

LA-UR-

*Approved for public release;
distribution is unlimited.*

Title:

Author(s):

Intended for:



Los Alamos National Laboratory, an affirmative action/equal opportunity employer, is operated by the Los Alamos National Security, LLC for the National Nuclear Security Administration of the U.S. Department of Energy under contract DE-AC52-06NA25396. By acceptance of this article, the publisher recognizes that the U.S. Government retains a nonexclusive, royalty-free license to publish or reproduce the published form of this contribution, or to allow others to do so, for U.S. Government purposes. Los Alamos National Laboratory requests that the publisher identify this article as work performed under the auspices of the U.S. Department of Energy. Los Alamos National Laboratory strongly supports academic freedom and a researcher's right to publish; as an institution, however, the Laboratory does not endorse the viewpoint of a publication or guarantee its technical correctness.



SEU Characterization of Xilinx SRAM-based FPGAs

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

Mike Wirthlin, Brian Pratt, Patrick Ostler
Brigham Young University

Outline

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

Background:

Field-Programmable Gate Arrays

- **FPGAs are a type of programmable device where the user's circuit is implemented in programmable logic and programmable routing**
- **There are a few basic categories of FPGAs:**
 - One-time programmable: Actel, Aeroflex
 - Flash reprogrammable: Actel
 - SRAM Reprogrammable: Altera, Xilinx
- **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**

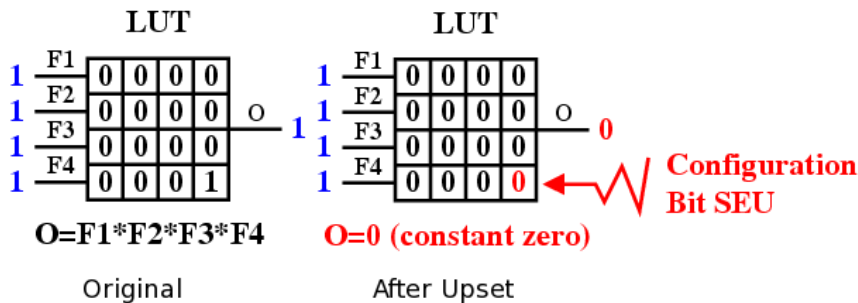
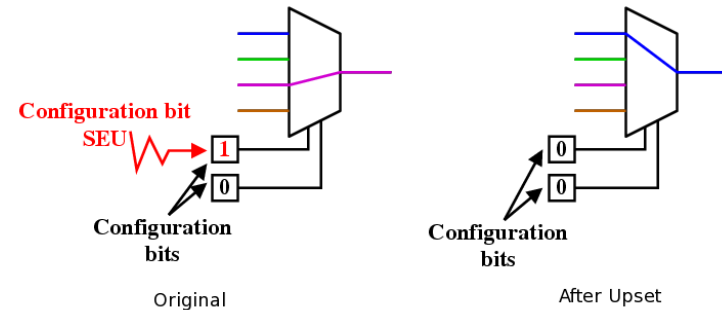
[1] G. M. Swift, "Virtex-II static SEU characterization," Xilinx Radiation Test Consortium, Tech. Rep. 1, 2004.

[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

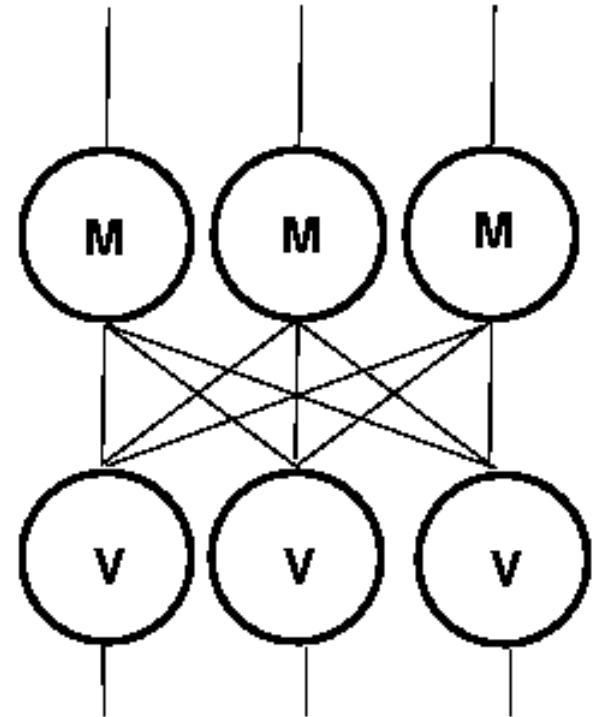
- Both the circuit and the circuit state are stored in SEU-susceptible SRAM.
 - An SEU could change the circuit functionality
 - An SEU could change intermediate processing values
 - An SEU could cause the device to become temporarily non-functional



