LA-UR-04-5272

Approved for public release;
distribution is unlimited.

Title: PRESENTATION:
Persistent Errors in SRAM-based FPGAs

Author(s):
D. Eric Johnson1,
Keith S. Morgan1,
Michael J. Wirthlin1,
Michael P. Caffrey2,
Paul S. Graham2
1Brigham Young University, Provo, UT 84602
2Los Alamos National Laboratory, Los Alamos, NM 87545

Submitted to:
2004 MAPLD International Conference
Ronald Reagan Building and International Trade Center
Washington, D.C.
September 8-10, 2004

Persistent Errors in SRAM-based FPGAs

Eric Johnson[1], Keith Morgan[1,2],
Michael Wirthlin[1], Michael Caffrey[2],
Paul Graham[2]

[1] Department of Electrical and Computer Engineering,
Brigham Young University
Provo, UT

[2] Los Alamos National Laboratory
Los Alamos, NM

This work was supported by Los Alamos National Laboratory, U.S. Department of Energy (LAUR-04-5272)
Deployable Adaptive Systems Project (DAPS), http://rcc.lanl.gov

Presentation Outline

- Overview
- Definitions
- Persistent Configuration Upsets
- Detecting Persistent Configuration Bits
- Persistent Measurements & Results
- Conclusions

Overview

- FPGAs are commonly used in radiation harsh environments, which can cause Single Event Upsets (SEUs)
- SEUs have the potential to *modify* an FPGA design, causing incorrect behavior
- Goals of project
  - Investigate the undesirable effects of configuration SEUs
  - Gain insight into SEU mitigation costs

Definitions

- ***Configuration upset*** – Single Event Upset (SEU) within the configuration memory of an FPGA
  - can potentially *modify* an FPGA design
- ***Error*** – incorrect result caused by an SEU

- ***Sensitive bit*** - configuration bit which, when upset, results in an error
- ***Non-sensitive bit*** - configuration bit which, when upset, does not result in an error

Persistence

- Severity of errors caused by configuration upsets of sensitive bits varies
- We introduce a new classification of sensitive bits based on error severity
- Classification:
  - ***Non-persistent*** sensitive configuration bits
  - ***Persistent*** sensitive configuration bits

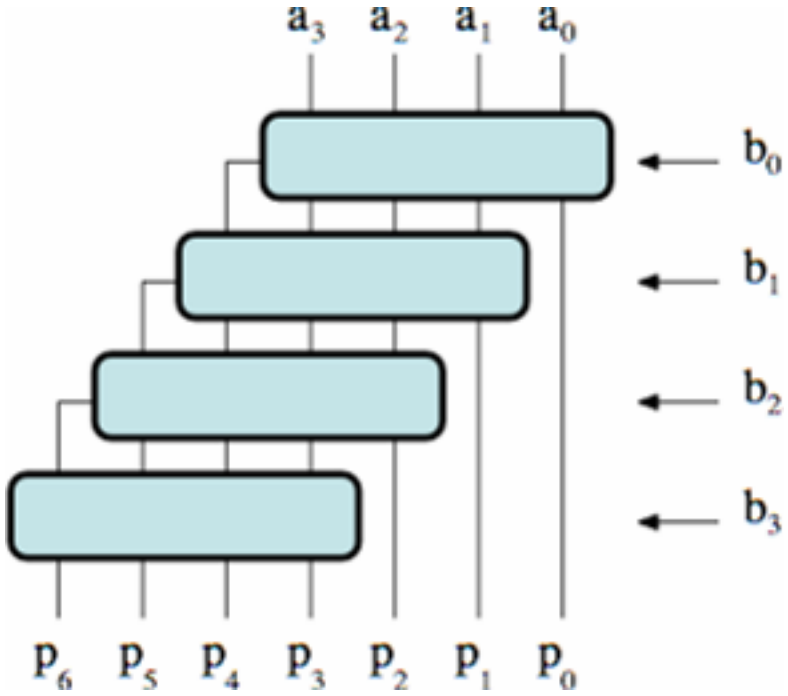
Non-persistence

- ***Non-persistent error*** - given enough time and the employment of an SEU correction strategy such as partial reconfiguration, these types of errors will eventually flush out of a system without the need of a reset (i.e. temporary errors)
- ***Non-persistent bit*** - a sensitive configuration bit which, when upset, results in a non-persistent error

