Xilinx Kintex-UltraScale Field Programmable Gate Array Single Event Effects (SEE) Heavy-ion Test Report

NASA Electronic Parts and Packaging (NEPP)

Melanie Berg – Principal Investigator: AS&D
Hak Kim, Anthony Phan, Christina Seidleck: AS&D
Ken Label: NASA/GSFC
Michael Campola: NASA/GSFC

Test Dates: 10/2016TAMU and 3/2017TAMU

Revision Number	Submission Date	Comment
KintexUltraScale_test_report_rev0	06/2017	Original submission
KintexUltraScale_test_report_rev0	09/2017	Update lot date code

Table of Contents

1. INTRODUCTION

This is an independent investigation that evaluates the single event destructive and transient susceptibility of the Xilinx Kintex-UltraScale device. Design/Device susceptibility is determined by monitoring the device under test (DUT) for Single Event Transient (SET) and Single Event Upset (SEU) induced faults by exposing the DUT to a heavy ion beam. Potential Single Event Latch-up (SEL) is monitored throughout heavy-ion testing by examining device current. This device does not have embedded mitigation. Hence, user implemented mitigation is investigated using Synopsys mitigation tools.

The objectives of this study are the following:

- Analyze flip-flop (DFF) behavior in simple designs such as shift registers. Compare SEU behavior to more complex designs such as counters. Evaluate the data trends. This analysis will help in extrapolating test data to actual design.
- Analyze global route behavior – clocks, resets.
- Analyze mitigation insertion schemes (various triple modular redundancy (TMR) schemes).
- Study potential SEL and subsequent destructive events.

2. BACKGROUND

Testing an SRAM based FPGA requires the user to test SEU behavior in the following device components: Configuration hardness, data path susceptibility, global structures and hidden logic (Single Event Functional Interrupt (SEFI)) susceptibility.

2.1 Cross Sections

SEU Cross Sections (σ_{seu}) characterize how many upsets will occur based on the number of ionizing particles to which the device is exposed. Generally a σ_{seu} is calculated by counting the number of observed upsets during irradiation and normalizing it (dividing by) the particle fluence:

General Terminology for heavy-ion testing:

- Flux: Particles/(sec-cm^2)
- Fluence: Particles/cm^2

$$\sigma_{SEU} = \frac{\text{Number of Upsets}}{\text{Fluence}} \quad (1)$$

σ_{seu}s are calculated at several linear energy transfer (LET) values (representing a particle spectrum) as in equation (1). Testing with a low flux is imperative with the Xilinx SRAM-based FPGAs due to the complexity of the device versus the accelerated rate of exposure.

2.2 Divide and Conquer Test Approach

An FPGA is a complex device with a variety of components containing varying SEU susceptibilities. Accordingly, tests are conducted such that components are isolated in order to develop an understanding of each of the component's SEU error responses. NEPP categorizes FPGA components and their susceptibilities as follows:

$P(fs)_{error}$: system or design σ_{seu}

$P_{configuration}$: Configuration σ_{seu}

$P(fs)_{functionalLogic}$: data path σ_{seu}

P_{SEFI}: hidden logic and global route σ_{seu}s

All σ_{SEU}s are calculated independently. It is important to note that configuration upsets (if the configuration bit is used by the design and enabled) will cause an upset in the functional operation. However, we denote this type of upset as a configuration upset not a functional logic upset. Functional logic upsets are categorized as malfunction that occurs due to SEUs affecting DFFs or combinatorial logic elements in the programmable user space.

$$P(fs)_{error} \propto P_{Configuration} + P(fs)_{functionalLogic} + P_{SEFI} \quad (2)$$

2.3 Configuration Test and Analysis

Because the configuration is static, static SEE tests are generally performed. This requires the following steps during testing:

- Configure the DUT
- Place the DUT in a radiation environment (e.g. particle accelerator),
- Remove the radiation environment (e.g. stop the accelerator beam),
- Read-back the configuration and count the number of bits that are upset.
- Read-back includes configuration latches and embedded memory cells (BRAM). Because the Virtex-5 has radiation hardened configuration latches and unhardened BRAM, special care must be taken to differentiate between the two when counting upset events.

$$\sigma_{seu} = \frac{\#bit\_errors}{(fluence) * \#ConfigurationBits} \quad (3)$$

$$\sigma_{seu} = \frac{\#bit\_errors}{(fluence) * \#BRAMBits} \quad (4)$$

Tests that are targeted for configuration susceptibility (i.e., $P_{configuration}$), are static and do not use scrubbing. $P_{configuration}$ is calculated using equation (3).

BRAM tests are part of configuration testing and hence are also static tests– BRAM information is obtained during the configuration readback process. P_{BRAM} is calculated using equation (4). Such tests only explore the susceptibility of BRAM bits. However, there is additional logic utilized when BRAM is used within a design. Additional (dynamic) testing must be performed to determine the error signatures and design σ_{seu}s for a more accurate characterization of BRAM usage.

Configuration upsets may or may not cause a system level error. If the configuration bit, that has changed state, is being used by the design and is in an enabled path of logic, then there is a chance that the changed state will cause system malfunction. Dynamic testing is required to investigate configuration upsets and their impact on system level functionality; i.e., this is not considered a configuration test – it is considered a dynamic functional test. In general, Configuration SEUs that directly affect active circuitry have error signatures that resemble temporary stuck faults. Although the configuration bit can be corrected (e.g. via scrubbing), the state of the impacted circuit will not be corrected. It is up to the system to correct the state of behavior after all configuration SEUs are corrected/scrubbed.

2.3.1 Dynamic Test and Analysis

Dynamic testing concentrates on evaluating the SEU susceptibility of the design under analysis (DUA) during accelerated radiation testing. Such testing requires the DUT to be operating and its outputs to be monitored during irradiation. Deviations from expected values are reported and noted as error responses.

This is a challenging study because, as previously mentioned, the complexity of the FPGA devices has various components with different error signatures and susceptibilities. The component contribution to overall design σ_{seu}s was given in equation (2). If tests are not constructed to isolate these components during the test and analysis phase, radiation data can be convoluted and difficult to process.

3. DEVICES TESTED

3.1 Device Specifics

Xilinx UltraScale FPGAs are high performance devices. They are fabricated on a high-k metal gate (HKMG) TSMC 20nm planar HPL (high performance low power) process. The DUT used in this investigation is the **XCKU040-2FFVA1156 with lot date code 1509.** Figure 1 shows a DUT mounted to a test-system daughter board.

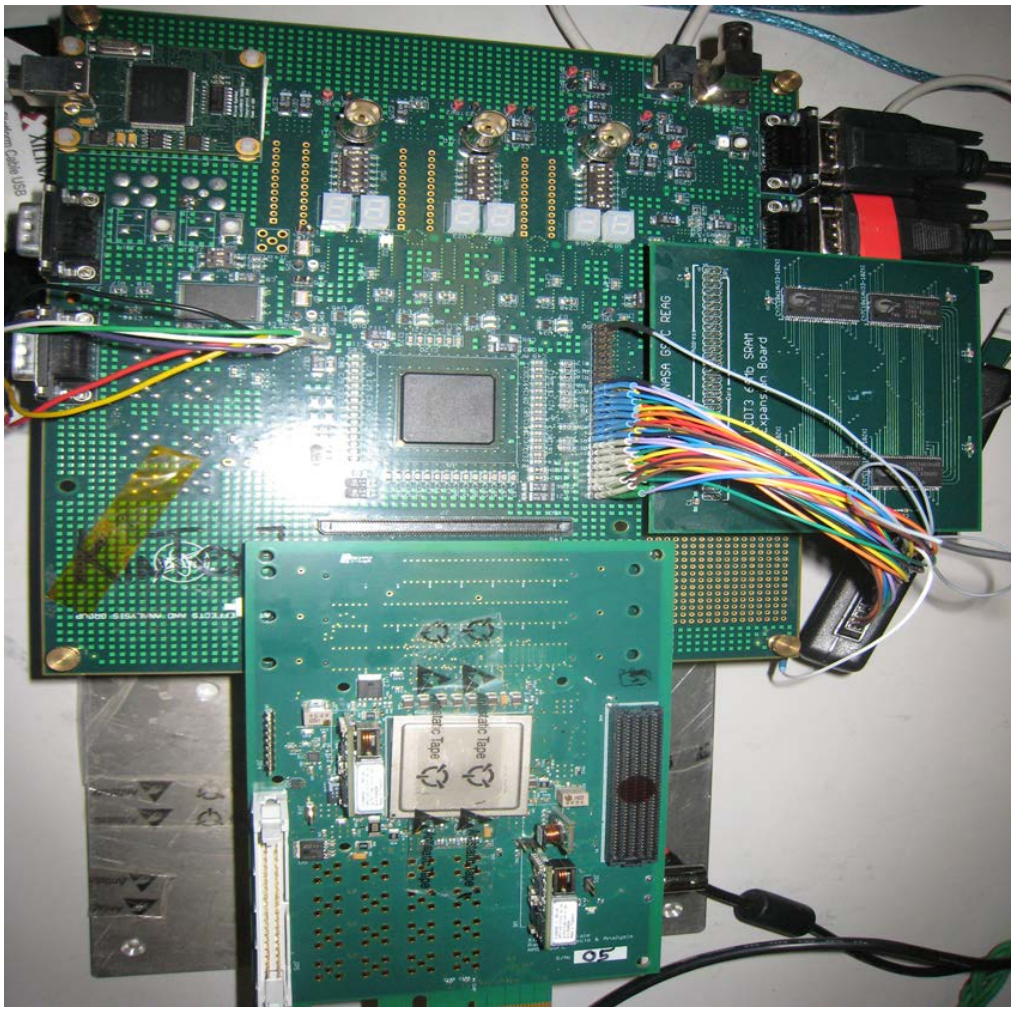


Figure 1: Mounted Xilinx Kintex UltraScale on the NEPP test-system daughter card.

The UltraScale family is the next generation of FPGA devices following the commercial Xilinx-7 series. I/O interfaces are significantly more robust. There is no embedded mitigation. However, additional gate-count better allows the user to insert mitigation into the design. There is no embedded processor. However, the user can embed a soft-core.

Table 1 lists the features of the Kintex-Ultrascale **XCKU040-2FFVA1156** DUT.

Table 1: General **XCKU040-1LFFVA1156I** Features

	Kintex-Ultrascale		**Virtex UltraScale**	
Type	GTH	GTY	GTH	GTY
Quantity	16-64	0-32	20-60	0-60
Maximum Data Rate	16.3Gb/s	16.3Gb/s	16.3Gb/s	30.5Gb/s
Minimum Data Rate	0.5Gb/s	0.5Gb/s	0.5Gb/s	0.5Gb/s

Key Applications	Backplane PCIe Gen4 HMC	Backplane PCIe Gen4 HMC	Backplane PCIe Gen4 HMC	100G+Optics Chip-to-Chip 25G+ Backplane HMC

3.2 NASA DUT Preparation

NASA has five populated boards with **XCKU040-2FFVA1156**. The parts (DUTs) were thinned using mechanical etching via an Ultra Tec ASAP-1 device preparation system (illustrated in Figure 2). The Ultra Tec ASAP-1 system is a proven process used successfully on several previous generation Xilinx parts. The thickness goal is 120 ums or below.

Figure 2: Ultra Tec ASAP-1 device preparation system

4. TEST HARDWARE (TEST SYSTEM)

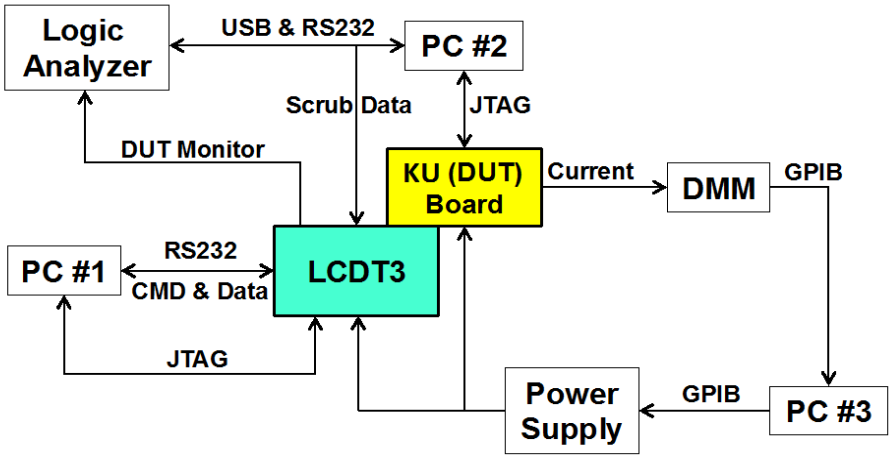


Figure 3: Low Cost Digital Tester 3 (LCDT) & Kintex-UltraScale Daughter Board

Figure 3 is an illustration of the NASA NEPP Kintex-UltraScale test system. It uses the NEPP low

cost digital tester (LCDT) and a daughter card with a mounted Kintex-UltraScale (Kintex-UltraScale (DUT) board). PC's and logic analyzers are used for control and monitoring of the test system. The following list provides an overview of test components (with respect to Figure 3) and their role within the test system:

- LCDT3
  - Handle All Kintex-UltraScale Operation Modes and Execution Routines.
  - Collect All Data from Kintex-UltraScale Board, analyze data and Report the results to PC #1.
- PC #1
  - Configure LCDT3 via JTAG (Joint Test Action Group).
  - Send Commands LCDT via RS232.
  - Receive Data from LCDT via RS232.
- PC #2
  - Configure Kintex-UltraScale via JTAG.
  - Readback Kintex-UltraScale configuration data after irradiation.
  - Send Kintex-UltraScale configuration data for DUT configuration scrubbing via USB & RS232.
  - Run and display logic analyzer capture via USB.
- PC #3
  - Control DC Power Supply via GPIB.
  - Collect current readings from DMM via GPIB.
- Logic Analyzer
  - Monitor Kintex-UltraScale operation status.
- Power Supply
  - Provide power to both LCDT3 & Kintex-UltraScale b.
- DMM
  - Scan Kintex-UltraScale supply current measurement
  - VCCINT, VCCO, VCCAUX, VCCMGT , VTxRx
- Kintex-UltraScale DUT
  - Although there are various components on this board (as illustrated in Figure 4), only the mounted Kintex-UltraScale device is subjected to the heavy-ion beam

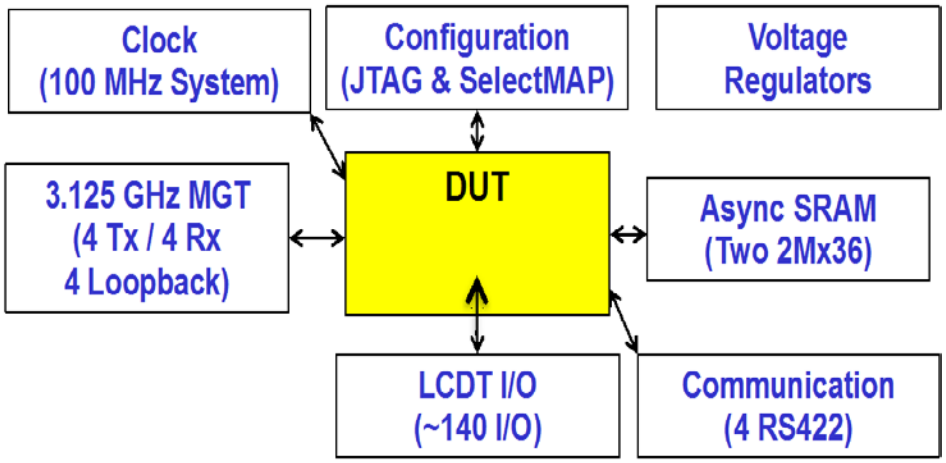


Figure 4: Daughter Board with mounted DUT

4.1 DUT Board Active Components and Schematics

Table 2 lists the active components on the daughter board.

Table 2: Daughter Board Active Component List

Part	Manufacturer	Manufacturer's Part Number
XCKU040-2FFVA1156	Xilinx	**XCKU040-2FFVA1156**
LD49300PT10R	STMicroelectronics	LD49300PT10R
SN74LVC1T45DBVR	Texas Instruments	SN74LVC1T45DBVR
SI53306	Silicon Labs	SI53306
LD39300PT18R	STMicroelectronics	LD39300PT18R
LD39300PT25R	STMicroelectronics	LD39300PT25R
D12S05020	Delta Electronics	D12S05020
PTR08060W	Texas Instruments	PTR08060WVD
LD39300PT-R	STMicroelectronics	LD39300PT-R
LD49300PT12R	STMicroelectronics	LD49300PT12R
SG3225VAN200.000000M	EPSON	SG3225VAN200.000000M
ASEMPLV-156.250MHZ	Abracon LLC	ASEMPLV-156.250MHZ

The following figures (Figure 5 through Figure 15) are the DUT schematics.