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 has the potential to increase a device's current draw.
- For high reliability/availability requirements, mitigation and repair of SEUs are 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

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
- **Static testing of FPGAs determines the worst case scenario for the sensitivity of the device’s programming data to radiation-induced upsets.**
- **Dynamic testing indicates that typically only 1-20% of the static cross-section will affect the function of a user circuit.**

Static Testing Methodology: A Continuous Capture System

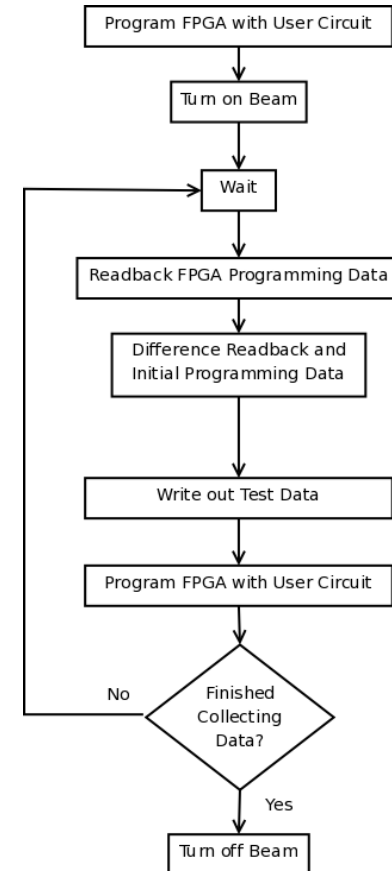
■ Advantages:

- 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

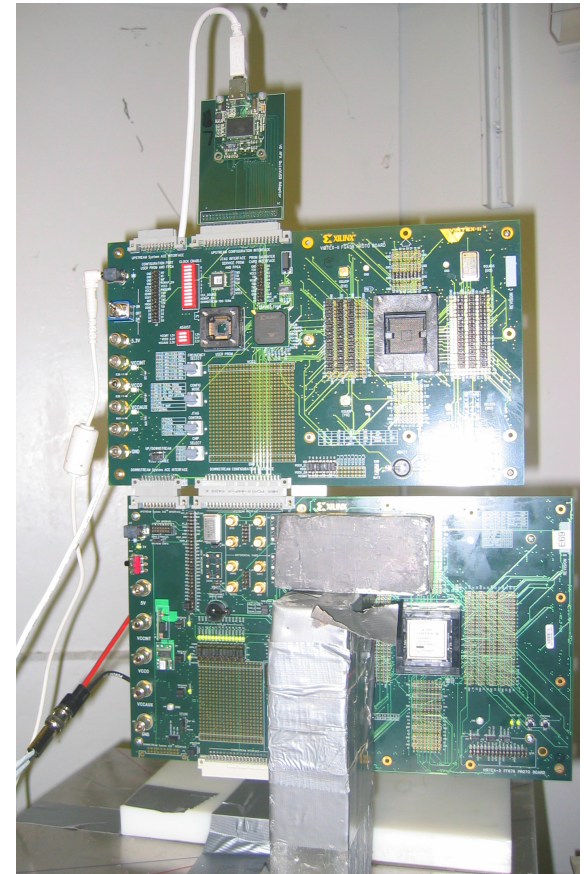
- **Even with these disadvantages, LANL and the XRTC both use this methodology**



Continuous Capture Methodology

Static Testing: 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 configuration 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 doesn't record detailed information
 - Slow sampling of configuration data due to software



Virtex-5 Test Fixture

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]