Non-Persistent Bit Example

- A simple example of non-persistence is a 4-bit feed-forward multiplier design

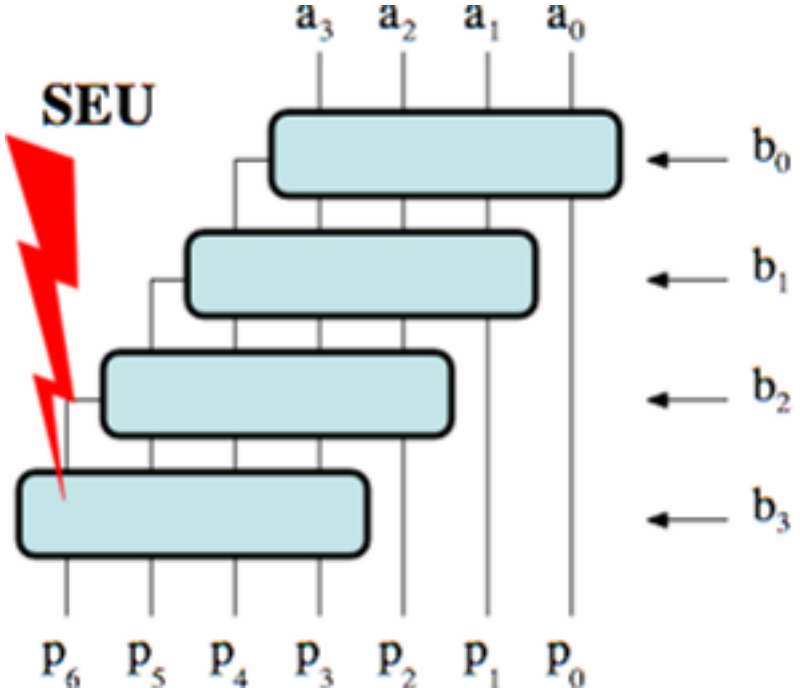
cycle	a	b	p
0	0x2	0xb	0x16
1	0xa	0x1	0x0a
2	0x6	0x9	0x36



Non-Persistent Bit Example

- When upset, the multiplier gives incorrect results

cycle	a	b	p
0	0x2	0xb	0x16
1	0xa	0x1	0x0a
2	0x6	0x9	0x36
3	0x2	0xb	0x96
4	0xa	0x1	0x8a
5	0x6	0x9	0xb6