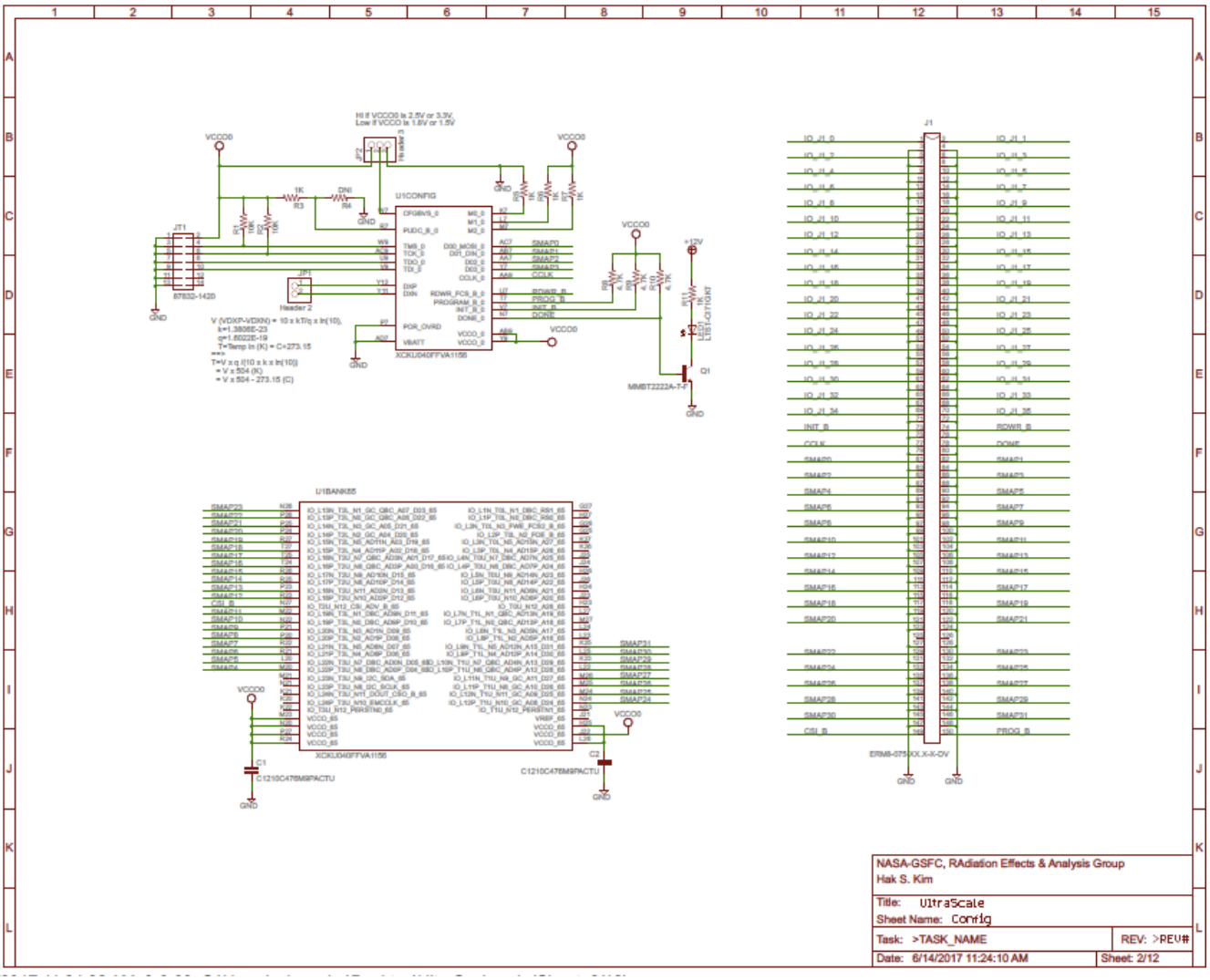


Figure 5: DUT Schematic Sheet 2

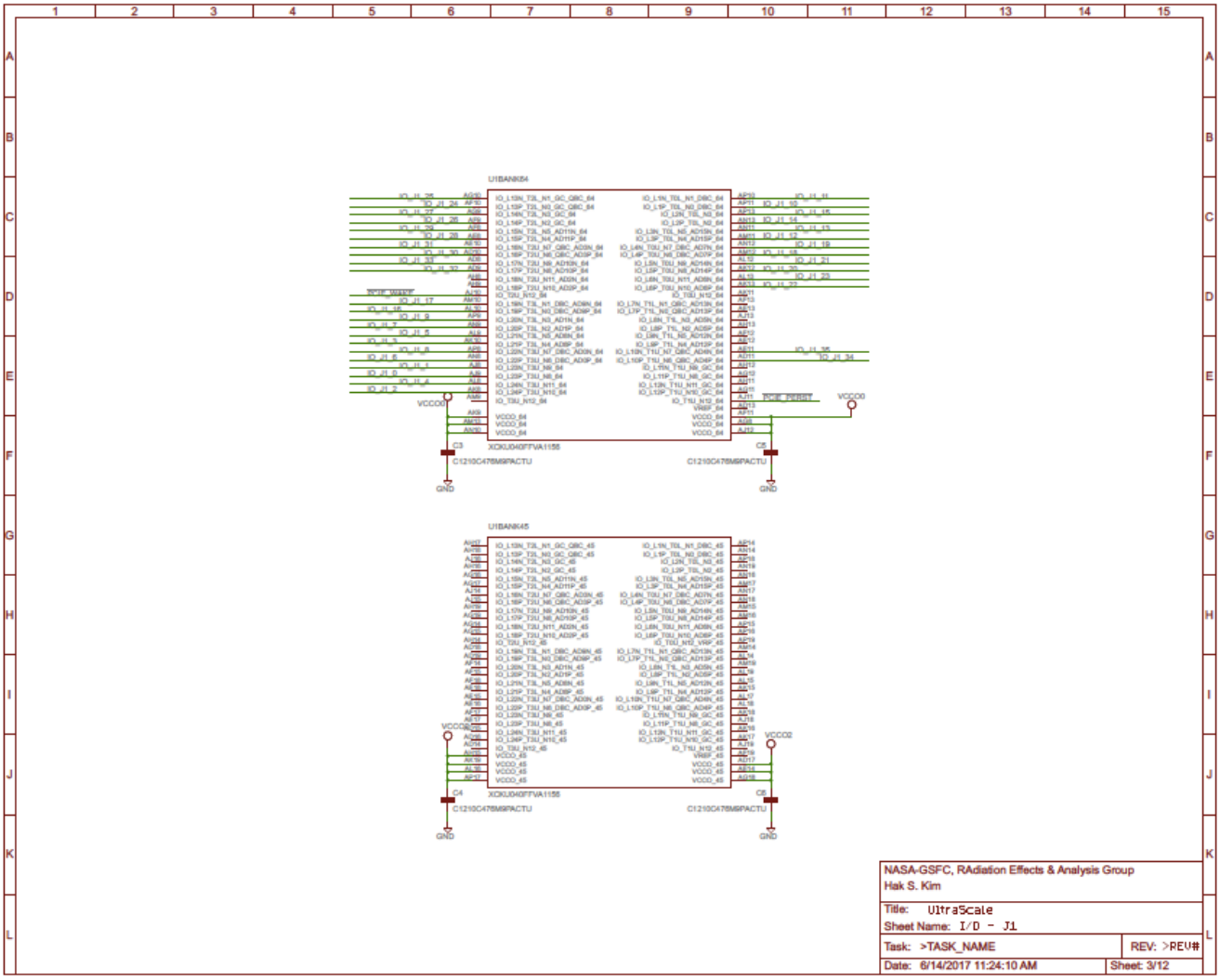


Figure 6: DUT Schematic Sheet 3

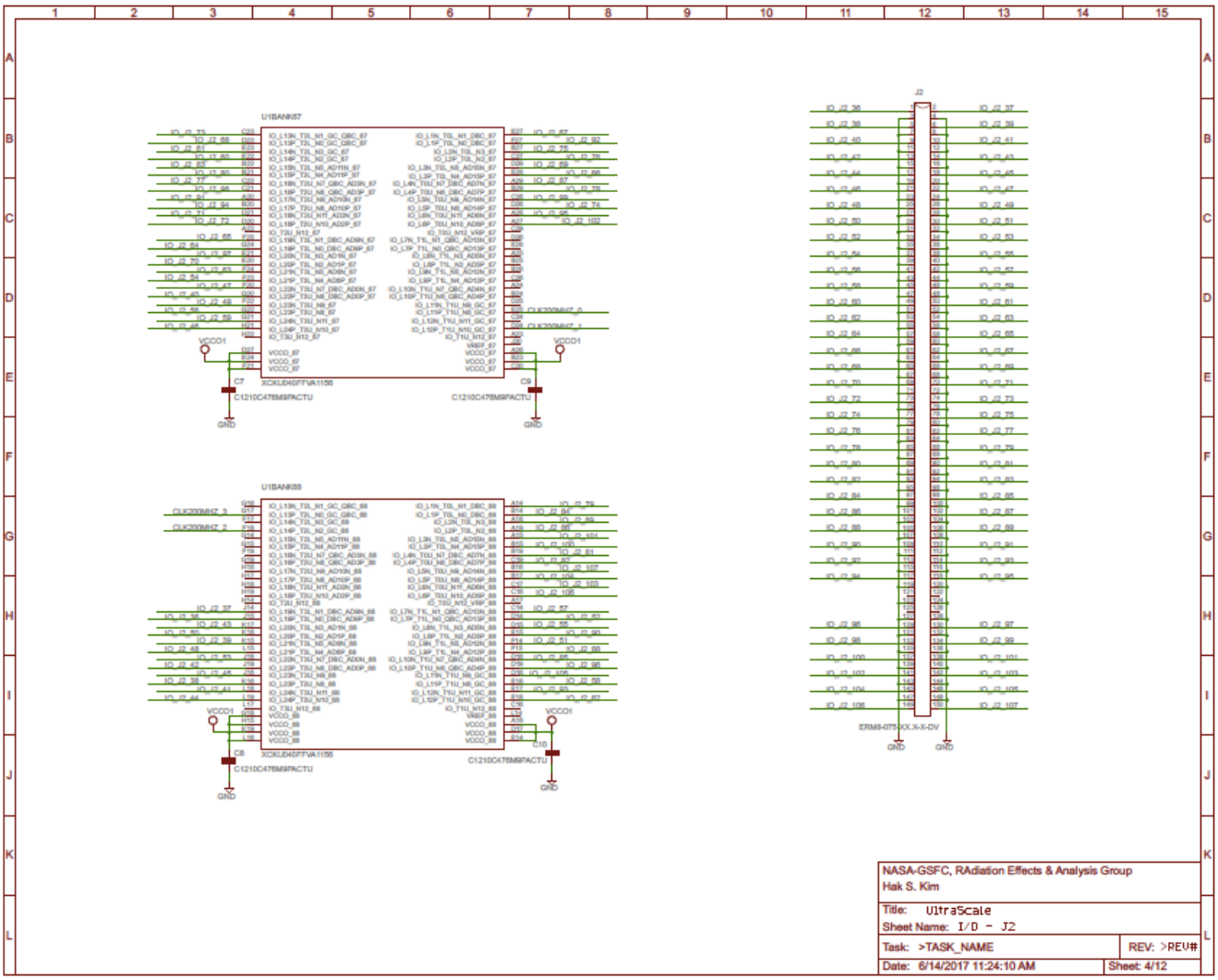


Figure 7: DUT Schematic Sheet 4

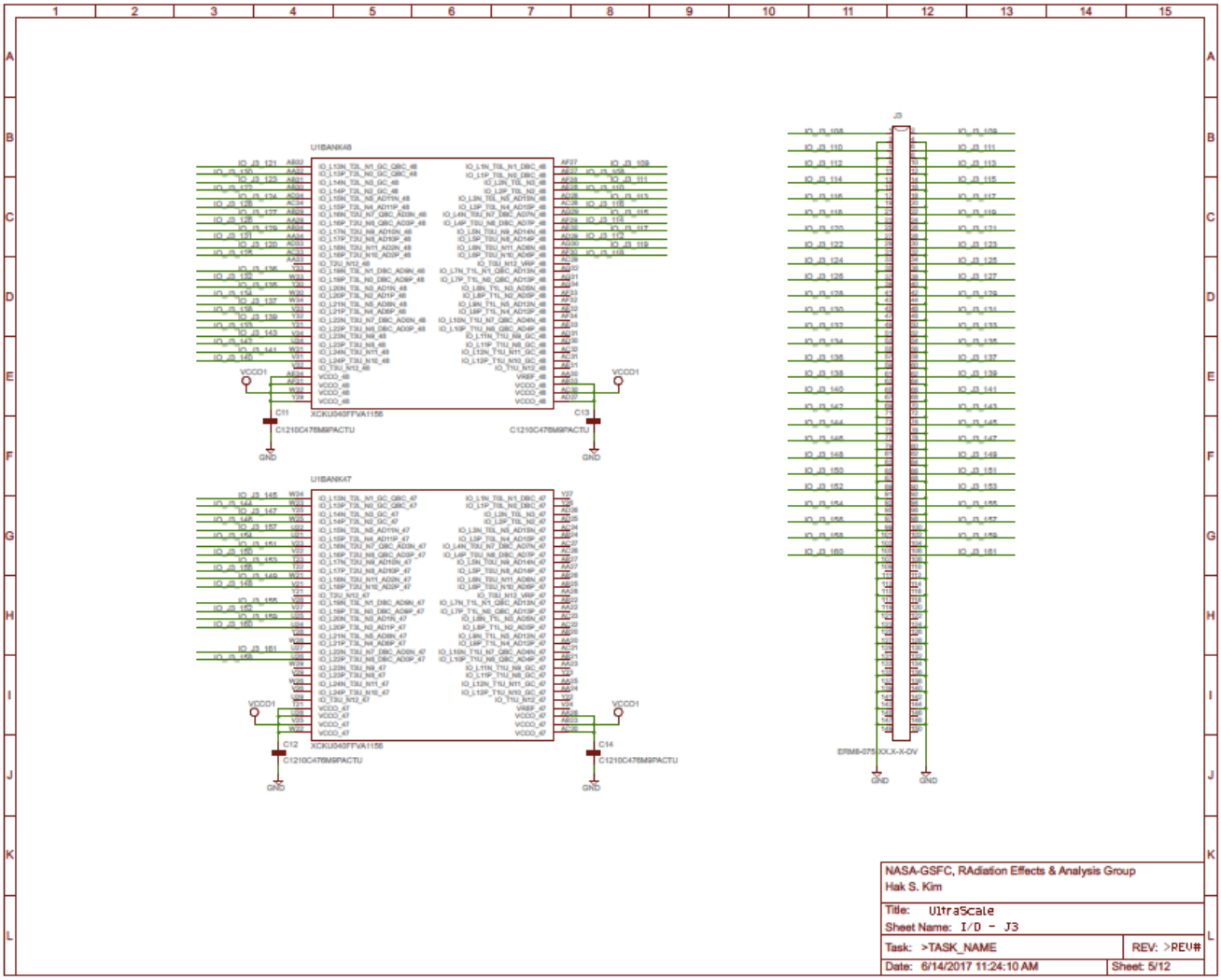


Figure 8: DUT Schematic Sheet 5

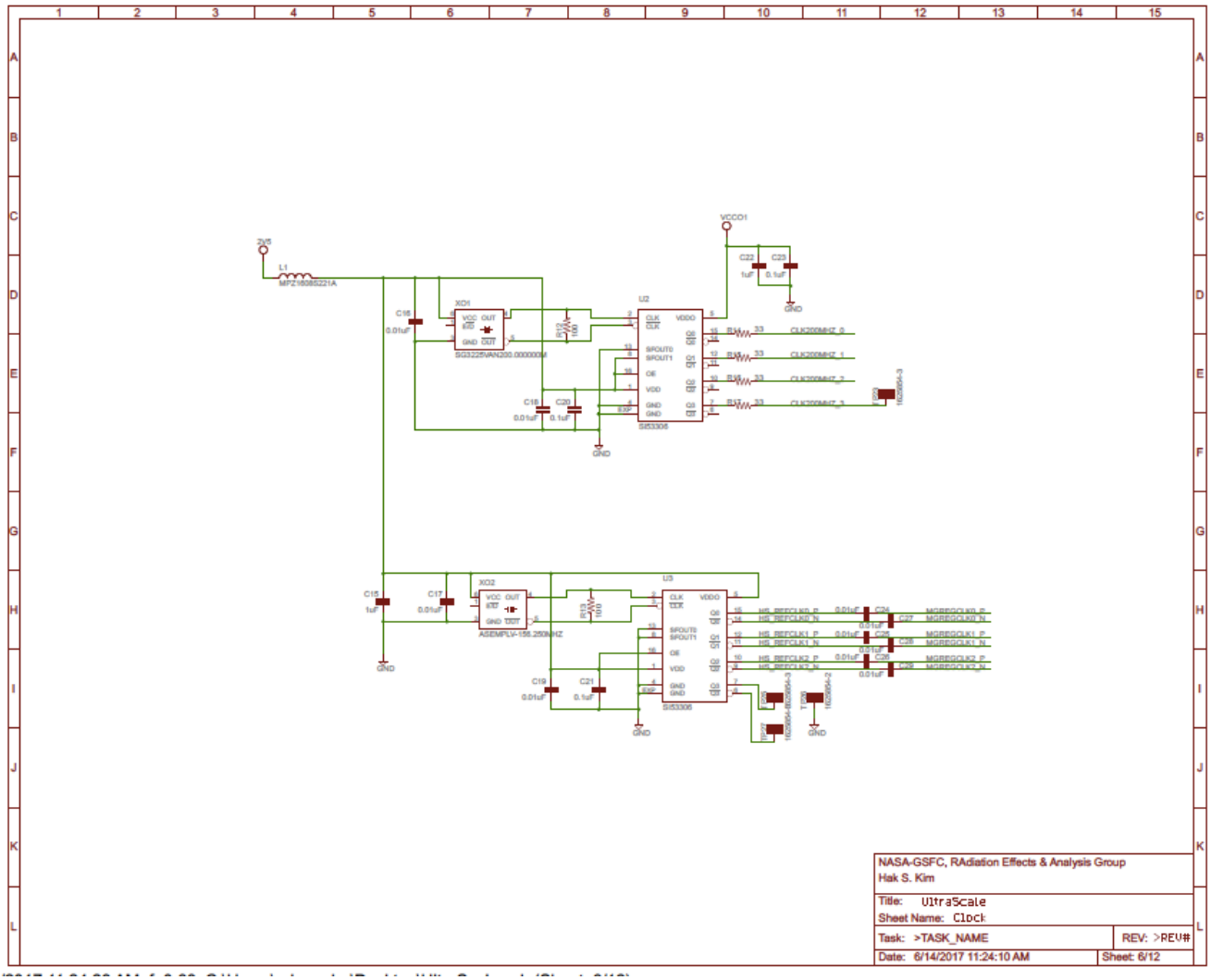


Figure 9: DUT Schematic Sheet 6

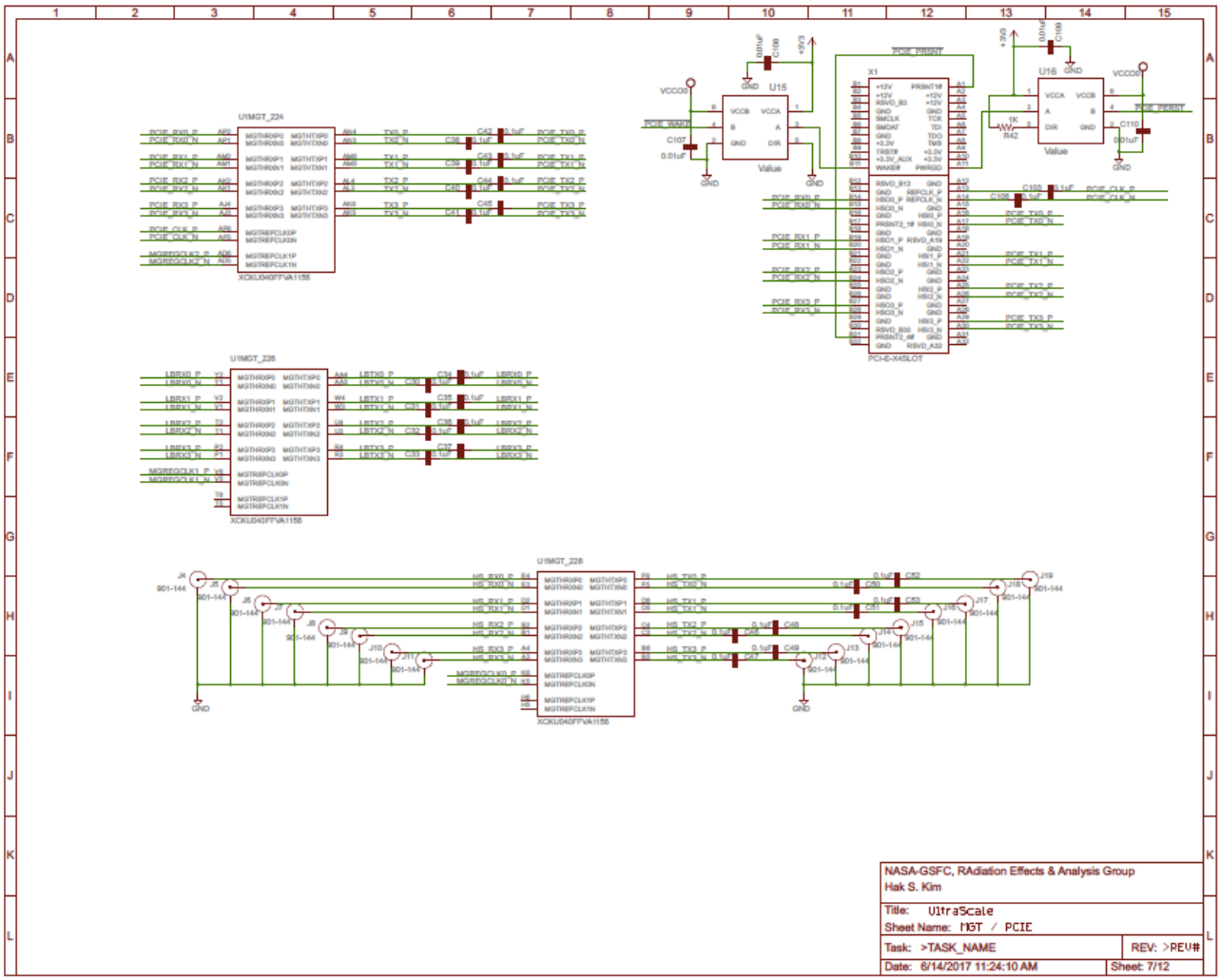


Figure 10: DUT Schematic Sheet 7

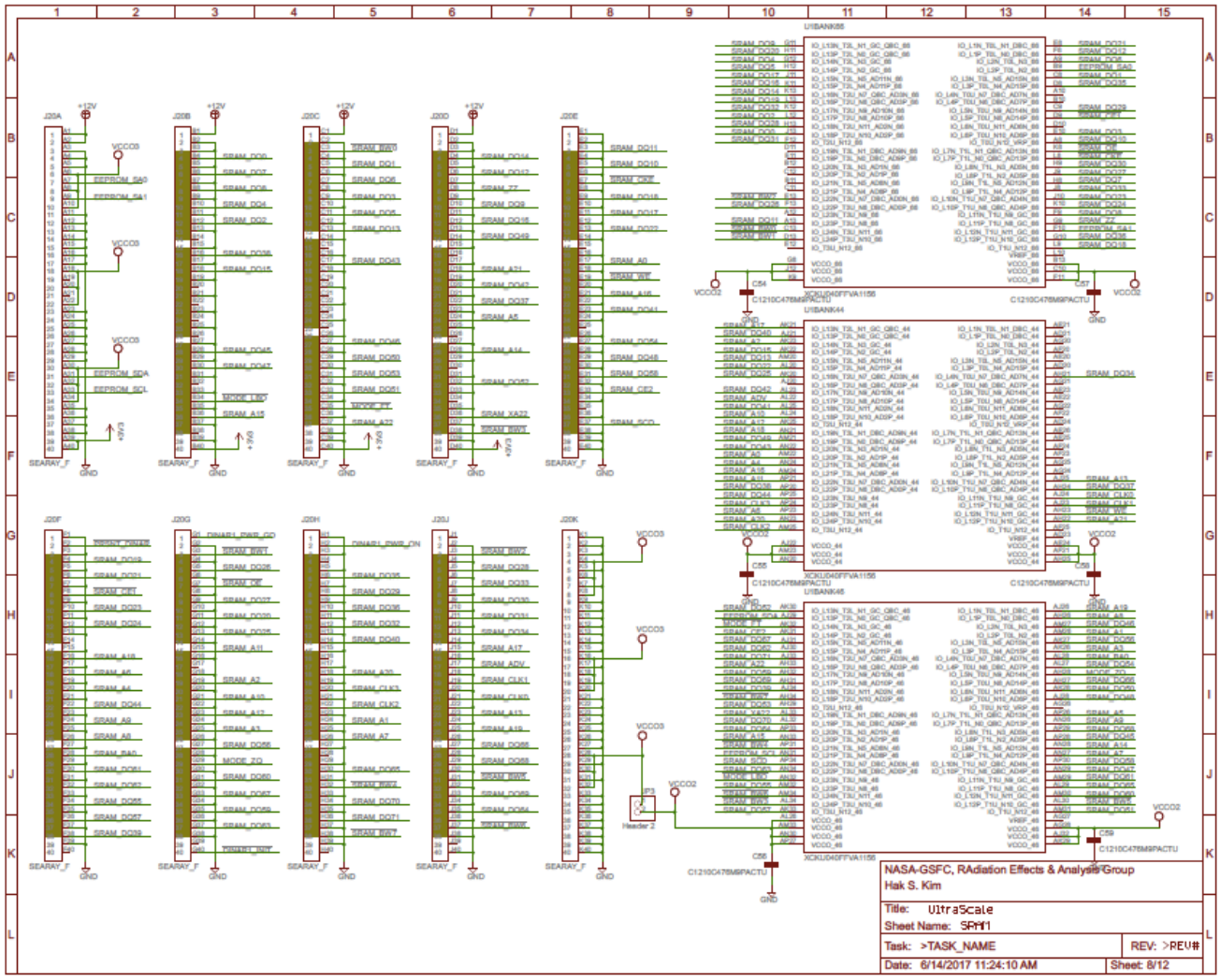


Figure 11: DUT Schematic Sheet 8

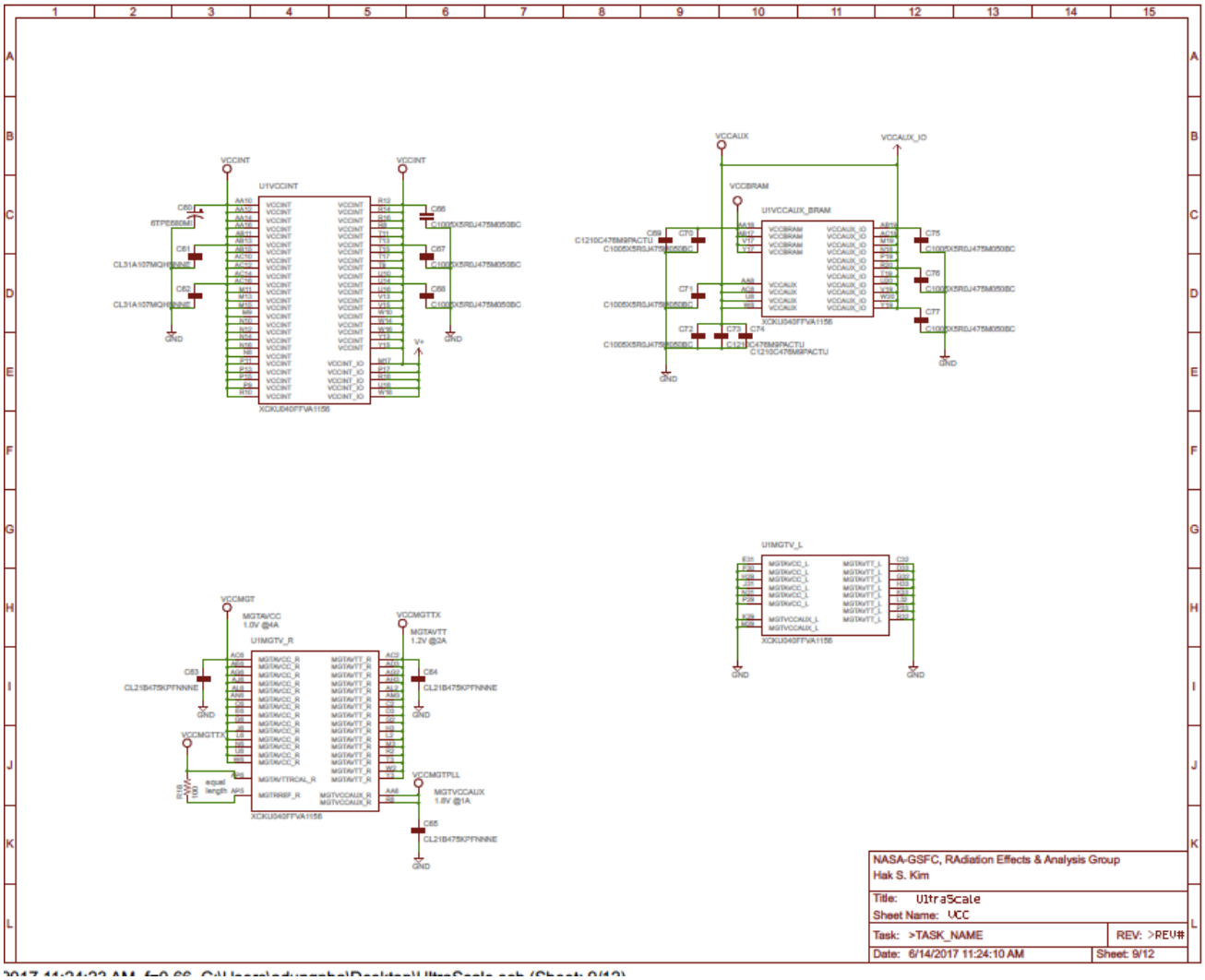


Figure 12: DUT Schematic Sheet 9

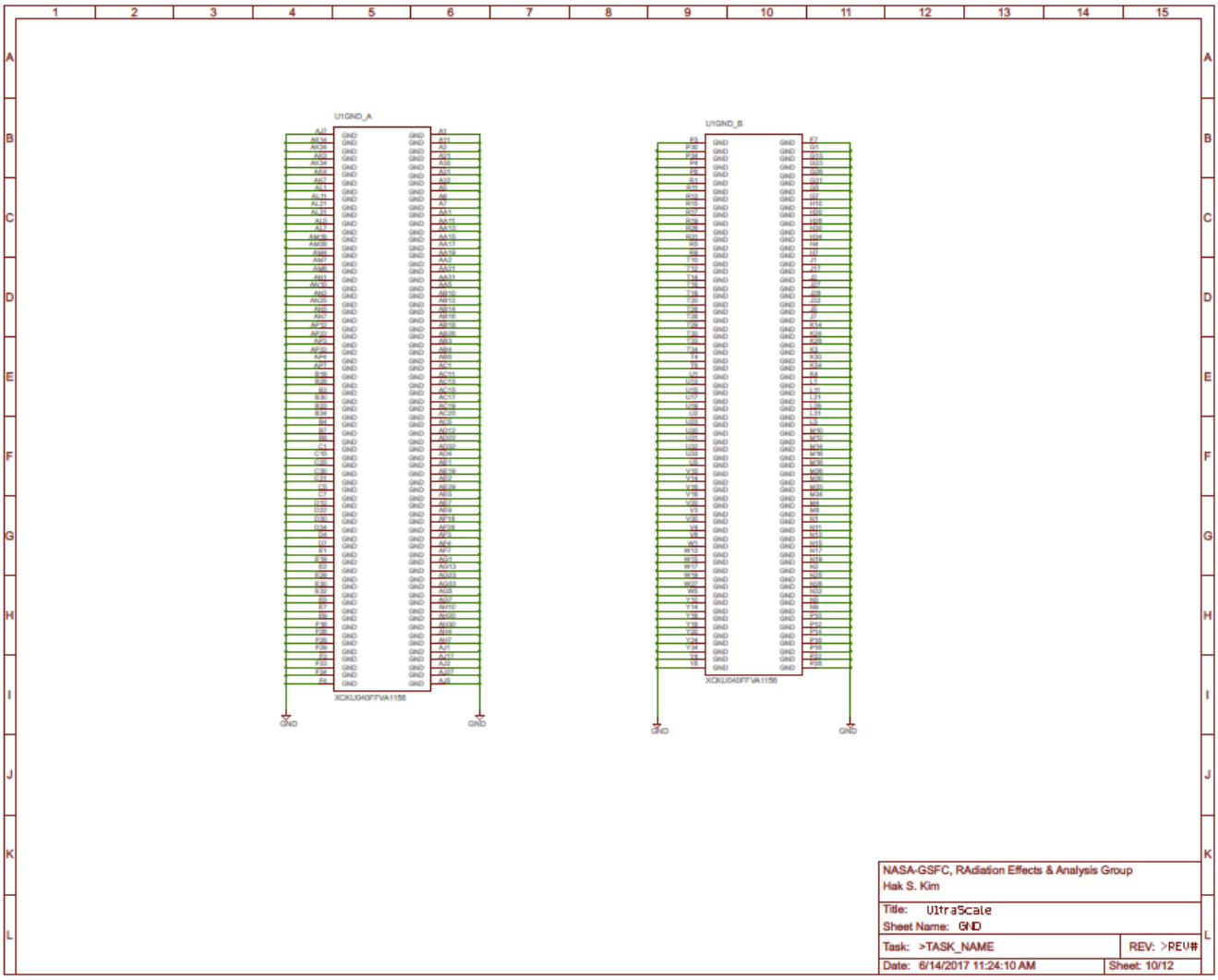


Figure 13: DUT Schematic Sheet 10

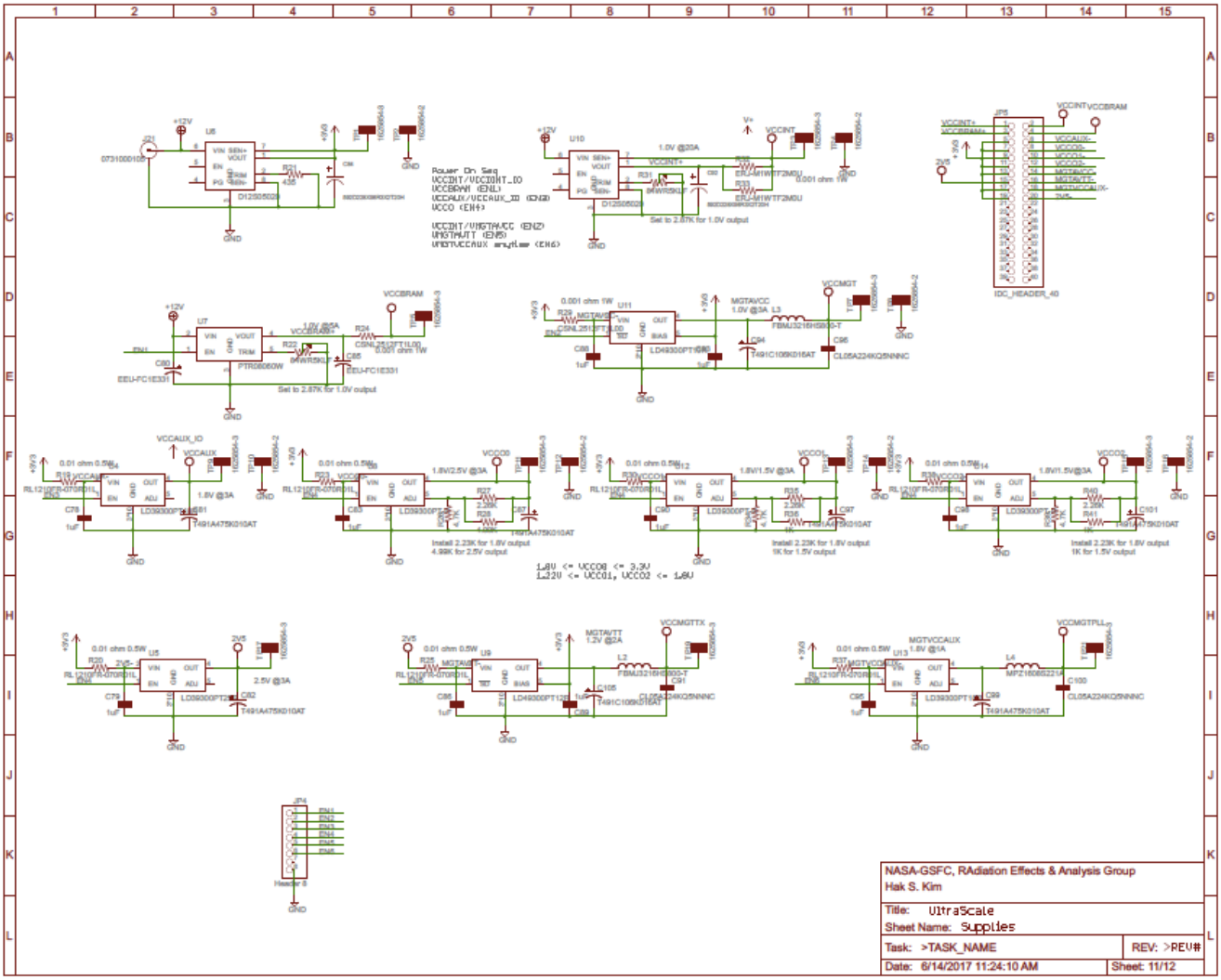


Figure 14: DUT Schematic Sheet 11

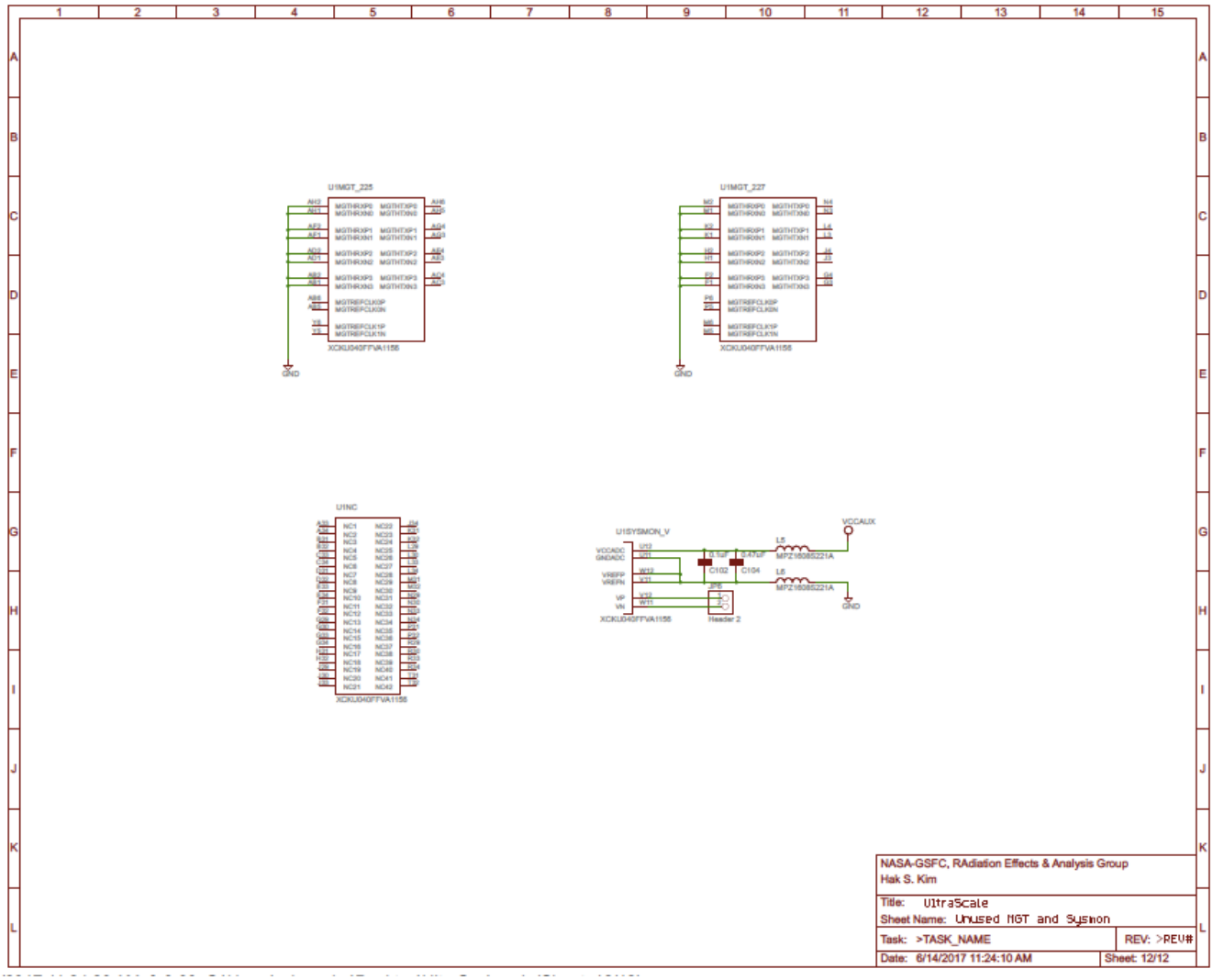


Figure 15: DUT Schematic Sheet 12

4.2 Pictures of the Test System

Figure 16 and Figure 17 are pictures of the Kintex-UltraScale populated DUT board connected to the LCDT tester.

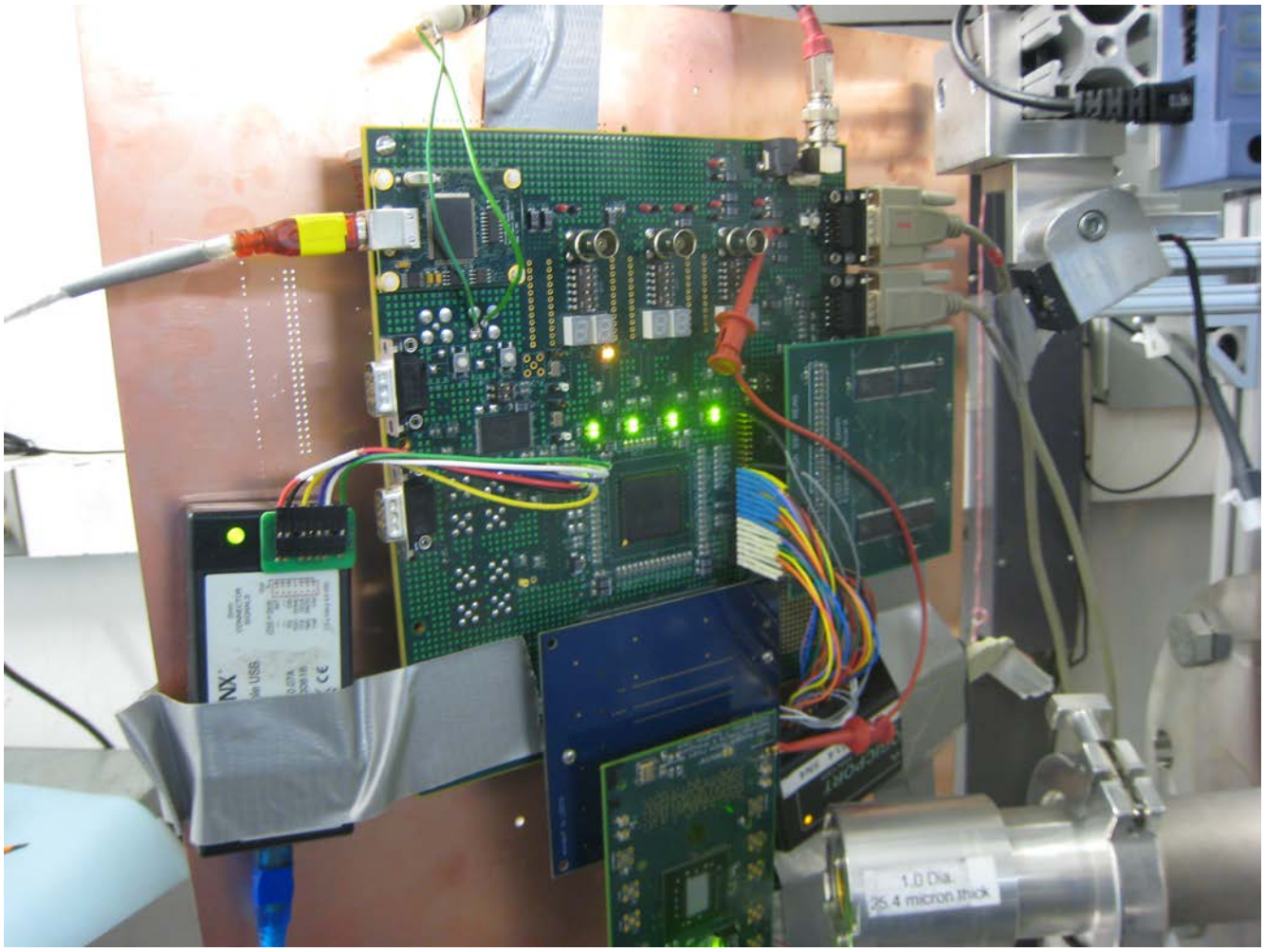


Figure 16: Test Setup - Connected LCDT3 and Daughter Card (1)

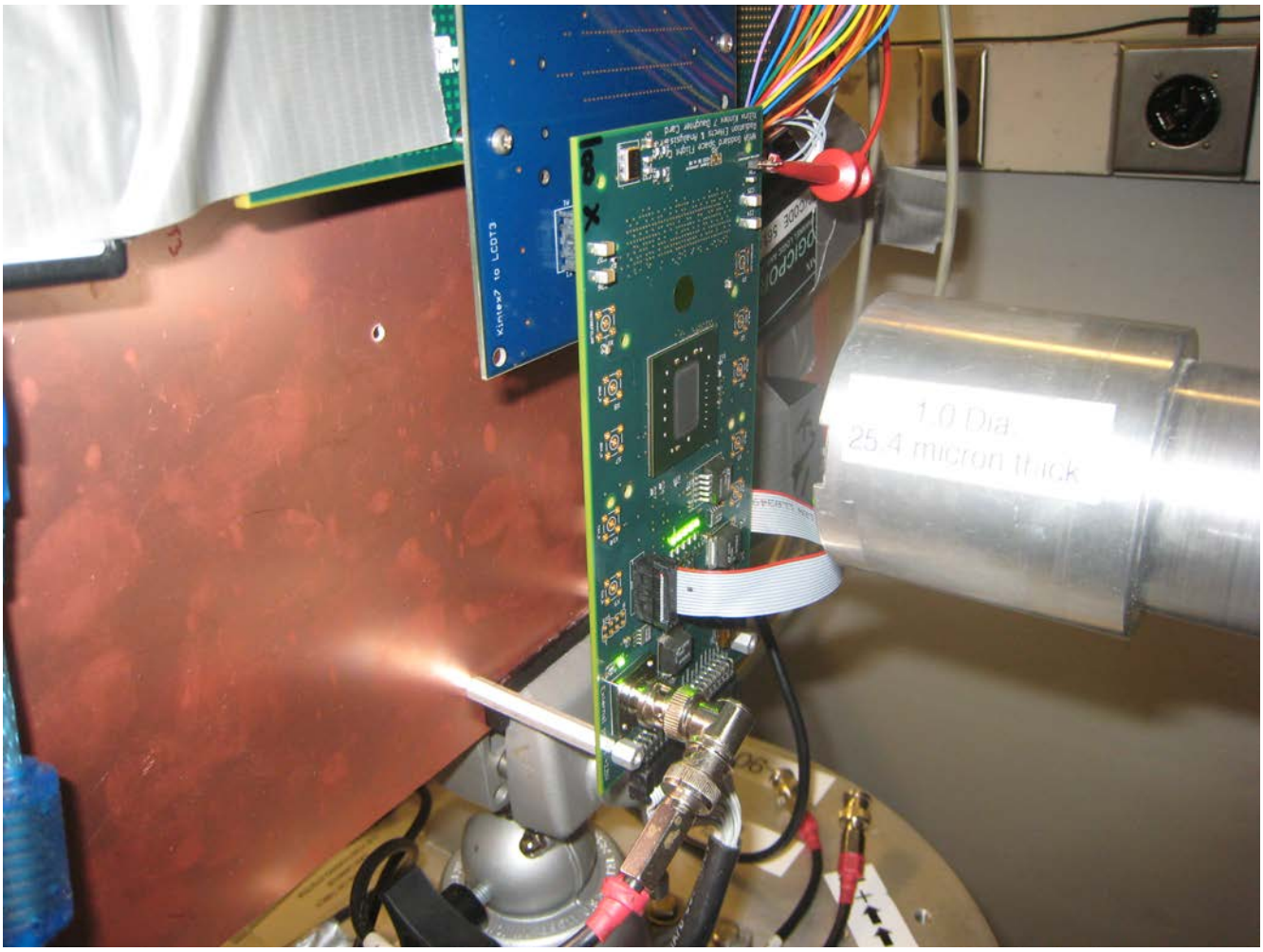


Figure 17: Test Setup - Connected LCDT3 and Daughter Card (2)

5. TEST PROCEDURES AND BEST PRACTICES

5.1 Functional Control

Types of DUT functional input control: clocks, resets, and data inputs.
Concerns and Challenges:

- Synchronizing inputs and managing skew between inputs. Challenging with high frequencies.
- Operating the device in a realistic manner:
  - Do not over-load the device with unrealistic stimulus during radiation testing. If the device is operating in states that would never occur, then radiation data will not be characteristic.
  - Do not under-load the device during radiation testing. If the device is underperforming, this means that a large amount of circuitry is not operating. This produces operational states with a large amount of logic masking; consequently, radiation data will not be characteristic.

5.2 Monitoring Functionality

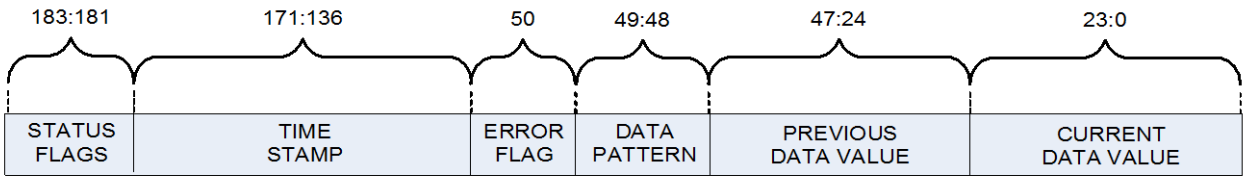
The following are general practices for monitoring FPGA functionality during heavy-ion testing.

- Compare DUT outputs to expected values
  - Visually (only recommended as a supplement); i.e., watching the error indication on the error detection equipment (e.g., logic analyzer);
  - Custom comparison circuitry (Low Cost Digital Tester: LCDT).
- Differentiate upset types: e.g., clock tree SET, flip-flop (DFF) SEU, combinatorial logic (CL) captured SET, or configuration faults.
- Count SEUs (upset statistics): After the upsets have been detected and differentiated, they need to be counted. The higher the number of upsets, the better the statistics.

5.3 Automated Data Capture and Messaging

The following are general practices for data capture during heavy-ion testing.

- Reliable data capture:
  - Follow synchronous design rules – which include how to capture asynchronous signals.
  - Determine minimal sampling frequency (when applicable).
  - Understand the limitations of the automated test equipment with respect to the DUT (e.g., memory-storage space, I/O voltage, I/O interface, and speed).
