LA-UR-

Approved for public release;
distribution is unlimited.

Title:	
Author(s):	
Intended for:	

Form 836 (7/06)

FMAC
FPGA Mission Assurance Center

FPGA Testing and Trends

Heather Quinn, Paul Graham, Keith Morgan, Jim Krone, Michael Caffrey
Los Alamos National Laboratory

Mike Wirthlin, Brian Pratt, Patrick Ostler
Brigham Young University

Acknowledgements

This work was funded by the DOE under the JAS, SOPRANO and DAPS projects, the Rhino project through a NASA ROSES grant, and the DOD under the FPGA Mission Assurance Center.

Outline

- **Background**
- **Static Testing**
  - Test Methodology: Test Fixtures, Angular Testing, Multiple-bit Upset Testing, and Micro-SEFIs
  - Analyzing Test Data: Data Cleansing, Analysis
  - Test Results
- **Dynamic Testing of Mitigated Circuits**
  - Modeling Tools
  - Fault Injection Tools
  - Accelerator Testing
  - Test Results

Background:
Field-Programmable Gate Arrays

- **In this talk, we will focus only the testing of SRAM-based, reprogrammable FPGAs, where logic is implemented in lookup tables and the routing uses programmable switches**
  - Xilinx is the preferred vendor, because they have published several reports verifying latchup-immunity [1] [2]
- **For the rest of the talk, the term "FPGA" will be used to indicate only the Xilinx reprogrammable, SRAM-based FPGA**

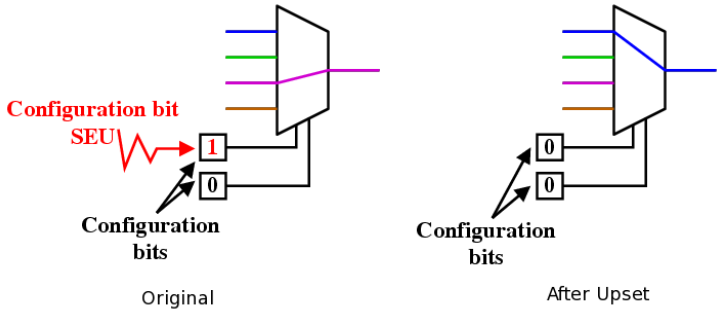
[1] G. M. Swift, "Virtex-II static SEU characterization," Xilinx Radiation Test Consortium, Tech. Rep. 1, 2004.
[2] G. Allen, G. Swift, and C. Carmichael, "Virtex-4VQ static SEU characterization summary," Xilinx Radiation Test Consortium, Tech. Rep. 1, 2008.

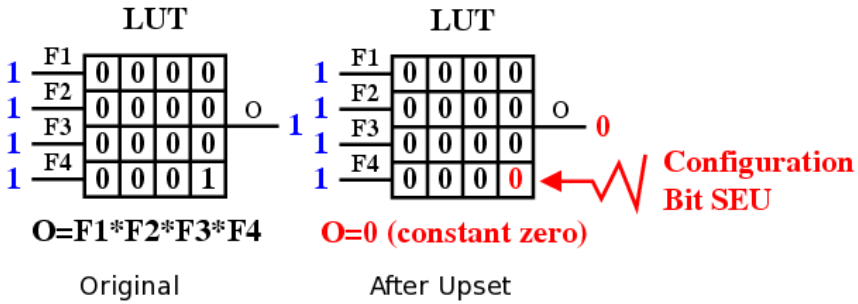
Background:
SRAM FPGAs in Space

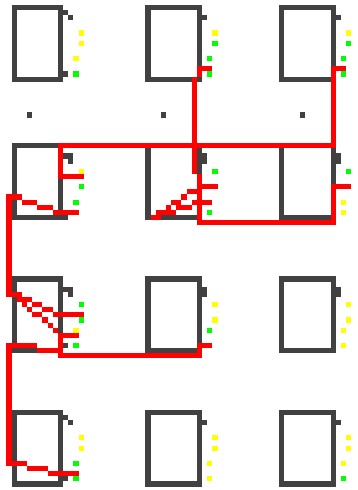
- **Both the circuit and the circuit state are stored in SEU-susceptible SRAM.**
  - An SEU could change the circuit functionality
  - An SEU could change the circuit routing
  - An SEU could change intermediate processing values
  - An SEU could cause the device to *temporarily* become non-functional
- **All SEUs on FPGAs are non-destructive, but can make fault-tolerant computing challenging**
- **Starting in V4, starting to see SETs in "hard" cores: DCMs, DSPs**

Background:
Failures that Affect Circuit Functionality

- **Routing Vulnerabilities**
  - Mux select lines change values
  - Pips and buffers open or short
- **Logic Vulnerabilities**
  - LUT value changes
  - LUT control bit changes
- **Tie-off Vulnerabilities**
  - Implicit logic constants: half-latches
  - Explicit logic constants: constant LUTs and VCC posts







Background:
Failures that Affect Circuit State

- **Maintaining state is difficult**
  - SEUs in circuit functionality can affect state
  - SEUs in user memory (flip-flops, BlockRAM) can affect state
- **Routing of global signals is particularly vulnerable**
  - Clock and reset trees provide a large target for SEUs
  - Most common method for handling logical constants elevates them to global signals

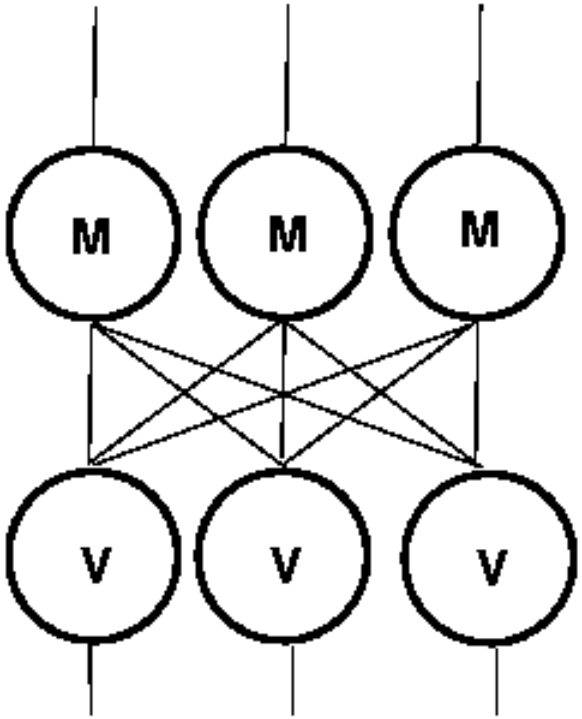
Background:
Failures that Affect Device Functionality

- **SEFIs that affect all devices [1][2]:**
  - JTAG TAP Controller: numerous failure states
  - SelectMAP: unable to read from or write to SelectMAP interface
  - Power-on-reset: configuration is cleared and DONE pin is driven low
- **Virtex-4 specific SEFIs:**
  - FAR and Readback SEFIs that mimic SelectMAP SEFIs
  - Global Signal SEFI: umbrella SEFI that covers SEFIs in Global Set/Reset, Global Write Enable and Global Drive High signals
  - Scrub SEFI: SEU in the control logic while performing on-line reconfiguration, causes a high current state

[1] G. M. Swift, “Virtex-II static SEU characterization,” Xilinx Radiation Test Consortium, Tech. Rep. 1, 2004.
[2] G. Allen, G. Swift, and C. Carmichael, “Virtex-4VQ static SEU characterization summary,” Xilinx Radiation Test Consortium, Tech. Rep. 1, 2008.

Background:
Mitigation and Repair Methods

- **Even a single SEU *could* cause the circuit to output bad data**
- **Accumulating SEUs increases the likelihood that the output data is corrupted and increases a device's current draw**
- **Mitigation and repair of SEUs is needed**
  - To date, best option for mitigation SEUs is to mask them through triple-modular redundancy (TMR)
  - On-line reconfiguration, called *scrubbing*, used to remove SEUs
  - Off-line reconfiguration used to remove SEFIs
- **Testing the mitigation scheme is necessary**



