BL-TMR and Mitigation Approaches for FPGAs

Mike Wirthlin
BYU

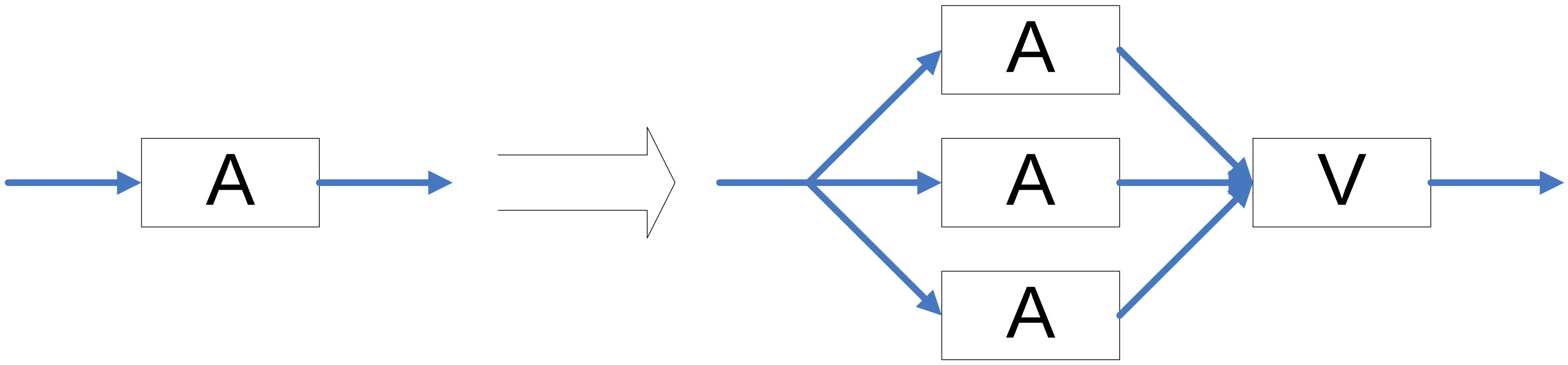
Los Alamos NATIONAL LABORATORY

CHREC NSF Center for High-Performance Reconfigurable Computing

1. TMR Overview

Triple Modular Redundancy (TMR)

- A form of N Modular Redundancy
  - *Triplicate hardware resources*
  - *Majority Vote on hardware outputs*

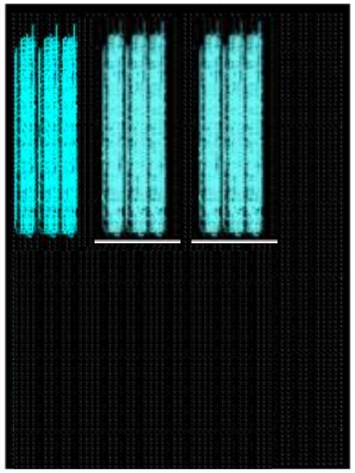


- Tolerates any *single* fault
  - *Tolerates many multiple fault combinations*

TMR Granularity

System Level

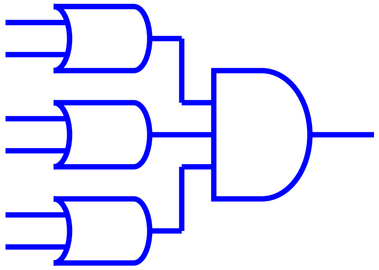
Device Level



Module Level

```
process(clk_int_a)
begin
   if clk_int_a'event and clk_int_a='1' then
       locked_d_a <= locked_a_int;
       if (all_locked_a = '0') then
          all_locked_a <= (locked_d_a and
                locked_d_b and locked_d_c);
       else
           all_locked_a <= tmr_voter(
             locked_d_a, locked_d_b,
             locked_d_c);
       end if;
   end if;
end process
```

RTL Level

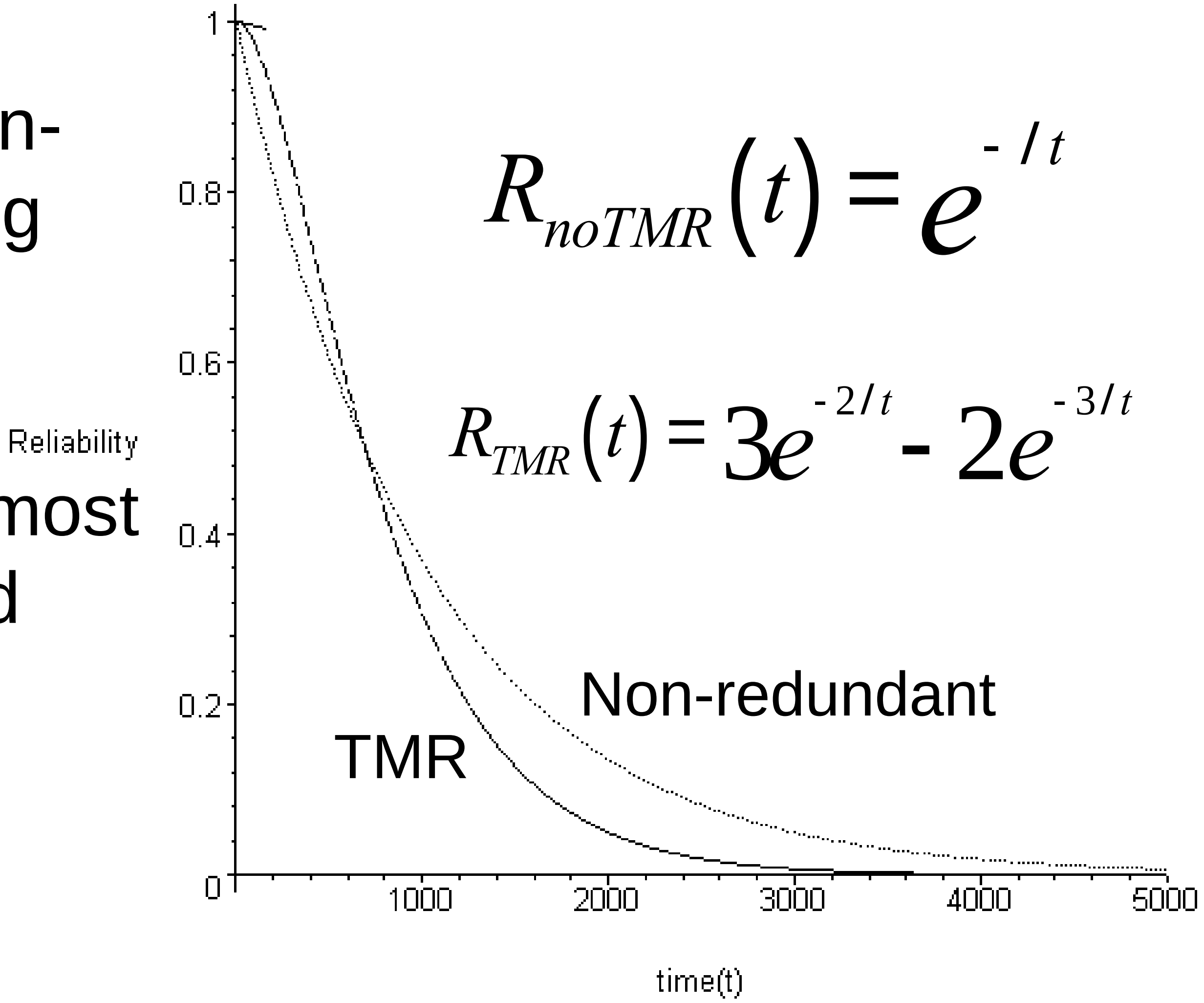


Logic Level

TMR Reliability

- TMR has lower reliability than non-redundant for long mission times

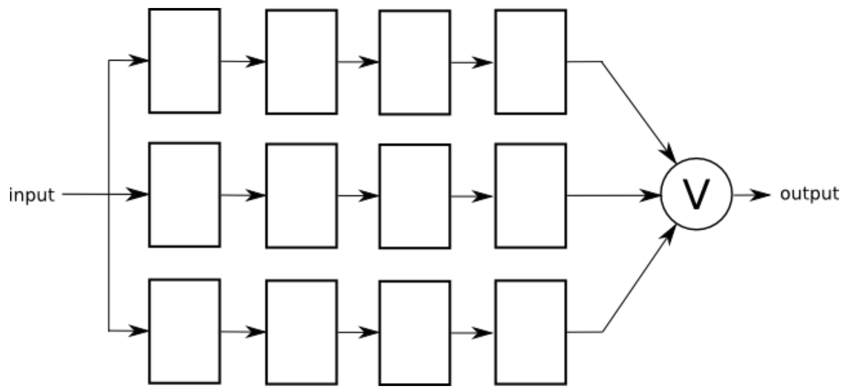
- Effective TMR almost always is coupled with “repair”

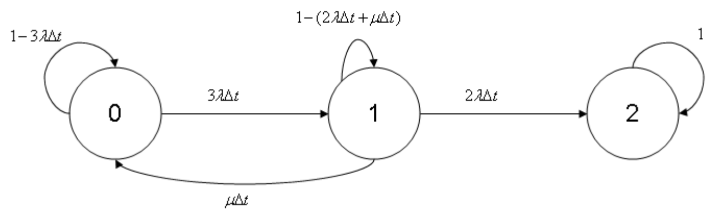


$$R_{noTMR}(t) = e^{- /t}$$

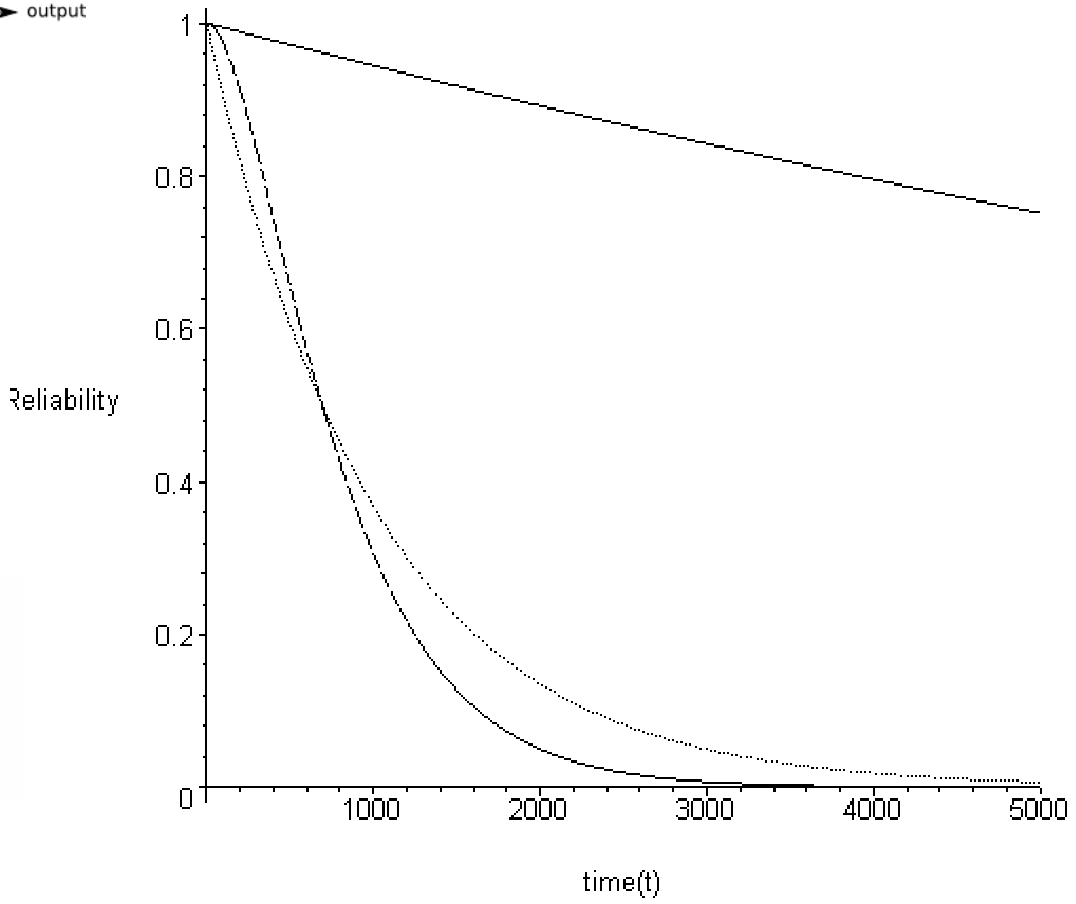
$$R_{TMR}(t) = 3e^{-2/t} - 2e^{-3/t}$$

TMR + Repair = Very Reliable!

