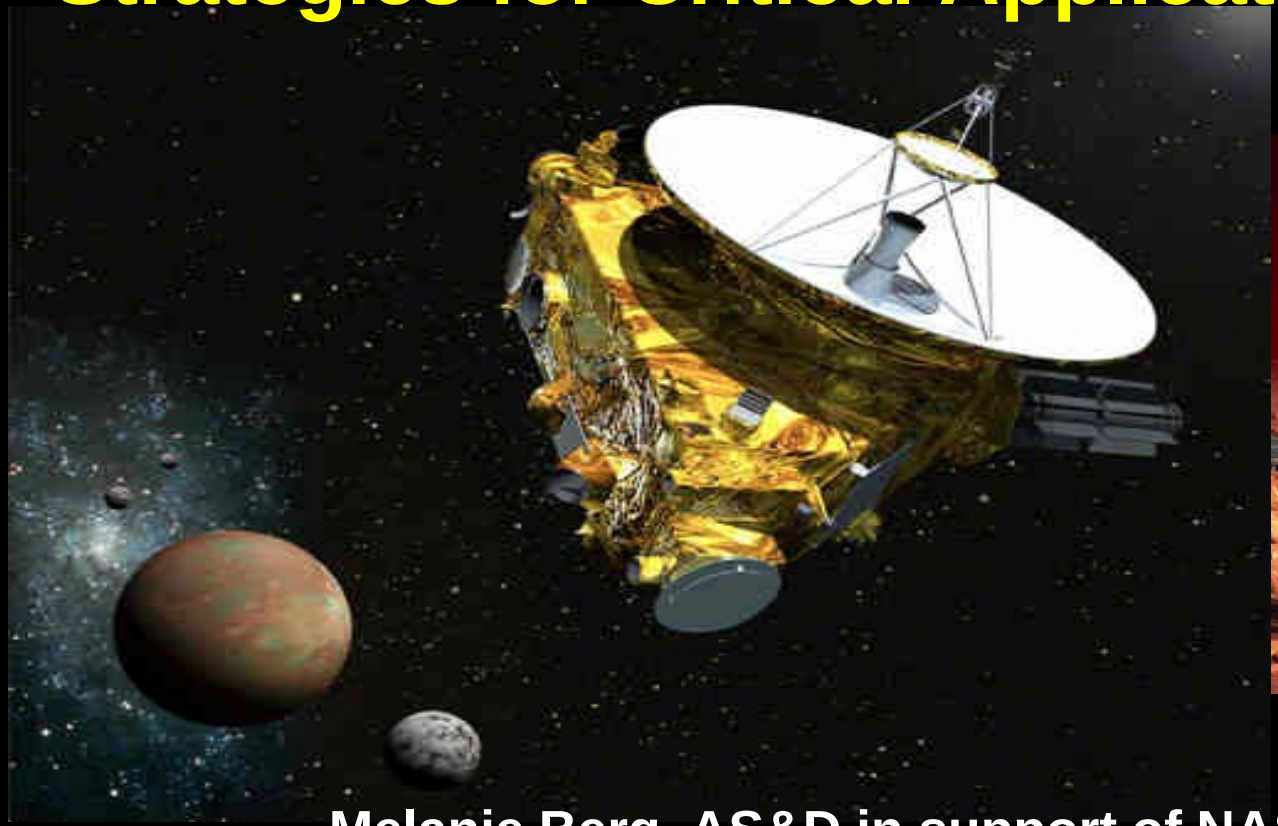
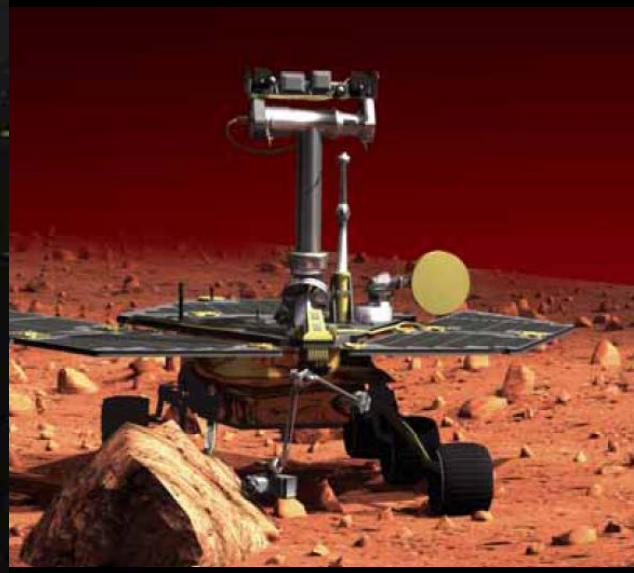
New Developments in Error Detection and Correction Strategies for Critical Applications





