



# Reliability Modeling Tools for Improving Space System Design: Case Studies and Results

**Heather Quinn, Paul Graham**  
**Los Alamos National Laboratory**



Operated by Los Alamos National Security, LLC for NNSA

UNCLASSIFIED

LA-UR-09-03283

Slide 1



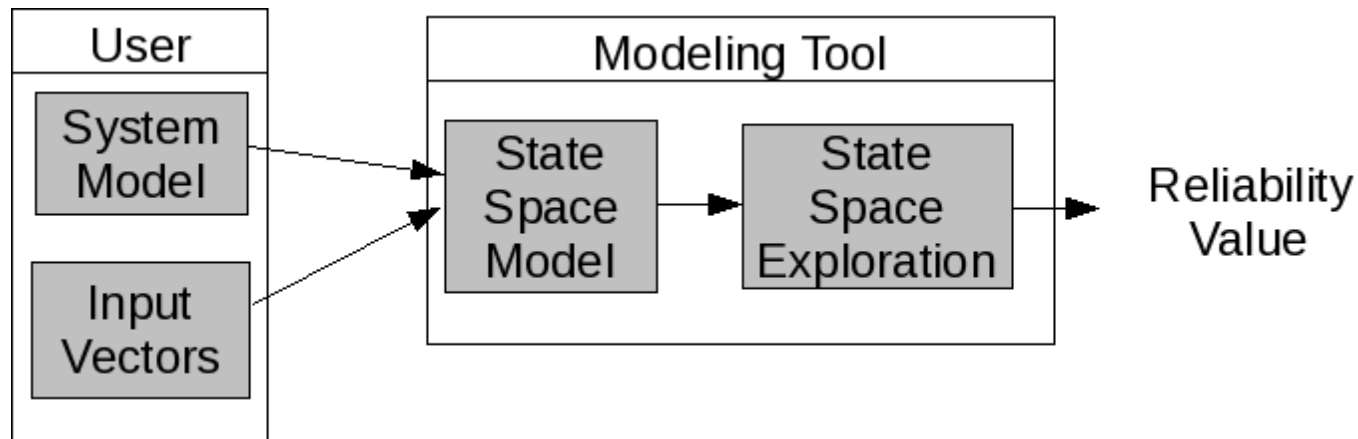
# Overview

---

- **Reliability Analysis Tools**
- **Scalable Tool for the Analysis of Reliable Circuits**
- **Case Studies**
- **Future Enhancements**

# Introduction

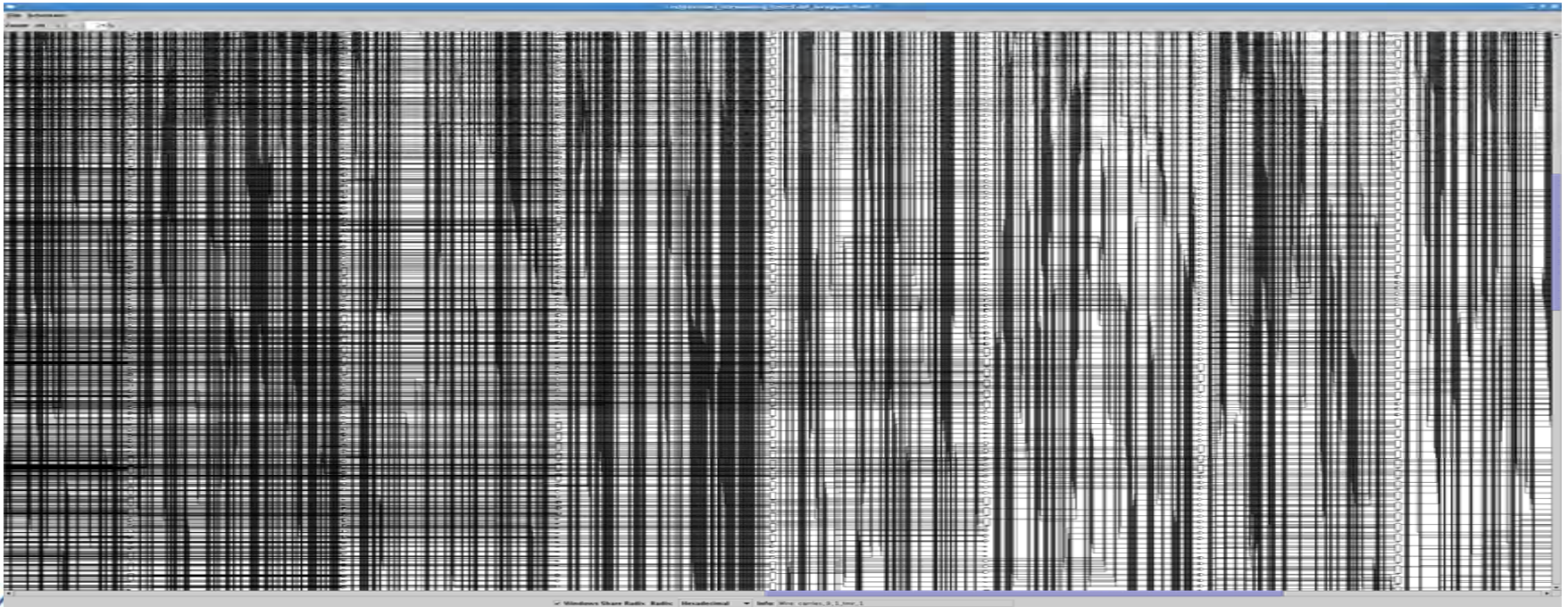
- **Reliability analysis/modeling tools are automated tools for assessing the reliability of a system**
  - System designer “composes” the system model and identifies input vectors
  - Tool determines the reliability



