A Test Methodology for Determining Space-Readiness of Xilinx SRAM-based FPGA Designs

Heather Quinn, Paul Graham, Keith Morgan, Michael Caffrey, and Jim Krone

Los Alamos National Laboratory

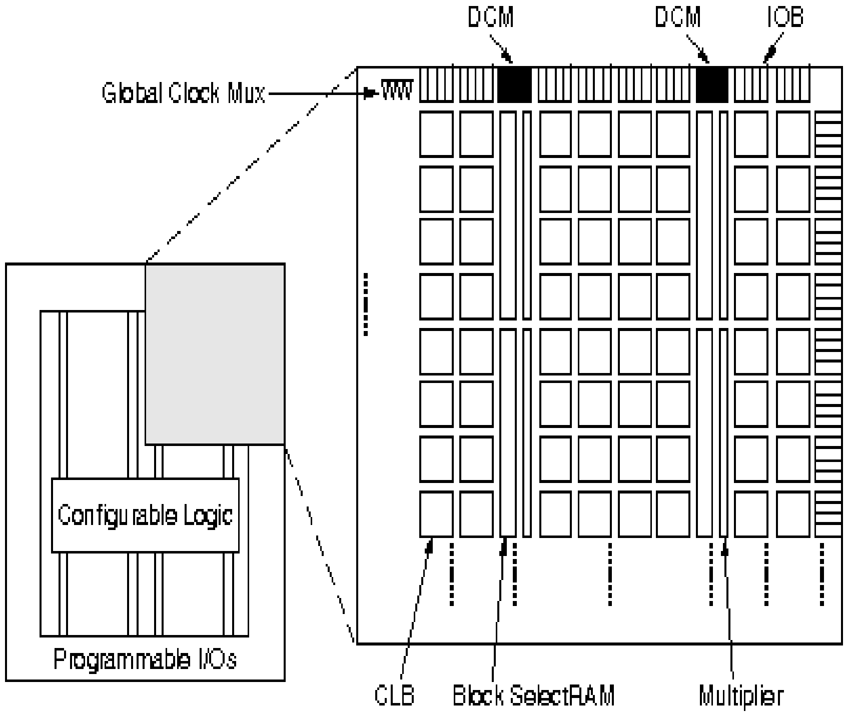
LA-UR-08-05601

Overview

- Background on using and testing SRAM-based FPGA designs for space
- Modeling tools
- Fault injection tools
- Radiation experiment validation
- Conclusions

Introduction:
Field-Programmable Gate Arrays in Space

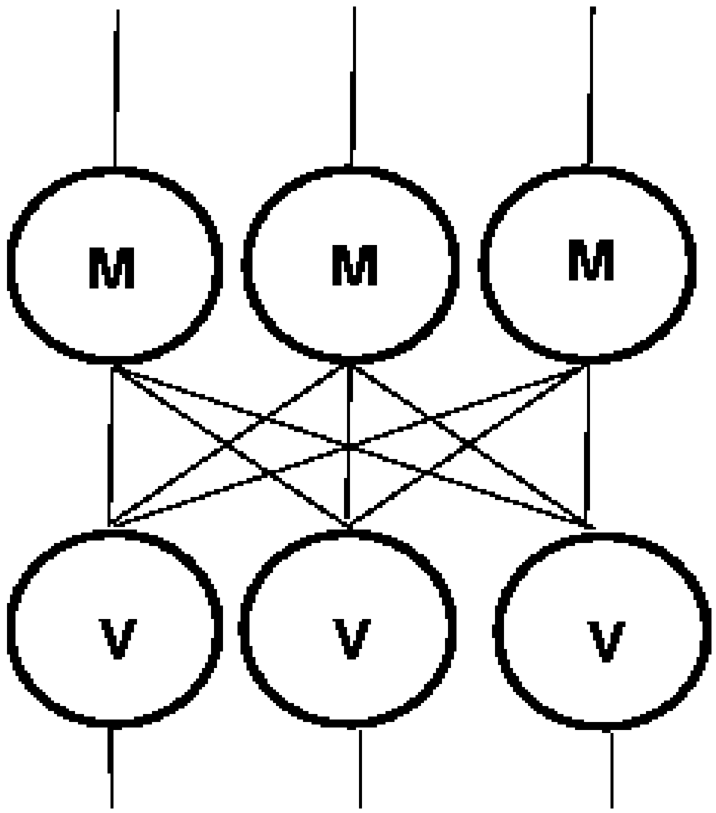
- **FPGAs are a type of programmable logic device that implements user designs in programmable logic and routing.**
- **Two types of FPGA technology:**
  - Radiation-hardened, anti-fuse technology that is one time programmable, and
  - Radiation-tolerant, SRAM technology that is reprogrammable.
- **Radiation-induced faults, such as single-event upsets (SEUs), can make fault-tolerant computing difficult for SRAM-based FPGAs.**
- **The advantages of on-orbit reconfiguration and current generation technology often out-weigh the disadvantage of SEUs.**