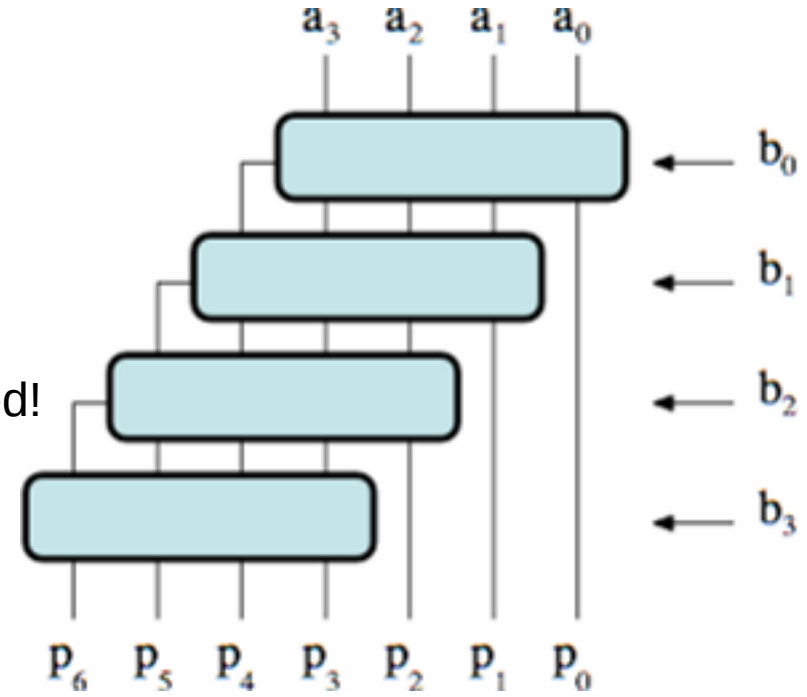


Non-Persistent Bit Example

- When the SEU is repaired, the multiplier operation returns to normal

cycle	a	b	p
0	0x2	0xb	0x16
1	0xa	0x1	0x0a
2	0x6	0x9	0x36
3	0x2	0xb	0x96
4	0xa	0x1	0x8a
5	0x6	0x9	0xb6
6	0x2	0xb	0x16
7	0xd	0x3	0x27

Fixed!



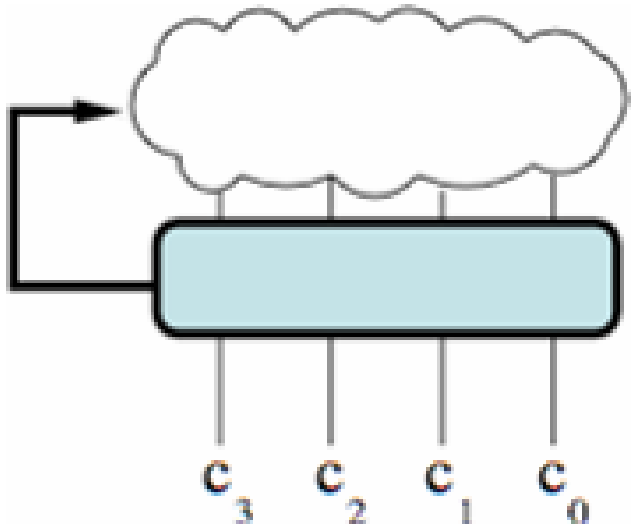
Persistence

- ***Persistent error*** – regardless of the SEU correction scheme employed, these errors will never flush out of a system
  - Only with additional intervention, for example through a reset signal, will the system ever recover
- ***Persistent bit*** – a sensitive bit which, when upset, results in a persistent error

Persistent Bit Example

- A simple example demonstrating persistence is a 4-bit counter design

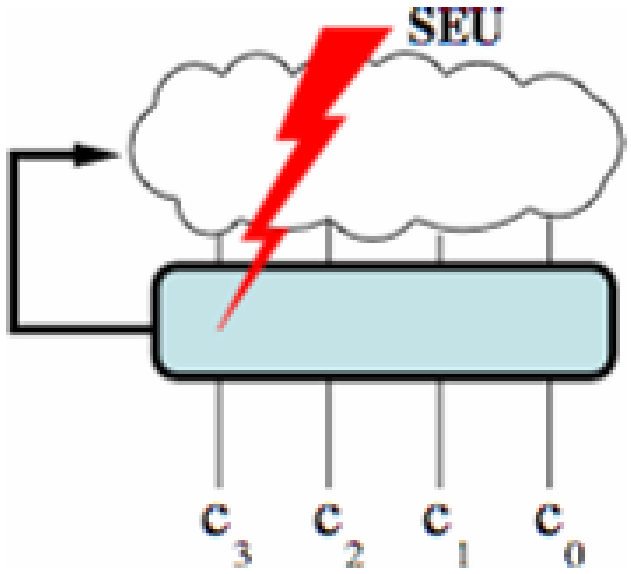
cycle	c
0	0x0
1	0x1



Persistent Bit Example

- When upset, the counter gives incorrect results

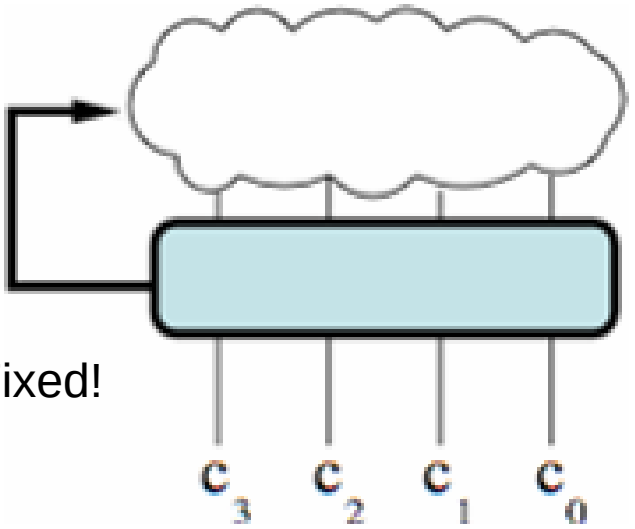
cycle	c
0	0x0
1	0x1
2	0xa
3	0xb



Persistent Bit Example

- Even after the SEU is repaired through partial reconfiguration, the counter continues to give incorrect results