## Benefits:

# Quantitatively Assess System Reliability

- In large systems, determining the reliability of a system with thousands of components is difficult, especially when mitigation is used
- The tools allow the designer to focus on the model and the design and not a complicated reliability calculation



*25% of a Design Schematic for an Image Processing Circuit*

# Benefits:

## System Exploration

---

- **Once system designers can quantitatively assess reliability, then they can explore design options to optimize reliability**
- **For example:**
  - Could changing the architecture increase reliability?
  - Could changing the implementation increase reliability?
    - Microscopically: different arithmetic implementations
    - Macroscopically: Reorder computation to avoid catastrophic cancellation
  - Could adding mitigation methods help?
    - Many mitigation methods are domain-specific and only work under certain circumstances, such as state machine encoding when the probability of a failure in the encoder and decoder is low

## Benefits:

### Reduce Testing Time

---

- For many devices, there is both a static and workload-specific dynamic cross-section in the radiation environment
- The dynamic cross-section might not equal the static cross-section
  - Complete device cannot be used: dynamic < static
  - Device has SETs: dynamic > static
  - For many devices it could be a combination of both derating the cross-section for utilization and uprating the cross-section for other effects
- The dynamic cross-section is different for each workload and testing or lack of testing can be very costly
  - Modeling allows the designers to analyze incremental design work for reliability
  - Final design is verified at the accelerator

## Currently Available General Reliability Tools

---

- Probabilistic model checking: PRISM
  - Symbolic model checking: ETMC<sup>2</sup>
  - Markov model analysis: CARE, CARMS, CARSA, MKV, etc.
  - Binary decision diagram analysis: BuDDy, Bidy, JavaBDD, etc.
  - Fault tree analysis: CARE-FTA, Fault-Tree+, Relex Fault Tree, etc
- 
- More information found at: <http://www.enre.umd.edu/tools.htm>

# Limitations of Reliability Modeling Tools

---

- **Most reliability analysis tools need both a model and an input vector set as inputs**
  - Input models are often in a proprietary language to the tool
  - Designers might not know the input vector set or the input vector set might be all possible input combinations – If the input bus is N-bits wide, the input vector space is  $2^N$
- **For many tools the model and the input vector set are combined to create a state space of all the possible state transitions in the system**
  - This state space grows exponentially with the number of possible input combinations
  - This state space has to be fully explored to determine the reliability of the circuit.
  - For realistic systems and circuits this computation is infeasible due to computational complexity – could leverage combinatorial optimization but would need to know how that affects the answer
- **Since these tools are not optimized for our work, there is no way to use VHDL, EDIF, or other circuit/board models as the input model.**
  - No translation tools exist to convert to these modeling languages.
- **The model is only as good as our ability to understand reality and for the model to match reality**

# Does This Mean that Reliability Tools Are Too Hard to Use?

---

No

That means that as a community  
we need to invest in designing  
tools that focus on our problems