Xilinx Virtex-II SRAM-based FPGA

Introduction:
Fault-Tolerant Computing with SRAM-Based FPGAs

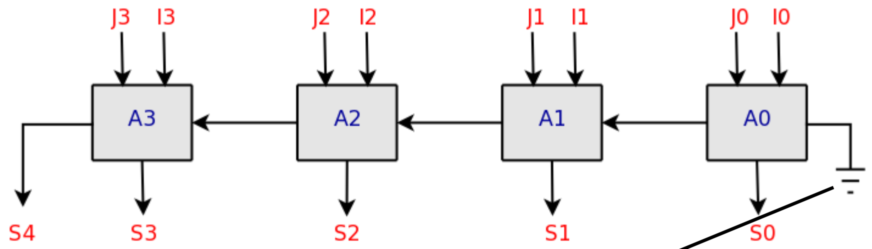
- **SEUs on SRAM-based FPGAs can cause faults in the programmable logic or routing, affecting the circuitry and the circuit state.**
- **With full triplication of signals, user logic, and voters (triple-modular redundancy), either all or most of the SEUs can be masked.**
- **User designs are often still sensitive to SEUs either through errors that occurred in the application of TMR or because full triplication is not possible**
  - User designs must be tested to determine whether TMR has been applied properly, whether sensitivities caused by untriplicated logic is reasonable, and whether the user design will meet availability requirements for the mission.

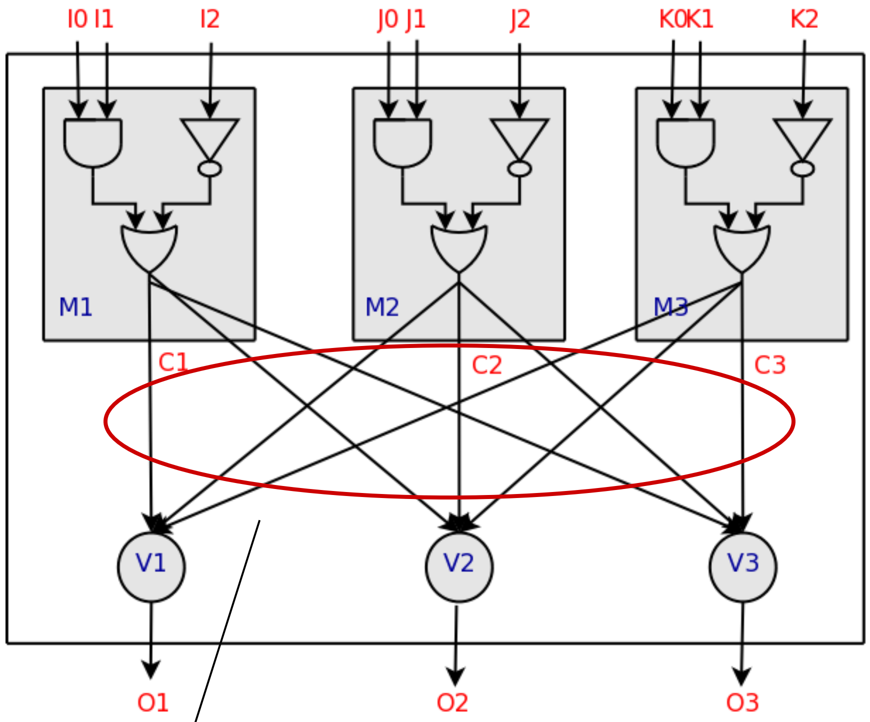


Triple-Modular Redundancy Application to an FPGA User Design

Introduction:
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.