cycle	c
0	0x0
1	0x1
2	0xa
3	0xb
4	0xc
5	0xd
6	0xe
...	Error



Fixed!

Persistence Analysis

- Conceptually we can categorize persistence and non-persistence by design styles
  - Feed-forward and datapath sections of a design typically do not contain persistent bits
  - Control and feedback sections typically do contain persistent bits
- However, in order to effectively apply mitigation techniques, an accurate analysis of persistence vs non-persistence is necessary
- For this purpose we have developed a persistence analysis tool

Fault Injection Tool to Locate Sensitive Bits

- Developed by BYU and Los Alamos
- Injects faults into the configuration memory of the design under test
- Classifies bits (sensitive vs non-sensitive) based on the design behavior
- Shown to predict FPGA design behavior in a radiation environment to within 98%

Johnson, Caffrey, Graham, Rollins, Wirthlin. "*Accelerator Validation of an FPGA SEU Simulator*", IEEE Transactions on Nuclear Science, Volume 50, Issue 6, December 2003, pp. 2147-2157.

Persistence Analysis Tool

- We have developed a testing method based on our fault injection tool testbed for identifying the persistent bits within an FPGA design
  - Search for sensitive bits
  - Determine if those sensitive bits are persistent

Persistence Detection: Timeline

◆ Simulator upsets configuration bit.

▲ Simulator repairs configuration bit. Persistence test begins.

■ End of persistence test.

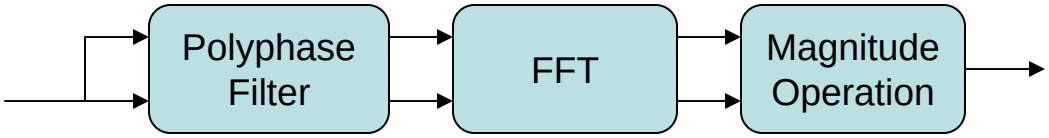
X Scrub time.

Y Flush time.

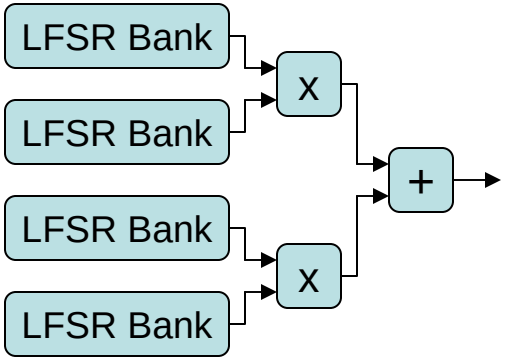
Z Persistence observation time.