# LANL's Research into Reliability Tools: Scalable Tool for the Analysis of Reliable Circuits (STARC)

---

- **Currently supports Xilinx SRAM-based FPGAs and their space reliability issues, as well as other structural logic designs**
- **Currently:**
  - Available as an alpha through GUL
  - Uses EDIF circuit representation for the input model
  - Does not use input vector sets
- **Working on:**
  - Adding Xilinx Design Language (XDL) circuit representation to provide placement-related information
  - XDL is textual description of the low-level FPGA implementation of a design that indicates both the FPGA-specific resources used and their location

# Scalable Tool for the Analysis of Reliable Circuits (STARC)

## Issues Addressed

---

- **Instead of probabilistic model checking analysis, STARC uses a hierarchical, combinatorial reliability analysis approach.**
  - Since circuits tend to be hierarchically arranged into sub-circuits, STARC can take advantage of the circuit structure
    - Without input vector sets, reliability analysis can be reused (“memoized”).
    - For circuits with a lot of sub-component reuse, the computation is greatly reduced, which allows the computation to easily scale to realistic circuit sizes.
    - The STARC state space grows linearly with the number of unique circuit sub-components in the circuit.
      - Even in large circuits with no sub-component reuse, the state space will still be much smaller than with probabilistic model checking.
  - Without input vector sets the reliability analysis provides a worst-case estimate, since it cannot determine if certain states are never reached.
    - For example, a 4-input LUT that has had an SEU change one value is still correct for 15 out of 16 input combinations.
    - STARC will consider it unreliable for all input combinations.

# Scalable Tool for the Analysis of Reliable Circuits (STARC)

## Extensible Architectures

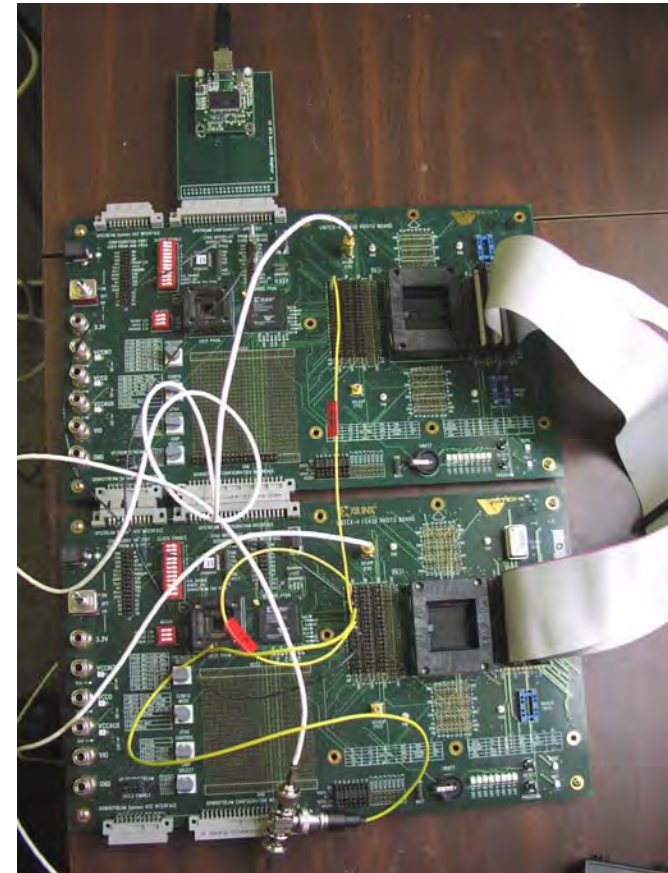
---

- **STARC analysis is also portable to different architectures.**
- **Architecture-specific models of resource reliability provide the basis for the reliability analysis.**
  - Currently, within the tool, models already exist for Xilinx Virtex family FPGAs SEU-based reliability and nano-scale CMOS yield-based reliability.
  - In the future, would like to extend to RHBD and RHBP reliability issues.
  - Designers could also import their own models.
- **By keeping the models separate from the calculation, designers could experiment with how the architecture affects the reliability**
  - Less important when comparing Virtex-1 to Virtex-4.
  - More important when looking at nano-scale architectures.