Melanie Berg, AS&D in support of NASA/GSFC

Melanie.D.Berg@NASA.gov

Keh LaBel, NASA/GSFC

To be published on seemapld.org originally presented at 2016 Single Event Effects (SEE) Symposium and the Military and Aerospace Programmable Logic Devices (MAPLD) Workshop.

Acronyms

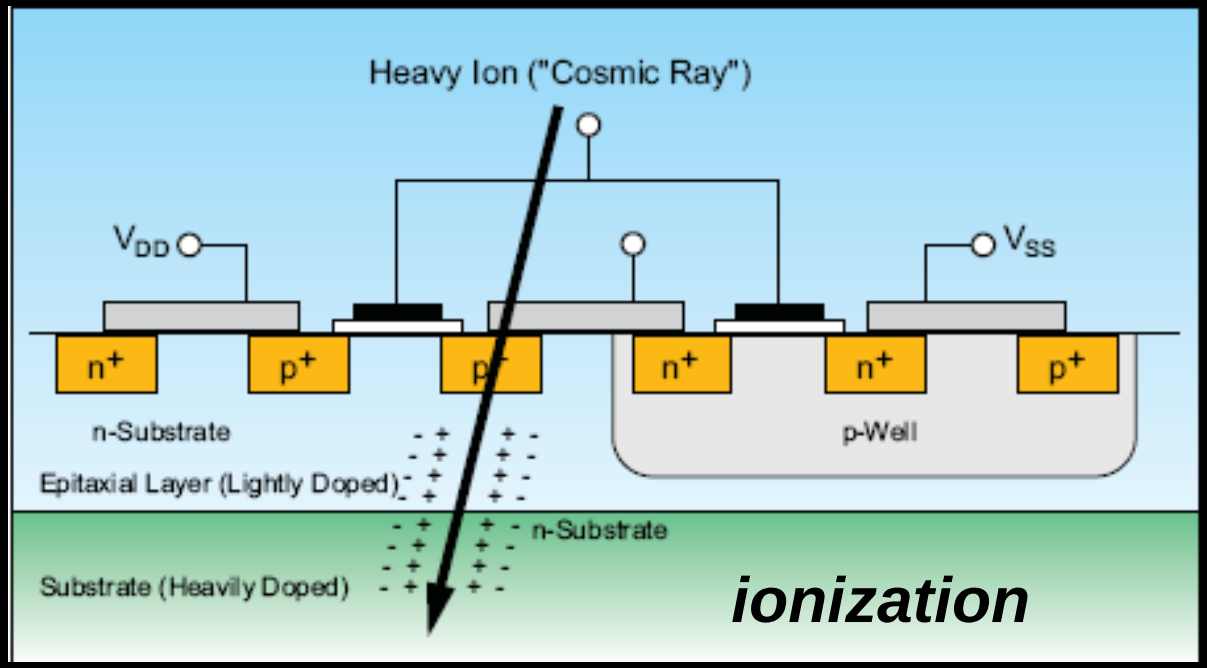
- Application specific integrated circuit (ASIC)
- Block random access memory (BRAM)
- Block Triple Modular Redundancy (BTMR)
- Clock (CLK or CLKB)
- Combinatorial logic (CL)
- Configurable Logic Block (CLB)
- Digital Signal Processing Block (DSP)
- Distributed triple modular redundancy (DTMR)
- Dual interlocked cell (DICE)
- Edge-triggered flip-flops (DFFs)
- Equivalence Checking (EC)
- Error detection and correction (EDAC)
- Field programmable gate array (FPGA)
- Gate Level Netlist (EDF, EDIF, GLN)
- Global triple modular redundancy (GTMR)
- Hardware Description Language (HDL)
- Input – output (I/O)
- Linear energy transfer (LET)
- Local triple modular redundancy (LTMR)
- Look up table (LUT)
- NASA Electronic Parts and Packaging (NEPP)
- Operational frequency (*fs)*
- Power on reset (POR)
- Place and Route (PR)
- Radiation Effects and Analysis Group (REAG)
- Single event functional interrupt (SEFI)
- Single event effects (SEEs)
- Single event latch-up (SEL)
- Single event transient (SET)
- Single event upset (SEU)
- Single event upset cross-section (σ_{SEU})
- Static random access memory (SRAM)
- System on a chip (SOC)
- Temporal redundancy (TR)
- Total ionizing dose (TID)
- Windowed shift register (WSR)