- Once erroneous data are captured, they should be packaged and stored (e.g., sent to a host PC). Example of package fields:
  - Timestamp,
  - Expected value, and
  - Received value.



Field	# of Bits	Description
Current Value	24	Current captured data (cycle N)
Previous Value	24	Previous captured data (cycle N-1)
Data Pattern	2	Unused: data pattern is always checkerboard
Error Flag	1	Unused
Time Stamp	32	Cycle counter. Must multiply by the DUT frequency to convert to time. Used to determine error burst sequences
Status	3	Indicates type of error record: "001" is a timeout – one of the shift clocks not detected "011"Out of timeout – all shift clocks are recovered "000" Error or non-error – current value does not equal previous value "010" Debug check – command was sent to check value settings

Figure 18: Sample Message from the LCDT to the Host PC. Message describe the SEU and Includes a Timestamp

5.4 Monitoring DUT Power

The following are general practices for monitoring DUT power.

- Use of power supply monitors.
- Use of an automated monitor/capture system is beneficial. Provides the ability to store and perform post processing on power data. Great for identifying particular error signatures.
- Power glitching or Single Event Latch-up (SEL) can cause the system to cease operation or be damaged. Hence it is best practice to separate test vehicle power from DUT power.
- It is also ideal to have current limiting circuitry for the test vehicle and the DUT.