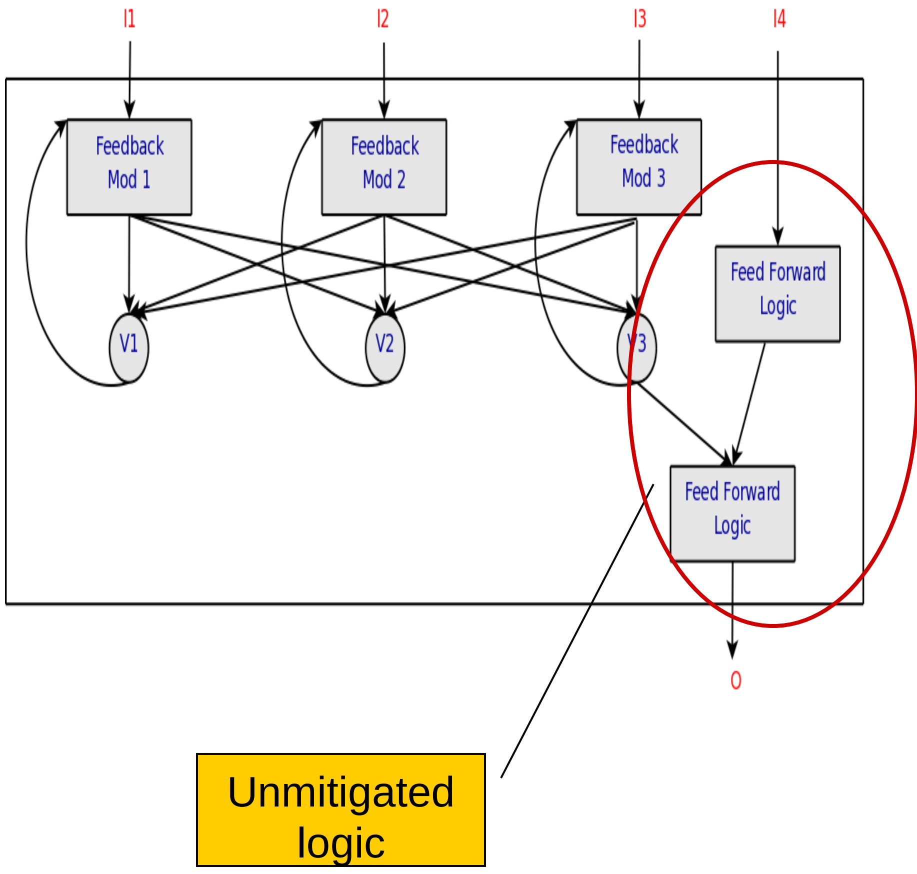


Logical constant provides the ground

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

Introduction:
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**



Introduction:
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 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.

Reliability 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.

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

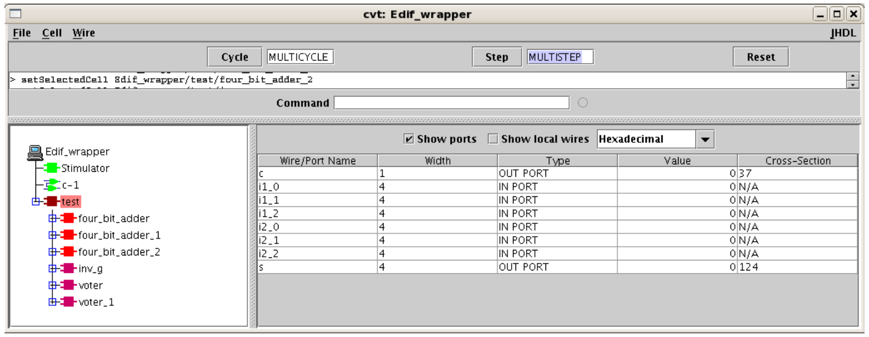
Reliability Modeling Tools:
The Scalable Tool for the Analysis of Reliable Circuits (STARC) (1 of 2)

- **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.
- **Without input vectors, STARC calculates the worst-case failure rate.**

Reliability Modeling Tools:

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

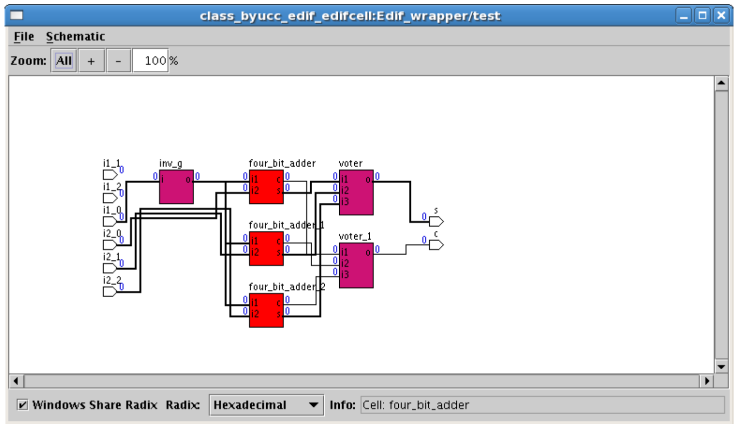
- **STARC was designed specifically to help designers find problems in TMR-protected designs.**
  - The mitigated partition is analyzed to determine if three modules are present and equivalent and if three voters are present.
  - The unmitigated partition is analyzed to determine the quantity of sensitive 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.



```
> setSelectedCell Edif_wrapper/test/four_bit_adder_2
```

MULTICYCLE · MULTISTEP · ✓ Show ports · ☐ Show local wires · Hexadecimal

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



✓ Windows Share Radix · Radix: Hexadecimal · Info: Cell: four_bit_adder

UNCLASSIFIED

Operated by Los Alamos National Security, LLC for NNSA

LA-UR-08-05601

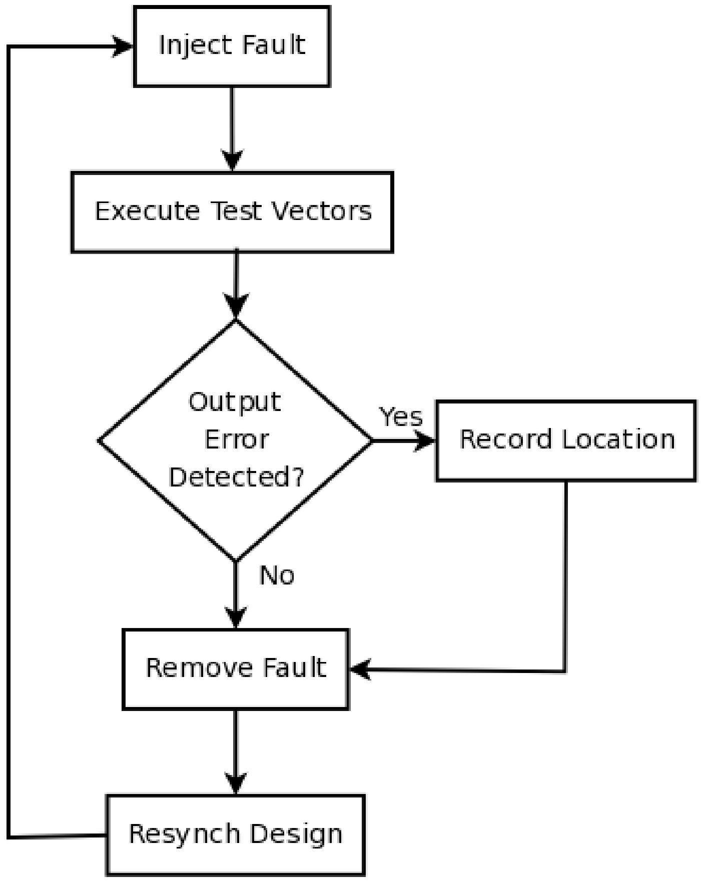
Fault Injection:
Background and Related Work

- **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 70-90% coverage of accelerator 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.

- 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).

Fault Injection:
Software Test Fixture

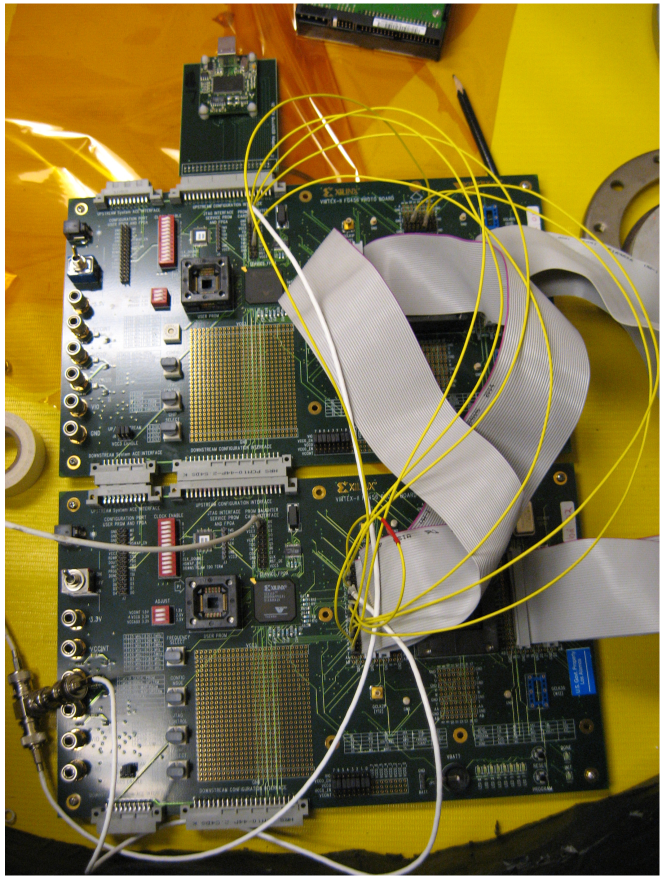
- **Standard algorithm for many test fixtures**
  - Can support a variety of hardware test fixtures.
- **Good test coverage is dependent on the number of input test vectors used.**
  - Without user-provided input vectors covering the input test vector set and maintaining good speed is a challenge.
 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