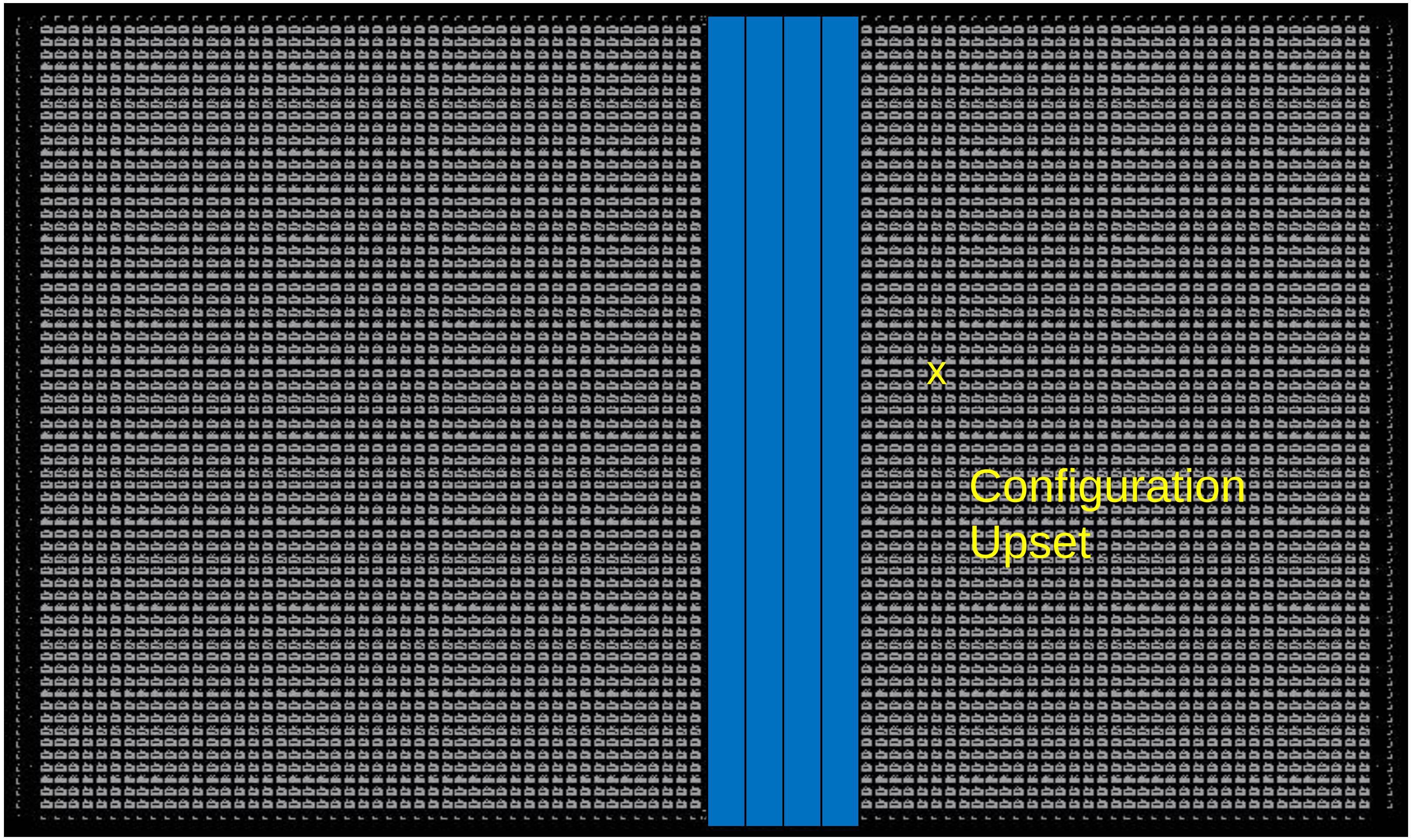




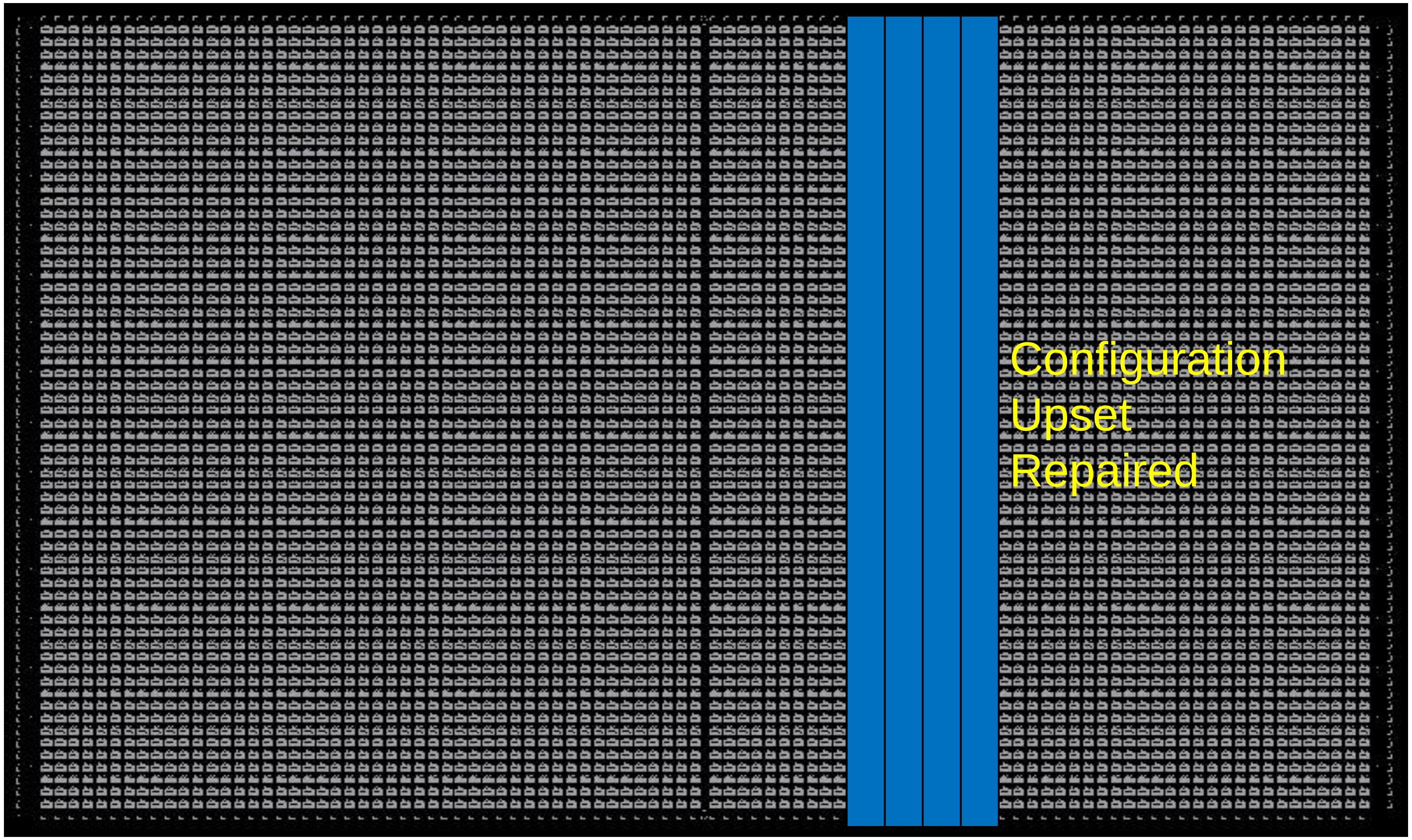
$$\begin{aligned} R(t) &= 1 - p_2(t) && (44) \\ &= \frac{(\mu+5\lambda)sinh(\frac{1}{2}t\sqrt{\mu^2+10\lambda\mu+\lambda^2})e^{-\frac{1}{2}(\mu+5\lambda)t}}{\sqrt{\mu^2+10\lambda\mu+\lambda^2}} \\ &\quad + \cosh(\frac{1}{2}t\sqrt{\mu^2+10\lambda\mu+\lambda^2})e^{-\frac{1}{2}(\mu+5\lambda)t} && (45) \end{aligned}$$



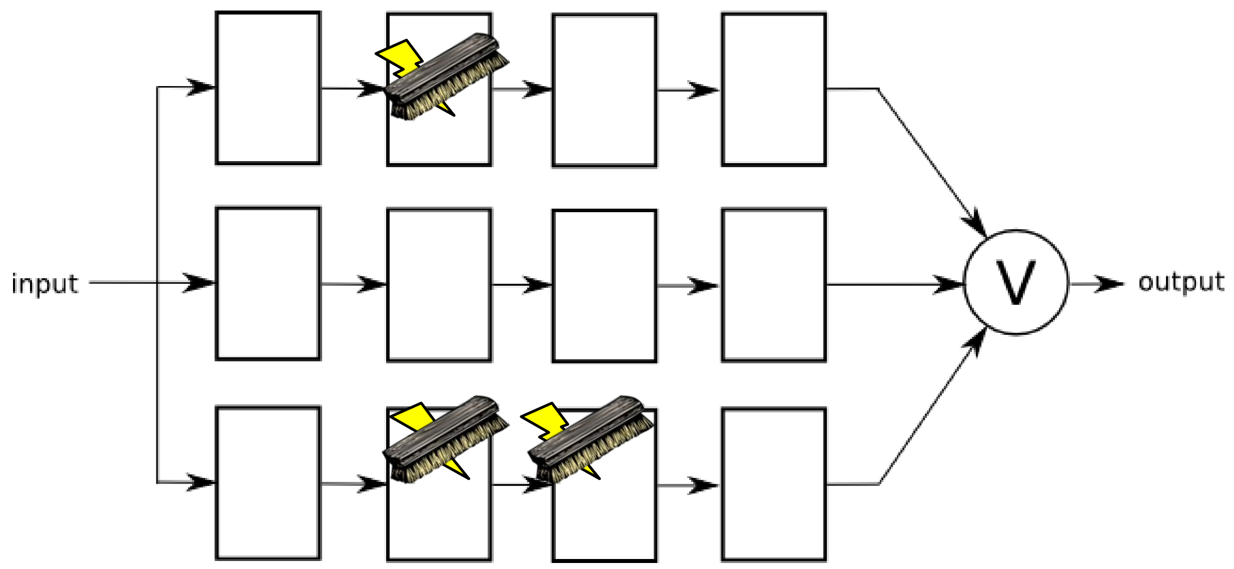
FPGA Configuration "Repair"



