to book
2. Main parameters for OBC, EPS + others

1.
- Provide a plan to the facilities. Maris to share presentation.
- Proton to be used
- Fluence, Flux

Test log for DEVICE X at FACILITY @DATE

The DUT is the Device X w serial nr xxxxx
The beam size is 4 cm side square
Beam size (cm2 4

Dose calculations based on PSTAR data for stopping powers for protons in Si					
Energy (MeV)	30	70	100	120	200
LET (MeVcm2 /(	14,6	7,611	5,836	5,118	3,682
Dose (rad/p)	2,35E-07	1,22E-07	9,34E-08	8,19E-08	5,89E-08

Run	Time Start	DUT #	Test Name	Test Type	Energy (MeV)	Flux (p/s)	Fluence (p)	Time (s)	Dose in Si (rad)	Actual Fluence (p)	Actual Fluence (p/cm2	SEU	SEFI	SEL	Comments
1	12/12/2022 0:00	1	test desc 1	1	200	1,00E+07	1,00E+11	10000	5891,20	3,10E+08	7,75E+07				
2	12/12/2022 0:00	1	test desc 2	2	200	1,00E+07	[illegible]	10000	5891,20	1,00E+09	2,50E+08				
						Fluence	2,00E+11	Dose	11782,40						
			5 minutes for energy change					600							
3	12/12/2022 0:00	1	test desc 1	1	70	2,00E+07	1,00E+11	5000	12177,60	5,00E+09	1,25E+09				
4	12/12/2022 0:00	1	test desc 2	2	70	2,00E+07	1,00E+11	5000	12177,60	5,00E+09	1,25E+09				
						Fluence	2,00E+11	Dose	12177,60						
			5 minutes for energy change					600							
5	12/12/2022 0:00	1	test desc 1	1	120	1,40E+07	1,00E+11	7143	8188,80	5,00E+09	1,25E+09				
6	12/12/2022 0:00	1	test desc 2	2	120	1,40E+07	1,00E+11	7143	8188,80	5,00E+09	1,25E+09				
						Fluence	2,00E+11	Dose	16377,60						
						Total fluence	6,00E+11	Total time (h)	12,63	Total dose (krad)	40,34				

https://docs.google.com/spreadsheets/d/1AoZgCZNGMFnen3oLY1BoZdAzL0INI-Oz/edit#gid=1755237176

- Start with establishing the fluence required, then the flux (estimate)
- Establish the cross-section of the area required for testing
- Energy?
- Consider change of DUT, downtime etc

Establishing the fluence:

- Choose main device (SOC, FPGA)
- Find data on it
- Establish the proton cross section (probability of having an upset, cm^2)
- Flux = proton/s/cm^2
- Board level test:
- Component level:
  - 100 events
  - cross section/fluence
- Consider OS/software

- IEEE xplore, TNS, Google Scholar, cross section
- Energies:
    - 200MeV is standard
    - Suggest going to level 35MeV
- Wobble curve: probability/energies.

- Consider component level testing. Entire board in test set-up but concentrate the beam device by device.
- Decide on what we want to “test” during the beam
    - SEFIs program stops functioning. Not nominal behaviour

Outcome of results:

- What’s the outcome of the test? A cross section that we’ve more confidence in
- Read/Write memories
- Before, write patterns
- Radiate
- Then read memory and count the upsets

During the Test:

- OBCs:
    - Memories
- Considerations for release of testing:
    - Hard remote restart
    - Soft remote restart
- SEFI
    - Catch all for non-nominal behaviours
- SEU (Single event Upset)
    - Memory bitflip
- Latchup
    - Protect with power supply current limit to avoid current surge
    - Device will hang
    - Measure voltage and currents
- Reading errors
- Flagging number of upsets
- How can we observe exactly what happened? And log it?
- Measure voltage and currents (latchups)
- Test is destructive - order more boards!