Fault Injection Algorithm

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

UNCLASSIFIED

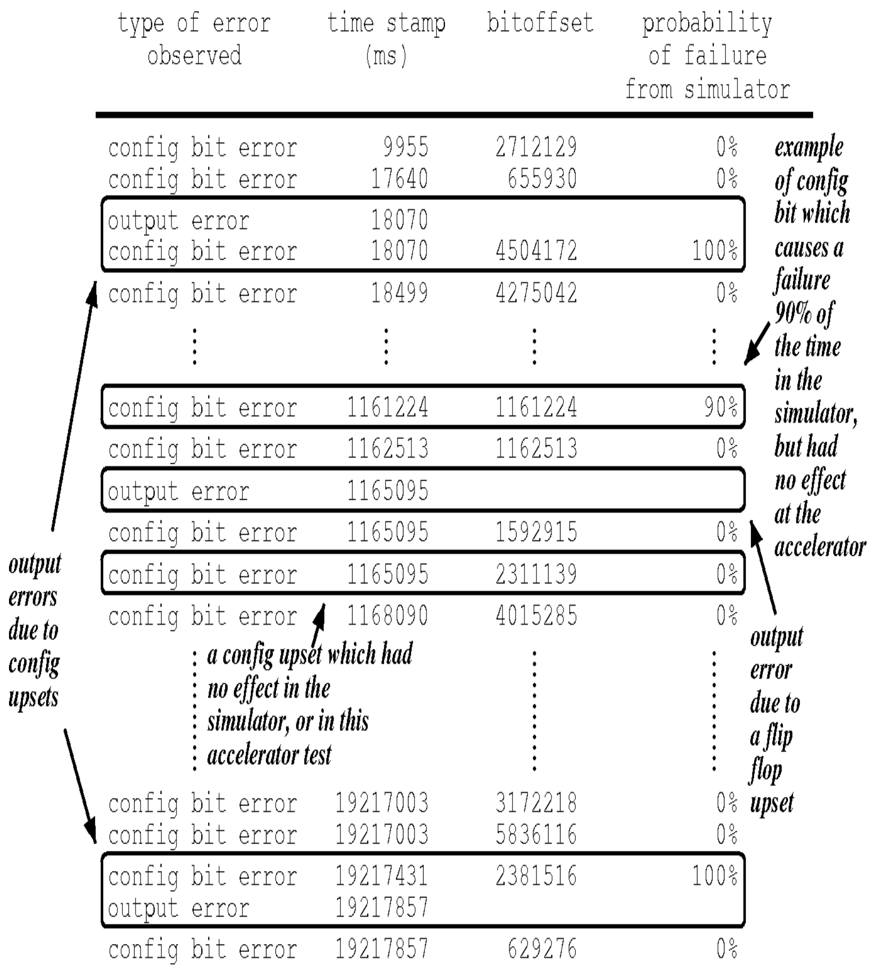
LA-UR-08-05601

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.

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.
- **Some times errors cannot correlated to fault injection data:**
  - Areas of the device not fault injected, such as user memory.
  - Multiple independent upsets
- **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%
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%
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%

example of config bit which causes a failure
90% of the time in the simulator, but had no effect at the accelerator
output errors due to config upsets
a config upset which had no effect in the simulator, or in this accelerator test
output error due to a flip flop upset

LA-UR-08-05601

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.

Conclusions

- **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.**