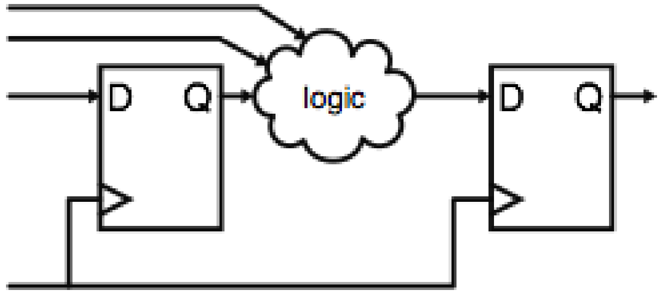
FPGA Configuration "Repair"

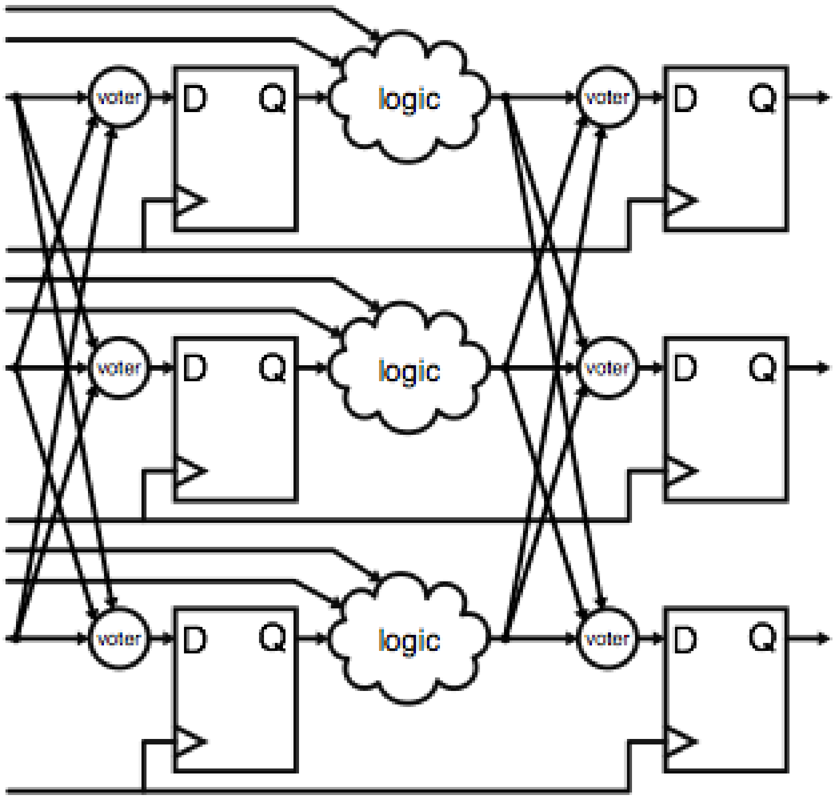


TMR & Scrubbing Example

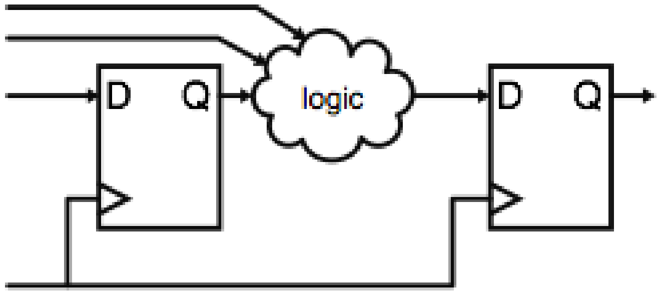


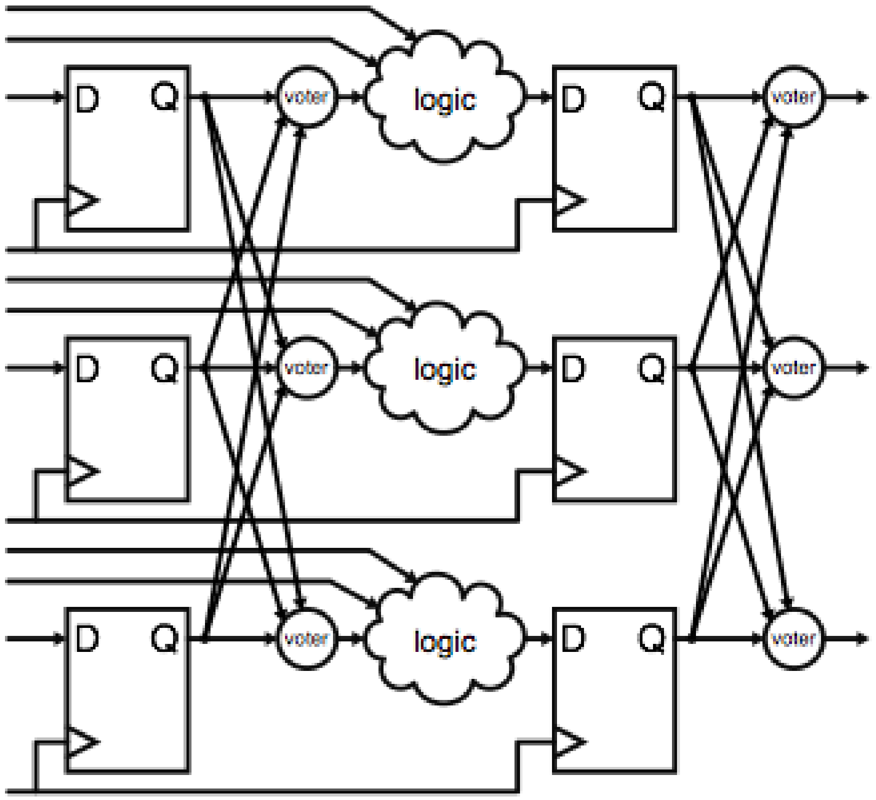
Voters Before Flip Flops



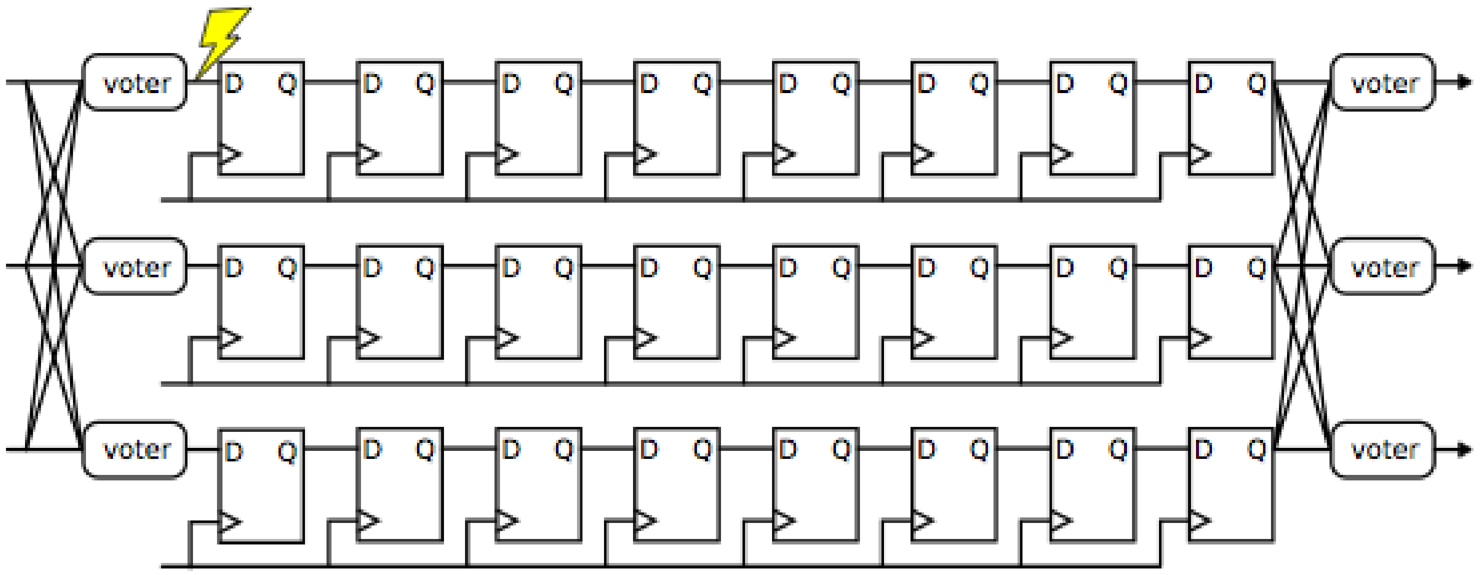


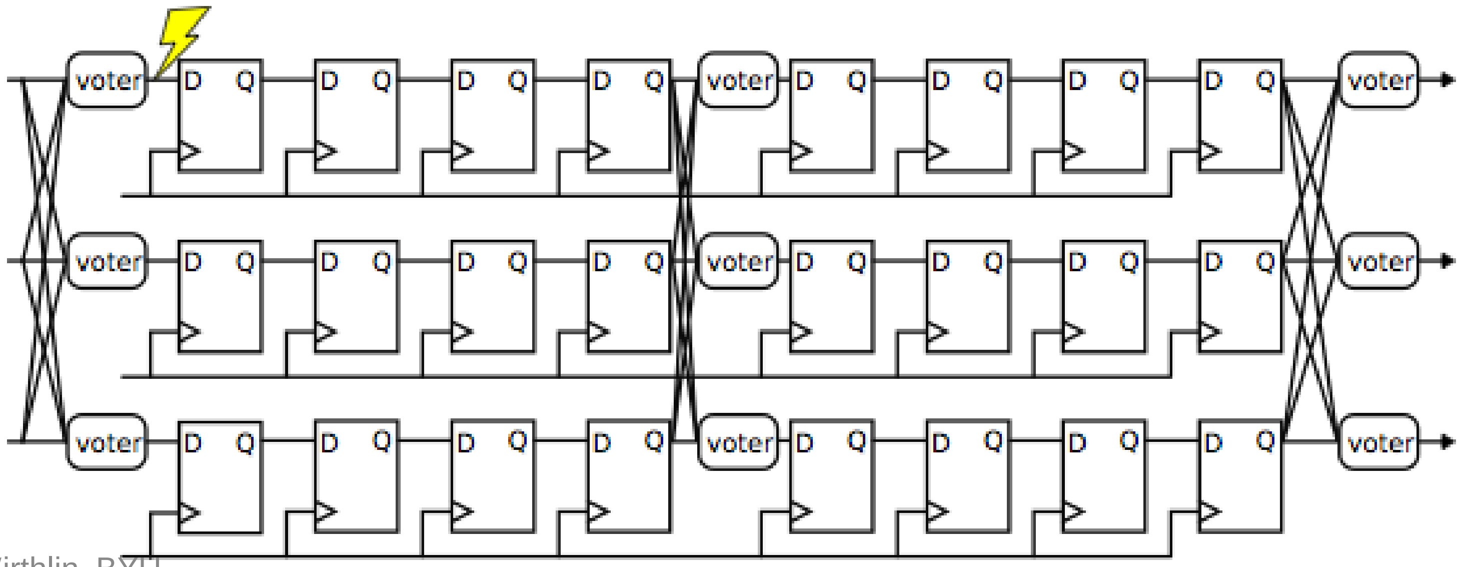
Voters After Flip-Flops





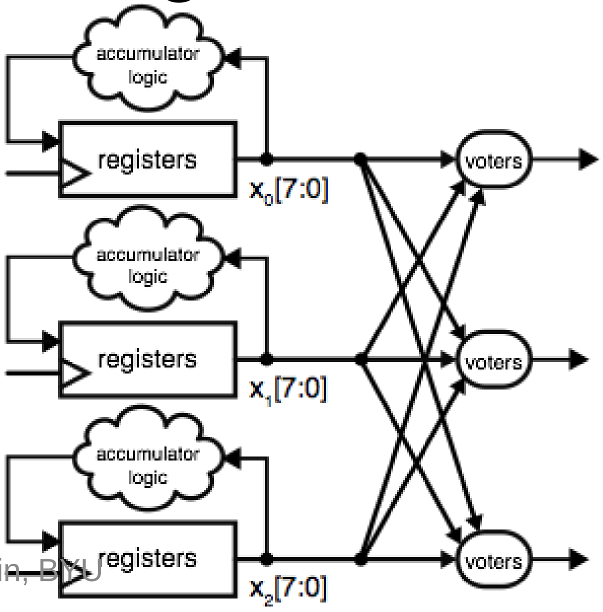
More Frequent Voting

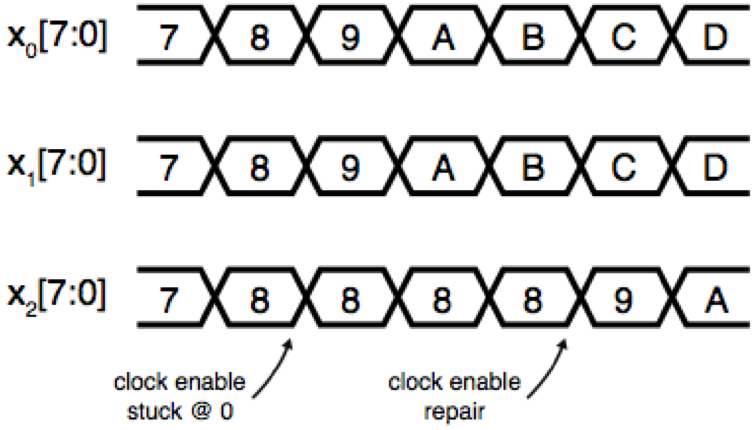




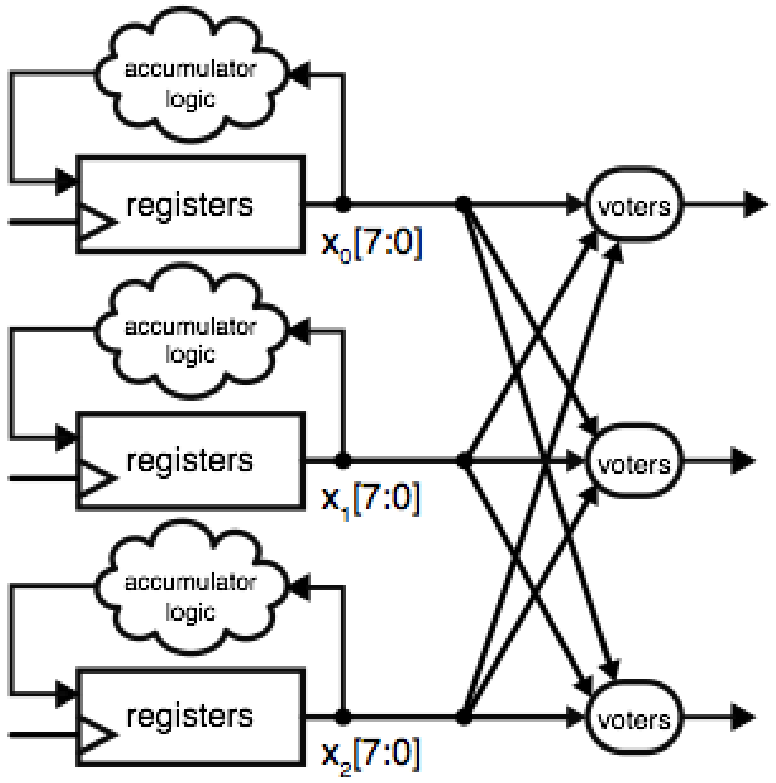
TMR Synchronization

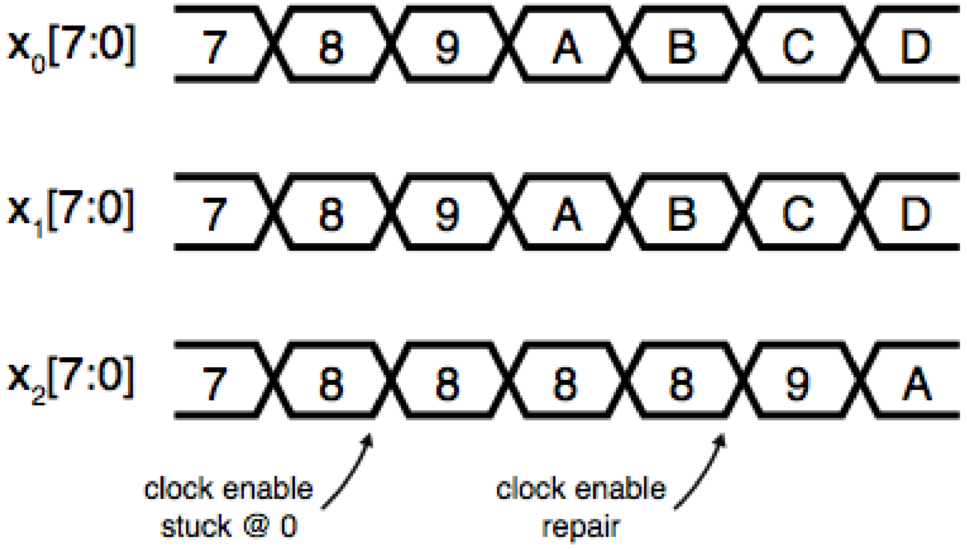
- Fault repair through scrubbing
  - *Fixes the cause of the error*
  - *Does NOT fix the state of the circuit*
- State of circuit *must* be synchronized to working circuits