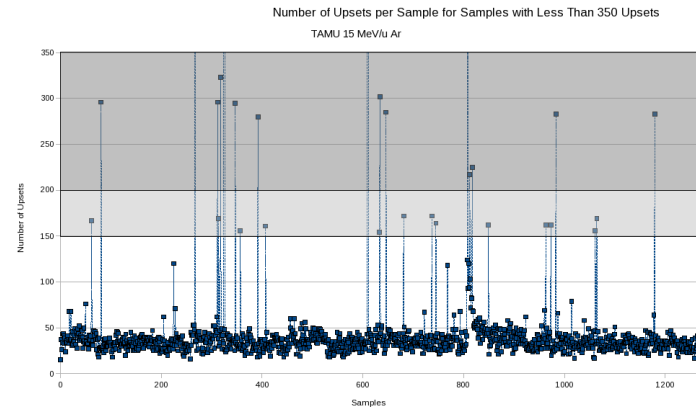
[1] Quinn, H., P. Graham, M. Wirthlin, B. Pratt, K. Morgan, M. Caffrey, J. Krone. "A Test Methodology for Determining Space-Readiness of Xilinx SRAM-based FPGA Devices and Designs. submitted to *IEEE Transactions on Instrumentation and Measurement*, Oct. 2008.

Analyzing Static Test Data: Overview

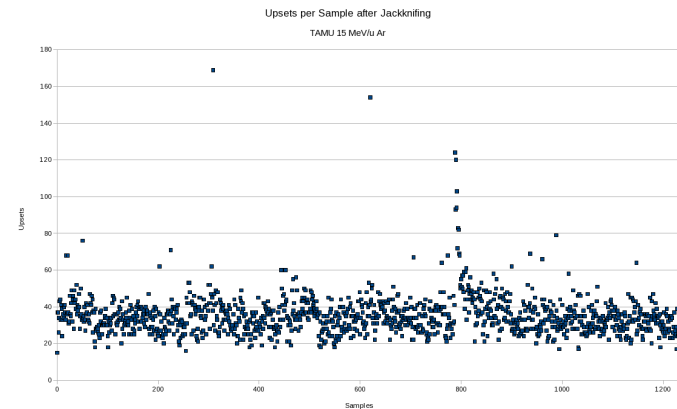
- **Each sample is analyzed separately: statistics are accumulated but samples are not analyzed as an aggregate**
- **Three step process:**
 1. Classify data statistically
 2. Correlate data to physical locations
 3. Analyze data for MBUs and affected memory cell type

Analyzing Static Test Data: Statistical Data Classification

- Initial classification is made by thresholding upset count to identify “obvious” SEFIs vs. other events
- Second-level classification of more subtle SEFIs or anomalies vs. SEUs 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



After Initial Classification of Data

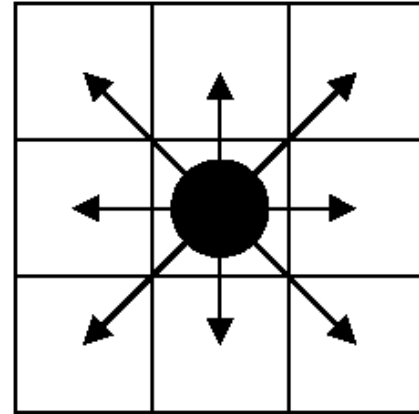


After Secondary Classification of Data

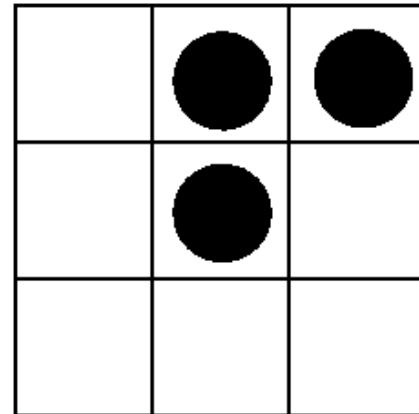
[1] B. Efron and R. Tibshirani, An Introduction to the Bootstrap.
Chapman and Hall/CRC, 1993.