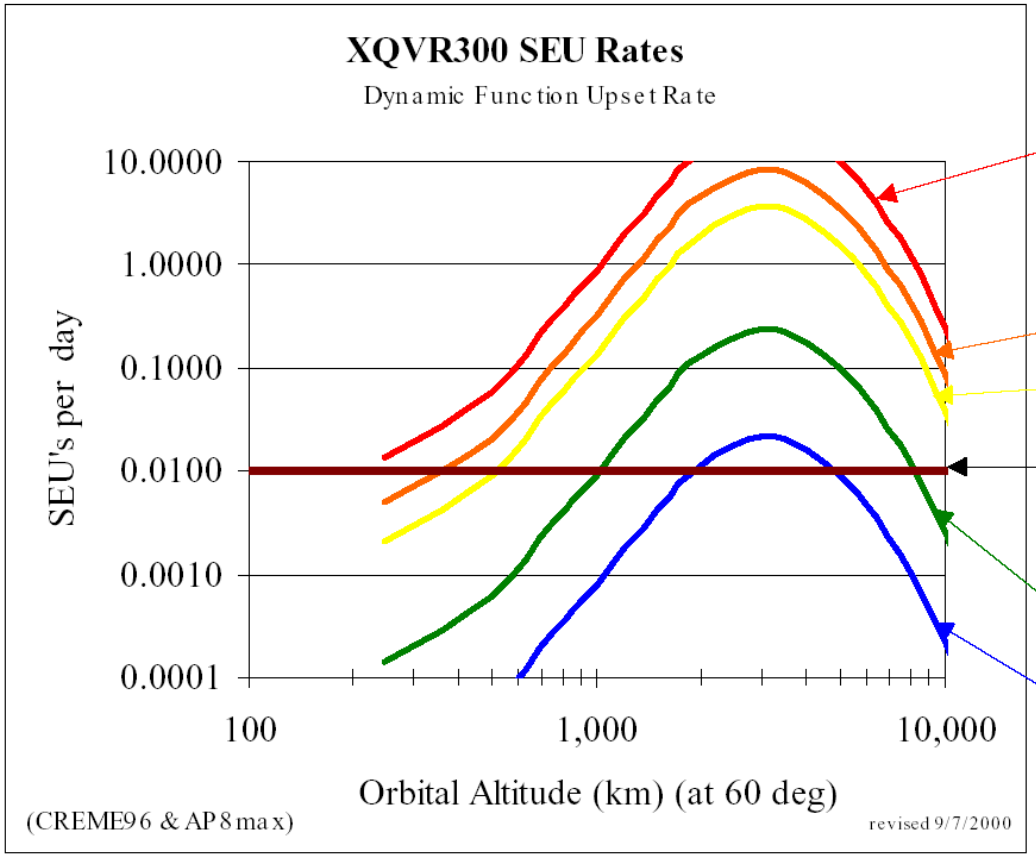
TMR-Protection for a Circuit Module

[1] K. Morgan, M. Caffrey, P. Graham, E. Johnson, B. Pratt, and M. Wirthlin, "SEU-induced persistent error propagation in FPGAs," IEEE Transactions on Nuclear Science, vol. 52, no. 6, pp. 2438 – 45, 2005.

[2] "Xilinx TMRTool user guide," on web: http://www.xilinx.com/products/milaero/ug156.pdf.

[3] C. Carmichael, M. Caffrey, and A. Salazar, "Correcting single event upsets through virtex partial configuration: Application note 216," on web: http://www.xilinx.com, 2000.

Background:
SRAM FPGA Radiation Effects Characterization Error Rates



Upset rate estimates with No mitigation:

Calculated worst-case, by including all bits

Measured dynamic upsets without design mitigation:

with No bitstream cause

with bitstream cause

IEEE 1156.4-1997 Standard

Upset rates Using mitigation techniques:

Benefit of TMR & PRC

Newest data including TMR, scrub, and suppressing unused nodes

Improvement of 1100x compared to worst-case

FPGA Background Summary

- Both the circuit functionality and state are in SEU-susceptible memory
- SEUs can change the LUT function, the routing, tie-offs, the user memory, or cause the device to malfunction
- TMR can be used to mask SEUs that affect circuit functionality and circuit state
- On-line reconfiguration can be used to remove SEUs that affect the circuit functionality and state, and to keep SEUs from accumulating
- Off-line reconfiguration can be used to remove SEFIs that affect the device's functionality

Outline

- Background
- **Static Testing**
  - Test Methodology: Test Fixtures, Angular Testing, Multiple-bit Upset Testing, and Micro-SEFIs
  - Analyzing Test Data: Data Cleansing, Analysis
  - Test Results
- Dynamic Testing of Mitigated Circuits
  - Modeling Tools
  - Fault Injection Tools
  - Accelerator Testing
  - Test Results

Static and Dynamic Testing of FPGAs

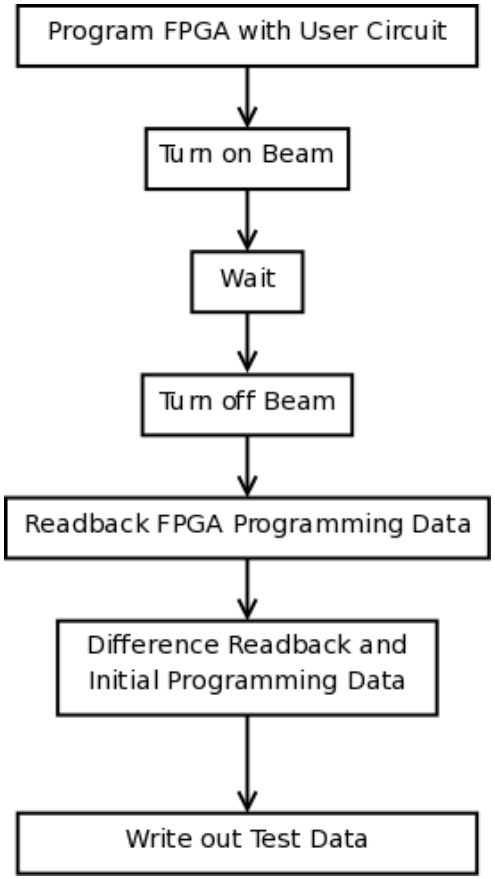
- **“Typical” definitions of static and dynamic testing:**
  - Static: unbiased, unclocked
  - Dynamic: biased, clocked
- **In both static and dynamic testing of FPGAs the device is clocked and biased**
  - Static: programming memory upsets observed, but input and output vectors not used for functional testing
  - Dynamic: input and output vectors used, errors in output vectors detected for functional testing
- **For the XRTC, dynamic testing is used to group various testing: resource-specific testing (DCMs, IOBs, FFs), and some designs (TMR-Protected MicroBlaze)**
- **For LANL, the circuit is the focus for dynamic testing: circuit will highlight a specific fault that we are interested in (DCEs, Half Latches) or circuit is expected to be used in space.**

Static Testing:
Test Methodologies

- **Two basic concepts between taking data:**
  - Complete capture methodologies that take one sample between turning the beam on and off
  - Continuous capture methodologies that take many samples between turning the beam on and off
- **There are advantages and disadvantages to both methodologies**
- **Not all test fixtures can handle both types of methodologies**

Static Testing Methodologies:
Complete Capture Systems

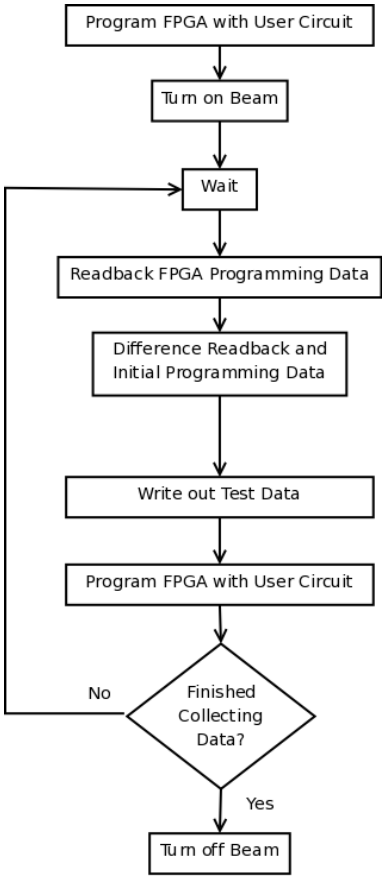
- **Advantages:**
  - Can easily determine the amount of ions per sample
  - Easiest to implement
- **Disadvantages:**
  - Time-consuming human interaction with beam
  - Optimizing the beam interaction often means taking a lot of data per sample or testing at a low flux, leading to problems with data analysis or wasting more time



Complete Capture Methodology

Static Testing Methodologies:
Continuous Capture Systems

- **Advantages:**
  - Time-consuming human interaction with beam minimized
  - Can optimize the amount of data per sample
  - Can test at higher fluxes
- **Disadvantages:**
  - Not all upsets are recorded:
    - Upsets that occur between reading back and programming the bitstream will be lost
  - Hard to determine the amount of fluence per sample
    - Fluence must be adjusted to remove the “unrecorded” upsets
- **Even with these disadvantages, LANL and the XRTC both use this methodology**



Continuous Capture Methodology

Static Testing:
Test Fixtures

- **Test fixtures consist of both hardware and software components**
- **Hardware provides the support for reading (*readback*) and writing (*programming*) the FPGA**
  - There are a number of commercial and custom options available that support one or both reconfiguration ports [1-4]
- **Software implements the FPGA-specific part of the test methodology**
  - Controlling when to read or write to the device
  - Differencing data
  - Saving data to disk
  - Software options including iMPACT or custom code

[1] G. Allen, G. Swift, and C. Carmichael, "Virtex-4VQ static SEU characterization summary," Xilinx Radiation Test Consortium, Tech. Rep. 1, 2008.