Agenda

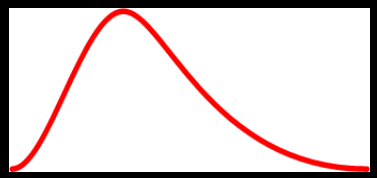
- Single Event Upsets (SEUs) in Digital Devices.
- Single Event Upsets and FPGA Configuration.
- Single Event Upsets in FPGA Data Paths.
- Fail-Safe Strategies for Critical Applications.
- Dual Redundancy:
  - Lockstep and
  - Separate systems.
- Cold Sparing.
- Triple modular redundancy (TMR):
  - Block TMR (BTMR),
  - Local TMR (LTMR),
  - Distributed TMR (DTMR), and
  - Global TMR (GTMR).
- Fail-Safe State Machines.

SEUs in Digital Devices

Although there are many sources of FPGA malfunction, this presentation will focus on SEUs as a source of failure.



Single event transient: SET



If an SET gets caught by a memory element, then it becomes an SEU

SEUs versus Total Ionizing Dose (TID)

- **The two are commonly confused.**
- **TID:**
  - **It is measured in rads and causes gradual degradation of device performance.**
  - **In other words, TID is dose that can cause device failure from exposure to ionizing particles (mostly protons and electrons) over time.**
- **SETs and SEUs have nothing to do with dose over time.**
  - **One particle's passage through a sensitive region of a device.**
  - **Causes ionization and can cause a transistor to change it's state.**

How SEUs Affect FPGAs

- **SEU and SET error signatures vary between FPGA devices:**
  - **Temporary glitch (transient),**
  - **Change of state (in correct state machine transitions),**
  - **Global upsets: Loss of clock or unexpected reset,**
  - **Route breakage (no signal can get through), and**
  - **Configuration corruption.**
- **The question is how to avoid system failure and the answer depends on the following:**
  - **The system's requirements and the definition of failure,**
  - **The target device and its surrounding circuitry susceptibility,**
  - **Implemented fail-safe strategies,**
  - **Reliable design practices,**
  - **Radiation environment, and**
  - **Trade space and decided risk.**

FPGA SEU Categorization as Defined by NASA Goddard REAG:

SEU cross section: σ_{SEU}

$$P(fs)_{error} \propto P_{Configuration} + P(fs)_{functionalLogic} + P_{SEFI}$$

Design σ_{SEU}

Configuration σ_{SEU}

Functional logic σ_{SEU}

SEFI σ_{SEU}

Sequential and Combinatorial logic (CL) in data path

Global Routes and Hidden Logic

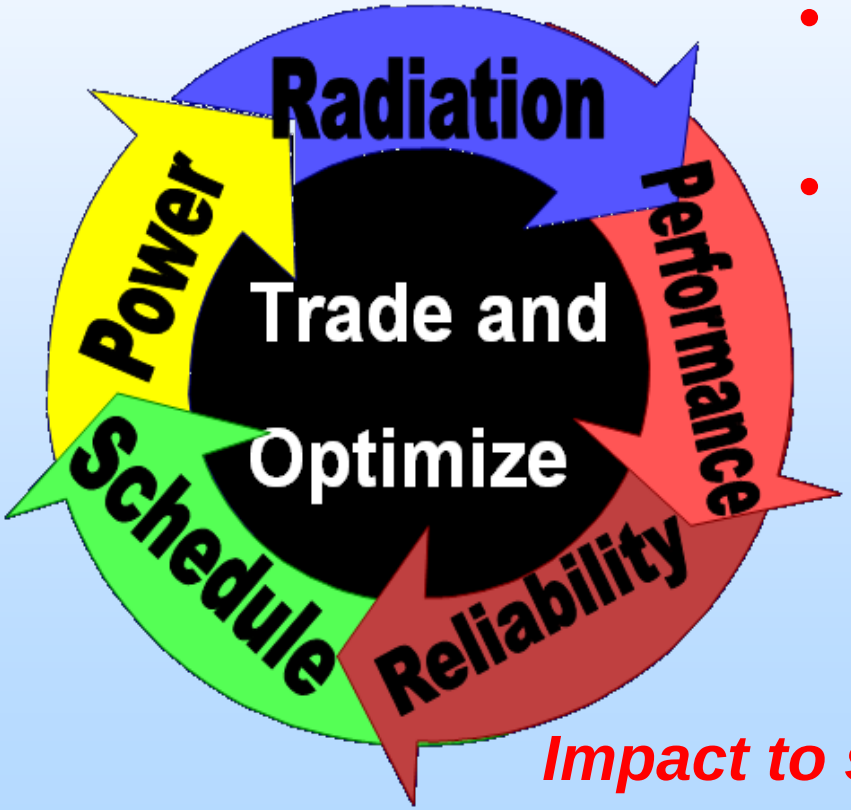
SEU Testing is required in order to characterize the σ_{SEU}s for each of FPGA categories.

To be published on seemapld.org originally presented at 2016 Single Event Effects (SEE) Symposium and the Military and Aerospace Programmable Logic Devices (MAPLD) Workshop.

Preliminary Design Considerations for Mitigation And Trade Space

Determine Most Susceptible Components:

$$P(fs)_{error} \propto P_{Configuration} + P(fs)_{functionalLogic} + P_{SEFI}$$



- **Does the designer need to add mitigation?**
- **Will there be compromises?**
  - **Performance and speed,**
  - **Power,**
  - **Schedule**
  - **Mitigating the susceptible components?**
  - **Reliability (working and mitigating as expected)?**

Impact to speed, power, area, reliability, and schedule are important questions to ask.

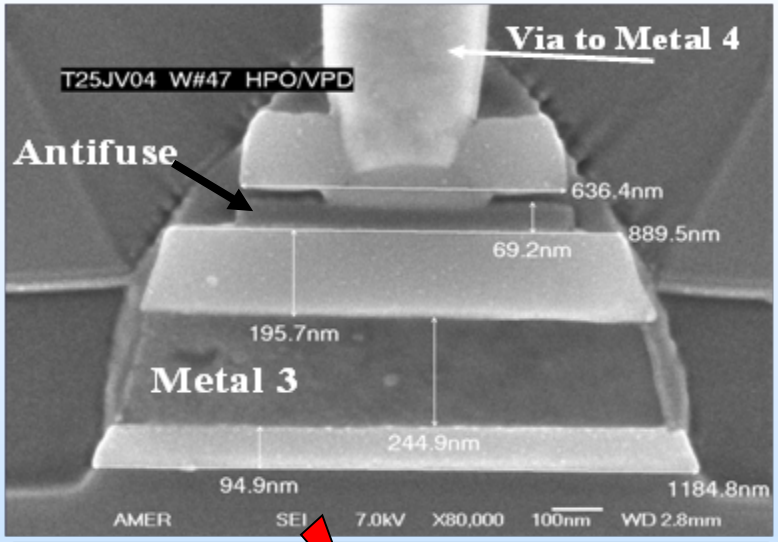
To be published on seemapld.org originally presented at 2016 Single Event Effects (SEE) Symposium and the Military and Aerospace Programmable Logic Devices (MAPLD) Workshop.