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



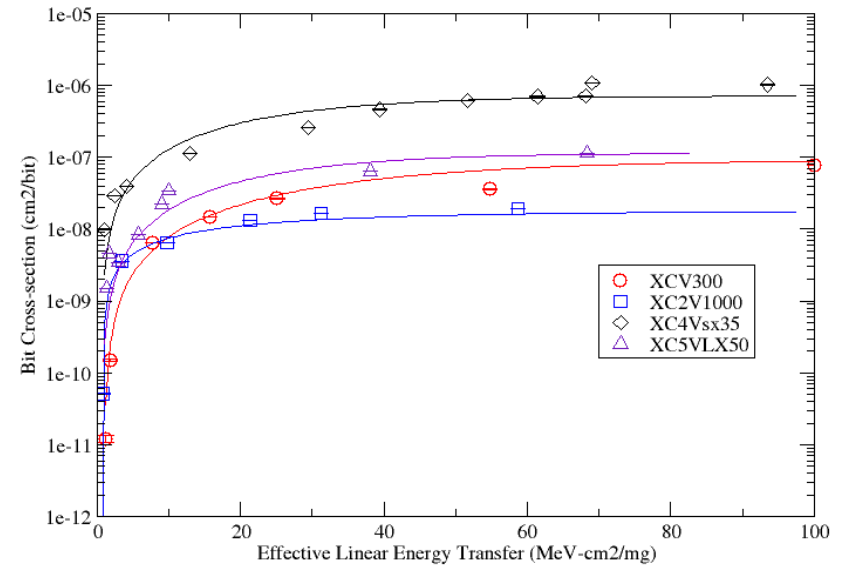
Multiple-Bit Upset Cluster

Static Test Results: Bit Cross-Sections

Device	Energy (MeV)	σ_{bit} (cm ² /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 Bit Cross-Sections



Heavy Ion Data

Static Test Results:

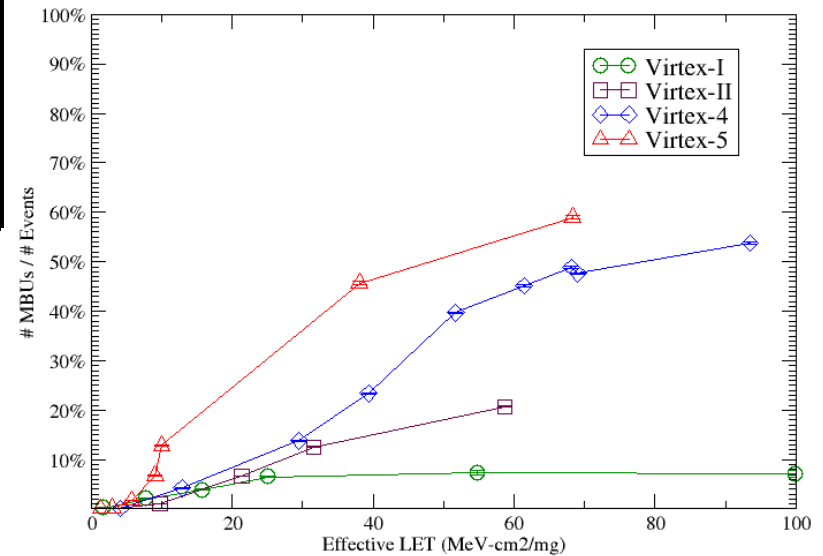
MBU Data

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

Percentage of MBUs Out of All Events in Heavy Ion



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 the 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:

Testing FPGA User Designs

- **Current “gold standard” is to do pre-launch testing of user designs through radiation experiments at a particle accelerator.**
 - Usually done at proton and heavy ion accelerators.
 - 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**
 - Expensive to exercise all failure modes, and
 - Hard to correlate errors to flaws in the user design.
- **Designers need faster, cheaper and more thorough 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:

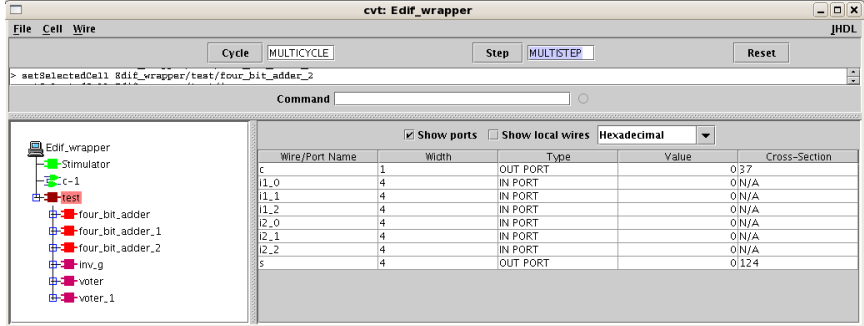
The Scalable Tool for the Analysis of Reliable Circuits (STARC) (1 of 2)

- **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.**

Dynamic Testing:

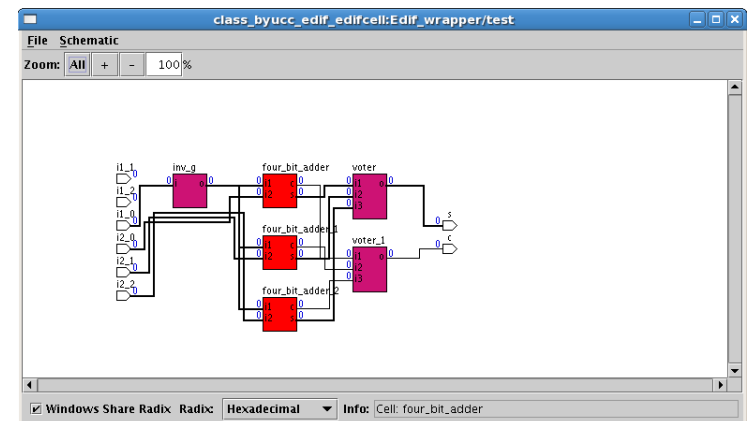
The Scalable Tool for the Analysis of Reliable Circuits (STARC) (2 of 2)

- **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.



The screenshot shows the 'cvt: Edif_wrapper' tool interface. It includes a menu bar (File, Cell, Wire), a toolbar with 'Cycle' (MULTICYCLE), 'Step' (MULTISTEP), and 'Reset' buttons. A command line contains the text: '> setSelectedCell Edif_wrapper/test/four_bit_adder_2'. Below the command line is a tree view showing the circuit hierarchy: Edif_wrapper, Stimulator, c-1, test, four_bit_adder, four_bit_adder_1, four_bit_adder_2, inv_g, voter, and voter_1. To the right is a table with columns: Wire/Port Name, Width, Type, Value, and Cross-Section.

Wire/Port Name	Width	Type	Value	Cross-Section
c	1	OUT PORT		0.37
i1_0	4	IN PORT	0/N/A	
i1_1	4	IN PORT	0/N/A	
i1_2	4	IN PORT	0/N/A	
i2_0	4	IN PORT	0/N/A	
i2_1	4	IN PORT	0/N/A	
i2_2	4	IN PORT	0/N/A	
s	4	OUT PORT		0.124



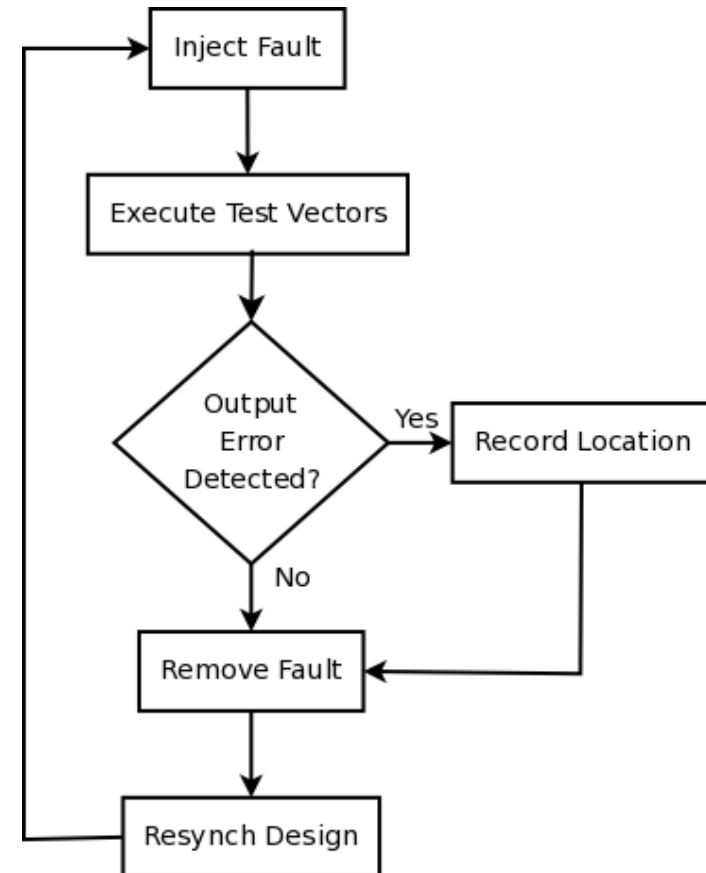
Dynamic Testing:

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 70-97% agreement with accelerator testing (difficulty: ensuring the same circuit state in both accelerator and bench-top testing)
 - Possible to fault inject to all of the configuration, except the user flip-flops.
- **While LANL and BYU designed the first extensive 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. (In fact, it is best to have fault injection capability in flight hardware for best results).
- 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 aerospace programmable logic devices conference (MAPLD08).