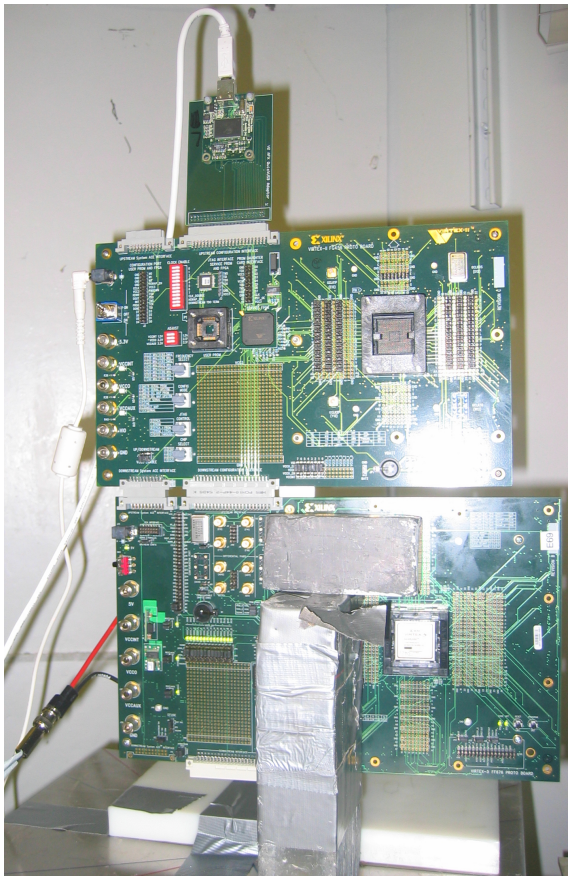
[2] J. George, R. Koga, G. Swift, G. Allen, C. Carmichael, and W. Tseng, "Single Event Upsets in Xilinx Virtex-4 FPGA devices," in Radiation Data Workshop of the Nuclear and Space Radiation Effects Conference, 2006, pp. 109–113.

[3] M. Berg, C. Perez, and H. Kim, "Investigating Mitigated and Nonmitigated Multiple Clock Domain Circuitry in a Xilinx Virtex-4 Field Programmable Gate Arrays," in the Single-Event Effects Symposium, 2008.

[4] Quinn, H., P. Graham, M. Wirthlin, B. Pratt, K. Morgan, M. Caffrey, J. Krone. "A Test Methodology for Determining Space-Readiness of Xilinx SRAM-based FPGA Devices and Designs. submitted to *IEEE Transactions on Instrumentation and Measurement,* Oct. 2008.

Static Testing:
LANL Test Fixture

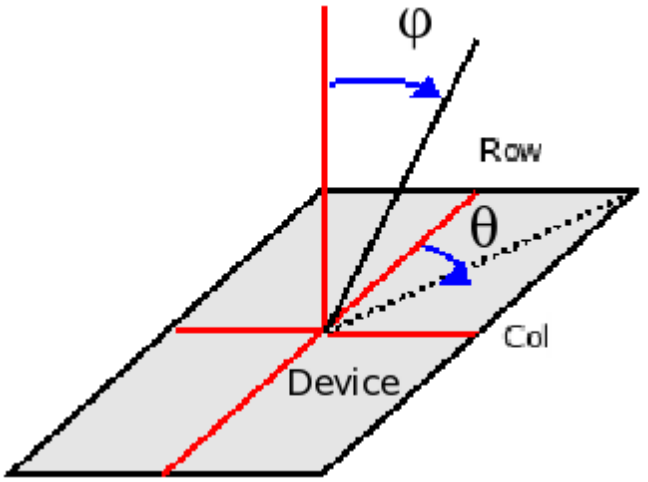
- **Hardware:**
  - 2 Xilinx AFX Development Boards (1 device-specific board, 1 V-II for controlling the test)
  - USB card to connect to laptop
- **Software:**
  - Controls off-line reconfiguration and readback through SelectMAP port
  - Saves differential bitstreams
  - Minimal statistics
- **Advantages:**
  - Flexible: Can develop test fixtures for new parts in 1-2 weeks
  - Small: Can carry in a suitcase
  - Cheap: Boards cost significantly less than fabing a custom board
- **Disadvantages:**
  - SEFIs: Currently does not record detailed information
  - Software sampling: slower than hardware sampling



Virtex-5 Test Fixture

Static Testing:
Angular Testing

- **Angular testing is often used to increase the beam energy by increasing the amount of silicon the ion traverses**
- **The unique layout of these devices creates different data sets for different θ's**



Θ and Φ Relative to Device

Static Testing:
Multiple-Bit Upset Testing

- **Because MBUs can violate the assumption that only one error exists in the system at a time, it is possible to defeat TMR-protected circuits with MBUs.**
- **During static testing, we test devices for MBUs to determine the likelihood of MBUs to affect TMR-protected devices**
- **If there is too many upsets/readback, it is possible that the MBU data set will be contaminated with *coincident single-bit upsets***
  - Must determine the co-incident SBU rate while testing for MBUs [1]

[1] Quinn, H., P. Graham, M. Wirthlin, B. Pratt, K. Morgan, M. Caffrey, J. Krone. "A Test Methodology for Determining Space-Readiness of Xilinx SRAM-based FPGA Devices and Designs. submitted to *IEEE Transactions on Instrumentation and Measurement,* Oct. 2008.

Static Testing:
Micro-SEFIs

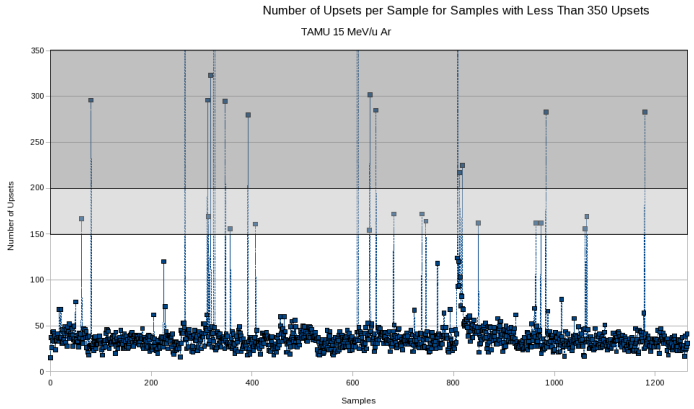
- **A micro-SEFI is a not completely understood phenomena that affects the Virtex family parts**
  - Has been observed in the V-II, V-4, and V-5
  - Has been increasing in cross-section for each new device
  - Seems to locally reconfigure the device, but not certain how
  - Often causes 300-500 upsets, but likely is design-dependent
- **Because the micro-SEFI is probably not caused by an SEU to the configuration memory cell, we eliminate the micro-SEFIs from the data set**
  - The easiest way to eliminate this data is statistically, so must tune the number of upsets/readback to make micro-SEFIs statistical outliers

Analyzing Static Test Data:
Overview

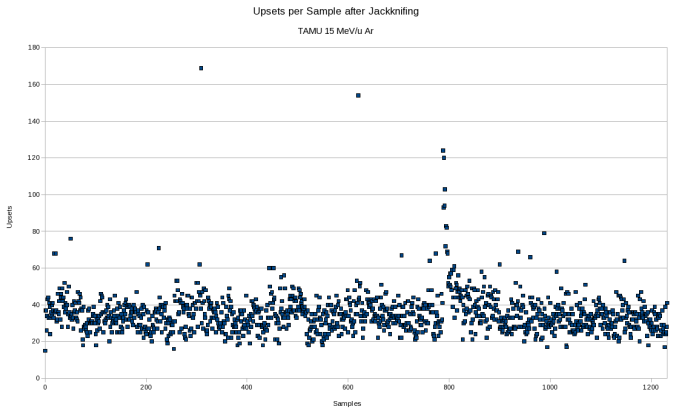
- **Each sample is analyzed separately: statistics are accumulated but samples are not analyzed as an aggregate**
- **Three step process:**
  1. Event classification
  2. Correlate data to physical locations
  3. Analyze data for MBUs and affected memory cell type

Analyzing Static Test Data:
Event Classification

- **Remove all of the SEFIs from the data**
- **First cut of data is made by thresholding upsets**
- **Second cut of data is made by Jackknifing [1]:**
  - Examine small windows of data to determine statistical outliers
  - Unlike moving averages, able to easily adapt to how the beam fluctuates both microscopically and macroscopically
  - Algorithm tuned to accept data with a low standard deviation
- **Third cut of data is made during the analysis to remove samples with "unusually large MBUs"**



After First Cut of Data

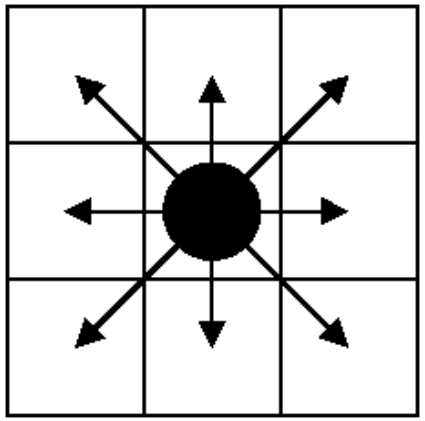


After Second Cut of Data

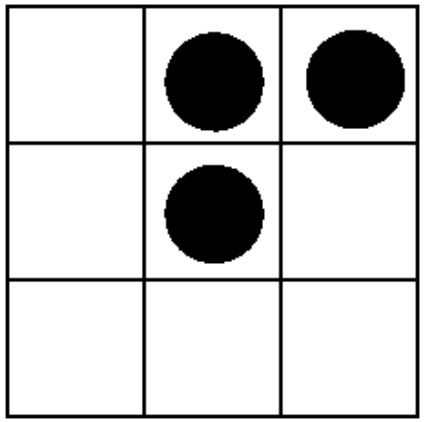
[1] B. Efron and R. Tibshirirani, An Introduction to the Bootstrap. Chapman and Hall/CRC, 1993.