Facilities:

PSI Switzerland https://www.psi.ch/en (Cheaper)
Holland PTC Netherlands https://www.hollandptc.nl/

1000EUR per hour baseline

Oriol’s Notes

https://docs.google.com/spreadsheets/d/1AoZgCZNGMFnen3oLY1BoZdAzL0INI-Oz/edit#gid=1755237176

- Flux (1e7), Fluence and Energy is mandatory.
- Find a proton cross section (probability of having an offset) cm^2, we can estimate the Fluence
- Cross Sections = Events / Fluence
- Finding some statistics
- Include some margin + change of devices (accessing the beam, etc.)
- Energy: 200 MeV (highest), 20 MeV (lowest) - depending on the facility.
- Outcome of the test: probability vs energy (that’s why we need to try different levels of energy).
- Maris will send some standards (not ECSS) as a guideline.
- PSI: 9 by 9 cm.
- Maybe targeting device by device (we can add a bigger board but just targeting a specific device).
- Program: getting a number of offsets, lots of events will be happening. Restarting the board, checking the memory, etc.
- Bring more than 2 devices to the test (maybe 3)

Meeting with Maris 04/08/23

Points to discuss

from notes document (DATAC radiation notes)

1. What energies should we consider? Near event-threshold? Please clarify what is mean by near-event threshold with respect to the test energies
2. During the test, which events should we be monitoring for?
    a.
3. During the test, what operations/actions should the DUT be performing? (Routine ops? Specific operations susceptible to SEE? Memory Tests?)
    a.
4. How can we detect errors? (Event triggers? Comparing outputs to expected values?)
    a.
5. How can we throttle current supply if DUT becomes unstable?
    a.
6. Any example test plans that we can use as a template?
7. What inputs to DUT?
    a. Commands, clocks, resets, data inputs
    b. Utilise full capability of FPGA whilst remaining within realistic/nominal ops
8. Any advice on whether to “read-back” during or after irradiating?
9. FPGA analysis seems very complex - is it suitable to reduce this complexity by carrying out the following top level plan:
    a. OBC Test:
        i. SoC (AMD/Xilinx Zynq™ 7030 FPGA)
            1. OBC to perform nominal operations
            2. Specific tests:
                a. Memory Test (bit flips)
                b. Other?
            3. During testing, monitor upsets and record type (how is this achieved?
        ii. Watchdog (ATSAME51J20A-AF):
            1. OBC to perform nominal operations
            2. Specific tests:
                a. Memory Test (bit flips)
                b. Other?
            3. During testing, monitor upsets and record type (how is this achieved?
    b. EPS A1 Test:

Two Main Test Subjects:

1. <u>FPGA Fabric</u>
    a. LUTS
    b. CRAM
    c. BRAM
    d. other
2. <u>FSW (ARM Core)</u>
    a. Q Sorting
    b. Check Sum
    c. other

Fluence

- XS *or* **σ**= Cross-section, probability of event occurring (cm^2)
- φ = fluence (protons / cm^2)
- # events = number of events
- Cross section is calculated:
  $\sigma = \frac{\# Events}{\varphi}$
- Typical σ = $1x10^{-4}$ cm^2
- Therefore:
  $\varphi = \frac{\# Events}{\sigma} = \frac{100}{1x10-4}$ = $1x10^6$ p cm^{-2}
- **Choose fluence = $\mathbf{1x10^6}$** p cm^{-2}
- **Minimum fluence = $\mathbf{1x10^4}$** p cm^{-2}

Flux

- Φ (Flux) = protons per cm2 per second (p cm^{-2} s^{-1})
- $\Phi = \frac{\varphi}{t}$ where *t* = *time* (s)
- Assume $1x10^{-8}$ cm^2
- Design test where we mneasure but flips
- Write 1000 bits, subject DUT to radiation, read.
  Example: if 2 bits fail, σ = $\sigma = \frac{1000}{\varphi \times 2}$
- Flux should be chosen such that the DUT doesn't crash instantly
- Use low flux to allow time for an event to happen

Calculations:

- Fluence determined by # errors
- Rule: Don't go shorter than 8hrs. At least 2 days testing time required (16 hrs minimum)
- $1x10^6$ - $1x10^7$ max flux
- Choose lower energies
- See where cross-section drops off: Plot: σ/energy