Analyzed Designs

- Snapshot Recorder

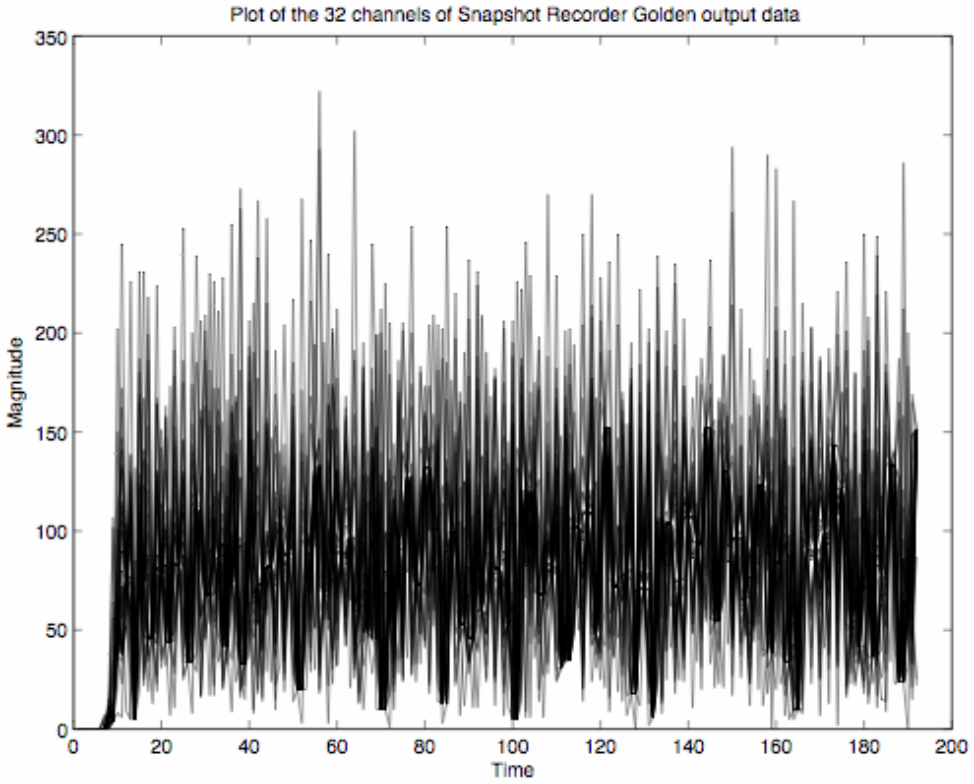


- Synthetic Design - LFSR banks driving a Multiply-Adder tree



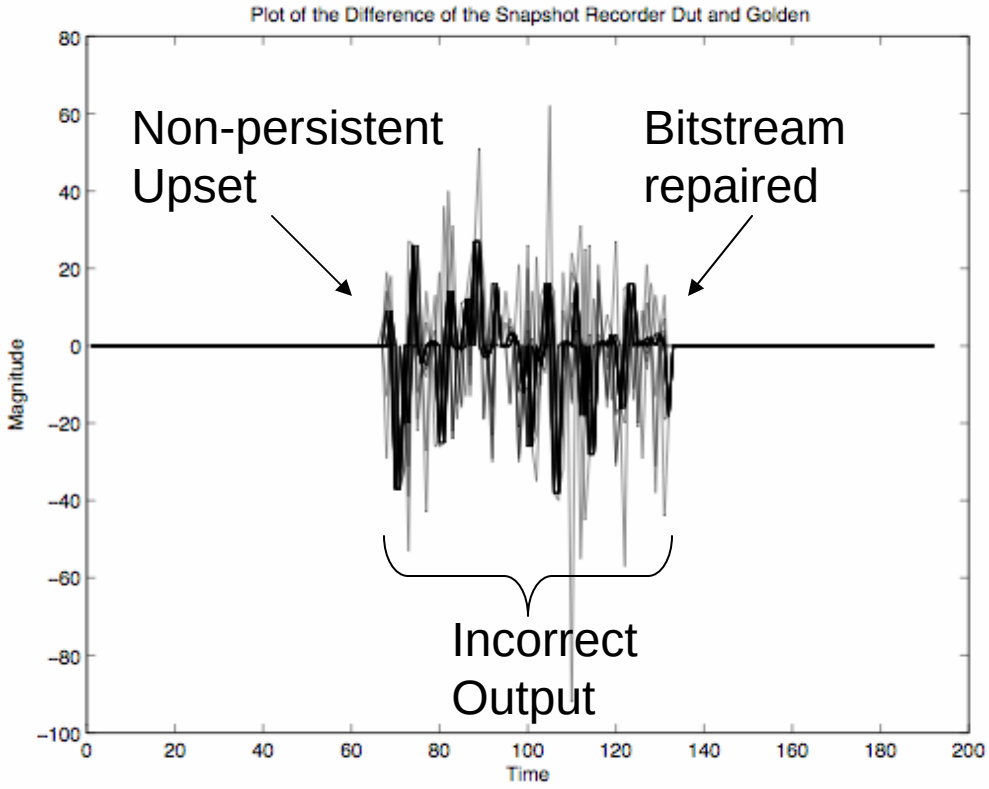
Example: Snapshot Recorder

- A polyphase filter that separates incoming data to 32 channels. The filter is followed by a magnitude operation
- This figure shows for a given set of input data the magnitude vs time plot of the 32 different channels, overlaid on top of each other