# Scalable Tool for the Analysis of Reliable Circuits (STARC)

## Cost/Benefits

- **Accelerators: one application, 2 upsets/sec, UC-Davis**
  - V1000: 33.5 days, \$403,000
  - 2V250: 11 days, \$132,000
  - 4VSX35: 69 days, \$828,000
  - 5VLX50: 86 days, \$1,032,000
- **SEU Emulators: one application**
  - Cost: \$6-12,000 (Boards and PC)
  - Time: .5-3 hours/run, multiple runs
  - >90% agreement with proton tests
- **Reliability tools**
  - Cost: PC, no boards
  - Time: minutes-hour



*Virtex-II SEU Emulator*

## FPGA-Specific Design Issues Addressed: Hardness Assurance Issues with TMR-Protected Designs

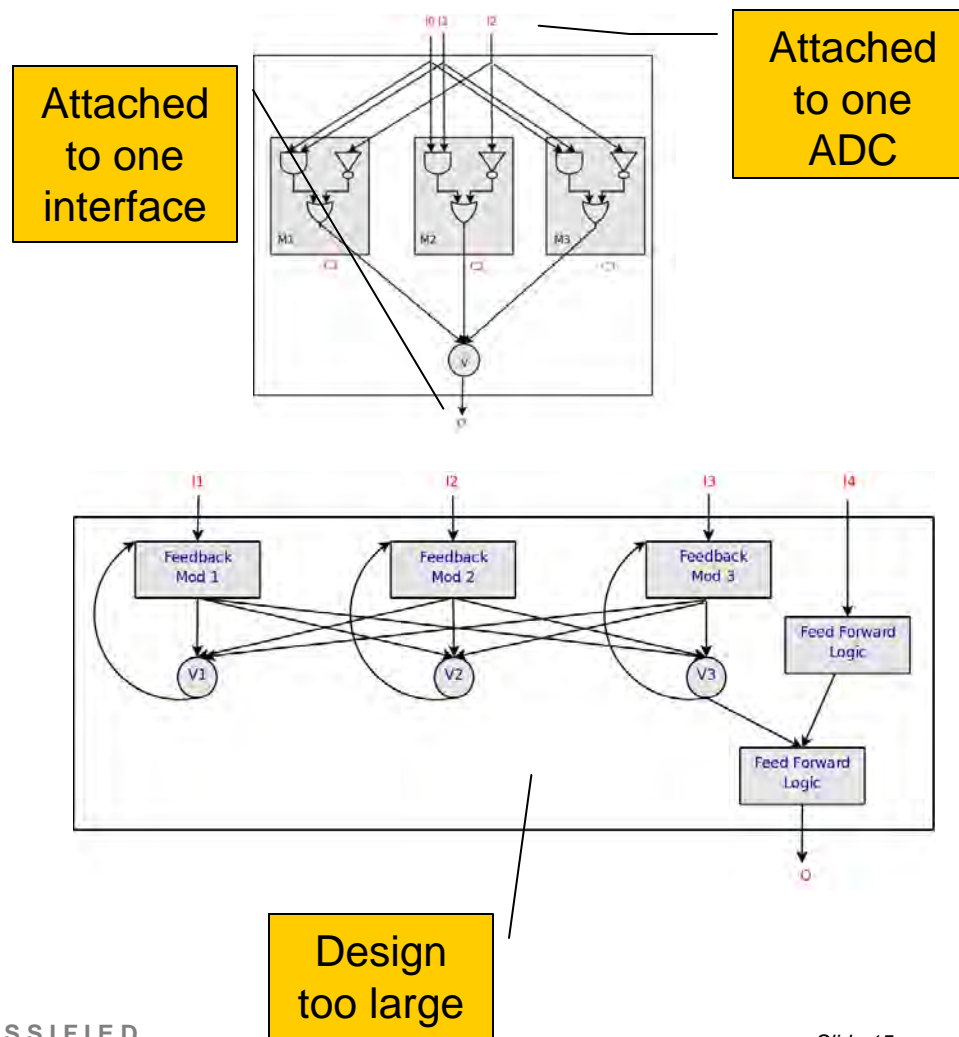
---

- **Circuit design influences from not using automated TMR tools, such as Xilinx's TMR Tool or BYU's BL-TMR**
  - Redundant modules partially or completely removed during synthesis
  - Persistence issues with feedback loops
- **Device constraint influences**
  - Not enough input or output pins to completely triplicate signals
  - Not enough device space to triplicate logic
- **Architectural Influences**
  - Logical constants not extracted
  - Design placement

Often times these problems are not the fault of the designer, but the fault of using design flow tools that are not designed for the space domain