5.5 Scrubbing Overview

Scrubbing is the act of simultaneously writing into FPGA configuration memory as the device's functional logic area is operating with the intent of correcting configuration memory bit errors. The following are challenges and concerns that should be addressed when implementing a scrubbing system:

- Understand scrub-rate to accelerated configuration upset rate ratio. Accelerated configuration upset rate will be dictated by device sensitivity and flux. Too many upsets in the system (due to accelerated flux) can cause unrealistic behavior… these can lead to obtaining unrealistic σ_{SEUS}!
- Can manage the accelerated upset rate by varying flux.
- Scrub as fast as possible – i.e., Make sure scrubbing can keep up with your upset rate.
- Our scrub rate for the Xilinx Kintex-UltraScale is once every 100 ms (10 Hz).
- NEPP uses Blind Scrubbing.
- Read-back after a test with scrubbing should have a minimal number of configuration-bit upsets (excluding un-scrubbable bits)… unless a SEFI occurs.
- The act of scrubbing must be validated prior to testing (that uses a scrubber). The verification process includes:
    - Starting functional operation.
    - Starting the scrubber.
    - Use the scrubber as a fault injector. A large number of configuration bits are injected to save time.

5.6 Investigating SEFIs

We look for particular error signatures to determine SEFI occurrence:

- Read-back of configuration is mostly logic '0' – assume a Power On Reset (POR) glitch.
- Unable to connect to the device to readback – assume problem in the configuration interface.
    - Hidden (to user) internal state machines that control interface communication can be in error.

  - Configuration registers can be in error.
- Global SEFIs in functional logic – not performed during static read-back.
  - Reset correction: clock tree or reset tree (global routing).
  - Configuration correction: configuration bit upset – not considered a SEFI.

P_{SEFI} is calculated using equation (5). Test until upset occurs and record the fluence.

$$\sigma_{SEFI} = \frac{1}{(fluence)} \quad (5)$$

The following sections describe the construction of the LCDT including communication interfaces with the DUT and user PCs.

6. LCDT ARCHITECTURAL OVERVIEW

As previously mentioned, the test system consists of a Mother Board (FPGA Based Controller/Processor) and a daughter board (containing DUT and its associated necessary circuitry). The DUT controller/processor is instantiated as a sub component within the LCDT. A block diagram is illustrated in Figure 19.

The objective of this DUT Controller/processor is to supply inputs to the DUT Device and perform data processing on the outputs of the DUT. The LCDT communicates with a user controlled PC. The user PC interface is LAB-VIEW. LAB-VIEW code has been designed to send user specified commands to the motherboard and receive information from the motherboard. Please see Documents: “LCDT” and “General Tester” for further information concerning the LCDT functionality. The LCDT is connected to the DUT as shown in the following Block Diagram.

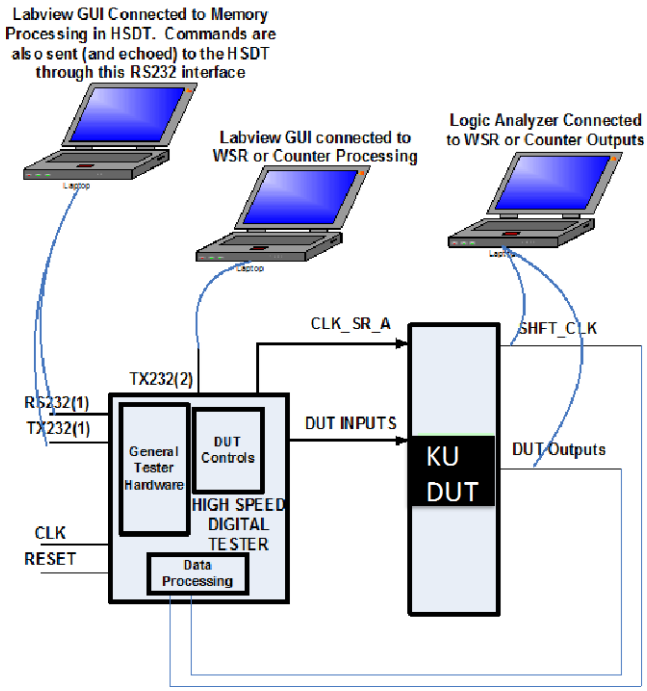


Figure 19: System Level Tester Architecture for the DUT configured as a counter.

6.1 LCDT flow for Scrubbing the DUT

It is the responsibility of the tester to scrub the DUT configuration during accelerated testing. By definition, particle flux during accelerated testing is magnitudes faster than the flux in space. The repercussion of high particle flux is the potential for unrealistic upset events – especially to soft areas of a DUT. Regards to scrubbing, if particle flux is relative high as compared to the upset rate and scrub rate, then the accumulation of configuration upsets can cause circuitry to break during testing, that would not normally break in space. Hence during accelerated testing it is required that the scrubber performs as fast as possible. In response, NEPP uses the blind scrubbing technique. The blind scrubbing technique for accelerated testing is as follows:

- Create a scrub file. The scrub file is created from the .bin output file provided by ISE. The command header and footer of the .bin file is changed specifically to perform scrub. The configuration area of the .bin file is unchanged.
- Download the configuration scrub file to Tester on-board SRAM. The Download is implemented using a USB interface.
- Upon scrub command, the tester reads SRAM and writes each data item as required by an 32-bit SelectMap interface. The SRAM contains the golden configuration and does not

attempt to correct via error detection and correction schemes. Alternatively, since the correct configuration value is known and written into configuration, the blind scrubber is not compromised or limited by multiple bit upsets.

- The scrubber never stops and is performed at 25MHz per 32-bit data word.

There are four areas of the configuration that cannot be scrubbed:

1. LUT SRLs or DRAMs: Generally LUTs contain static configuration settings. However, LUTs can alternatively be used as design latches and can hence change value during operation. The design latches can be in the form of Shift register latches (SRLs) or distributed RAM cells (DRAM). There are times when the design will map into these special structures. In that case, the user cannot scrub into these structures. To avoid scrubbing into these structures the user must set the GLUTMASK bit to 0 in the Kintex-UltraScale CTL0 register.
2. BRAM: Not all designs use embedded RAM (BRAM). However, if they do, then the BRAM cannot be scrubbed. This can be accomplished by stopping the scrubber prior to reaching BRAM area.
3. Capture bits: Configuration capture bits are used to read current values of designated DFFs. They are primarily used for chipscope usage. However, designers are able to manipulate these configuration bits at will. No scrubber will scrub into these areas. Hence during read-back the bits can list as upset, but really have no impact on the design.
4. Hidden logic: There are some configuration bits that control hidden logic and cannot be scrubbed by any method. These bits will readback as in error if hit during irradiation.

Within blind scrubbing, there are two types of scrubbers implemented: (1) Scrub every word of the configuration or (2) scrub all words except SRLs, and BRAM. Option (1) is used on designs that contain no SRLs and do not use BRAM. This enhances the tests by reassuring the user that the scrubber is working during irradiation because when reading back the configuration, the number of upsets is minimal (usually between 0 and less than 10 – not including capture bit upsets)

6.2 RS232 communication from the LCDT to the Host PC

All RS232 communication from the LCDT to the host PC is prefaced with a header. Information from the LCDT to the Host PC is one of the following listed in Table 3: an alive-timer, a command echo, or an Error Record.

Table 3: A list of the LCDT to Host PC RS232 Header bytes. Only the LCDT uses header information. The host PC sends pure commands to the LCDT without headers.

Header	Description
00 FA F3 20	Alive Header No data bytes follow (i.e. only the header is sent from the LCDT to the PC)
00 FA F3 22	Command Echo. 4 data bytes follow that represent the command that was previously sent

	from the Host PC to the LCDT.
00 FA F3 21	Data Error Record: 23 bytes follow.

6.3 RS232 communication From the Host PC to the LCDT

Communication from the host PC to the LCDT does not contain a header. Information sent from the host PC to the LCDT are commands and are all 4 bytes in length The interface is controlled by a user GUI designed with LabView software.

6.3.1 User GUI

Commands are sent by typing specific values into Labview fields or controlling Labview on/off buttons listed on the screen. GUI Figures will be provided

6.3.2 User Interface and Command Control

The User controls the tests via a LABVIEW interface running on a PC. The PC communicates with the LCDT with a RS232 serial link. The format of communication is a command/Data 4-byte word.

Table 4 : Summary of Commands Used in DUT Counter Tests

Command #	Command	D0	D1	D2	Description
01	Reset LCDT	n	n	n	Resets SIRF-S
03	Reset Counter	N	N	n	RESETS Counters
04	Start Counter Testing	N	N	n	Sends a reset pulse and then starts the DSPs and counter array tests
02	Start Counter Testing	N	N	N	
A0	clock divider	y	y	n	D0(LSB) and D1 (MSB) are the Clock frequency divider of 100mhz. The synthesized clock will be sent to the DUT counters as their system clock. Default=0 (no division)

Table 5: Summary of Commands Used in DUT Scrubbing

Command #	Command	D0	D1	D2	Description
99	Reset scrubbing control logic	N	N	N	Gets the scrubbing control logic ready to start scrubbing
06	Start Scrubbing	n	n	n	Run all architectures
26	Stop Scrubbing	N	N	N	
E	Inject Error to Configuration memory	N	N	N	
79	Start Address for Error injection	Y	Y	Y	(address reflects data word of 16-bits)
7A	End Address for Error injection	Y	Y	Y	(address reflects data word of 16-bits)
7E	Flip Every bit between Start Address and End Address given in commands 79 and 7A	Y	Y	Y	When this is inactive, error injection is performed one bit at a time between Start address and end address given in commands 79 and 7A. When active, the tester will flip every configuration bit within the address range between start address and end address.

7. DUT TEST STRUCTURES AND DYNAMIC ACCELERATED TESTING

In this study, we start with simple test structures; increase complexity per test structure; study trends; and then try to make sense out of the convoluted data obtained from complex test structures.

Tests used scrubbing because of the accelerated test environment.

Test Structure Considerations Taken from the NASA Electronics Parts and Packaging (NEPP) FPGA SEU Test Guidelines: https://nepp.nasa.gov/files/23779/fpga_radiation_test_guidelines_2012.pdf .

Referencing the FPGA SEU Test Guidelines manual, for simple test structures, they should have the ability to flush SEU's (post-detection)… i.e., test circuits should have the ability to keep working after an SEU occurs to increase statistics of each beam run. This helps to increase statistics during testing. In addition, the tester (for these simple test structures) should be robust enough to

resynchronize to new DUT expected values caused by SEUs; i.e., expected values should be dynamic based off of impact of SEUs. These statements are not necessarily implementable in a complex system. However, in shift registers and counters; i.e., flushable test systems, it is feasible.

Table 6: Overview of Test Structures

Test Structure	Frequency Range	SEU Flow-Through (Flushable)
Counters (Count Array): Parallel independent counters with a variety of mitigation strategies.	50 MHz	Yes
Global routes	50 MHz	N/A

7.1 Counter Array

During SEU testing, it would be ideal to be able to monitor every element of a complex design for every cycle. This is generally not feasible because it would require a DUT interface to the test vehicle for every internal DFF node. Therefore, more creative designs and interfaces must be developed so operation during irradiation is unrestricted (fast, continuous, and unobstructed) yet node observation is maximized. Just as important, the tester must be fast enough and robust enough to capture and process the data supplied by the DUT. Processing integrity is very important. Dropped or incorrectly processed data can drastically change error cross sections.

For the SEU testing of the Kintex-UltraScale Counters, a simple yet effective interface was used (snap-shot bank of registers).

7.1.1 Counter Array Implementation

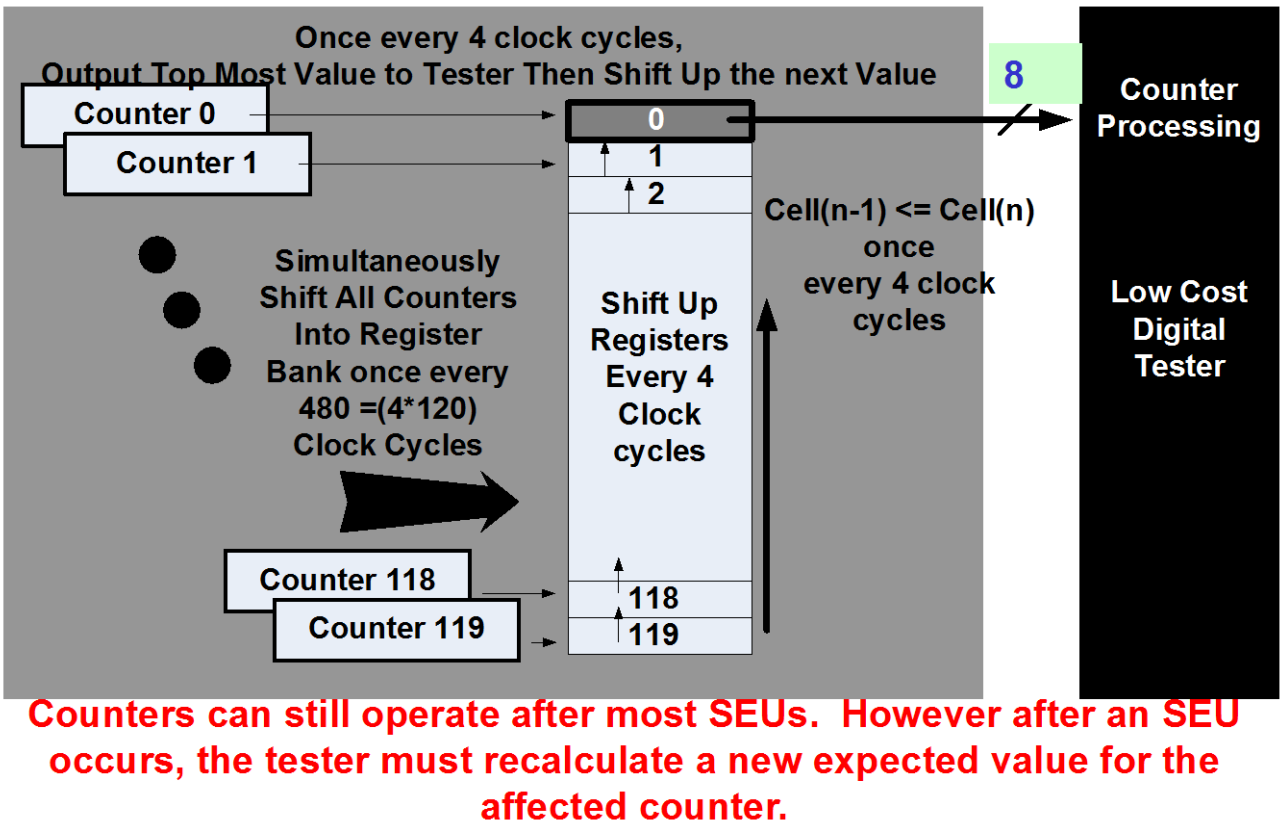


Figure 20: Schematic of the 8-bit Counters and their Output Selection Logic. In this case, the output selection logic is a Snap-shot shift register (Shifts up counter values to the output registers every 4 cycles). The DUT uses a 8-bit counter scheme with 200 counters labeled “counter 0” through “counter 199”.

The counter array developed by NEPP is illustrated in Figure 20. The array contains 200 counters that were 8-bits wide. Because it is impossible to simultaneously output 200 by 8-bits, requiring 1600 outputs, an output scheme had to be employed that would not compromise the number or speed of the counters yet ensure that each counter is an observable node. Conventional thinking would suggest employing a multiplexer that sequences through the array and selects one of the 1600 counters to be output at a time. Unfortunately, the function of the multiplexer, selecting 200 items, requires many levels of combinatorial logic that can be problematic during radiation testing and will slow down the operation of the circuit. Such a large block of logic can potentially mask the primary objective – i.e., which is characterizing counter SEU susceptibility. Therefore, a novel output methodology had to be established.

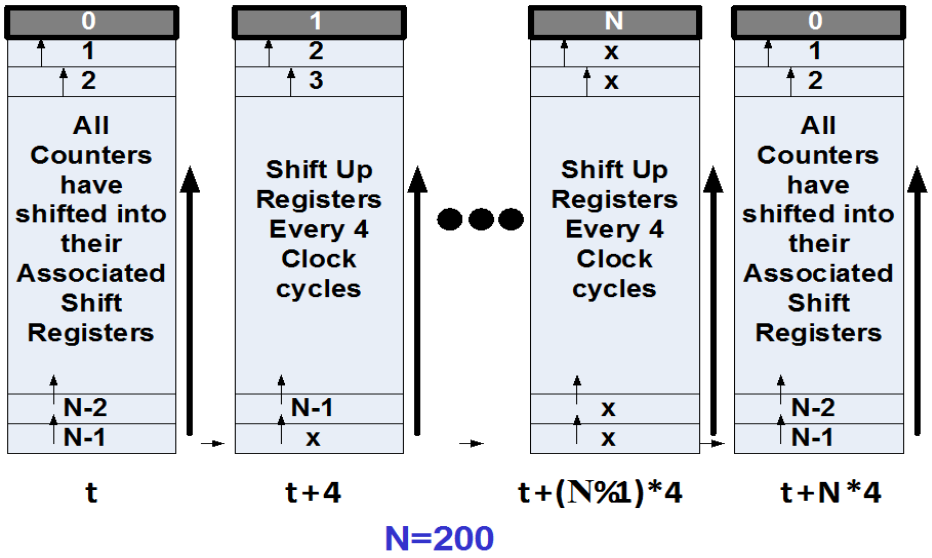


Figure 21: Counter Shift Register Cycles inside a snap-shot bank; Values in Snap-shot shift registers represent counter labels at a given moment in time. Regarding this figure, if there is an x with the Snap-shot register, then it is considered a "don't-care" state. N=200 for the Kintex-UltraScale Radiation Test.