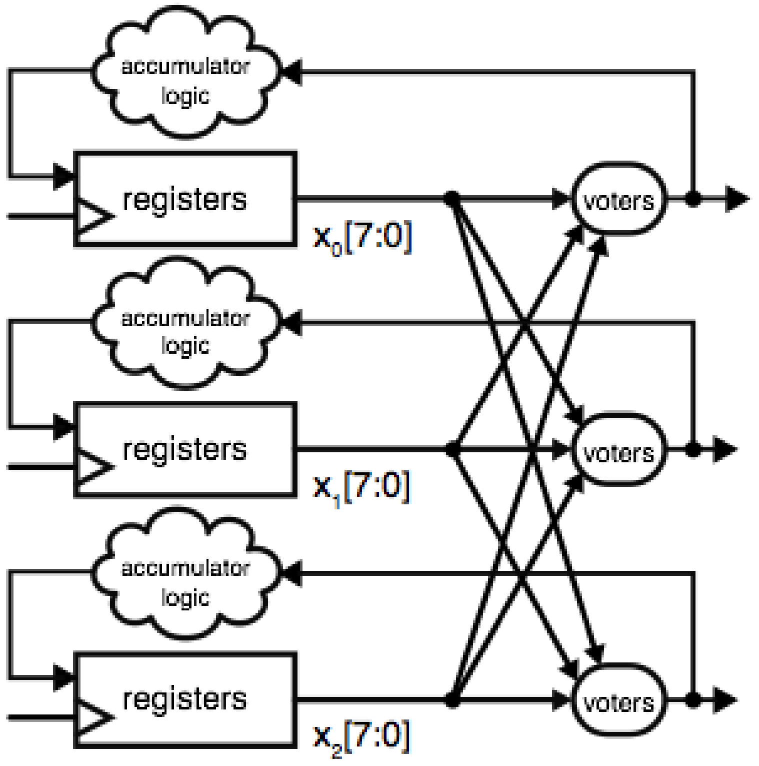


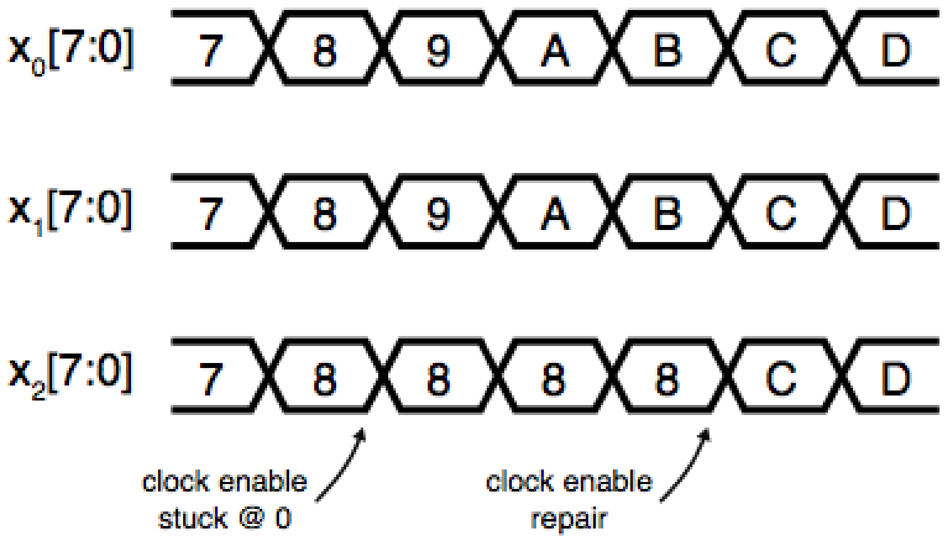
Synchronizing Voters



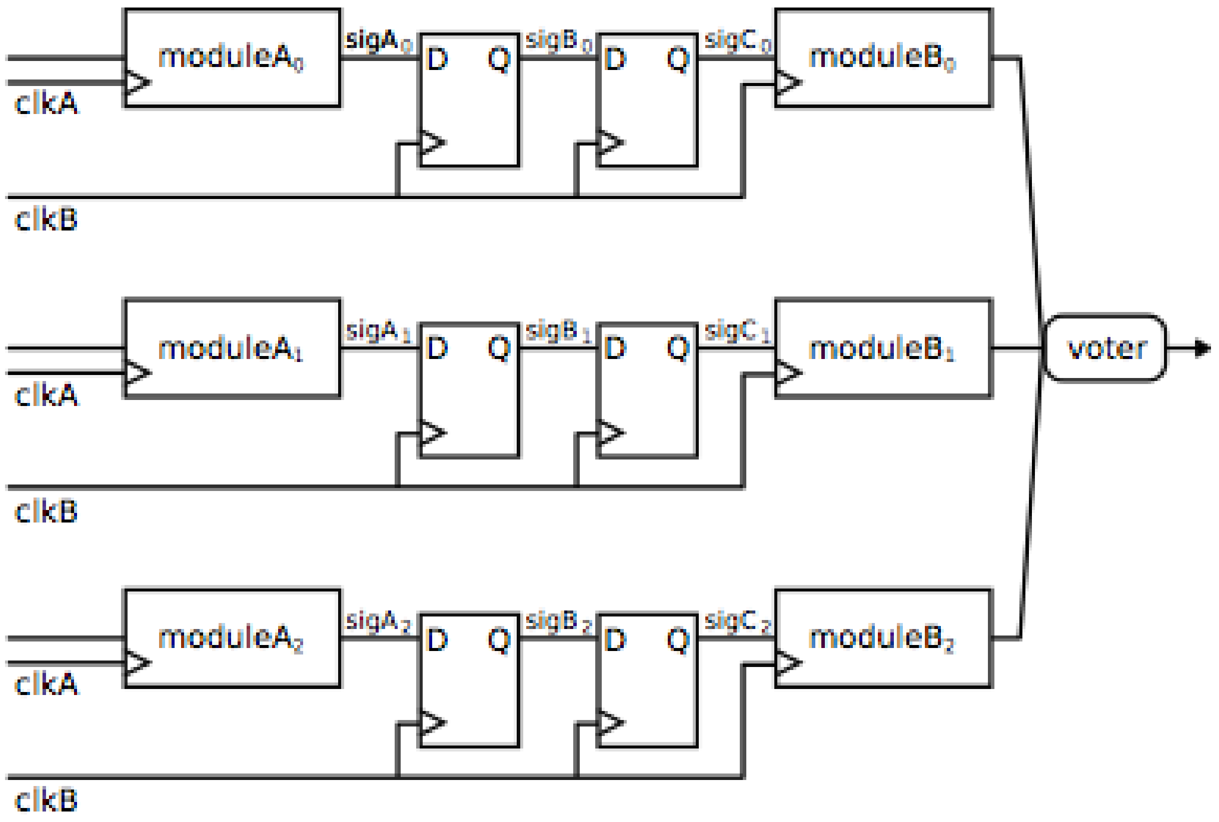


Synchronizing Voters





Clock Domain Crossing



Partial TMR

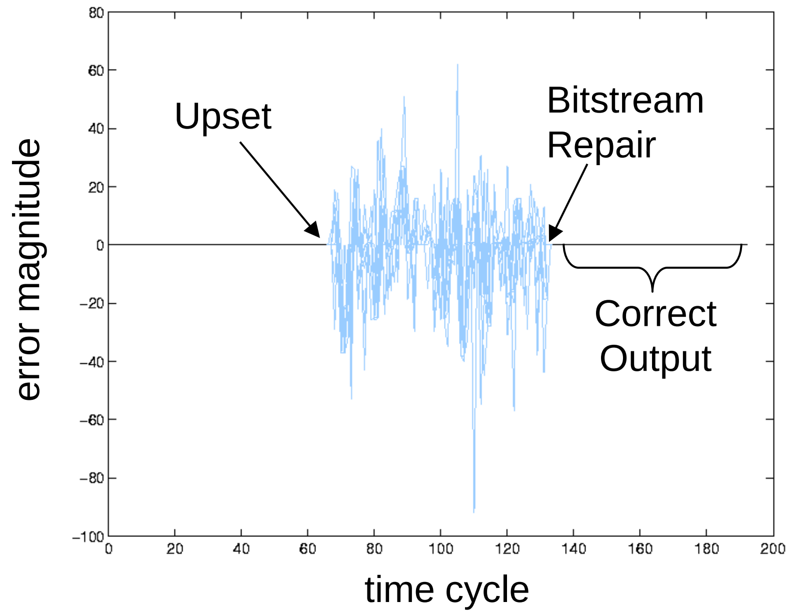
- TMR may be applied selectively
  - *Failures in some circuit areas cause more harm than others*
  - *Some circuit areas are protected by other SEE mitigation techniques (TMR not needed)*

- Challenge: deciding where to apply TMR
  - *Circuits with feedback (state machines)*
  - *Circuits with high "functional influence"*