Analyzing Static Test Data:
Physical Correlation

- **Upsets are translated to the physical layout:**
  - Bits that are within each other's adjacency neighborhood are classified as MBUs
  - FPGA resource (CLB, BRAM, etc) is determined
- **Statistics on size, frequency and memory cell type affected are determined**
- **Analyzed data is used to determine cross-section**



Adjacency Neighborhood

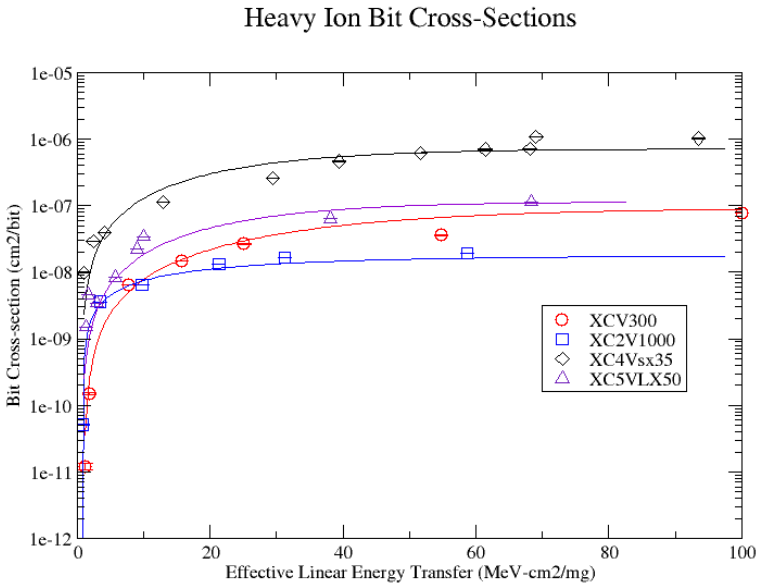


Multiple-Bit Upset Cluster

Static Test Results:
Bit Cross-Sections

Device	Energy (MeV)	σ_{bit} (cm²/bit)
XCV1000	63.3	$1.32 \times 10^{-14} \pm 2.69 \times 10^{-17}$
XC2V1000	63.3	$2.10 \times 10^{-14} \pm 4.64 \times 10^{-17}$
XC4VLX25	63.3	$1.08 \times 10^{-14} \pm 2.71 \times 10^{-17}$
XC5VLX50	65	$7.57 \times 10^{-14} \pm 1.35 \times 10^{-15}$
XC5VLX50	200	$1.07 \times 10^{-13} \pm 5.37 \times 10^{-16}$

Proton Data



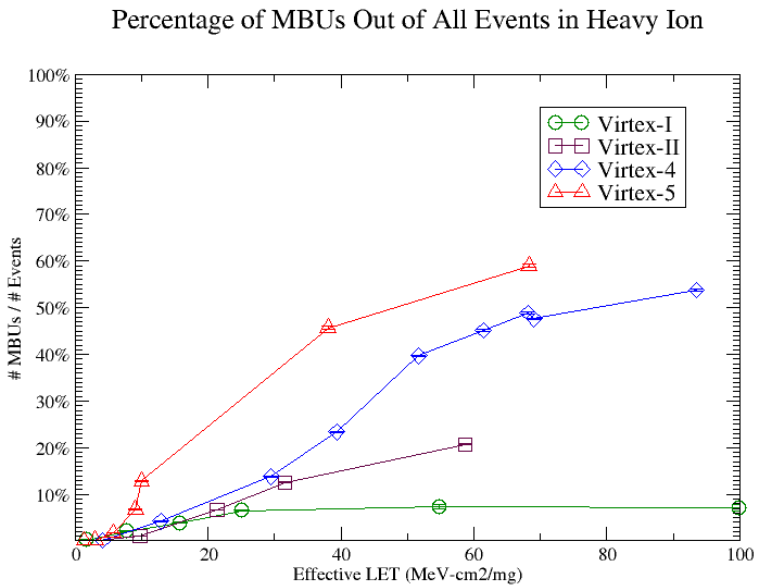
Heavy Ion Data

Static Test Results:
MBU Data

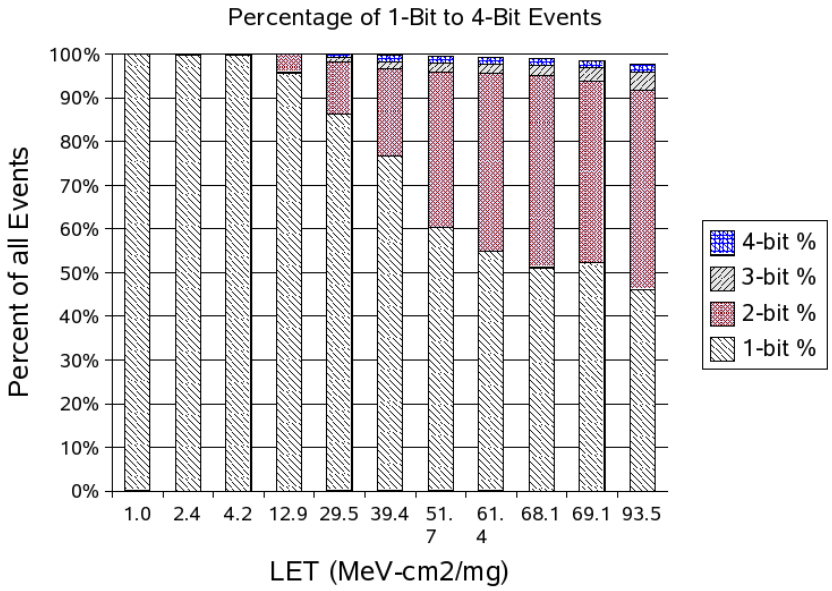
Device	Energy (MeV)	1-Bit Events	2-Bit Events	3-Bit Events	4-Bit Events
XCV1000	63.3	99.96 %	0.04%	0.00%	0.000%
XC2V1000	63.3	98.42%	1.16%	0.01%	0.001%
XC4VLX25	63.3	96.44%	2.99%	0.05%	0.005%
XC5VLX50	65	94.23%	5.43%	0.30%	0.03%
XC5VLX50	200	89.86%	8.79%	0.92%	0.43%