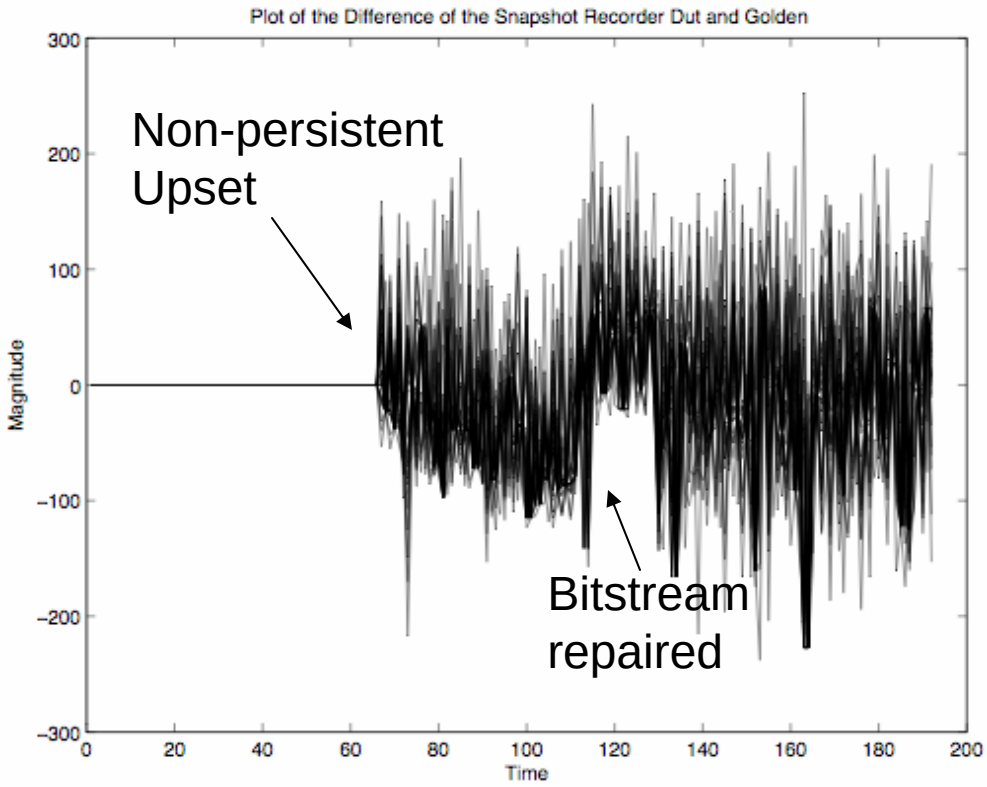
Non-Persistent Example: Error = Golden - DUT

- Plot of difference between golden output and DUT output for one bit-upset
- This illustrates a non-persistent bit
- The system recovers shortly after the SEU is repaired



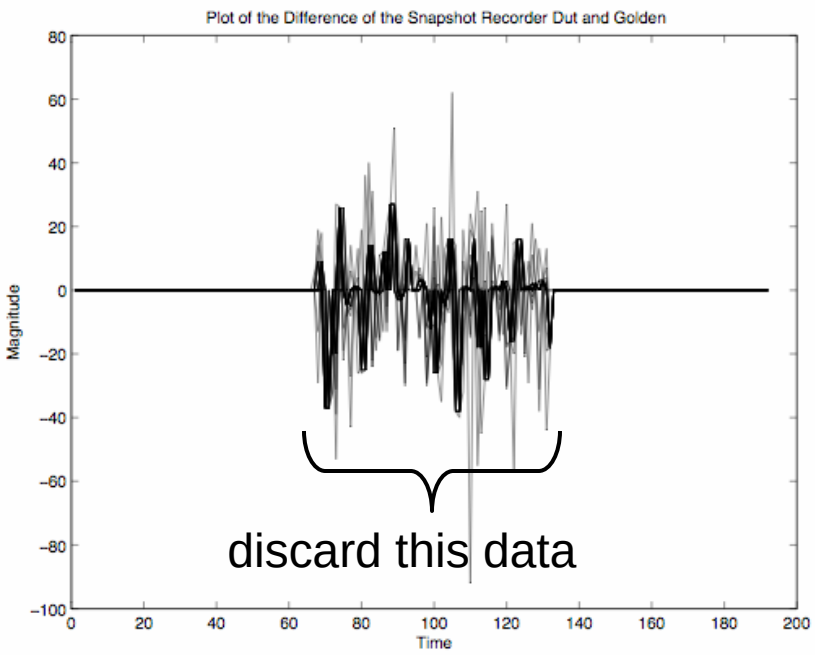
Persistent Example: Error = Golden - DUT