As an alternative, a "snap-shot" solution was implemented. With this methodology, each counter is captured simultaneously at a given time into a bank of snap-shot registers. The number of registers is equivalent to the number of counters (i.e. each counter has its own snapshot register). This is illustrated in Figure 21. The top of the register bank (register 0) is the only register that is accessible by the tester and is 8- bits wide. Subsequently, the DUT to tester interface is simplified.

Figure 20 and Figure 21 illustrate the utilization of the snapshot shift register for each clock cycle. The nomenclature pertaining to the counter circuits throughout this document is as follows:

- n: counter label number
- k: snapshot cycle. First cycle out of reset all of the counter values are snapshot (shifted over) to the shift up register bank. k is 0 for this cycle. The next snapshot of counter values is 1200 clock cycles later and k will increment to 1.
- $X_{n,k}$ is the counter-n value that was snapshot into the shift up register for snapshot cycle k.

Coming out of reset, each counter has an initial value (X_{n0}) equal to the counter label number (n), e.g. Counter 0 has a reset value of 0 ($X_{0,0}$=0) and counter 199 has a value of 199 ($X_{199,0}$=199). As the circuit comes out of reset, the counter values are loaded into its corresponding snap-shot register. The counters continue to increment simultaneously as the shift registers shift counter values up every 4 clocks cycles – illustrated in Figure 21. The purpose of the snap-shot register is to output all counters to the tester. The snap-shot registers are loaded and then each value is shifted up so each counter-value can reach register 0 (the output window to the tester). After all loaded counters residing in the snap-shot bank have reached register 0 (τ+4N= once every 4*200 = 800 clock cycles), all of the counters are reloaded into the snap-shot shift register bank.

As a summary, the key of the snapshot output scheme, is that the shift register array has now replaced a huge multiplexer. The benefits are as follows:

1. Counter upsets can easily be identifiable.

2. Counters are incrementing and changing state every cycle. Hence maximum performance can be tested.
3. If a counter becomes upset, it will stay upset and it will eventually be captured during the snap shot period Counters are continuous and are not interrupted due an elaborate output scheme
4. Routing complexity is exclusive to just the counter array
5. The shift register architecture allows for high speed counter testing. A large multiplexer creates long paths of combinatorial logic and significantly slows down system speed.

The state space of the DUT should be deterministic and traversable. Pertaining to Equation (6), for a 8-bit counter running at 25MHz and a shift up period of once every 4 clock cycles, it will take a little less than 1s for every state to be reached for all counters. Radiation tests generally last for several minutes. Hence, counter states are considered fully traversed within each test run.

$$\frac{2^8}{f_s} = \frac{256}{100MHz} = 2.56us \quad (6)$$

7.1.2 Counter I/O Interface and Expected Outputs

Table 7: DUT1 WSR Outputs.

I/O Name	Bits	Direction wrt to DUT	Description
Counter_Shift_CLK	1	OUT	Counter shift Clock
COUNTER	8	OUT	8-bit Counter output
CLK	1	IN	Clock to counter circuitry
RESET	1	IN	Reset to counter circuitry

Table 7 lists the I/O for the counter DUA. The DUT receives a clock (CLK) and a reset(RESET) from the tester. The DUT responds with two primary outputs:

- Counter: 8-bit counter output of DUT. This is the output from the top snap-shot register that is forwarded to the tester.
- Counter_Shift_CLK: This output is ¼ the speed of the CLK (input from the tester). It is ¼ the speed indicating the shift speed (as previously mentioned, the snap-shot register shifts once every 4-CLK cycles).

The expected output (COUNTER) is purely an increment by 1 starting at value 0. The first COUNTER_Shift_CLK output will pertain to counter 0, followed by counter 1 after 4-CLK cycles, counter 2… up to counter 199. After the value of counter 199 has reached the top of the snap-shot register and is output to the tester, a new snap-shot is performed (all counters are shifted into their associated shap-shot register). After the new snap-shot load, the top of the snap-shot register bank now has the value for counter 0. Hence, COUNTER_OUT (top of snap-shot register) will provide the tester loaded value of counter 0.

With respect to the separate counters, Counter(n) is output every 800 cycles (800 cycles= 1 snapshot cycle). Therefore the counter values that represent each counter will increment by 800 for each snapshot cycle.

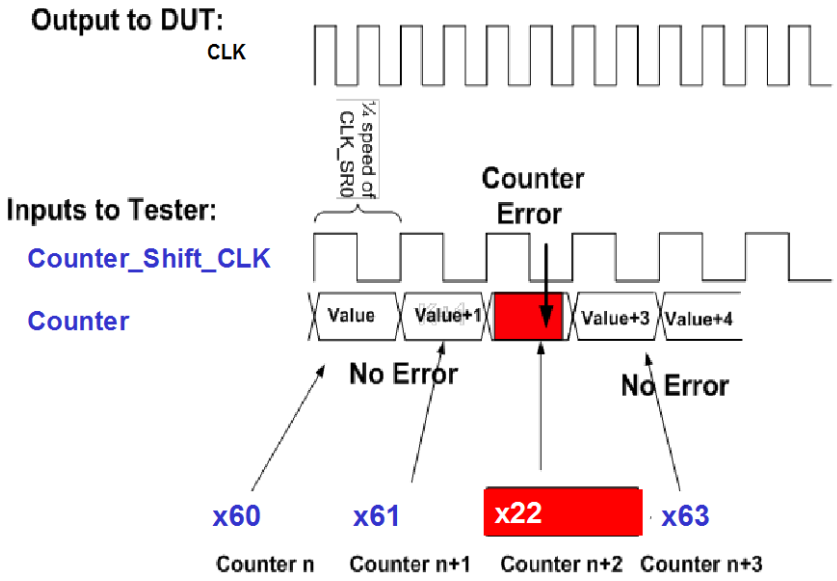


Figure 22: Typical SEE Counter Outputs. Each output represents a value from a different counter in the array. Counter selection is sequential, hence, the counter number and the counter values all increment by 1 each Counter_Shift_Clk cycle.

•

7.1.3 Counter Array and LCDT Specifics

DUA contains the following:

- 200 8-bit counters
- 200 8-bit snapshot registers

All counters and snapshot registers are connected to the same clock tree and RESET. The clock tree is fed by the CLK input from the LCDT.

The LCDT will send the following signals to the DUT:

- 1 clock
- 1 reset

7.1.4 Processing the DUT Outputs during Testing

The outputs of the DUT are fed to the tester for data processing. The objective of the data processing is to capture data from the DUT, compare to an expected value, and report to the host PC if there is an error. The DUT system clock and reset signals are generated in the LCDT. The following describes how the counter expected values are calculated. As a note, each counter will have a uniquely stored expected value.

7.1.4.1 Counter Array data processing

All counters within the array independently and continuously increment every clock cycle. Their states are captured (snapshot) once every 800 (4*200) clock cycles into a snapshot array. The top most register of the snapshot array is the only interface to the tester.

In order to avoid metastable events due to an error in the output, data is registered twice before evaluation. Both the data and the counter number are expected to increment every 4 cycles and will wrap around at its boundaries as listed in Table 8.

Table 8 Counter Value and Counter Number Wrap around Boundaries

	Bit Width	Wrap Around Value
8 bit Counter	8	2^8-1 (after 2^8-1 next value is 0)

At the beginning of each snapshot cycle (at capture time), Each counter should have incremented by either 800MOD8 from the last snapshot capture. If not, then an error record is sent to the LCDT. The tester keeps a copy of each counter value during a snapshot cycle. This copy is kept around as a comparison point for the next snapshot cycle. During the next snapshot cycle, the old copy becomes: "Last counter value" and the current counter value (just captured) becomes the New counter value. The tester is expecting the increment as follows:

Expected Checks: New Counter Value = (Last counter value + 800) MOD 8 … for a 8-bit counter

If this doesn't occur, an error is registered. The following is a list of potential incorrect counter values and their corresponding sources of error:

- A single-bit (or multiple-bit) upset in a counter,
- An upset while the value is sitting in the snapshot register,
- Configuration SEU or multiple bit upset (MBU),
- Global upset, or
- Catastrophic event

Due to the complexity of this device, error signatures can be convoluted. Consequently, a significant amount of post processing on radiation data is required to differentiate the SEUs.

7.1.4.2 Counter SHIFT_CLK Processing

Regarding the SHIFT_CLK associated with the Counter Arrays, it is used to alert the tester that the counters are alive and new counter data is ready to be taken. The SHIFT_CLKs are always ¼ of the speed of the DUT system clock. SHIFT_CLK can be interpreted as an operation heart-beat.

Due to the interface delays and device latencies and in order to consequently decouple the DUT to tester timing restrictions, the DUT SHIFT_CLK is considered asynchronous to the tester and is sampled using the LCDT system clock (max 100 MHZ). Thus, the tester's sampling clock will always be 4 times as fast as SHIFT_CLK. The SHFT_CLK is fed into a metastability filter and an edge detect. This process takes 1 to 2 clock cycles of the sampling clock (detection will be delayed by 1 to 2 sampling clock cycles of the actual edge).

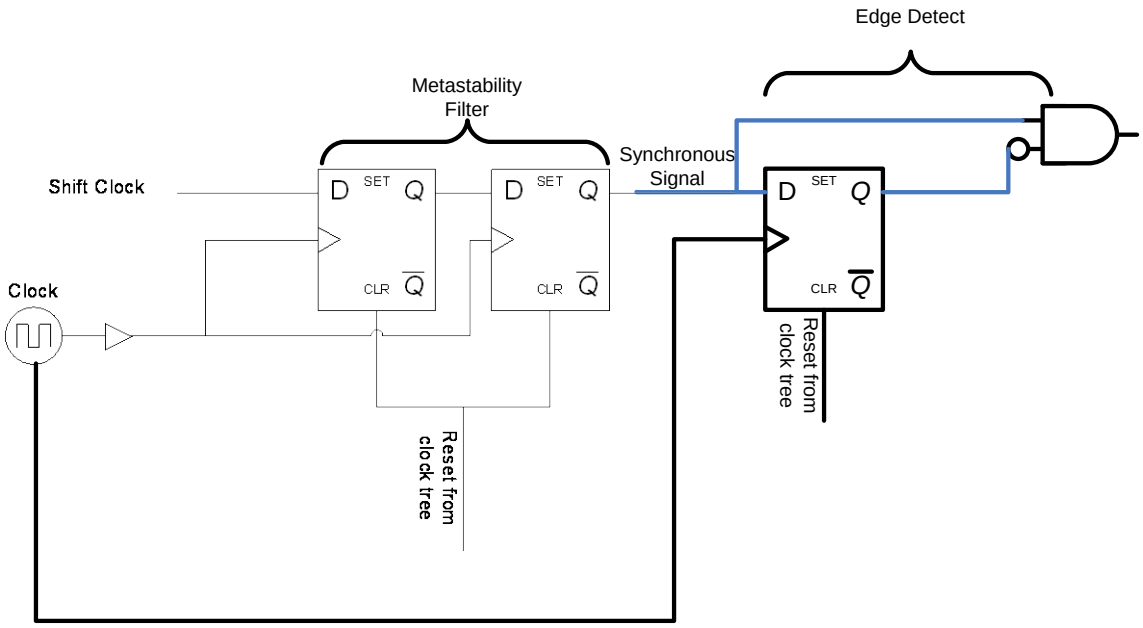


Figure 23: Shift_ClK Capture consists of a Metastability Filter and a Edge Detect

The SHIFT_CLK edge is expected to come at a frequency that is ¼ of the LCDT clock. If the edge is stopped, glitched, or missing, the event is reported to the host PC by the LCDT.

7.1.5 Counter Array Error Record

A significant amount of post processing is expected to be performed on this data. Subsequently, the error record should contain enough information to comprehend and differentiate between events.

Table 9: Counter Array Error Record Fields: Yellow indicates Fields generated from Tester capture of DUT Inputs. Other fields indicate values calculated internal to the tester

Field	Bits	Description
Data Cycle N	16	Current DUT output. Error: If it is not an increment of 1 from the previous counter value and not an increment of 2 from the data value received cycle (n-2) No error (recover from error): If it is not an increment of 1 from the previous value but is an increment of the data value received cycle (n-2) Otherwise: DUT is in a burst of error
Data Cycle N-1	16	Capture cycle n-1 DUT output (capture cycle n-1 is actually 4 LCDT clock cycles from Data cycle N)
Data Cycle N-2	16	Capture cycle n-2 DUT output (capture cycle n-2 is actually 8 LCDT clock cycles from Data cycle N)
Data Cycle N-3	16	Capture cycle n-1 DUT output (capture cycle n-3 is actually 12 LCDT clock cycles from Data cycle N)

Counter Number	16	LCDT local copy of expected counter number. (0 through 299)
Error Count		
Time Stamp	32	
Status flags	3	

7.1.6 Counter Array Post Processing

Each error record will be post-processed. Error records (SEUs) are differentiated via the following:

- Bit or multiple-bit upset: This example is illustrated in Figure 3. One counter-bit flips and the counter keeps incrementing as normal from that flipped state. This error signature is attributed to a data-path SEU.
- Snap shot register:
  - A snap-shot register bit incurs an SEU. In this case a counter will show up as incorrect for two consecutive snap-shot registers.
  - Snap shot register can either stop shifting- in this case the output values will remain constant or become noise
  - Snap shot register can skip a cycle, in this case the counter values will be off the number of skipped shift cycles from their expected values
- Burst of broken counters: Counters stop counting and their values either stay constant or become complete noise. This error signature is attributed to either a misconfiguration or a global upset
- Burst of bad counter values: counters still increment but have one or more bits in error for several snap-shot cycles.
  - If the error signature is regular and correctable within an order of milliseconds, this error signature is attributed to a configuration SEU that eventually gets scrubbed away
  - If the error signature only lasts for less than 2400 clock cycles (and is not a snap-hot upset), this error signature is attributed to a global structure. We designate this as a small burst
  - If the error signature only lasts for microseconds, this error signature is attributed to a global structure. We designate this as a global upset
  - If the error signature lasts for greater than a second, this error signature is attributed to a global structure. We designate this as a drastic global upset

$$\sigma_{COUNTERSEU} = \frac{\#CounterArray_{errors}}{fluence * \#Counter\_ARRAY\_DFFs} \quad (7)$$

8. MITIGATION

In many circumstances, field programmable gate array (FPGA) devices expected to reliably operate in radiation environments require some level of mitigation. It has been shown that with reconfigurable non-radiation hardened FPGA devices, triple modular redundancy (TMR) is the most reliable mitigation strategy. TMR is the process of (1) triplicating circuitry into three redundant components; and (2) performing a majority vote on the redundant components.

Four different base designs have been described: WSR, Counter Array, FIR, and FIRWFB. to be This study includes an investigation of the efficacy of mitigation insertion. The previously described based designs were evaluated without inserted mitigation and also with a variety of mitigation triple modular redundancy (TMR) schemes.

8.1 TMR

TMR is the process of triplicating a circuit and performing a majority vote on the triplicate components. There are several TMR strategies. The strategies are differentiated by: the type of logic that is triplicated and where the voters are placed. In order to implement TMR insertion verification, it is essential to recognize the design's chosen TMR strategy and to use the strategy's topological definition as a verification reference. There are four primary TMR schemes: block triple modular redundancy (BTMR –Figure 24), local triple modular redundancy (LTMR –Figure 25), distributed triple modular redundancy (DTMR –Figure 26) and Global triple modular redundancy (GTMR). GTMR has the same mitigation topology as DTMR. The only difference between DTMR and GTMR is with GTMR, the clocks are redundant (each TMR domain is on a separate clock tree); and with DTMR the clocks are singular.

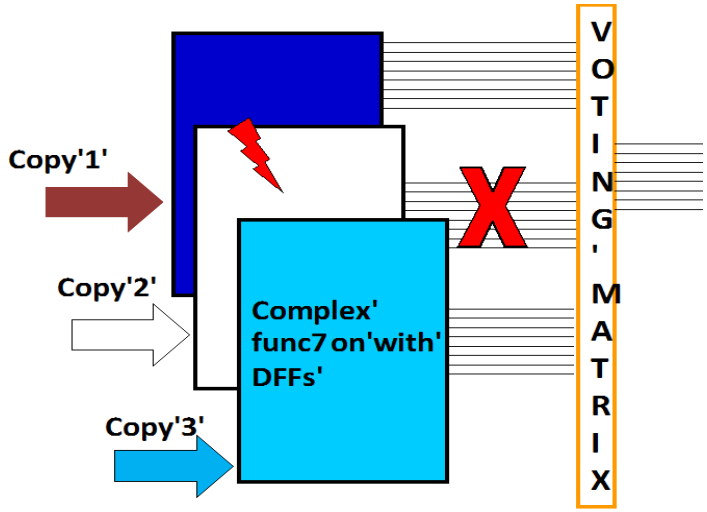


Figure 24: Block TMR (BTMR) Diagram: a complex function containing combinatorial logic (CL) and flip-flops (DFFs) is triplicated as three black boxes; majority voters are placed at the outputs of the triplet.