Proton Data

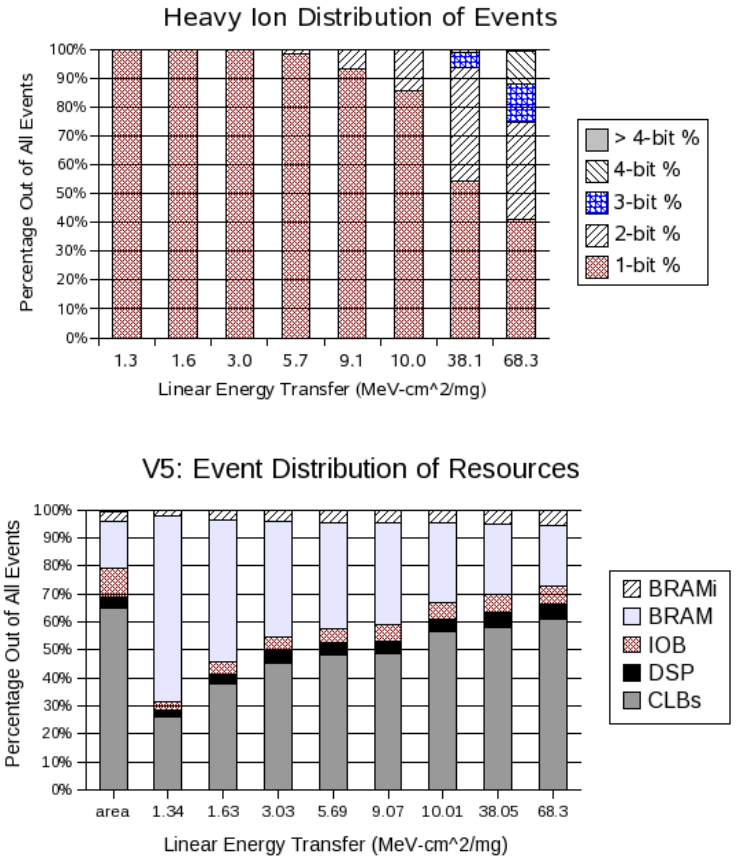
Heavy Ion Data



Static Test Results:
Distribution of Events

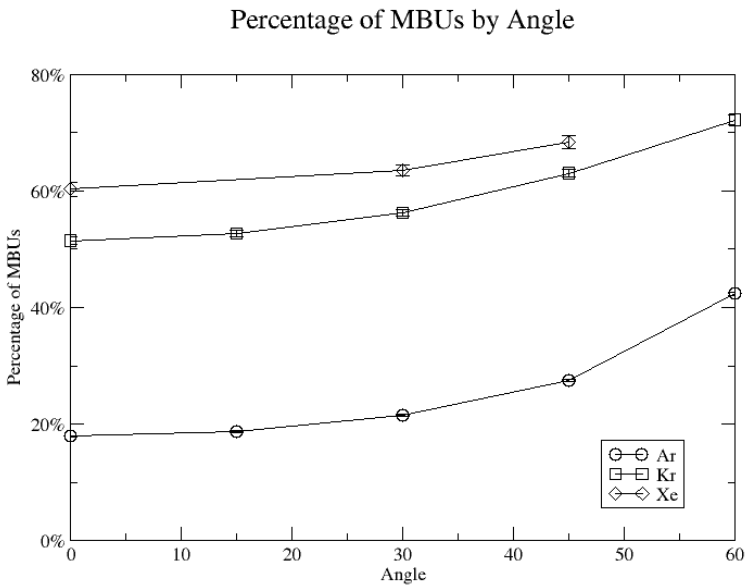


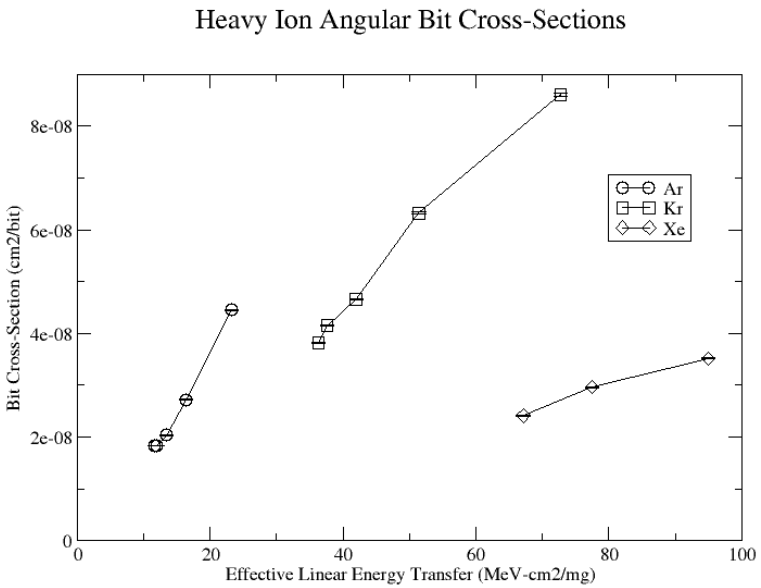
V4 Data



V5 Data

Static Test Results:
V5 Angular Data





MBUs

Bit Cross-Sections

Static Testing Summary

- **Two test methodologies: complete and continuous capture**
- **Various test fixtures available**
- **Post-data collection analysis of data can include event classification, correlation to physical layout, and MBU analysis**

Outline

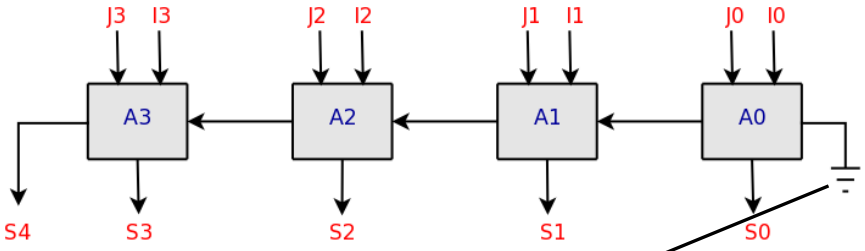
- Background
- Static Testing
  - Test Methodology: Test Fixtures, Angular Testing, Multiple-bit Upset Testing, and Micro-SEFIs
  - Analyzing Test Data: Data Cleansing, Analysis
  - Test Results
- **Dynamic Testing of Mitigated Circuits**
  - Modeling Tools
  - Fault Injection Tools
  - Accelerator Testing
  - Test Results

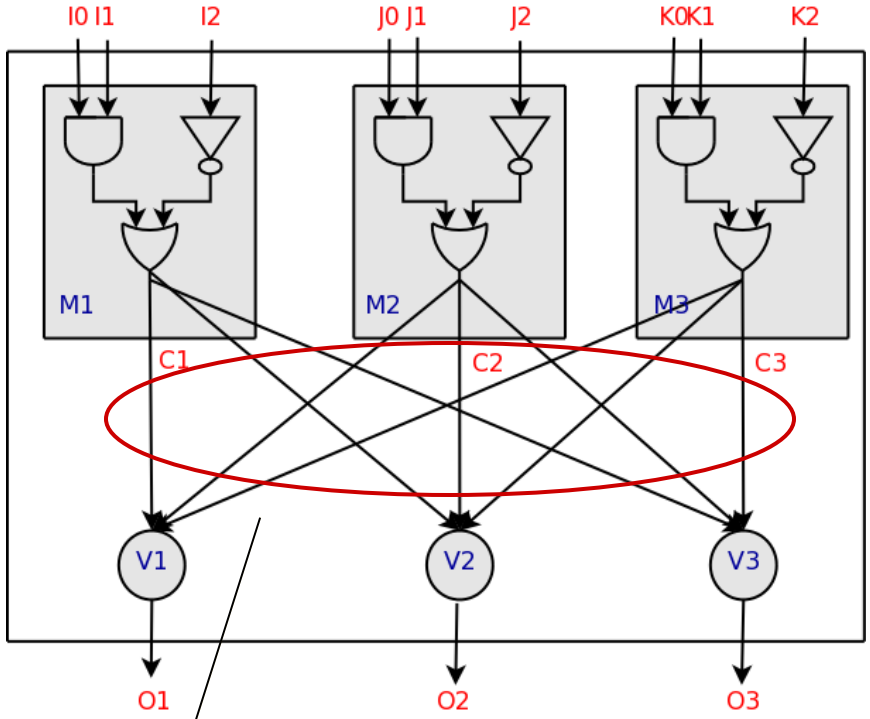
Dynamic Testing:
Introduction

- **Dynamic testing is necessary to determine:**
  - If mitigation was applied properly
  - If there are MBU-related or placement-related issues
  - What the remaining cross-section is on a partially mitigated design
  - Where are the location of *sensitive* bits in a partially mitigated design
  - If there is outstanding architectural issues that static testing did not determine, for example:
    - Cross-sections of memory not directly visible from programming interface
    - SEFI modes not visible through programming interface
- **Dynamic testing of mitigated circuits:**
  - Fully mitigated circuits could have placement-related issues
  - Partially mitigated circuits will have both placement-related issues and unprotected cross-section

Dynamic Testing:
Types of Errors Expected in FPGA User Designs (1 of 2)

- **Placement-related issues in fully TMR-protected designs**
  - Logical constants (that implement 1s and 0s) , or
  - The placement of the different TMR modules/voters in too close proximity.



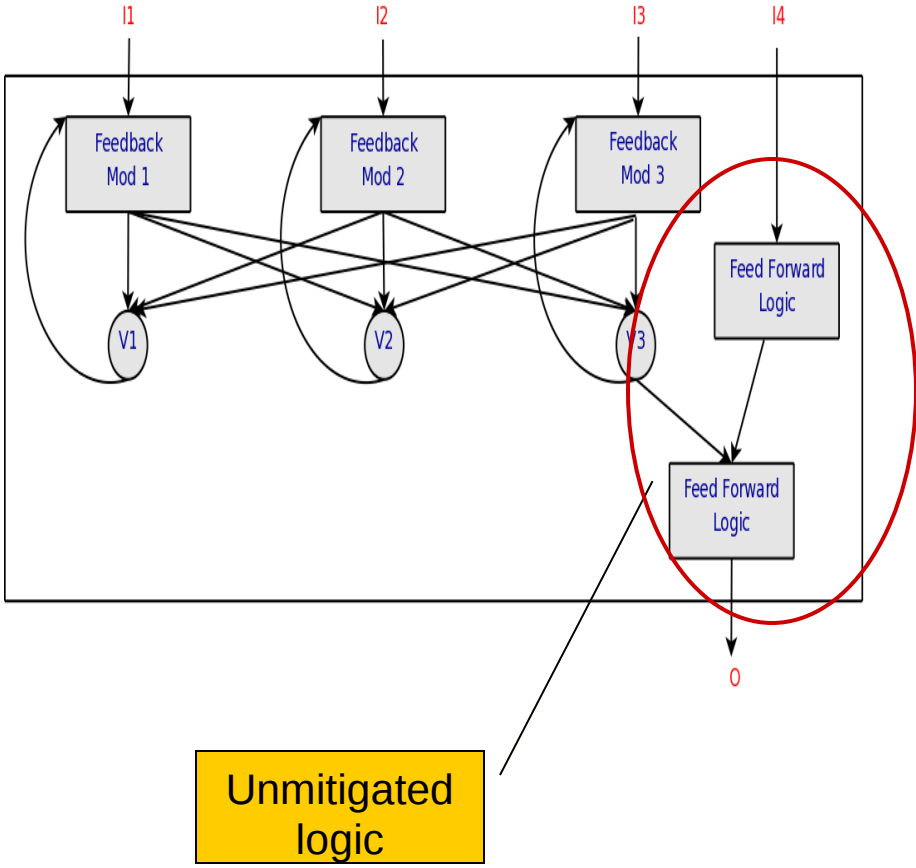


Logical constant provides the ground

An MBU in these routes could sever the modules from the voters.