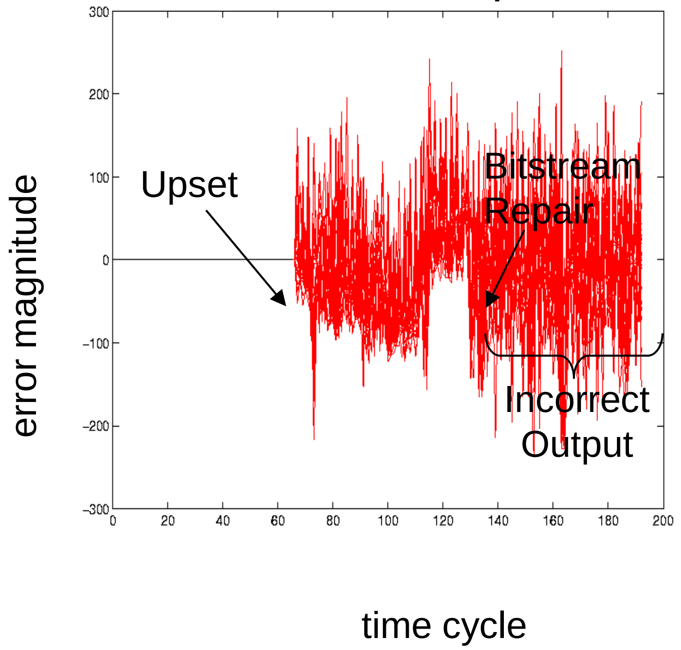
Persistent vs. Non-persistent Upset

- Some upsets repaired through *scrubbing*
  - Non-persistent upsets: repairable through scrubbing
  - Persistent upsets: requires reconfiguration

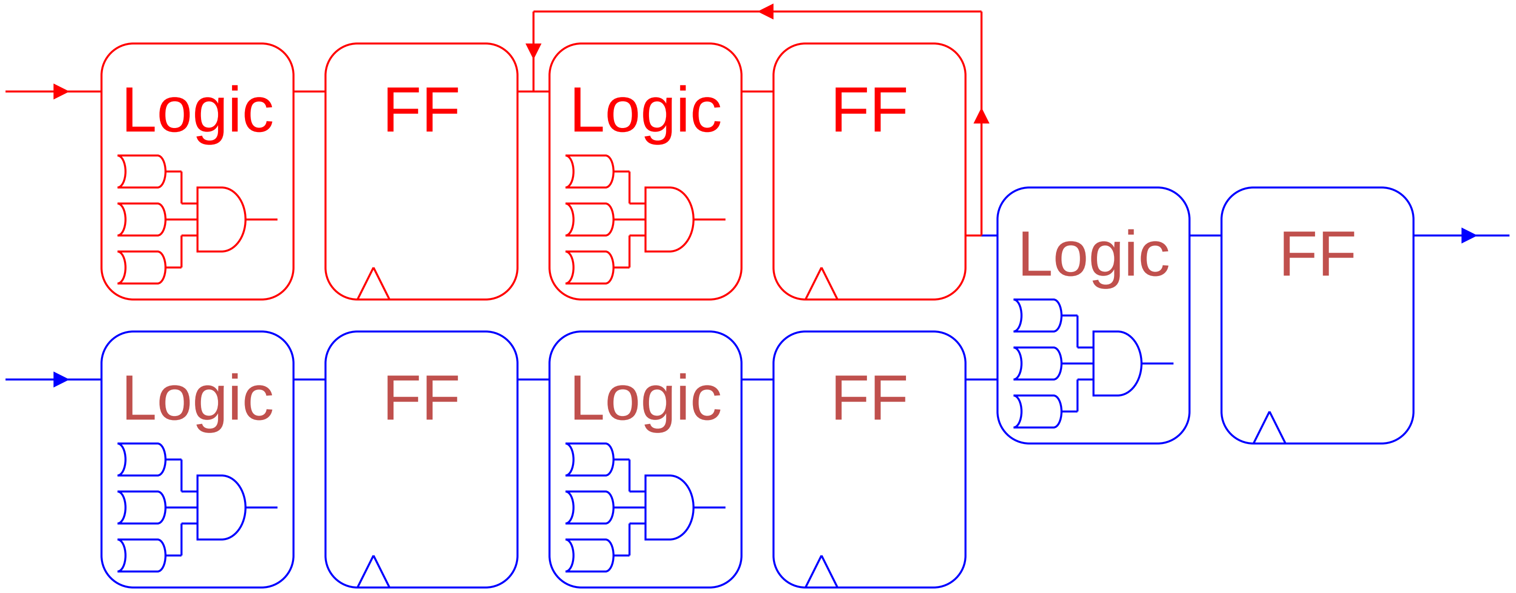
Non-Persistent Upset



Persistent Upset

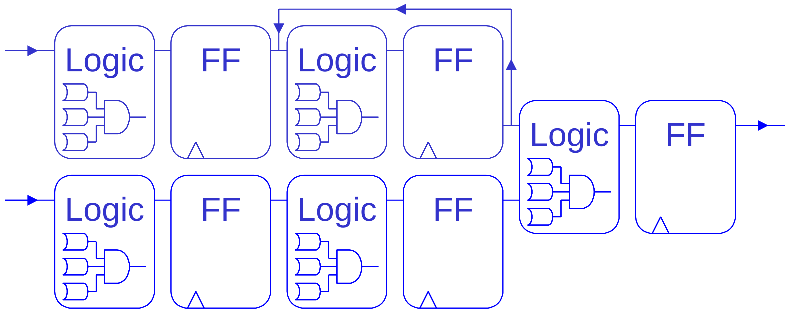


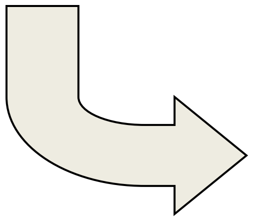
Persistent Circuit Structures

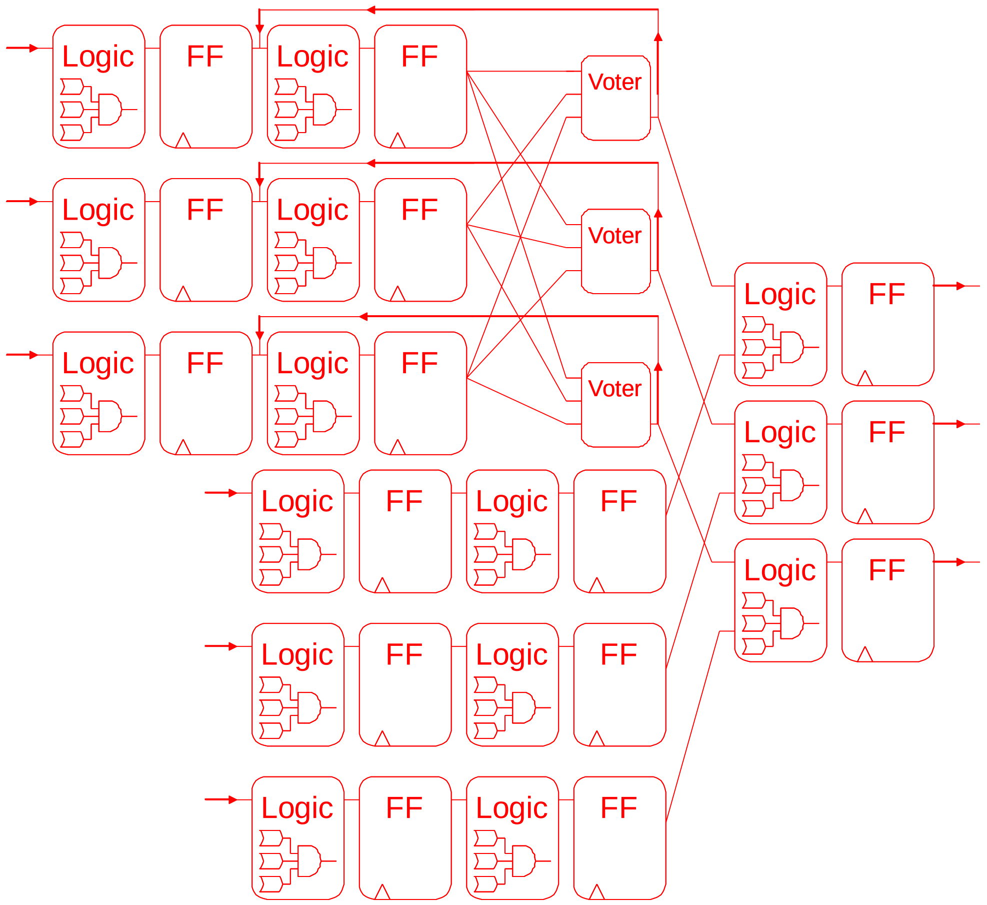


- Non-Persistent Structure – Feed-forward
- Persistent Structures – Contribute to feedback
- Partial TMR – Priority given to persistent structures