Single Event Upsets and FPGA Configuration

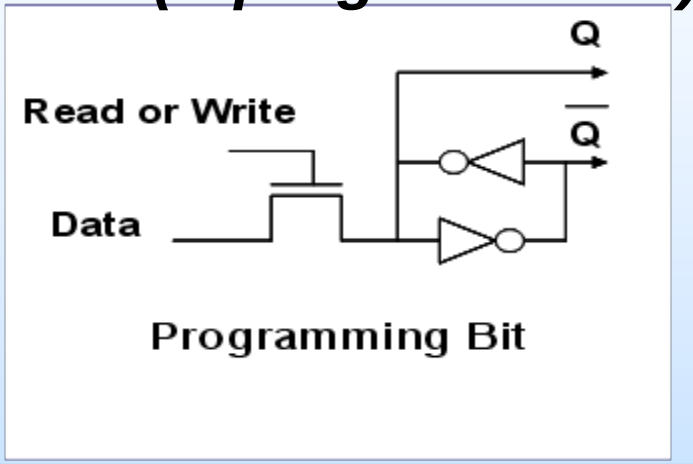
$$P_{configuration} + P(fs)_{functionalLogic} + P_{SEFI}$$

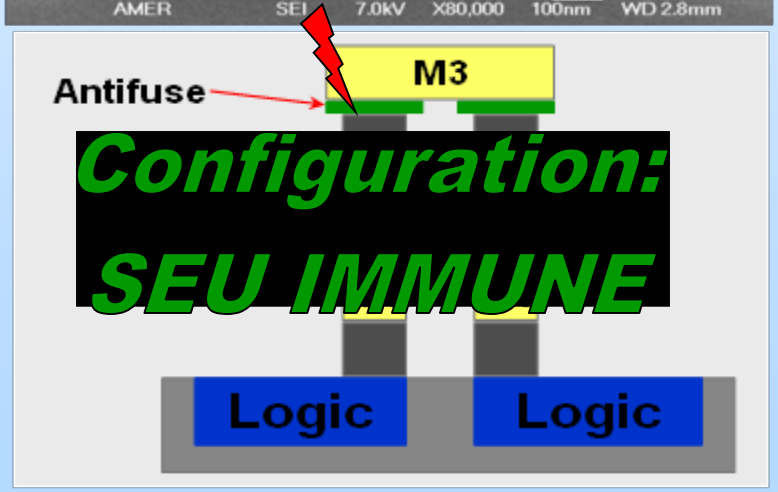
Programmable Switch Implementation and SEU Susceptibility

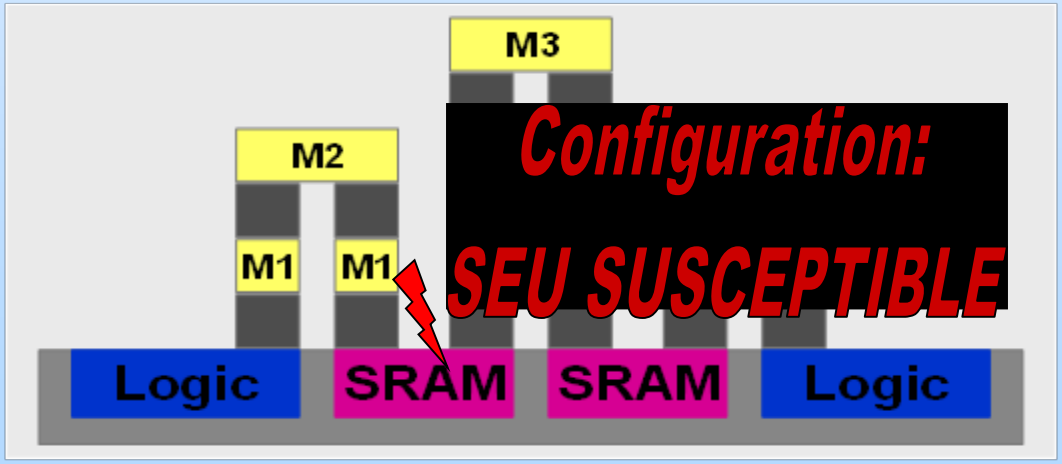
ANTIFUSE (one time programmable)



SRAM (reprogrammable)







To be published on seemapld.org originally presented at 2016 Single Event Effects (SEE) Symposium and the Military and Aerospace Programmable Logic Devices (MAPLD) Workshop.