Dynamic Testing:
Types of Errors Expected in FPGA User Designs (2 of 2)

- **Logic and routing in unmitigated portion of a partially TMR-protected designs will have**
  - Sensitivities in the logic can be affected by workload (input vector sets) but are static in quantity.
  - Sensitivities in the routing can be affected by placement and routing (shortening and lengthening paths), but can only be implemented by triplicating all unprotected logic.
  - Problem locations within a user design that cause SEUs to manifest output errors are called *sensitive bits*.
- **Placement-related issues in the TMR-protected portion**



Dynamic Testing:
Testing FPGA User Designs

- **Current "gold standard" is to do pre-launch testing of user designs through radiation experiments at a particle accelerator.**
  - Space-qualifying a design could take days worth of time and thousands of dollars at an accelerator.
  - Radiation-induced faults are statistical in nature which further complicates the time and expense of radiation-experiments
  - Hard to correlate errors to flaws in the user design.
- **Designers need faster, cheaper and more uniform methods of testing user designs.**
  - Modeling tools and fault injection tools can be useful in these regards, and
  - Radiation experiments used only to validate these results.

Dynamic Testing:
Modeling Tools Background and Related Work

- **Reliability analysis often done using modeling tools.**
  - Allows designers to focus on creating a model of the system, while the modeling tool handles the analysis.
- **Reliability analysis tools often use analytical, Boolean network or probabilistic systems.**
  - There are limitations in these tools, such as the transformation of circuit descriptions to intermediate probabilistic models and the computational complexity of analyzing large circuits.

[1] J. A. Abraham and D. P. Siewiorek, "An algorithm for the accurate reliability evaluation of triple modular redundancy networks," *IEEE Transactions on Computers*, vol. 23, no. 7, pp. 682–692, July 1974.

[2] S. Krishnaswamy, G. F. Viamontes, I. L. Markov, and J. P. Hayes, "Accurate reliability evaluation and enhancement via probabilistic transfer matrices," in *Design, Automation and Test in Europe (DATE'05)*, vol. 1. New York, NY, USA: ACM Press, 2005, pp. 282–287.

[3] C. Hirel, R. Sahner, X. Zang, and K. Trivedi, "Reliability and performability using SHARPE 2000," in *11th Int'l Conf. on Computer Performance Evaluation: Modeling Techniques and Tools*, vol. 1786, 2000, pp. 345–349.

[4] G. Norman, D. Parker, M. Kwiatkowska, and S. Shukla, "Evaluating the reliability of NAND multiplexing with PRISM," *IEEE Transactions on CAD*, vol. 24, no. 10, pp. 1629–1637, 2005.

[5] F. V. Jensen, *Bayesian Networks and Decision Graphs*. New York: Springer-Verlag, 2001.

[6] D. Bhaduri, S. K. Shukla, P. S. Graham, and M. B. Gokhale, "Reliability analysis of large circuits using scalable techniques and tools," *IEEE Transactions on Circuits and Systems - I: Fundamental Theory and Applications*, vol. 54, no. 11, pp. 2447 – 60, November 2007.

Dynamic Testing:
The Scalable Tool for the Analysis of Reliable Circuits (STARC) (1 of 2)

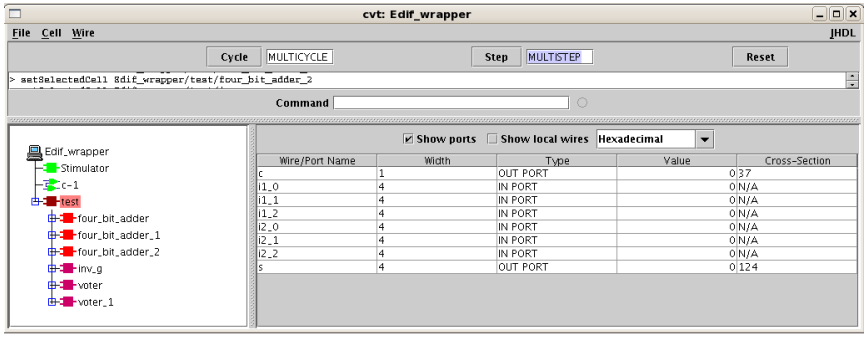
- **Despite being similarly named, the STAR tool [1] from Politecnico di Torino works at a much different level of abstraction than STARC**
- **STARC addresses the limitations of traditional reliability analysis tools by:**
  - Using the industry-standard Electronic Design Interchange Format (EDIF) circuit representations for the circuit model,
  - Not using input vector sets,
  - Using memoization to reduce computational complexity, and
  - Using combinatorial reliability calculations.
- **By using EDIF, the designer can address reliability problems early in the design process, even if the design is not complete, the design does not work, and the hardware is not available.**
  - There is no placement-related information available in EDIF.
  - Currently adding in ability to use XDL information for placement information
- **Without input vectors, STARC calculates the worst-case failure rate.**

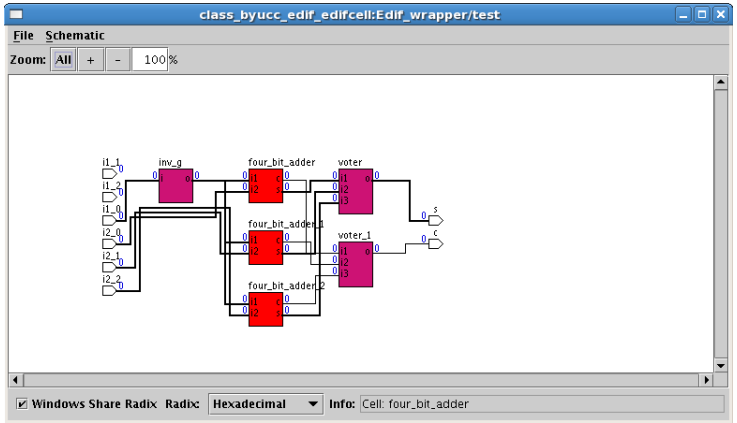
[1] L. Sterpone, M. Violante, R. H. Sorensen, D. Merodio, F. Sturesson, R. Weigand, and S. Mattsson, "Experimental Validation of a Tool for Predicting the Effects of Soft Errors in SRAM-Based FPGAs," Transactions on Nuclear Science, Vol. 54, No. 6, pp. 2576–2583, 2007.

Dynamic Testing:

The Scalable Tool for the Analysis of Reliable Circuits (STARC) (2 of 2)

- **STARC was designed specifically to help designers find problems in TMR-protected designs.**
  - The mitigated partition is analyzed to determine if three modules are present and equivalent and if three voters are present.
  - The unmitigated partition is analyzed to determine the quantity of sensitive bits.
- **The output of STARC provides warnings and information about the design, including**
  - Feedback loops that are improperly mitigated,
  - A list of unmitigated logic, and
  - Warnings about the use of single points of failures, logical constants and functionally nonequivalent modules.