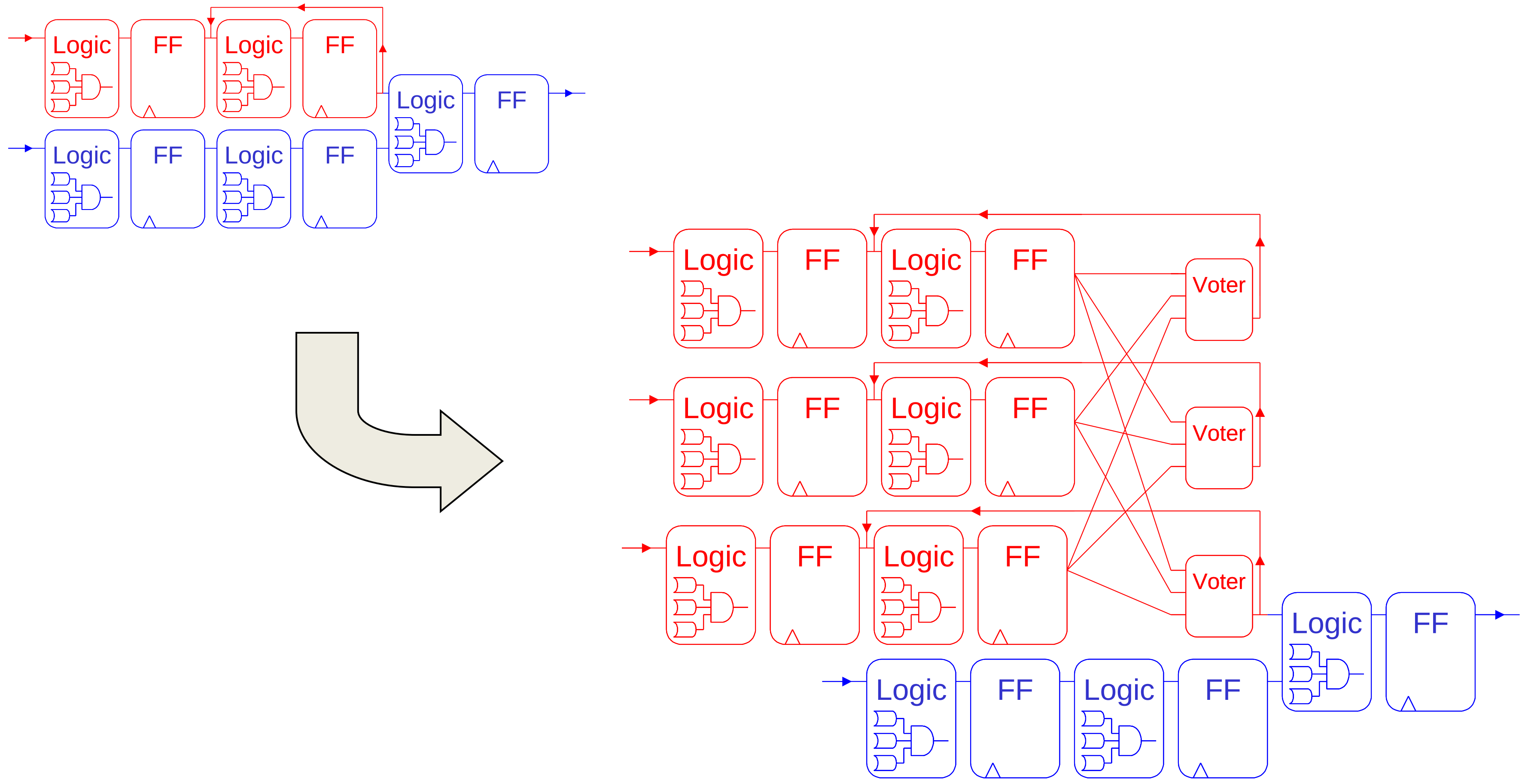
Full TMR


Logic
FF
Logic
FF
Logic
FF
Logic
FF
Logic
FF
Logic
FF




Logic
FF
Voter
Logic
FF
Voter
Logic
FF
Voter

Partial TMR



TMR Automation

- TMR is relatively easy to automate
  - *Analyze design*
  - *Replicate resources*
  - *Insert voters*
  - *Verify resulting circuit*

- Different Strategies for Automated TMR
  - *Netlist level*
  - *HDL Level*
  - *Selective/Partial*

- Several tools available for Automatic TMR

Automated TMR Tools

TMRTool

XILINX®

Precision® Hi-Rel

Mentor Graphics®

SYNPLIFY PREMIER
The Ultimate FPGA Implementation Platform

SYNOPSYS®

BL-TMR

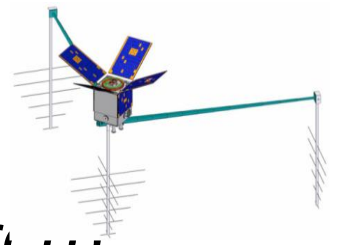
BYU™

(and other several other academic projects)

2. BL-TMR

BL-TMR

- BYU-LANL TMR Tool
  - *BYU-LANL Triple Modular Redundancy*
  - *Developed at BYU under the support of L Alamos National Laboratory (Cibola Flight Experiment)*
  - *Used to test TMR on many designs*
    - Fault injection, Radiation testing, in Orbit
  - *Testbed for experimenting with various application techniques (used for researc*



Mike Wirthlin, BYU

Ongoing Development

- Based on the success of BL-TMR, additional funding has been provided to extend BL-TMR for additional devices, environments, and address new problems
  - *Commercial companies concerned about SER rates*
    - Cisco Systems
  - *High Energy Physics*
    - Brookhaven National Laboratory (BNL), CERN
  - *Space system developers*
    - SEAKR systems, Sandia, LANL, Lockheed Martin

- Interest in BL-TMR is growing
  - *Commercialization currently under consideration*

BL-TMR (BYU/LANL TMR)

- EDIF data structure & API
  - *Parse, represent, and manipulate EDIF*
- Available tools:
  - *EDIF parser*
  - *Half-latch removal*
  - *SRL replacement*
  - *Feedback cutset tool*
  - *Full and partial TMR*
  - *Detection circuitry insertion*
  - *EDIF output*
- Project size
  - *~50 Java packages*
  - *350+ Java classes*
  - *478,401 lines of code*
  - *Includes contributions from CHREC member LANL*

```
[brian@tiger:test] java -cp ~/jars/BLTmr.jar
byucc.edif.tools.tmr.FlattenTMR ../no_tmr/synth/counters80.edf --
removeHL --full_tmr --technology virtex -p xcv1000fg680 --log
counters80.log

BLTmr Tool version 0.2.3, 12 Oct 2006
Search for EDIF files in these directories: [.]
Parsing file ../no_tmr/synth/counters80.edf
Removing half-latches...
Flattening
            Flattened circuit contains 3451 primitives, 3461
nets, and 13692 net connections
Processing: ASUF 1.0

Forcing triplication of instance safeConstantCell_zero

Analyzing design . . .
            Full TMR requested.
Triplicating design . . .
domainreport=BLTmr_domain_report.txt
            Added 1931 voters.
            3431 instances out of 3451 cells triplicated (99%
coverage)
            6862 new instances added to design.
            3431 nets triplicated (6862 new nets added).
            0 ports triplicated.
```

Tools and code available at: http://sourceforge.net/projects/byuediftools/

BL-TMR User Control

- Provides significant control to user
- Can be scripted for complex BL-TMR runs

```
Usage:
java byucc.edif.tools.tmr.FlattenTMR <input_file>
   [(-o|--output) <output_file>]
   [(-d|--dir) dir1,dir2,...,dirN ]
   [(-f|--file) file1,file2,...,fileN ]
   [--tmrSuffix suffix1,suffix2,...,suffixN ]
   [--full_tmr]
   [--tmr_inports]
   [--tmr_outports]
   [--no_tmr_p port1,port2,...,portN ]
   [--tmr_c cell_type1,cell_type2,...,cell_typeN ]
   [--tmr_i cell_instance1,cell_instance2,...,cell_instanceN ]
   [--no_tmr_c cell_type1,cell_type2,...,cell_typeN ]
   [--no_tmr_i cell_instance1,cell_instance2,...,cell_instanceN ]
   [--notmrFeedback]
   [--notmrInputToFeedback]
   [--notmrFeedBackOutput]
   [--notmrFeedForward]
   [--noInoutCheck]
   [--SCCSortType <{1|2|3}>]
   [--doSCCDecomposition]
   [--inputAdditionType <{1|2|3}>]
   [--outputAdditionType <{1|2|3}>]
   [--mergeFactor <mergeFactor>]
   [--optimizationFactor <optimizationFactor>]
   [--factorType <{DUF|UEF|ASUF}>]
   [--factorValue <factorValue>]
   [--low <low>]
   [--high <high>]
   [--inc <inc>]
   [--removeHL]
   [--hlConst <{0|1}>]
   [--hlUsePort <hlPortName>]
   [--technology <{virtex|virtex2}>]
   [(-p|--part) <part>]
   [--summary]
   [--log <logfile>]
   [--domainReport <domainReport>]
   [--writeConfig[:<config_file>]]
   [-h|--help]
   [-v|--version]

For detailed usage, try `--help'
```