Configuration SEU Test Results and the REAG FPGA SEU Model

$$P(fs)_{error} \propto P_{Configurat\ ion} + P(fs)_{functional\ Logic} + P_{SEFI}$$

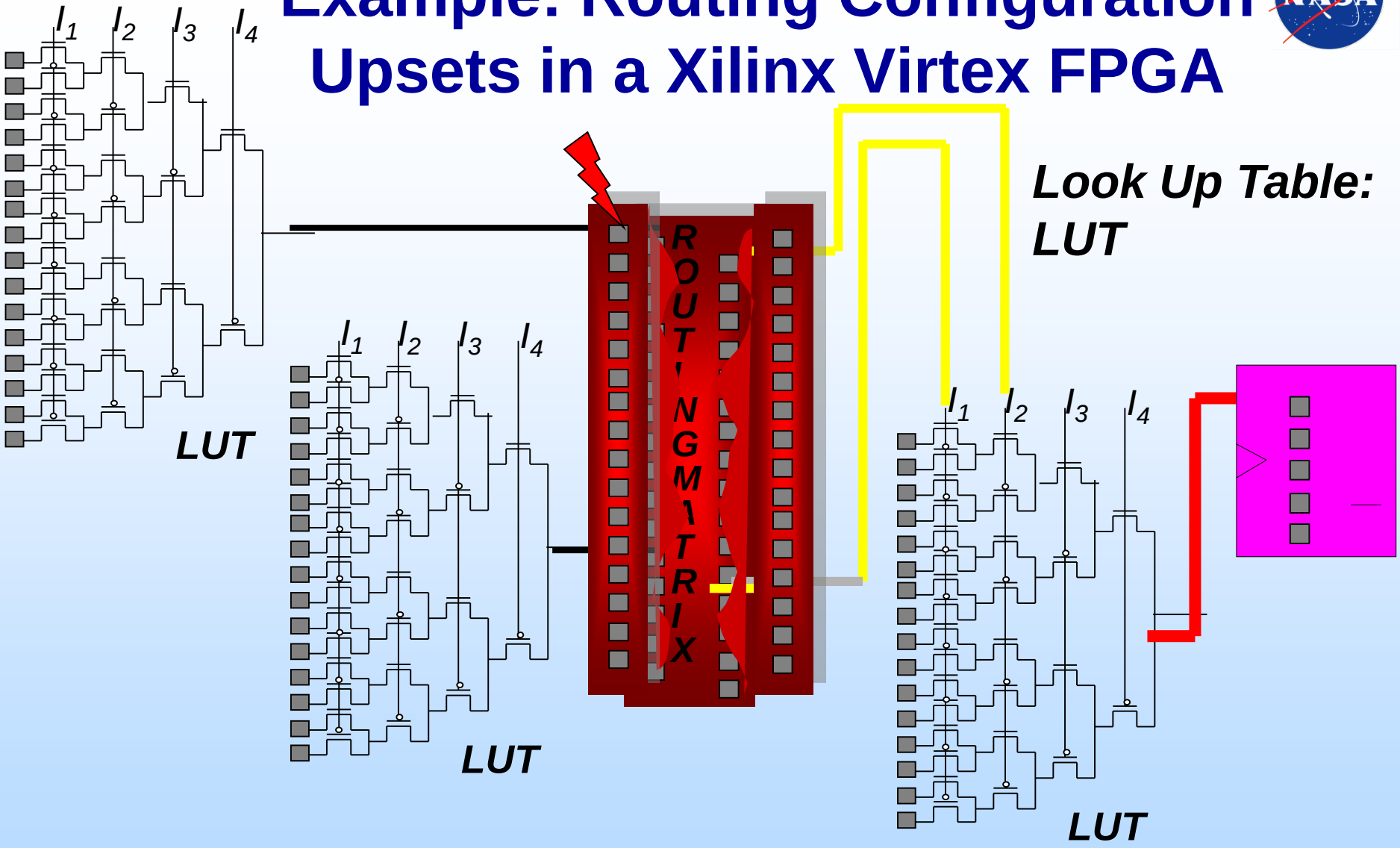
FPGA Configuration Type	REAG Model
Antifuse	$P(fs)_{error} \propto P(fs)_{functionalLogic} + P_{SEFI}$
SRAM **(non-mitigated)**	$P(fs)_{error} \propto P_{Configuration}$
Flash	$P(fs)_{error} \propto P(fs)_{functionalLogic} + P_{SEFI}$
Hardened SRAM	$P(fs)_{error} \propto P_{Configuration} + P(fs)_{functionalLogic} + P_{SEFI}$