- Plot Wable curve
- CREAM MC, give orbit
- Use σ from test with flux
    - **upsets/device/day (key parameter)**
    - **Use to determine “availability factor”**
    - **Used to determine if mitigation required or not (shielding, alternative devices etc)**

Parameters:

- Fluence: $1x10^6$p cm^{-2}
- Duration: 16 Hrs minimum
- Flux: $1x10^7$ p cm^{-2} s^{-1}
- Energies: 200, 150, 75, 40/30 MeV
- Events: 100
- If time too long, reduce events or choose more bits
- Choosing flux such that:
    - Testing not too long
    - No immediate errors
- Expected cross-section: $1x10^{-4}$
-

Linux:

- OS will hang
- Depends on software version
- Error reporting may occur
- Requirement to set-up “watch-dog” on CAN, UART, SSH to figure out when/how it crashed
- Automatic restarts are useful
- When device off, cross-section not to be included in calculation
- Upset = beam-stop
- OS running, health monitor ON, beam start event, important to use time stamp to know durations

Testing Tips:

- Allow time to swap devices
- 16 hrs minimum
- Test logs and time stamps are critical

Linux
SSD
- Bit flips
-

Discussion With Chris on Software Tests (29/08)

Memory Testing

- Bit Flip testing
- Collect diagnostics:
    - Reports bit-flips, (not reliable, depending how they occurred)
    - Error correction works in 100 byte blocks
    - More than 1 bit-flips across multiple blocks = issues
- Error correction corrects up to a certain number of bits but then detects but not correct more than that
- Beyond scheme, not sure if being corrected or not (lack of info prior to event)
- Robustness of correction is proved mathematically
    - For certain scheme, how many corruptions can it tolerate? (between memory refreshes)
    - DRAM = capacitors store 1s and 0s depending on state of charge. Cycles read-write.
    - Error correction on re-fresh
    - Assessing likelihood of errors between memory refreshes
- General rule:
    - Don't embed error correction inside of OS
    - Program executed lives in RAM, if RAM corrupted, program corrupted, can't trust it
    - Impossible for programme to assess its own corruption
    - ECC works on hardware level (more robust, moving trust from RAM to hardware)

SSD

- Some level of error correction
- SSD:
    - Block level read-write tests
    - Scan end-to-end reading and writing until error
    - Treat contents of SSD (blocks of data), fill each block with data, product of calculator unique for each memory block
    - Eg No. 4, run calc, expect data, read data from block 4, compare to expected
    - Can locate blocks with errors
    - Fill with predictable data, radiate, read data. Takes 20 minutes (read-write)
    - External system that talks to OBC, that runs all tasks from outside of OBC
        - GSE required
        - Don't rely on internal OBC monitoring, anything can go wrong

        - Ethernet to External PC, PC tells OBC what to do, stress tests, routines, scripts etc
        - What resource/support required
    - Development:
        - Set up PC to continuously test
        - Gather data during test, as uch as fast as possible
        - Gather data about during which ops, errors reported
        - Check that ethernet interfaces correct, check watchdog, e-can
        - All interfaces need harnessing out
        - External PC requires set-up, software, harnessing
        - Spacwire?
    - SSD functional lifespan will be reduced, should not be for space afterward.
    - Read/Write lifetime will be stressed
    - SSD NAND https://www.swissbit.com/en/products/nand-flash-products/managed-nand/#pcie-bga
    - Knowing limits of hardware is more valuable that writing robust software
-

OS (Linux)

Watchdog

Questions/Concerns:

- Assess single event effects
- Decide system or component level

Further Discussion 31/08/23

Notes:

OBC & EPS:

2 alleys:

- Test with Flight Software (if mature)
- Write specific tests for some of the parts
  - Memories
  - Processor
  - Caches
- Not mutually exclusive - best to perform both
- Specific tests dependant effort:
  - FSW is mature, FDIR part relatively new, has progress been made, or more mature FDIR?
    - Not- Comprehensive FDIR, what's frame of reference?
      - Relating to MANTIS
        - Automatic resets due to radiation, scheduled resets etc not implemented
        - We don't routinely reset for purpose of refreshing the memories
        - For mission similar to MANTIS (MENUT), functionality not required, maintains good uptime.
    - FDIR is present, watchdog, that runs code that is constantly pinging to check health status. If no response, it controls the power lines and resets OBC if necessary
    - Detecting SEU and resetting, less functionality here. Not implemented, not sure how to implement…
  - Use testing to help better the FDIR detection
    - Chris to think how we would approach this
- System level:
  - Monitor as many parameters as possible of the system
  - Count hangs (SEFI)
  - IN parallel, monitor memories and registers for SEU
- Specific memory test:
  - SSd, SD, eMMMC
  - Don't have to write full memory
  - Write specific pattern
  - Extrapolate
  - Has to be predictable pattern
  - Same for RAM?
    - What are we able to monitor?
      - Focus on what is possible,
- Running application
  - Monitor registers on device
  - Monitor caches

        - Linux safety feature (RAS)
- Chris proposing simple = one set of software
    - Is it normal to have specific software for memory testing alone?
        - Fill RAM, then read back?
    - Small working footprint, doesn't need memory
- Gianluca:
    - Using standard software, wrote simple program
    - Equip software to give visibility on all the components
- Nicola:
    - Software requires counting, address of error (stack bit or random error)
    - Visibility of error is important
- For SEU testing:
    - Record time, address, value (0 or 1), expected value
- Nicola suggests component level testing
- Chris would like to understand the failure modes of COTS parts
    - UNderstanding ways that it can fail can help with mitigation, is this a secondary goal?
- Maris commenting that FSW will change
    - Good to extrapolate test results:
        - See how software deals with errors
        - If write more data, I'll get x amount
    - Good to see failure patterns, then asses impact and how to mitigate (what approach)
- Try to compile a list of what we can monitor on-board
- Proton level testing is very fast (too fast)
- Application test will fail quite often:
    - Make sure we automate the set-up. If hangs, it resets itself (can't waste time resetting it ourselves)
- Chris: Is it a known value how much more intense proton testing is, compared to space environment? (Acceleration coefficient…)
    - This is determined by *flux* value
- Proton testing is volatile, can't run and leave, need a reactive test, robust testing, monitor everything
- Nicola: Tests at board level for entire board, can you reduce flux to give software time to reset:
    - Minimum flux (to keep beam alive), depends on facility
    - Maris recommending PSI
    - Nicola recommends
- 48hrs total time (friday 11pm - Monday 6am)
    - 16hrs testing time (2x8hrs slots) (starts late at night)
    - Plan on using beam continuously, have multiple people, 2 minimum
- Nicola: Should we be concerned about reflections?
    - Maris: Take into account if one component is being targeted, another may fail
- Chris: NOR Flash rely on it being robust
    - Maris: Block write, read will slowly fail due to accumulated dose
    - Annealing after removing from beam
    - If the device fails during the test, we won't have time to recover it. Have spare
- Maris:

    - Select number of errors (100)
    - Don't radiate longer than needed
    - Assume a certain margin
    - Quick test = seconds (if lucky) minutes (if very lucky)
- Chris:
    - Can we turn the beam on and off? Yes
    - Can we just let the system run for a period of time, beam off then check for errors? Yes (good test for memories)
    - eTAC (eDAC?) continuously read memory
    - Is there much point doing known quantity read and write for SD cards? Maris: this is what we want to monitor - bad blocks, anything that went wrong.
    - If the flash controller is doing bad block detection and re-mapping, writing and reading a known quantity may not be valid as it's already been fixed. We need to think about how to *see* the errors before the detection.
    - Information about what manger/FDIR is doing is useful
- Are events that have been corrected still accounted for in the cross-section calculation?
- Look for way to disable FDIR or learn what it corrected
- Need to characterise the cross-section for our environment
- We do component level tests using a built OBC, just monitor other components for failures at the same time. Scattering expected.
- Need to consider all of this for the EPS as well…

Next Steps:

- Think of tests we want to perform, based on what we can observe
    - Memories are easy
    - Signals
    - Caches (find way to monitor)
- Take at least two devices and spare SD cards
-

Goal of meeting:

Gain understanding of overall tests to be performed so that we're aware of what is required as a development.