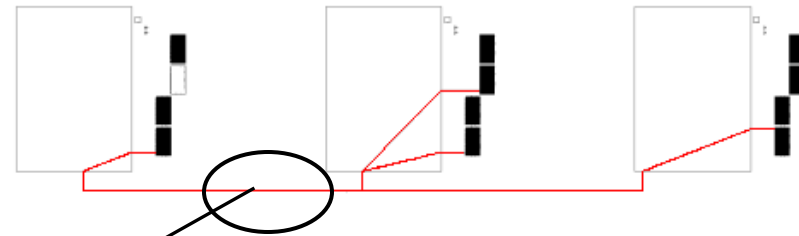
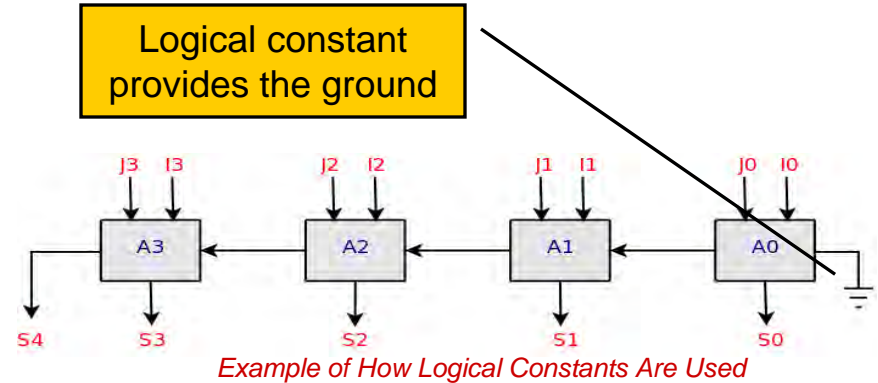
## Design Constrained Scenarios that Affect TMR

- Sometimes designers are unable to fully triplicate the design due to either area or pin issues.
- Triplicating input/output signals can be impossible due to pin constraints and can be difficult to manage due to skew.
- Triplicating all of the logic might not be possible due to the chosen device's size.
  - BL-TMR can automatically apply partial TMR for this scenario through prioritized redundancy based on device size.
- **Unprotected cross-section can be assessed using STARC, checks to make certain that TMR-protected portion has three voters and three functionally equivalent modules**



# Architectural Influences that Affect TMR: Logical Constants

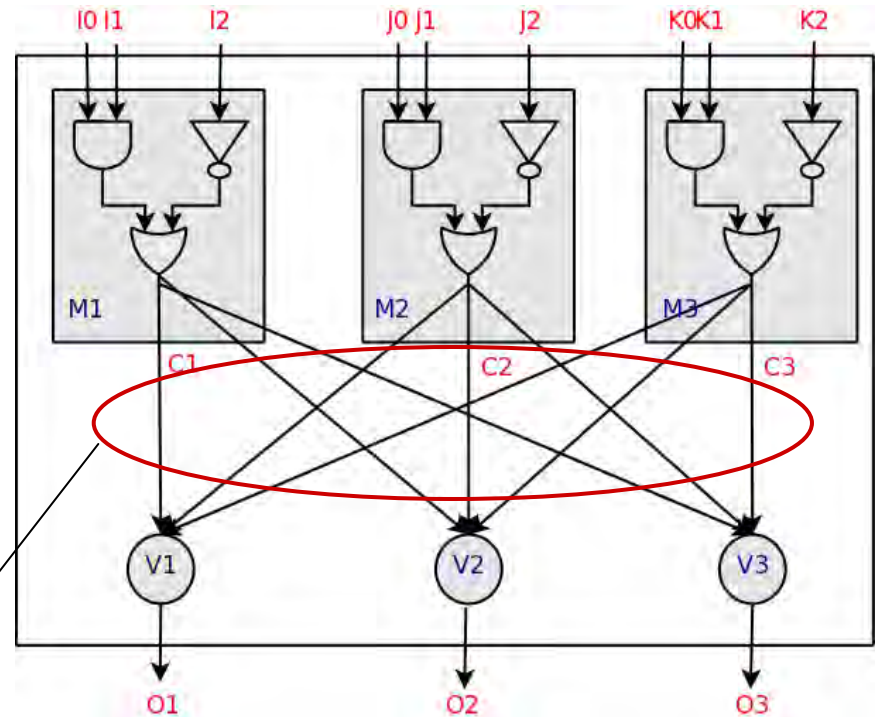
- Logical constants:**
  - Are needed to provide ones and zeros for constants in the circuit logic area efficiently
  - Are an artifact of translating VHDL to the FPGA architecture, when designs do not have all of the necessary signals tied off
  - Are SEU-prone and corrupt state in the circuit
- Easily taken care of by doing “half latch extraction” (misnomer: takes care of all types of logical constants) in the automated TMR tools or using LANL’s RADDRC tool.
- STARC checks to make certain the logical constants have been mitigated**