What Does The Last Slide Mean?

FPGA Configuration Type	Susceptibility Data-path: Combinatorial Logic (CL) and Flip-flops (DFFs); Global: Clocks and Resets; Configuration
Antifuse	Configuration has been designated as hard regarding SEEs. Susceptibilities only exist in the data paths and global routes. However, global routes are hardened and have a low SEU susceptibility.
SRAM **(non-mitigated)**	Configuration has been designated as the most susceptible portion of circuitry. All other upsets (except for global routes) are too statistically insignificant to take into account. E.g., it is a waste of time to study data path transients, however clock transient studies are significant.
Flash	Configuration has been designated as hard (but NOT immune) regarding SEEs. Susceptibilities also exist in the data paths and global routes (e.g., clocks and resets).
Hardened SRAM	Configuration has been designated as hardened (but NOT hard) regarding SEEs. Susceptibilities also exist in the data paths and global routes (e.g., clocks and resets).

To be published on seemapld.org originally presented at 2016 Single Event Effects (SEE) Symposium and the Military and Aerospace Programmable Logic Devices (MAPLD) Workshop.

Example: Routing Configuration Upsets in a Xilinx Virtex FPGA



Because multiple paths can pass through the routing matrix, this configuration can be catestrophic – i.e., break simple mitigation

To be published on seemapld.org originally presented at 2016 Single Event Effects (SEE) Symposium and the Military and Aerospace Programmable Logic Devices (MAPLD) Workshop.

Fixing SRAM-based Configuration…Scrubbing Definition

- **From SEU testing, it has been shown that the configuration memory of radiation un-hardened SRAM-Based FPGAs is highly susceptible to SEUs.**
- **We address configuration susceptibility via scrubbing: Scrubbing is the act of simultaneously writing into FPGA configuration memory as the device's functional logic area is operating with the intent of correcting configuration memory bit errors.**

Configuration scrubbing only pertains to SRAM-based configuration devices.

Warning!