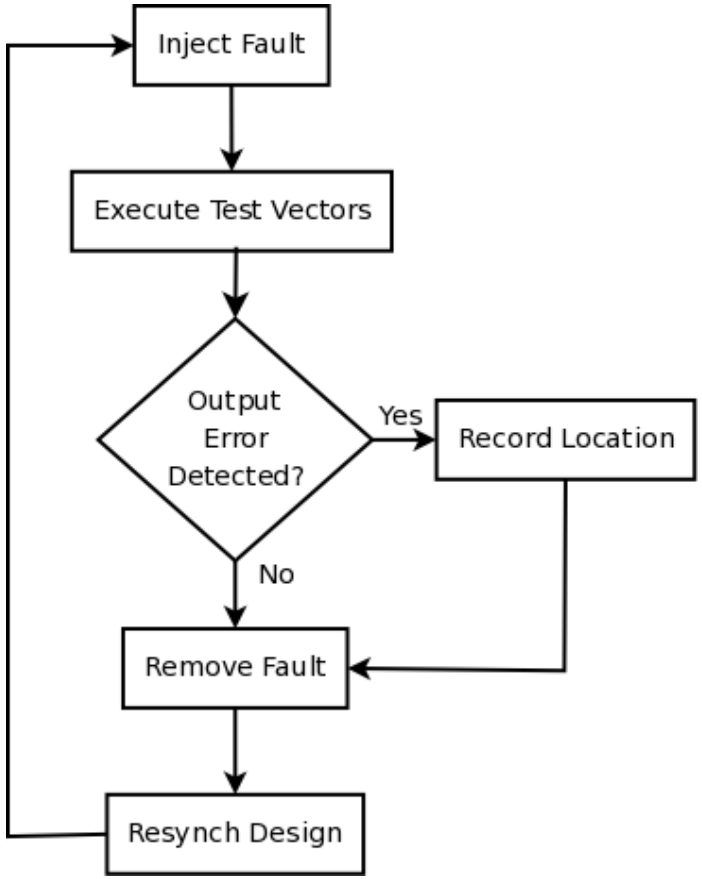
Dynamic Testing:
Fault Injection Background and Related Work

- **The reconfiguration ports for SRAM-based FPGAs can be used for fault injection by intentionally corrupting the configuration memory.**
  - Unlike modeling tools, uses actual hardware.
  - Can provide a 70-97% coverage of accelerator testing
    - Difficult to ensure the same circuit state in both accelerator and benchtop testing
  - Possible to fault inject to all of the configuration, except the user flip-flops.
- **While LANL designed the first FPGA fault injection tool, several now exist**
  - Each fault injection test fixture supports a specific device and a specific input/output, clock and reset structure
  - If fault injection test fixture matches the flight hardware, good predictor of on-orbit behavior.
    - Building fault injection capability into flight hardware would make the best test fixture

- M. Alderighi, F. Casini, S. D'Angelo, M. Mancini, S. Pastore, G. Sechi, and R. Weigand, "Evaluation of single event upset mitigation schemes for sram based fpgas using the flipper fault injection platform," in *Proc. of the 22th IEEE Int. Symp. Defect and Fault Tolerance in VLSI Systems (DFT07)*, September 2007, pp. 105–113.
- M. Berg, C. Perez, and H. Kim, "Investigating mitigated and nonmitigated multiple clock domain circuitry in a Xilinx Virtex-4 field programmable gate arrays," in Single-event effects symposium, 2008.
- G. Swift, C. W. Tseng, G. Miller, G. Allen, and H. Quinn, "The use of fault injection to simulate upsets in reconfigurable FPGAs," 2008, submitted to the military and aerespace programmable logic devices conference (MAPLD08).

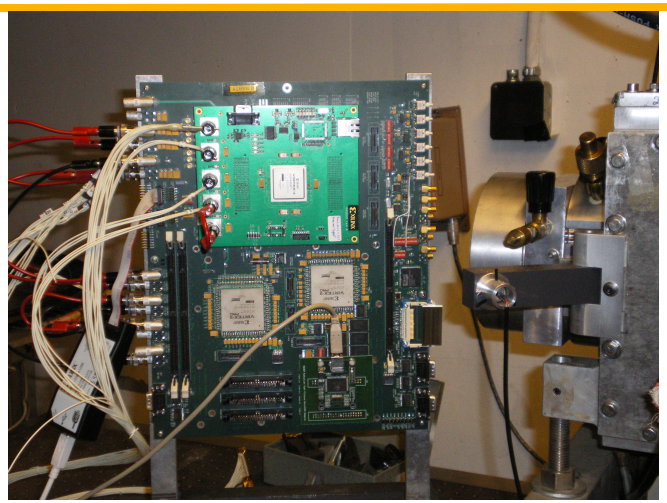
Dynamic Testing:
Fault Injection Software Test Fixture

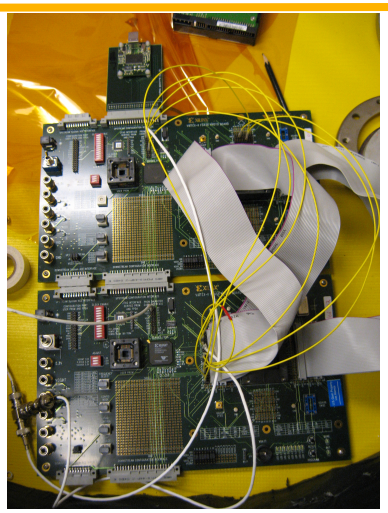
- **Standard algorithm for many test fixtures**
  - Can support a variety of hardware test fixtures.
  - If you can avoid the fault injecting into the SEFIs, can use on-line reconfiguration.
- **Good test coverage is dependent on the number of input test vectors used.**
  - Without user-provided input vectors covering the input test vector set and maintaining good speed is a challenge.
  - LANL test fixture uses random input vector generation and covers between 250,000-500,000 input vectors per test, which can be increased.
- **Resynchronizing design between injecting faults is important for proper fault attribution.**



Fault Injection Algorithm

Dynamic Testing:
Fault Injection Hardware Test Fixture





Multiple-FPGA Fault Injection Hardware Test Fixture

- **Two predominant test fixture designs:**
  - Single FPGA systems with more software control, and
  - Multiple FPGA systems that are more hardware controlled.
- **Single FPGA systems have simpler hardware, slower to determine output errors, not extensible to the accelerator test fixture.**
- **Multiple FPGA systems have more complex hardware, quicker to determine output errors, easily altered for the accelerator test fixture.**

Dynamic Testing:
Fault Injection Analysis

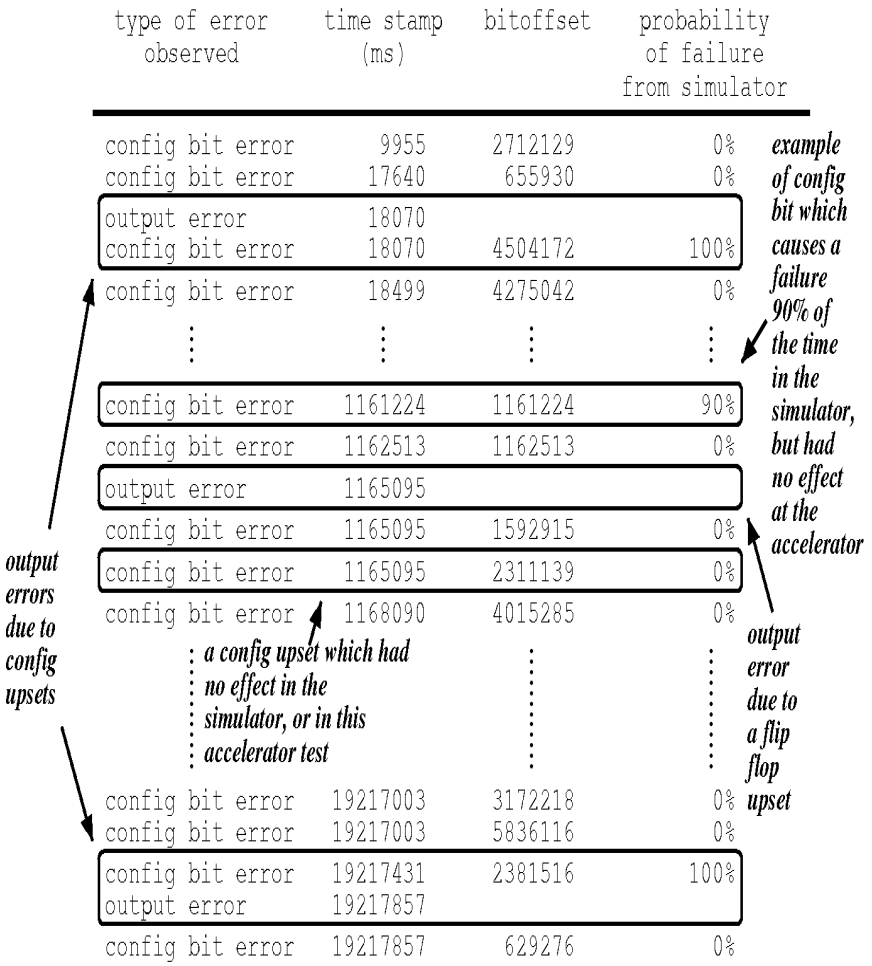
- **Ideally, a completely mitigated design should have no output errors. If output errors occur, the cause should be investigated.**
- **In a partially mitigated design, output errors are expected.**
- **Results from the fault injection testing must be analyzed to determine what part of the circuit failed**
  - Static testing analysis tool can be used to correlate readback location to physical location
  - FPGA editor can be used to correlate the physical location to the circuit
- **In a mitigated circuit, the designer needs to determine if an observable output error was caused by an unmitigated portion of the device or whether there is a problem in the mitigation scheme or whether there is an architectural problem**

Dynamic Testing:
Accelerator Testing Background

- **By performing accelerator testing after modeling and fault injection should already know the areas of the circuit and the locations of sensitive bits that cause output errors.**
- **Many fault injection test fixtures can be modified to serve as the accelerator test fixture – just remove the fault injection protocol!**
- **Controlling the rate of accelerator-induced faults very difficult**
  - The arrival time of faults is a Poisson random process.
  - Want to balance the chance of not getting a fault with the chance of getting too many faults for each loop of the test fixture algorithm.
- **Removing SEUs quickly is important**
  - Can use on-line reconfiguration for fastest response time.
  - Might need to use off-line reconfiguration for difficult to remove errors or SEFIs.