- Illustration of a persistent bit
- The system never recovers, even though the SEU was repaired



Mitigation Cost Savings

- For systems which can detect a configuration SEU and discard faulty data, no mitigation needs to be added for non-persistent bits
- Standard SEU mitigation is Triple Module Redundancy (TMR), which has prohibitive costs when applied exhaustively
- Large cost savings can be gained by applying such mitigation techniques only for persistent bits, rather than exhaustively



Persistence Test Results

- We have done extensive persistence testing on the following designs:

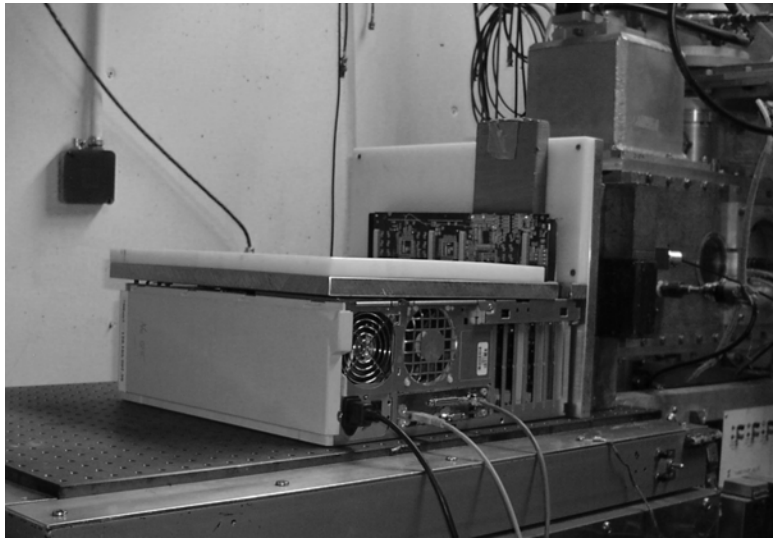
Design	Logic Slices	Sensitive Bits	Sensitivity	Persistent Bits	Persistence	Persistence Sensitivity Ratio
Snapshot Recorder	5,775	568,660	9.79%	12,382	.213%	.022
Synthetic Design	5,331	421,874	7.26%	59,369	1.02%	.141

Limitations

- The persistence simulation can only be run for a finite amount of time
  - False positives
    - In some cases, an error will disappear if only we wait longer
  - False negatives
    - An error may not have yet appeared because we have not waited long enough
- As such, we can not identify all ***persistent*** or ***non-persistent*** bits

Accelerator Testing

- We have conducted an accelerator test at Crocker Nuclear Laboratory to search for persistent bits and validate our tool
  - We are currently analyzing the data gathered from this test



Persistence: Conclusions

- We have developed a tool which can find and characterize the persistent errors in an FPGA design
- Certain designs may benefit greatly from the reliability-cost tradeoff made when tolerating non-persistent errors
  - Application dependent
- Not all designs need to be 100% bullet-proof
  - Our tool shows which sections of a design require persistent fault mitigation
  - Such analysis can reduce mitigation costs

Future Work

- Investigate design techniques which will mitigate the effects of persistent bits in FPGA designs
- Automatic application of these design techniques
- Investigate the tradeoffs to be had using these mitigation techniques
  - size vs reliability