Power network ties carry chains for three domains together

# Architectural Influences that Affect TMR: Design Placement

- Even in fully TMR-protected designs, TMR defeats due to multiple-bit upsets are possible
- The addition of the XDL circuit representation will allow STARC to assess these problems
  - Domains sharing CLBs is to the first order effect the problem, where clocks, or clocks and resets from different domains are switched



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

## Determining the Hardness Assurance of a TMR-Protected Design

---

- A designer could use fault injection if the circuit can be adapted to a fault injection tool.
- Accelerator testing is also possible, but much more expensive and harder to achieve uniformity and observability.
- The “reality” of how radiation affects an FPGA is straight forward
  - We understand how the device fails
  - We understand how the design plays a role in translating device failures into output data corruption
  - We understand and can detect problems with mitigated designs that lead to mitigated designs failing
  - This breadth of observability makes this problem a good candidate for modeling

## Case Study:

# Implications to TMR in Device-Constrained Scenarios

---

- **BL-TMR used to partially TMR two image processing algorithms**
  - Algorithm 1: edge detection
  - Algorithm 2: noise filtering
- **Four implementations of the two algorithms:**
  - Unmitigated
  - Partially mitigated (1): logic mitigated, no signals mitigated
  - Partially mitigated (2): logic, clock and reset mitigated; no data I/O signals mitigated
  - Fully mitigated
- **BL-TMR used to do partial and full mitigation**
  - Partial mitigation approaches chosen to illustrate how some designers might choose to use the tool
  - Not necessarily the best way to use the tool

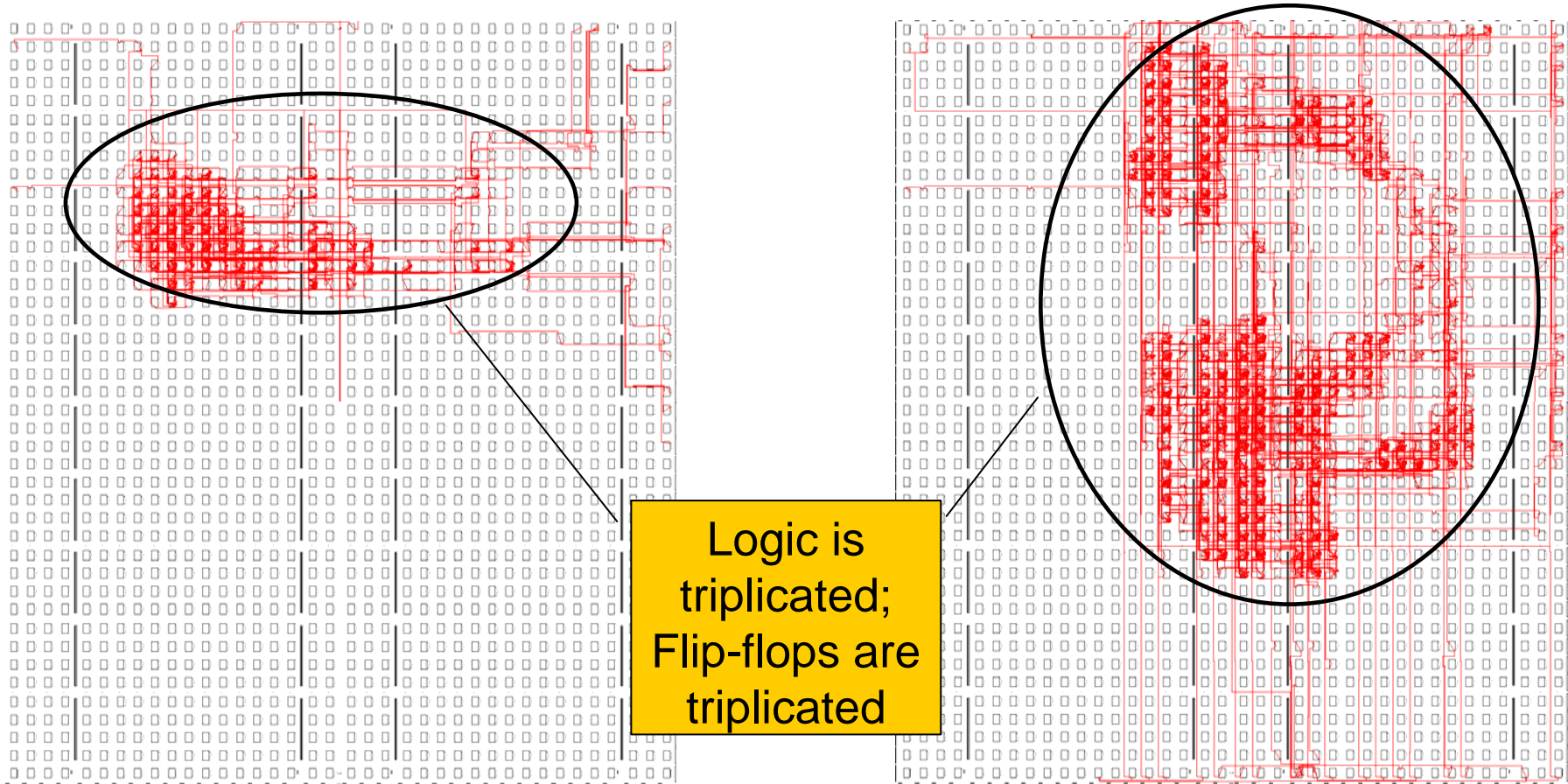
# Case Study:

## Results of Estimated Unprotected Cross-section

Implementation	Edge Detection (configuration bits)	Noise Filter (configuration bits)
No TMR	15,418	14,914
Partial TMR (1)	21,800	14,332
Partial TMR (2)	24	24
Full TMR	0	0

# Case Study:

## What happened with Partial TMR Implementation 1?

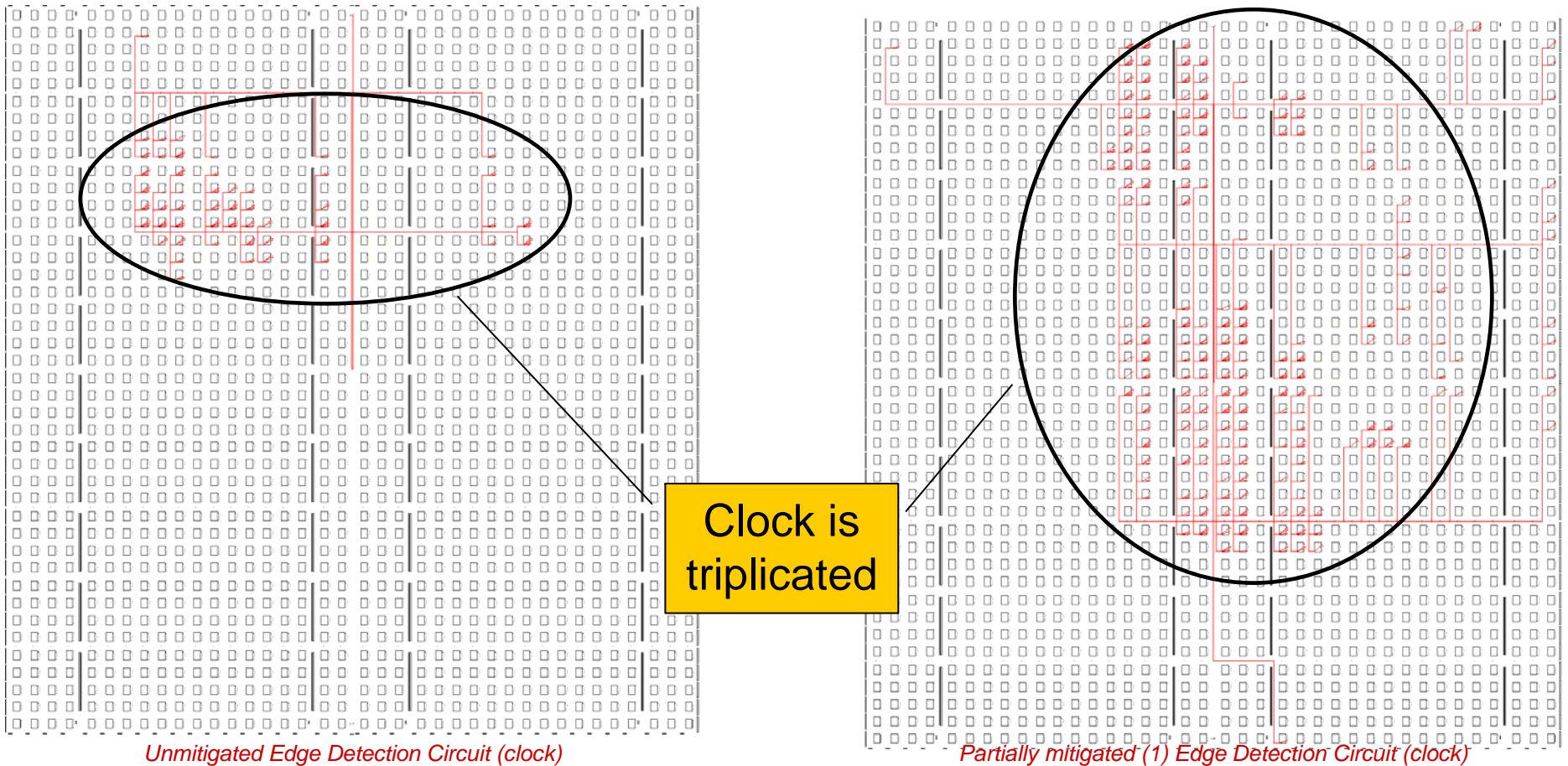


*Unmitigated Edge Detection Circuit (logic)*

*Partially mitigated (1) Edge Detection Circuit (logic)*

# Case Study:

## What happened with Partial TMR Implementation 1?



# Case Study:

## What happened with Partial TMR Implementation 1?

- **When the design was triplicated, the number of flip-flops was also triplicated**
  - Each flip-flop using the same clock, reset, and clock enable signal
  - The untriplicated global signals triplicated in size
- **All of the unprotected logic is in the routing network**

Design	Routing (cfg bits)	Logic (cfg bits)
Edge Detection	21,781	19
Noise Filter	14,313	19

# Case Study:

## Triplicating Signals

---

- **Triplicating clock and reset essential in TMR-protected FPGA designs, especially, for highly pipelined logic designs**
- **Do not have to triplicate input and output data signals**
  - Input data signals will triplicate the first time they are registered – register immediately after the input
  - Output data signals will become single string after the final voter – make the final voter the last piece of logic
  - For both designs the unprotected cross-section due to unmitigated data signals is 24 bits

## Future Enhancements:

# Providing an Interface to SEU Mitigation Tools

---

- **STARC was built on the same infrastructure as the BYU BL-TMR tool and the LANL RADDRC tool**
- **STARC could be used by these tools to address reliability concerns in a specific circuit**
- **For example:**
  - STARC determines that logical constants were not extracted and calls RADDRC to fix the design placement
  - STARC determines that the circuit does not meet reliability requirements and calls BL-TMR Tool to mitigate the circuit further

# Future Modeling Efforts:

## Multi-Objective Modeling Tools

---