Dynamic Testing: Fault Injection Software Test Fixture

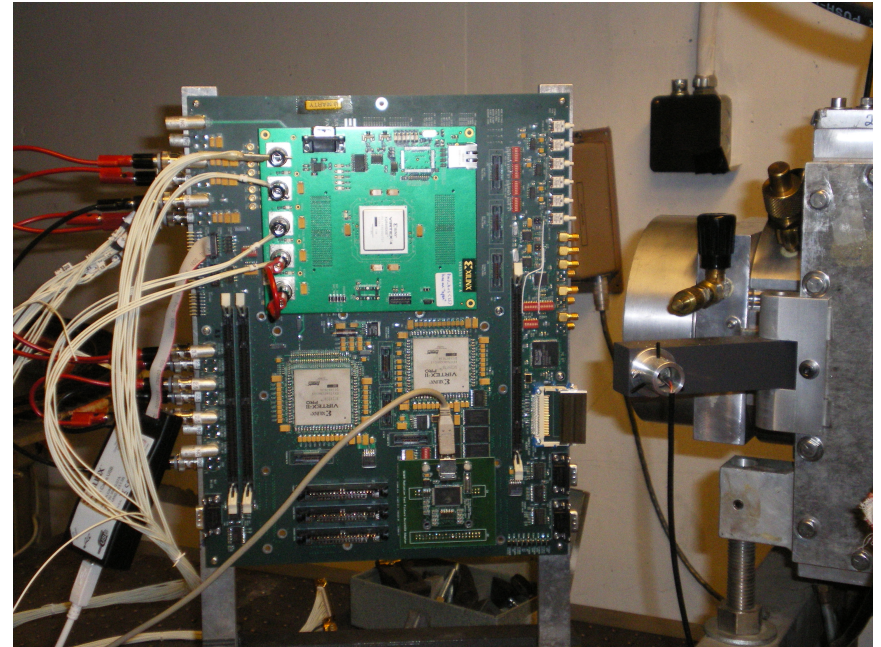
- **Standard algorithm for many test fixtures**
 - Can support a variety of hardware test fixtures.
- **Good test coverage is dependent on the number of input test vectors used.**
 - Without user-provided input vectors covering the input test vector set and maintaining good speed is a challenge.
 - LANL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



Fault Injection Algorithm

Dynamic Testing: 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.**



Multiple-FPGA Fault Injection Hardware Test Fixture (being used for accelerator testing)

Dynamic Testing:

Accelerator Testing Background

- **By performing accelerator testing after modeling and fault injection, the designer 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/SEFIs.

Dynamic Testing: Accelerator Testing Correlating Results

- **Correlating accelerator results to fault injection results can be tricky.**
 - Analyze the location of upset data in “windows” around the output error to try to match a location to the fault injection results (differences between when event occurred and when the results are sampled).
- **Some times errors cannot be 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.**

type of error observed	time stamp (ms)	bitoffset	probability of failure from simulator	
config bit error	9955	2712129	0%	<i>example of config bit which causes a failure 90% of the time in the simulator, but had no effect at the accelerator</i>
config bit error	17640	655930	0%	
output error	18070			
config bit error	18070	4504172	100%	
config bit error	18499	4275042	0%	
⋮	⋮	⋮	⋮	
config bit error	1161224	1161224	90%	
config bit error	1162513	1162513	0%	
output error	1165095			
config bit error	1165095	1592915	0%	
config bit error	1165095	2311139	0%	<i>output error due to a flip flop upset</i>
config bit error	1168090	4015285	0%	
⋮	⋮	⋮	⋮	
⋮	⋮	⋮	⋮	
⋮	⋮	⋮	⋮	
⋮	⋮	⋮	⋮	
config bit error	19217003	3172218	0%	
config bit error	19217003	5836116	0%	
config bit error	19217431	2381516	100%	
output error	19217857			
config bit error	19217857	629276	0%	