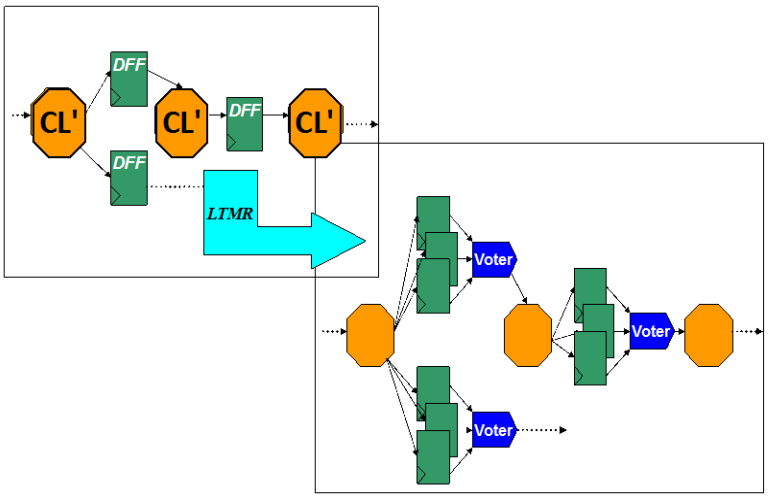


Figure 25: Local TMR (LTMR) Diagram: only DFFs are triplicated and data-paths stay singular; voters are brought

into the design and placed in front of the DFFs

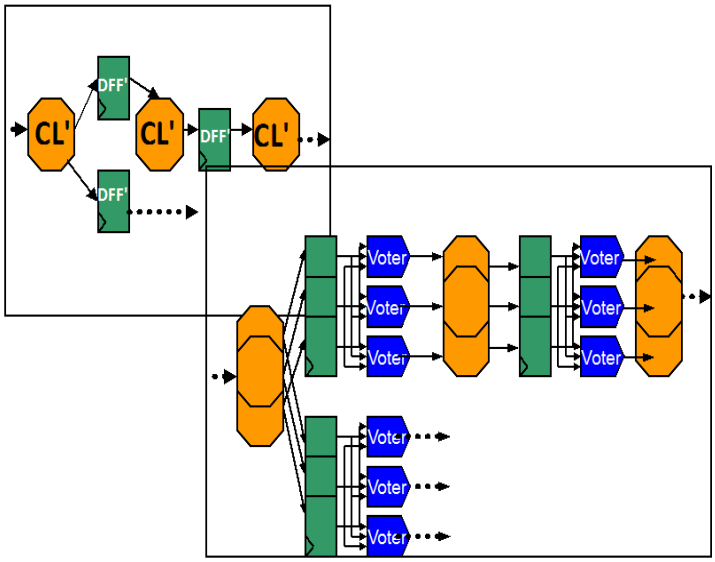


Figure 26: Distributed TMR (DTMR) diagram: the entire design is triplicated except for the global routes (e.g., clocks); voters are brought into the design and placed after the DFFs. DTMR masks and corrects most SEU upsets. Errors can be corrected if DFFs have voter-feedback

8.2 TMR and Mitigation Windows

When logic is triplicated, in a TMR scheme, each copied triplicate is referred to as a TMR domain. Hence for triple modular redundancy, there are three redundant domains of logic. We define a mitigation window to be three redundant domains of logic that terminates at their TMR majority voters. For BTMR, there is one mitigation window that contains three redundant copies of the design culminating with voters on the outputs. For LTMR, only the DFFs are triplicated and voted, hence the mitigation windows only encompass the DFFs. For DTMR and GTMR, the mitigation windows are more complex. For a full implementation of DTMR and GTMR, voters are placed after every DFF. Hence, for DTMR and GTMR, the smallest mitigation windows will encompass voter output from one stage of DFFs to the voter output of the next stage of DFFs. Mitigation windows are illustrated in Figure 27, Figure 28, and Figure 29.

If one or more SEU's are able to affect multiple domains that are contained in the same mitigation window, then the TMR scheme will break; i.e., the upset(s) will not be masked. In addition, in this case, correction will only be able to be handled by flushing the system (configuration scrub and perhaps a reset). Only SEUs that span multiple domains within the same mitigation window can break TMR. Figure 27, Figure 28, and Figure 29 illustrate examples of how SEUs in configuration memory can potentially affect the TMR scheme relative to mitigation windows.

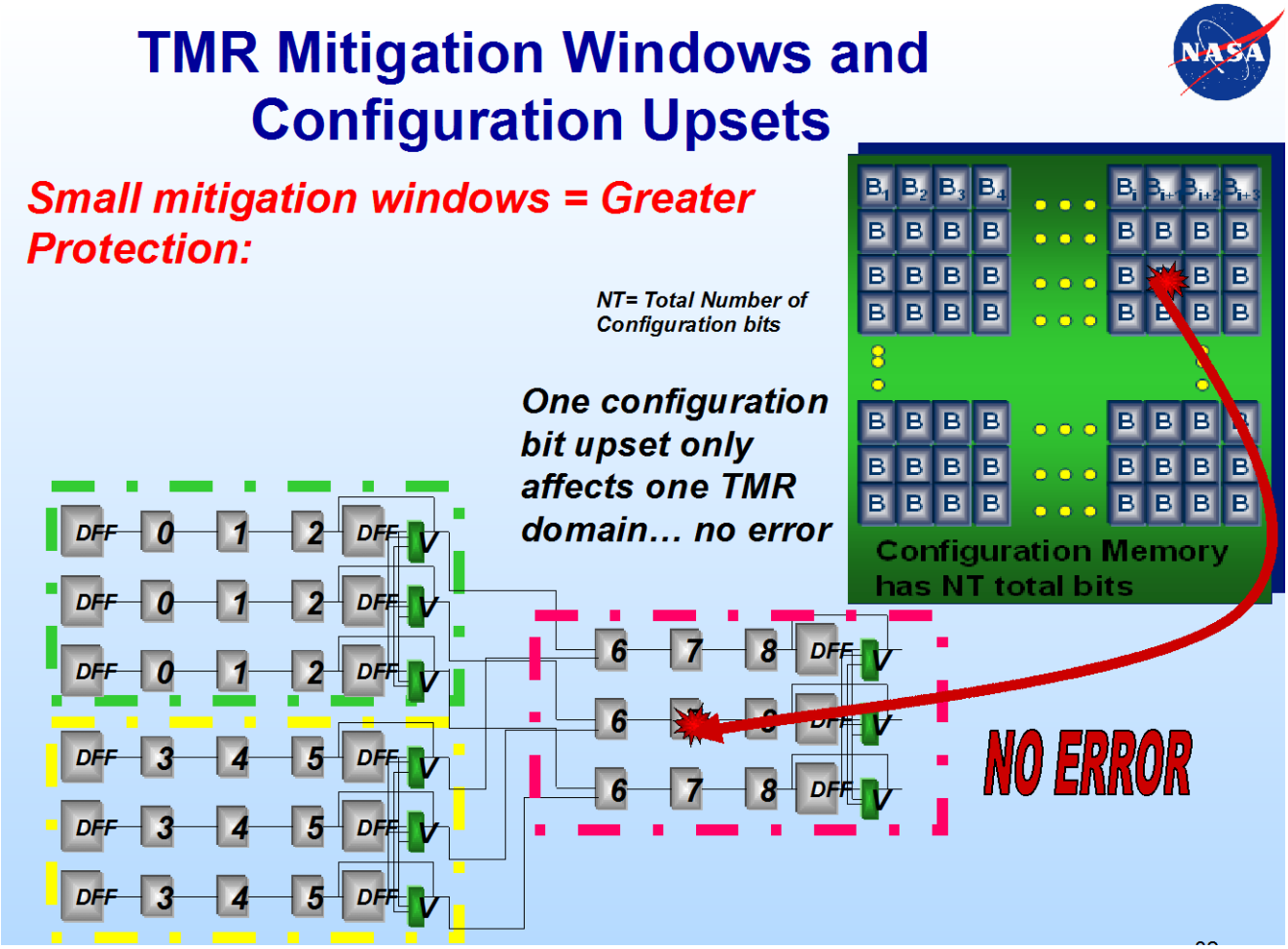


Figure 27: One configuration bit SEU that only affects one TMR domain will not cause system malfunction.

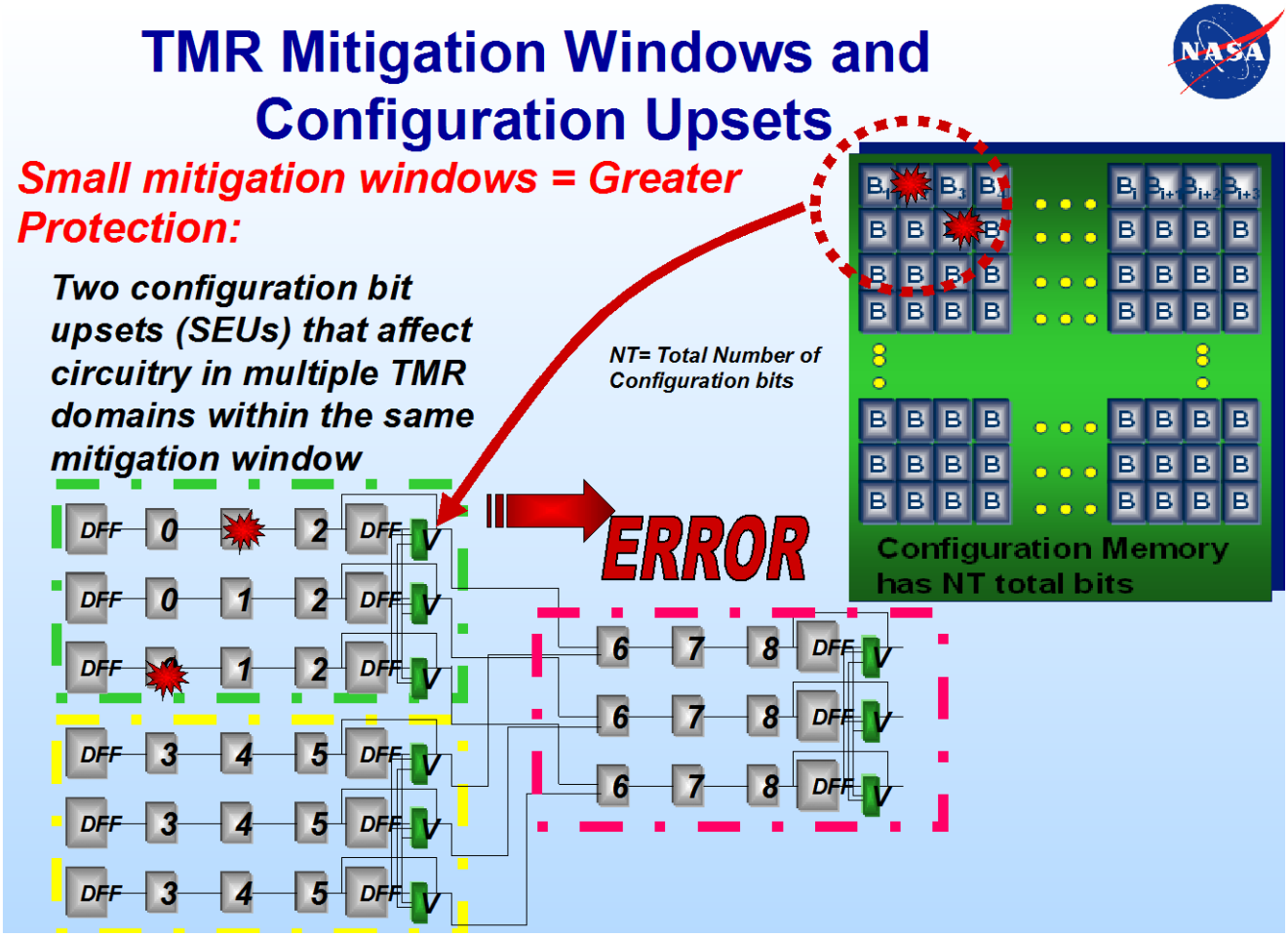


Figure 28: Two configuration bit SEUs that affect multiple TMR domains within the same mitigation window can cause system malfunction.

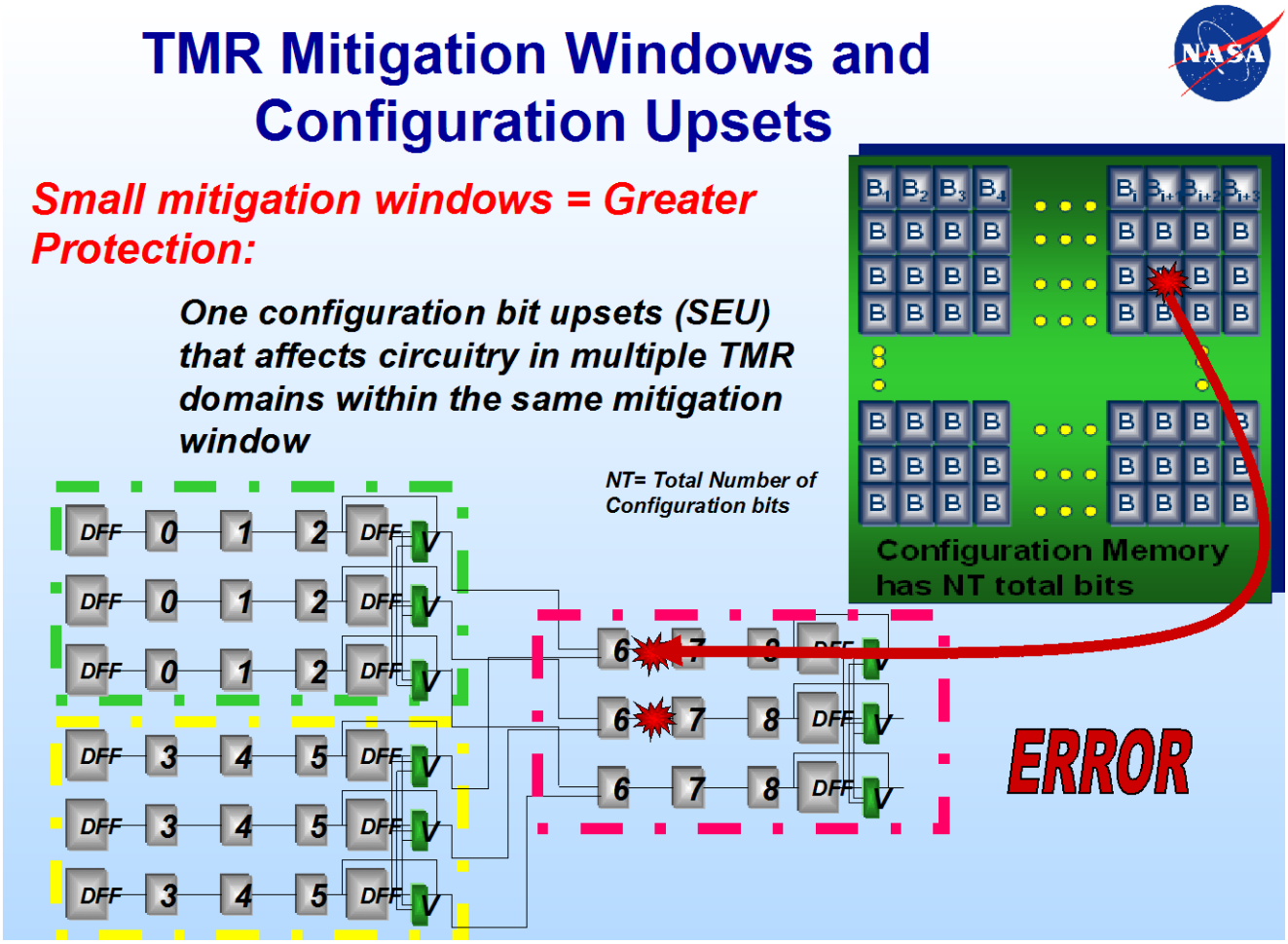


Figure 29: One configuration bit SEU that affects multiple TMR domains within the same mitigation window can cause system malfunction

8.3 DUAs and Partitioning

As previously mentioned, when logic is triplicated, in a TMR scheme, each copied triplicate is referred to as a TMR domain. If one or more SEU's are able to affect multiple domains that are contained in the same mitigation window, then the TMR will break; i.e., the upset(s) will not be masked. In addition, in this case, correction will only be able to be handled by flushing the system (configuration scrub and perhaps a reset).

A single SEU that can affect multiple TMR domains in a mitigation window is referred to as a single point of failure. In order to enhance TMR mitigation, single points of failure should be minimized. This is accomplished by partitioning the design such that the TMR domains share minimal logic. Partitioning is a good practice for BTMR, DTMR. It is not considered for LTMR.

Caution should be taken when implementing partitioning. Area, power, and speed will be compromised.

8.4 Summary of Tested DUAs

As previously mentioned, the base designs were evaluated with and without mitigation; and with and without partitioning. Table 10 lists the base designs with associated TMR and/or partitioning schemes.

Table 10: Designs used during Kintex-UltraScale heavy-ion accelerated testing.

Test Structure	Frequency Range
Counter Array No TMR	50MHz
Counter Array DTMR with partitioning	50MHz
Counter Array DTMR no partitioning	50MHz
Counter Array BTMR with partition	50MHz
Counter Array LTMR with partition	50MHz

8.5 TMR Heavy-Ion Accelerated Testing Data Analysis

As previously mentioned, SEU cross-sections are measured by the number of errors normalized by the number of particle exposure (particle fluence). See Equation (1). This is an appropriate measurement, when testing flushable designs because a significant number of upsets will occur per particle event space. However, when testing a system failure, there is generally one failure per test; and hence a relatively small number of upsets per event space. Consequently, we are using mean fluence to failure (MFTF) as a metric to compare TMR schemes.