Dynamic Testing:
Accelerator Testing Correlating Results

- **Correlating accelerator results to fault injection results can be tricky.**
  - Output error in log could precede or follow configuration bit error
  - Analyze the location of upset data in "windows" around the output error to try to match a location to the fault injection results.
- **Some times errors cannot correlated to fault injection data:**
  - Areas of the device not fault injected, such as user memory.
  - Multiple independent upsets
  - Data dependent errors
- **Accelerator test can be "played back" through the fault injector to determine the repeatability or the cause of the output error.**



Dynamic Testing:
Results

- **The circuit is an adder tree that was designed to highlight placement-related issues in triplicated designs.**
- **The user design was implemented for a Xilinx Virtex-II FPGA.**
- **The results from all three test methodologies:**

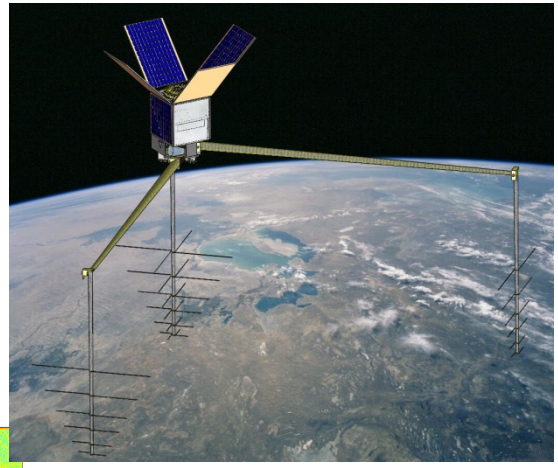
Test	Cost	Time	Results
STARC	$0	<1 minute	Testing determined the circuit was properly triplicated, but had placement-related issues.
Fault Injection	$6,000	14 hours	Testing found 285 single bit locations, 18,733 2-bit locations, 11,264 3-bit locations, and 19,464 4-bit locations that had placement-related issues that caused output errors.
Accelerator	$1,700 *	2 hours*	50 output errors observed; 16 output errors were attributed to placement-related issues and 88% were correlated to known fault injection locations.

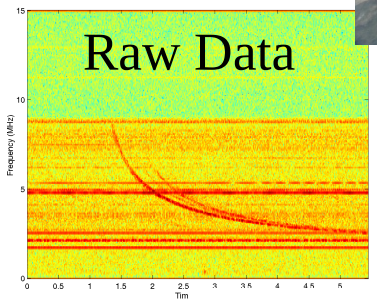
* Completing the single bit test would take 385 hours, 192 FPGAs, and $288,000.

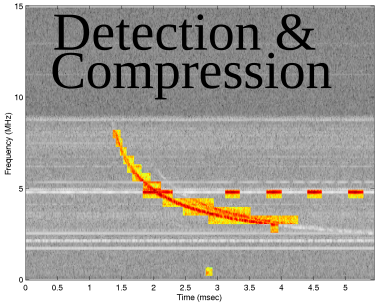
Dynamic Testing Summary

- **A three-tiered methodology was presented for FPGA user design testing:**
  - Modeling provides quick support to designers,
  - Fault injection provides uniform tests of the design on the hardware, and
  - Accelerator testing provides validation of results.
- **Results show that by using these three test methodologies together that the overall cost of accelerator testing can be minimized.**

Cibola Flight Experiment: Demonstration of Fast On-Board Processor with COTS Parts

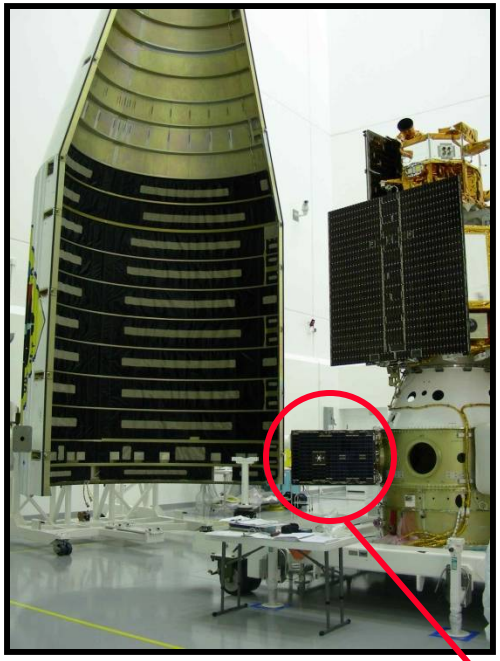


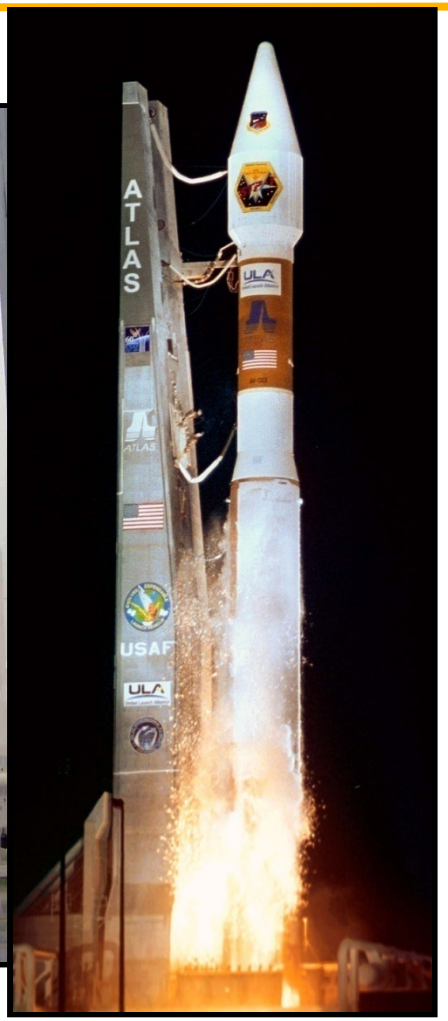




- **Launched March 2007**
- **Orbit: Circular 560 Km, 35.4 degree inclination**
- **Software Radio:**
  - Four channels, 20 MHz bandwidth each
  - Tunable from 100 to 500 MHz,
  - 3-board, 9 Xilinx Virtex FPGA 300-Gop/sec (peak) re-configurable computer (RCC)
  - 4-element antenna array

CFE Project Highlights: Space Launch by STP

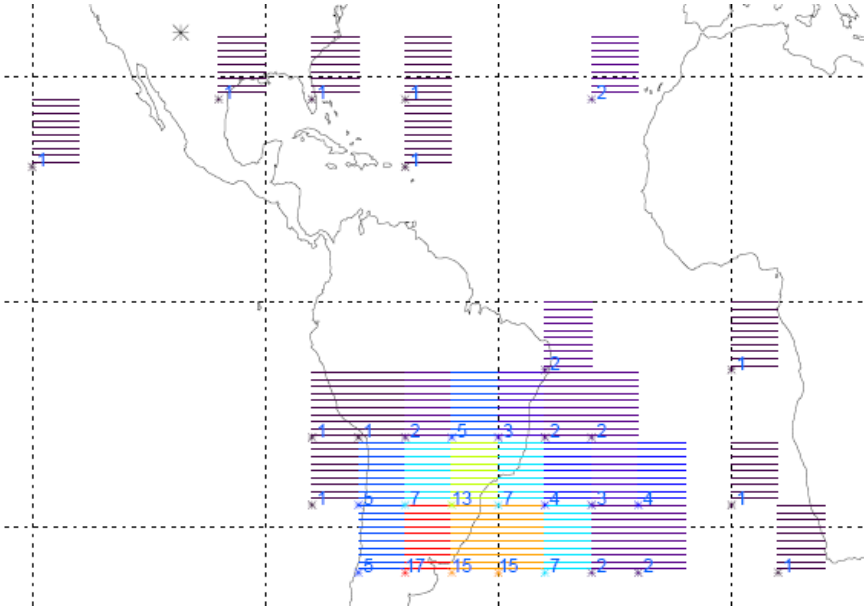




CFE mated to launch adapter

CFE Operational Successes to Date

- **>10,000 Experiments**
- **>24 uploads of new/modified applications**
- **From 03/01/07 to 04/15/08**
  - 12 GB State of Health Data
  - 181 GB Science Data
  - 1578.3 hrs of radio operations
- **SEUs Measured**
  - 141 SEUs / 565 device days = .25 upsets/device day
  - SEU mitigation strategy is working perfectly
  - >90% of upsets in South Atlantic Anomaly



Conclusions

- FPGAs will have SEUs on orbit, but errors can be masked through TMR and scrubbed from the device using reconfiguration
- Static testing will provide an understanding of the worst case scenario
- Dynamic testing is necessary to determine the vulnerabilities not visible through static testing, to determine whether mitigation schemes have been applied properly, to determine whether mitigation meets mission requirements