output errors due to config upsets

a config upset which had no effect in the simulator, or in this accelerator test

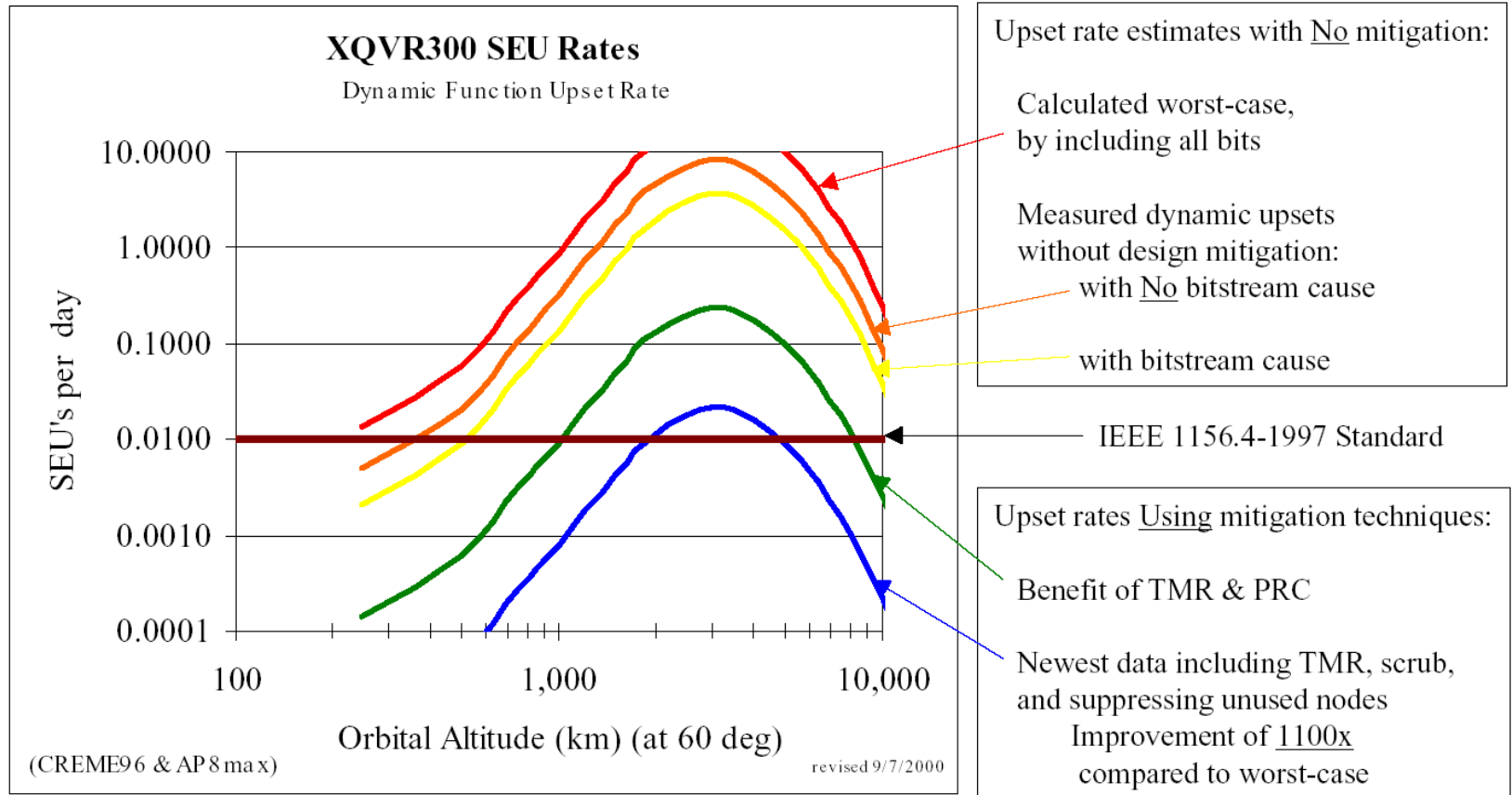
Dynamic Testing: Results

- 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 of beam time.

Dynamic Testing: SRAM FPGA Radiation Effects Characterization Error Rates



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, and top determine whether mitigation meets mission requirements.**
 - Multi-level approach
 - Automated design analysis: can be done early in the design cycle, quick results, does not take into account placement (yet)
 - Fault injection: requires hardware, can provide thorough coverage of programming data SEU sensitivities for individual bits and even MBUs, good test coverage may be challenging to achieve due to the large state space for input vectors and internal states
 - Accelerator testing: defacto standard, provides more realistic results but requires considerable cost to get good coverage for problems, hard to attribute errors to cause