- System malfunction is defined (for this study) as any output that is not in its expected state; i.e., all outputs must be in their expected states at all times for the systems in this study to be considered as functioning properly. Otherwise the system is considered to be malfunctioning.
- MFTF is the average exposure particle-fluence per design that will cause system malfunction. This is measured as the first upset that occurs. This can be challenging to measure because the beam will run longer than when the first error occurs:

  - There will be a significant delay between when the error occurs, when it is visually detected, and human response to stop the accelerated beam, or
  - The first error may be missed by human detection (however, will be in the digital log that is created from the LCDT to the Host PC).

- In order to capture the first error occurrence for MFTF calculations, synchronizers were designed to synchronize the LCDT (test system) to the beam activation signal. This was done to increase accuracy during data analysis. When the beam is turned on, the LCDT sends a status record (status = 6) to the Host PC. When the beam is turned off, the LCDT sends a status record (status =4) to the Host PC. Both records have a time stamp. The difference between the timestamps is the full time of beam exposure. It also is used as a reference mark within the LCDT to Host PC to indicate where SEU data (from beam exposure) begins and ends within the digital log. MFTF is calculated by multiplying the average flux by the time difference indicated.

MFTF, for all TMR schemes, is measured as the average fluence to first system failure. However, for BTMR we discriminate between first time to system failure and first time to TMR domain failure. A TMR domain failure is when one of the TMR copies fails; however, the failure is masked by the voted outputs and hence the system is still defined as operating correctly. We make this distinction because in some BTMR implementations, although the system is functioning correctly, it is required to bring the system down and flush the system as soon one or more of the domains are in error. Calculating the MFTF for each of the BTMR domains (first time to domain failure) assists in determining how often this sort of system will have to be serviced and flushed.

9. DUT ACCELERATED HEAVY ION TEST PROCEDURES

9.1 Summary of DUT-Tester operation.

Overview of performing an accelerated SEU test:

- Bias the device, turn on clocks, and toggle reset.
- DUA logic operates as tester captures DUA outputs and compare with expected counter pattern (verify no errors).
- Irradiate DUT.
- During irradiation: Tester reads DUT and compares to expected value:
  - If error during read, then the LCDT records that an error has occurred and sends the data value with timestamp to the PC.
  - If not done with test Goto 4, else goto 5.
- Stop Beam
- Reset Tester and DUT to prepare for next test

9.2 Running a Full Test

9.2.1 Files required for running a test

Independent of the DUT design under analysis (DUA), each test requires a DUT bit file (configuration file with extension ".bit"), DUT mask file (configuration mask file with extension ".msk"), DUT scrub file (extension ".scrub"), and a tester bit file (LCDT configuration file with extension ".bit").

In order to run a test, the tester-bit file is downloaded from the host PC, via Xilinx-impact (JTAG), to the LCDT3 configuration memory; the DUT-bit file is downloaded from the host PC, via Xilinx-

impact (JTAG), to the DUT configuration memory; The DUT-scrub file is downloaded from the host PC, via USB+LCDT3 control, to the LCDT3 on-board SRAM. After each test is finished, the DUT configuration is readback, via Xilinx-impact (JTAG). The DUT-msk file is required for the readback step.

9.2.2 Procedures for running a test

The following tables list the procedures required for running a test.

Table 11: Running a Counter Array Test

Action	Explanation
Program tester (download bit file to tester)	Via LCDT JTAG.
Program DUT (download bit file to DUT)	Via DUT JTAG.
Hardware reset	
Command 01 x x x	Via LabView: LDCT command Soft Reset.
Command 99 x x x	Via LabView: LDCT command Soft Reset.
Send Scrub file using quickUSB software	Via QuickUSB software.
Command A0 nn nn x	Via LabView: Speed Control.
Command 03 x x x	Via LabView: DUT Reset
Command 02 x x x	Via LabView: Start the tester/DUT system
Command 06 x x x	Via LabView: Turn on scrubber

10. HEAVY ION TEST FACILITY AND TEST CONDITIONS

Facility: Texas A&M University Cyclotron Single Event Effects Test Facility, 25 MeV/amu tune).
Flux: $1.0x10^2$ to $1.0x10^5$ particles/cm^2/s
Fluence: All tests will be run to 1 x 10^7 particles/cm^2 or until destructive or functional events occurred.
Test Temperature: Room Temperature

Power Supply Voltage: Variety of DUT voltages. See Datasheet.

Table 12: LET Table

Ion	**Energy (MEV/Nucleon)**	**LET (MeV*cm^2/mg) 0°**	**LET (MeV*cm^2/mg) 60 °**
He	25	.07	.14
N	25	.9	.18
Ne	25	1.8	3.6
Ar	25	5.5	11.0
Kr	25	19.8	40.0

10.1 Overview of Heavy-ion Accelerated Tests performed at Texas A&M

The following is a summary (list) describing tests structures and set-up conditions for heavy-ion testing at Texas A&M.

- Test structures: Counters arrays with a variety of mitigation strategies.
- Flux is kept low when required (below 100 particles per second).
- Appropriate fluxes were calculated based off of configuration upset rates and speed of scrubber. Hence, when starting tests at a particular LET, static configuration tests were run first in order to calculate configuration-upset rates at a given flux.
- Some tests were run with the scrubber on versus the scrubber off in order to determine if scrubbing would affect the system SEU rate. (Scrubbing did not).

11. RESULTS

11.1 Configuration Test and Analysis

After every test-run, the configuration memory is always read-back. Tests that were constructed specifically for configuration memory testing do not use scrubbing. The SEU cross-section results illustrated in Figure 30 are data gathered from no-scrub testing. Tests that have the scrubber turned on are expected to produce 0-to-10's of upsets after a beam run. If this is not the case, then either the scrubbing was broken or a POR SEFI occurred. Once again, this section only focuses on tests with the scrubber turned off. See section 2.3 for detailed information of configuration testing and analysis procedures.

Configuration data from heavy-ion tests are currently being analyzed and will be published in the next version of the test report. As a place marker, configuration data from ***David Lee et. al. "Single-Event Characterization of the 20 nm Xilinx Kintex UltraScale Field-Programmable Gate Array under Heavy Ion Irradiation,"*** ***https://www.osti.gov/scitech/servlets/purl/1263983.*** The data is shown in Figure 30.

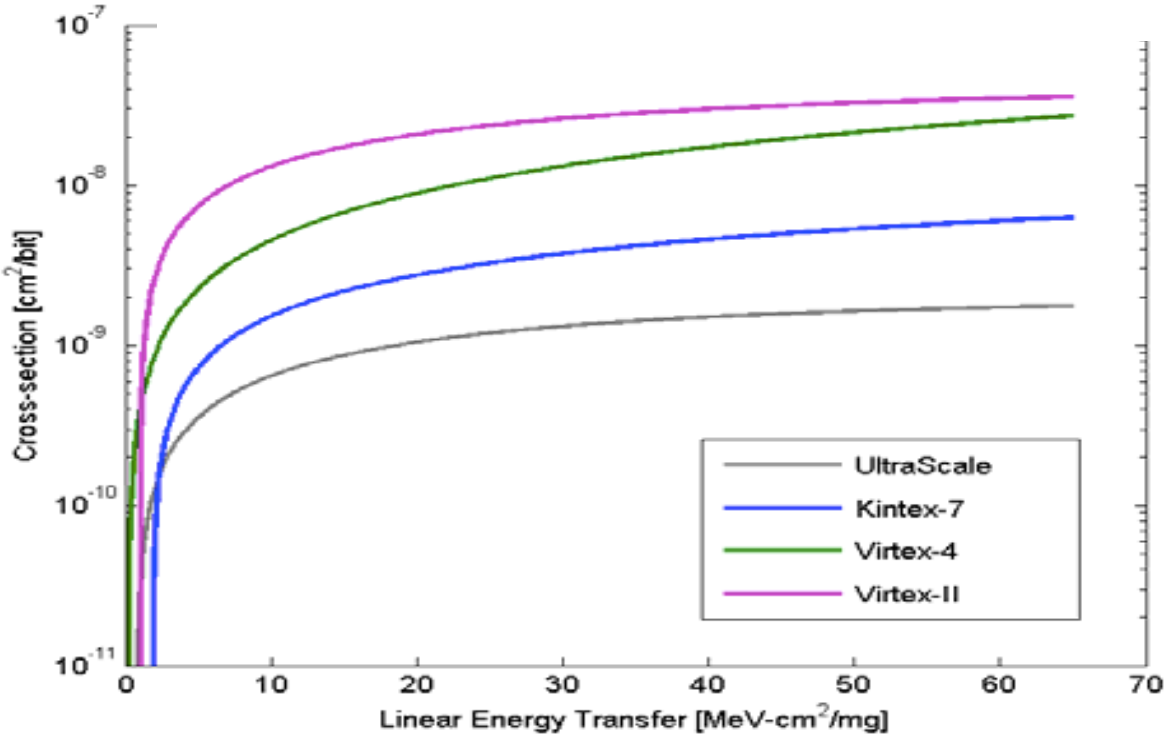


Figure 30: Previous testing Configuration Memory SEU Cross-Sections: ***David Lee et. al. "Single-Event Characterization of the 20 nm Xilinx Kintex UltraScale Field-Programmable Gate Array under Heavy Ion Irradiation," https://www.osti.gov/scitech/servlets/purl/1263983***

Based off of the configuration data illustrated in Figure 30, one can expect daily upsets in the configuration bits. It is important to emphasize that this event does not guarantee system failure. That will only occur if the configuration bit is upset while it is controlling active circuitry. If system failure is detected, 2 steps must be performed: 1. The configuration bit is corrected and then 2. The system's active circuitry is reset. As a clarification, correcting a configuration bit does not correct system state – system state can only be corrected by a reset or a flush.

11.2 Kintex-UltraScale SEL Results

The following lists the history of "SEL-like" events in Xilinx devices:

- Xilinx Virtex 2: Latchup-like events have been observed in flight. Most likely due to embedded half-latches in the device.
- Xilinx Virtex 5: Half-latches were removed. No latchup-like events observed during SEE testing or in flight.
- Xilinx 7-series: Is it SEL or latchup-like? Observed only on 7-series devices that contained 3.3V I/O. Devices that do not contain such I/O have no latchup-like events.
- Xilinx UltraScale series no latchup-like event observed.

11.3 Counter Array Results

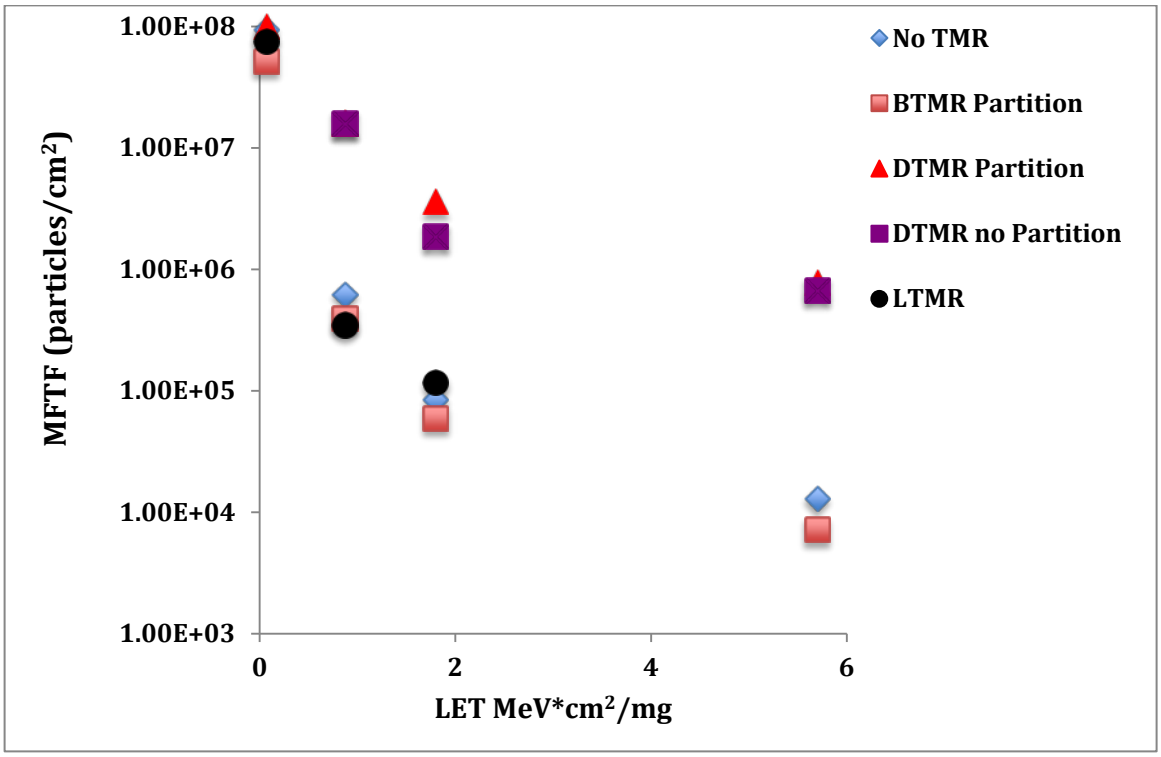


Figure 31: **Kintex-UltraScale Mitigation Study: Counter Arrays Mean Fluence to Failure (MFTF) versus LET**

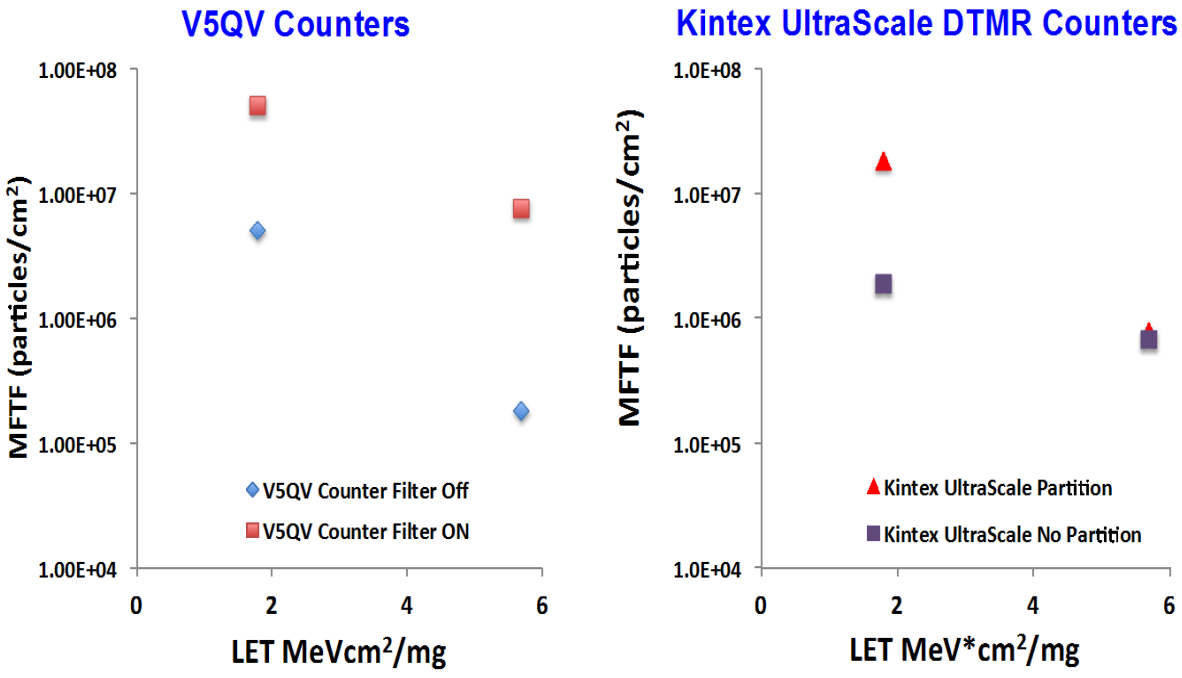


Figure 32: **Comparison of V5QV and Kintex UltraScale with Mitigation**

The mitigation study was conducted using Synopsys Premiere synthesis mitigation tools [4]. The following is a summary of mitigation data analysis:

- Mitigation study proves DTMR is the strongest mitigation scheme implemented in an SRAM-based FPGA.
  - However, for flushable designs BTMR might be acceptable.
  - LTMR is not acceptable in SRAM-based FPGAs for any design.
  - Partitioning may not be necessary.
- Although GTMR has been implemented in V5 families and earlier Xilinx device families, NEPP has suggested to avoid GTMR because clock skew is difficult to control.
  - In 2015-2016, via heavy-ion SEU testing, It has been observed in the Xilinx 7-series, that race conditions due to clock skew are unavoidable.
  - This is due to the speed of combinatorial logic and route delays in the 7-series versus earlier Xilinx FPGA device families.
- Synopsis tool has improved for simple designs. They are still working on IP core instantiations and other challenges.
- Mitigation and IP cores are still a major concern.

12. REFERENCES

[1] Melanie Berg et. al, "FPGA SEU Radiation Test Guidelines:" ***https://nepp.nasa.gov/files/23779/fpga_radiation_test_guidelines_2012.pdf***

[2] UltraScale Architecture Datasheet Overview: https://www.xilinx.com/support/documentation/data_sheets/ds890-ultrascale-overview.pdf

[3] UltraScale Configuration Architecture User Guide: https://www.xilinx.com/support/documentation/user_guides/ug570-ultrascale-configuration.pdf

[4] Symplify Premiere: https://www.synopsys.com/implementation-and-signoff/fpga-based-design/synplify-premier.html

[5] David Lee et. al. "Single-Event Characterization of the 20 nm Xilinx Kintex UltraScale Field-Programmable Gate Array under Heavy Ion Irradiation," https://www.osti.gov/scitech/servlets/purl/1263983