- **Providing designers an understanding of how reliability affects performance, power, and temperature**
- **For example:**
  - Performance modeling of a multi-core device indicates not all of the cores are being used. A tool determines whether there is enough margin in the power and temperature values to increase reliability through the use of spare cores
  - Reliability modeling determines that the mission requirements for the workload are met with a large margin. A tool determines whether mitigation can be removed such that more performance, less power or a smaller device can be used without violating mission requirements.
  - Architecture modeling determines (reliability, performance, power, temperature) tuples for a set of architectures using similar workloads, such that the designers know which architecture best matches mission requirements and workload.

# Future Enhancements:

## Modeling with Simulation Tools

---

- Reliability modeling tool simulates a “day in the life” of a payload/satellite to show probable on-orbit behavior and system reaction to radiation-induced upsets.
- Lifetime aging models determine how the accumulation of dose affects the payload/satellite over the course of the mission, including how the loss or degradation of devices affects computation.

# Conclusion

---

- **The benefits of using reliability analysis tools**
  - Allow the designer to determine reliability without using an accelerator
  - Allow the designer to focus on design more, and do more design exploration
  - When used in conjunction with accelerator testing and fault injection, designer will have a solid understanding of on orbit behavior
- **STARC is a reliability analysis tool from LANL that determines the reliability of FPGA designs**
  - Provides designers a quick reliability test that is fast enough to fit into the design flow
  - Case study shows how STARC can help designers spot non-intuitive design problems
- **Many future avenues to develop automated modeling and analysis tools for the space domain**



# Questions?



Operated by Los Alamos National Security, LLC for NNSA

UNCLASSIFIED

LA-UR-09-03283

Slide 29