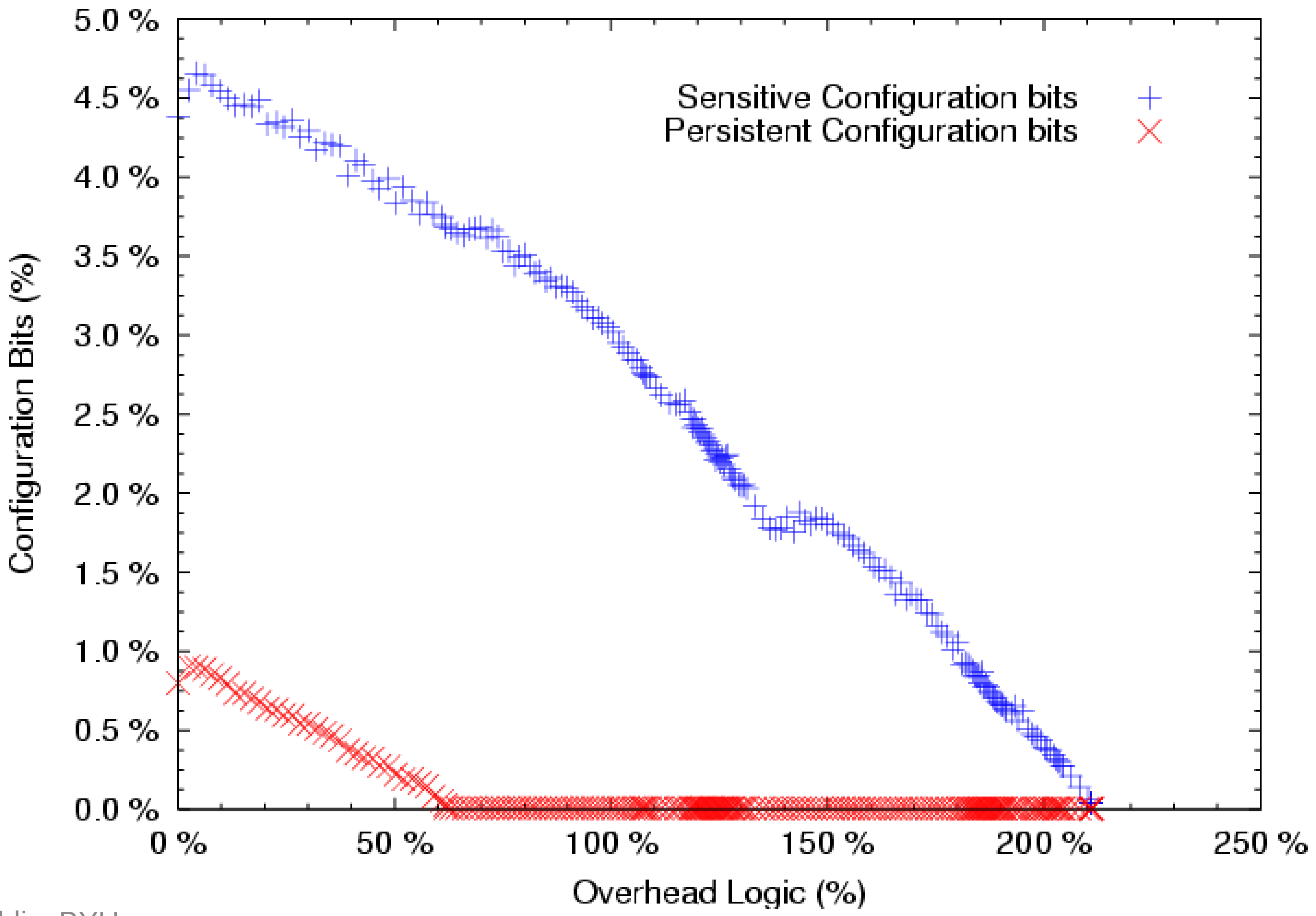
Sample Execution

```
[brian@tiger:test] java -cp ~/jars/BLTmr.jar byucc.edif.tools.tmr.FlattenTMR
../no_tmr/synth/counters80.edf --removeHL --full_tmr --technology virtex -p xcv1000fg680
--log counters80.log

BLTmr Tool version 0.2.3, 12 Oct 2006
Search for EDIF files in these directories: [.]
Parsing file ../no_tmr/synth/counters80.edf
Removing half-latches...
Flattening
          Flattened circuit contains 3451 primitives, 3461 nets, and 13692 net
connections
Processing: ASUF 1.0

Forcing triplication of instance safeConstantCell_zero

Analyzing design . . .
          Full TMR requested.
Triplicating design . . .
domainreport=BLTmr_domain_report.txt
          Added 1931 voters.
          3431 instances out of 3451 cells triplicated (99% coverage)
          6862 new instances added to design.
          3431 nets triplicated (6862 new nets added).
          0 ports triplicated.
```

Cost of TMR

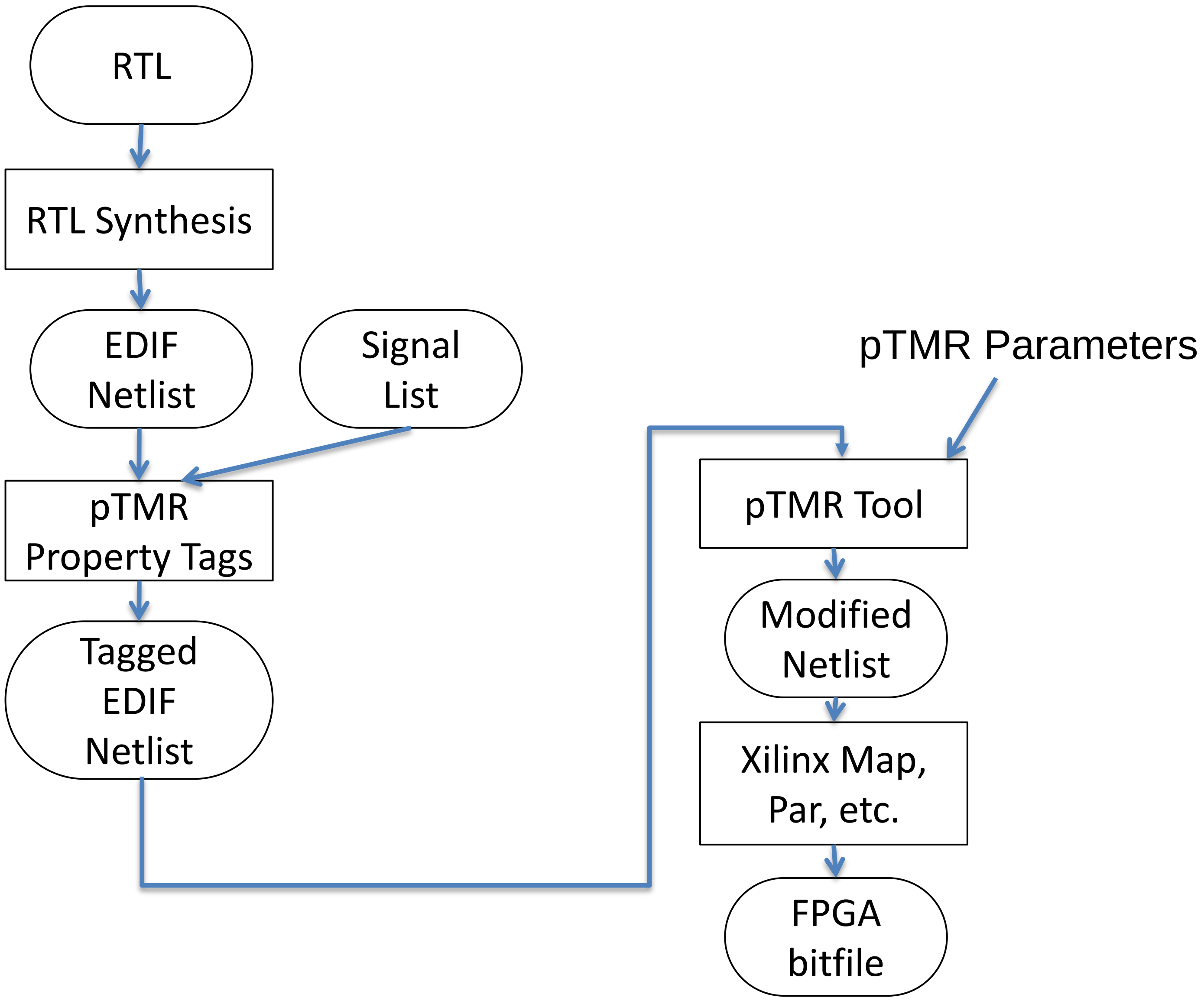
	Size Increase	Critical Path Before TMR	Critical Path After TMR	% Increase in Critical Path
blowfish	3.1X	28.3 ns	31.7 ns	12.0%
des3	3.4X	11.1 ns	13.6 ns	22.5%
qpsk	3.1X	80.0 ns	83.9 ns	4.9%
free6502	3.3X	29.6 ns	33.1 ns	11.8%
T80	3.3X	27.8 ns	33.7 ns	21.2%
macfir	3.9X	14.4 ns	19.5 ns	35.4%
serial_divide	4.1X	9.2 ns	12.2 ns	32.6%
planet	3.1X	10.9 ns	12.6 ns	15.6%
s1488	3.1X	9.9 ns	12.0 ns	21.2%
s1494	3.1X	10.4 ns	12.2 ns	17.3%
s298	3.1X	15.8 ns	19.1 ns	20.9%
tbk	3.9X	10.3 ns	12.9 ns	25.2%
synthetic	4.0X	9.9 ns	10.4 ns	5.1%
lfsrs	6.3X	9.0 ns	12.7 ns	41.1%
ssra_core	3.5X	6.1 ns	7.2 ns	18.0%
mean	**3.6X**	**8.17 ns**	**12.08 ns**	**16.0%**

BL-TMR Incremental Results



3. Design Flow

Design Flow


RTL
RTL Synthesis
EDIF Netlist
Signal List
pTMR Property Tags
Tagged EDIF Netlist
pTMR Parameters
pTMR Tool
Modified Netlist
Xilinx Map, Par, etc.
FPGA bitfile

pTMR Steps

1. Component Merging
2. Design Flattening
3. Graph Creation and Analysis
4. IOB Analysis
5. Clock Domain Analysis
6. Instance Removal
7. Feedback Analysis
8. Illegal Crossing identification
9. TMR Prioritization & Selection
10. Voter Selection
11. Netlist generation

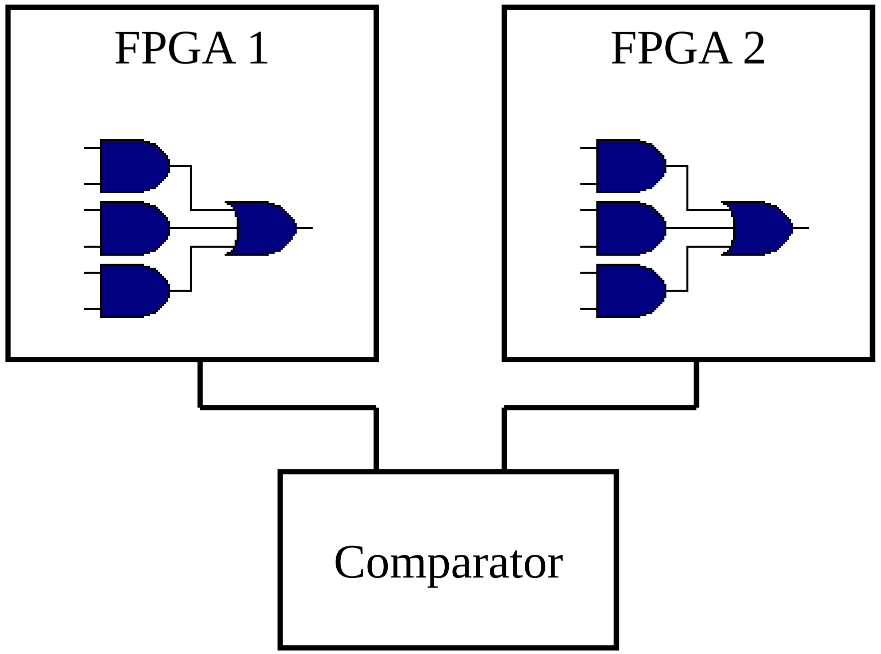
11. Netlist Generation

- Circuit generated from pTMR rules
  - *Cells triplicated*
  - *Voters inserted*
- Netlist created for new circuit

3. Verifying BL-TMR

Fault Injection

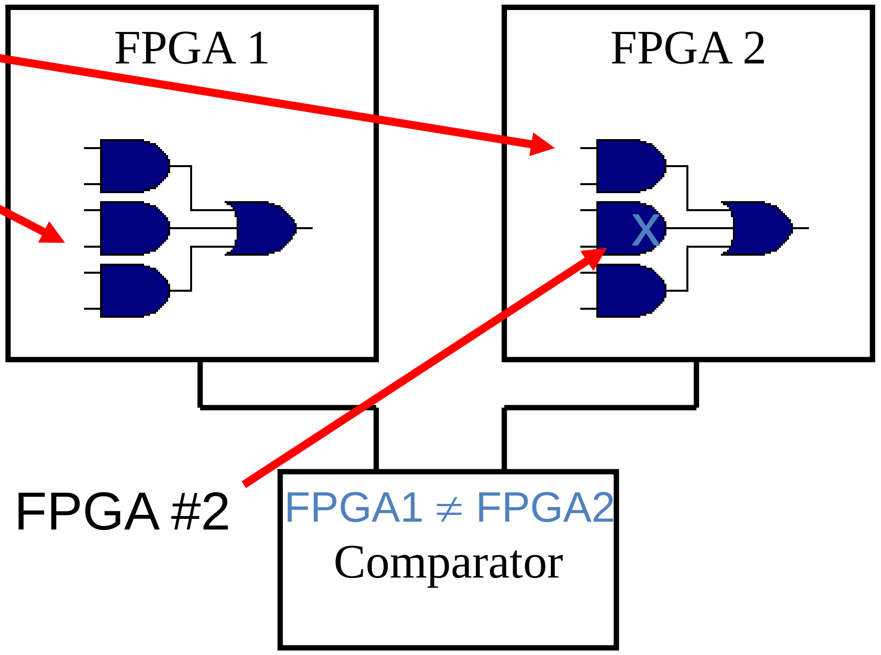
- Configure user design onto two identical FPGAs
- Compare results of two designs using Comparator FPGA
- Insert configuration SEUs into design under test (FPGA2) and compare results
- If discrepancies between FPGAs are found, record configuration error



SEU Insertion Example #1

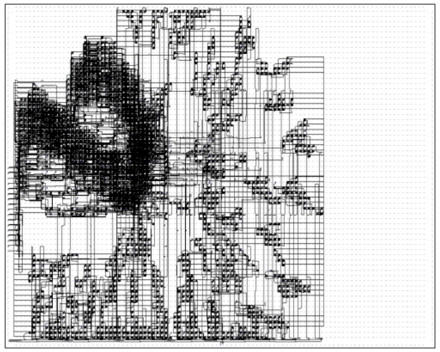
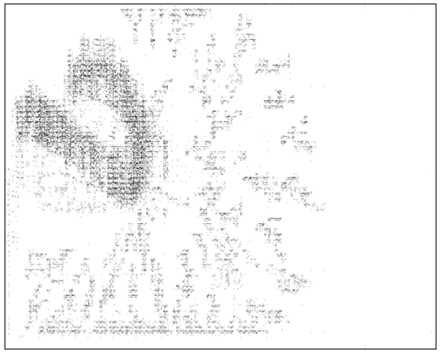
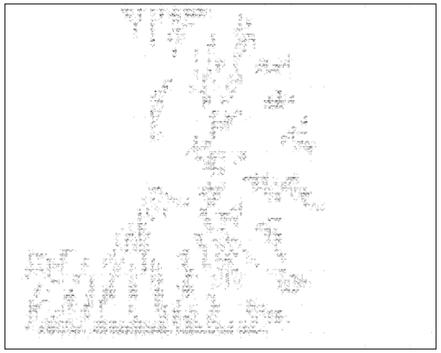
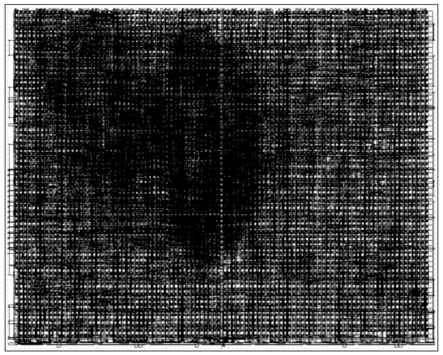
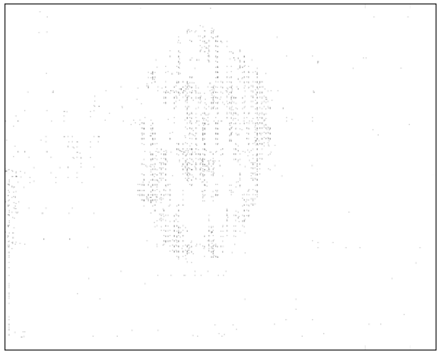
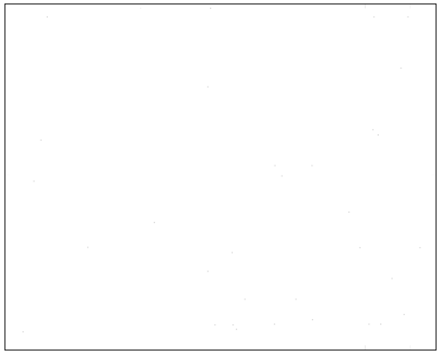
Apply test vector to circuit input

Insert configuration SEU into FPGA #2



Compare circuit results

Experimental Results – Design #2

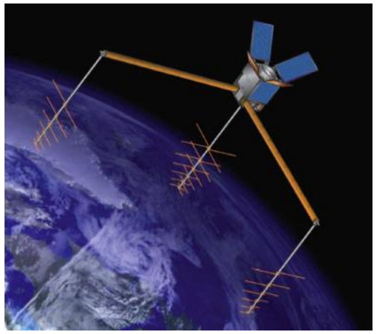
	FPGA Editor Layout	Sensitivity Map	Persistence Map
Unmitigated	 3,005 slices (24%)	 254,840 (4.39%)	 46,368 (0.80%)
Full TMR Applied	 12,165 slices (99%)	 2,395 (0.041%)	 671 (0.005%)

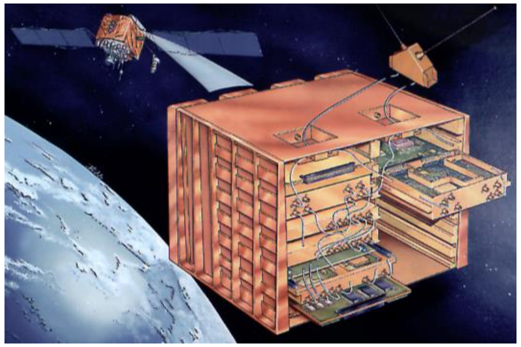
LANL Cibola Flight Experiment

- Los Alamos National Laboratory technology pathfinder
  - validate FPGAs for high performance computing
  - Investigate SEU behavior of Xilinx Virtex FPGAs

- Several BYU experiments validated in orbit
  - TMR (including BL-TMR tool)
  - Duplication with Compare
  - DRAM controllers

Cibola Flight Experiment
560 km, 35.4º inclination





Sandia MISSE-8

Under *direction* of Sandia National Laboratory

- BYU Experiments on ISS
  - *TMR PicoBlaze (Successful mitigation event!)*
  - *Smart signal detection*
  - *Reduced Precision Redundancy*
  - *BRAM Scrubbing & BRAM ECC*

Photo courtesy of NASA

Endeavor (STS-134)
May 16, 2012

V4 FX60

V5QV (SIRF)

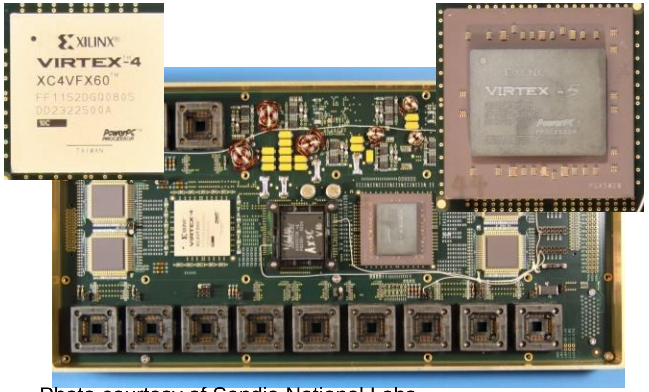


Photo courtesy of Sandia National Labs

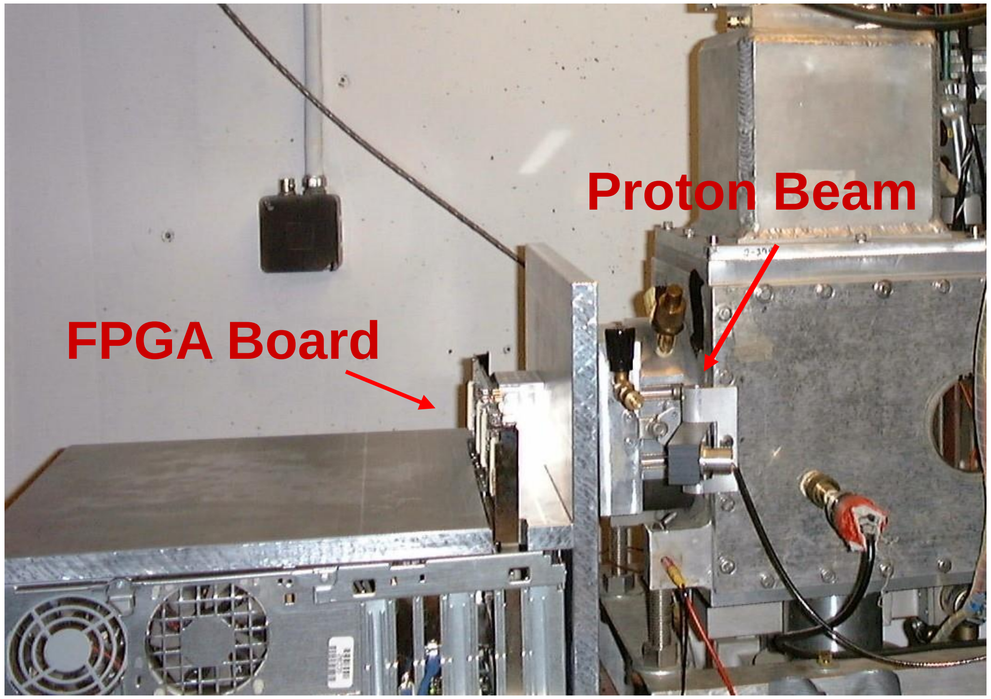
Photo courtesy of NASA

Radiation Testing

- Apply Ionizing Radiation to Design with TMR
  - *Verify accuracy of artificial simulator*
  - *Identify upset in non-configuration state*
  - *Identify other failure modes*

UC Davis, Crocker Nuclear Laboratory

- Medium-energy particle accelerator (76-inch cyclotron)
- 63 MeV proton source
- Flux: 1e7 particles/cm^2/second: (~1 upset/second)
- 16 hour test (~25,000 upsets)



5. TMR Summary

- Pros:
  - *Significant improvements in reliability*
  - *Easy to apply (limited design effort)*
  - *Can be applied selectively*
- Cons
  - *Requires significant hardware resources*
  - *Negative impact on timing*
  - *Difficult to verify*

Alternatives to TMR

- Exploit specific circuit structures/styles
  - *Memories, state machines, processors, etc.*
  - *Arithmetic structures*
- Detection+
  - *Detecting a fault quickly opens up many lower cost mitigation strategies*
- Temporal Redundancy
- Duplication with Compare

Future Plans

- Clock domain aware TMR
- Timing aware TMR
- Improved support for clock and I/O resources
- Integrated Duplication with Compare (DWC)
- More frequent voting
- NMR (5-MR, 7-MR, etc.)
- Support for New FPGA Architectures
- Improved verification (formal verification)
- GUI support
- Improved partial TMR selection (Algorithmic pTMR)

Questions?