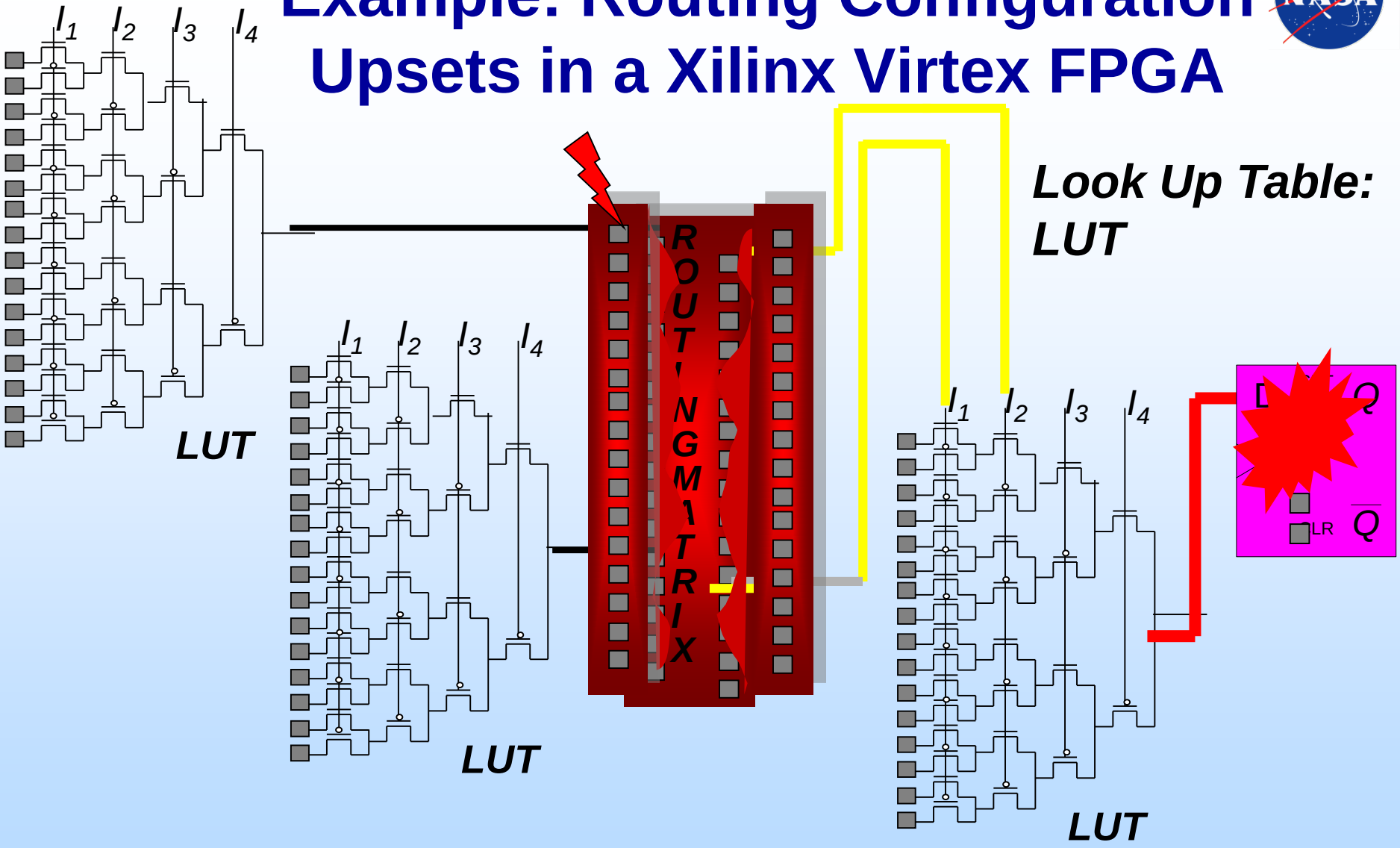
- **Fixing a configuration bit does not mean that you have fixed the state in the functional logic path.**
- **In order to guarantee that the functional logic is in the expected state after the configuration bit is fixed, either the state must be restored or a reset must be issued.**

Reliably getting to an expected state after a configuration-bit SEU (that affects the design's functionality) requires one of the following:

– ***Fix configuration bit + (reset or correct DFFs) or***

– ***Full reconfiguration.***

Example: Routing Configuration Upsets in a Xilinx Virtex FPGA



Configuration + design state must be corrected after a configuration SEU hit.

Single Event Upsets in an FPGA's Functional Data Path and Fail-Safe Strategies

$$P_{configuration} + P(fs)_{functionalLogic} + P_{SEFI}$$

Data-path SEUs and Their Affect At The System Level

- **Each data path in an FPGA device is a cascade of sequential and combinatorial logic.**
- **The occurrence of an SET or SEU does not definitively cause system error.**
- **Probability of a system error due to an SEU depends on many factors:**
  - **Probability of fault generation in a gate (SET or SEU).**
  - **Probability of error propagation – will the SET or SEU force the system's next state to be incorrect?**

Error Propagation in A Data-Path: SEU De-rating

SEUs usually occur between clock edges(during system next-state calculation): A system-level malfunction occurs if the event forces the system's next state to be incorrect.

- **Capacitive filtration: data-path capacitance can stop transient upset propagation; e.g.:**
  - **Routing metal or heavy loading.**
  - **If a transient doesn't reach a sequential element, then it most likely will not cause a system upset.**
- **Logic masking:**
  - **Redundancy and mitigation of paths can stop upset propagation.**
  - **Turned off paths from gated logic can stop upset propagation.**
- **Temporal delay: path delays can block temporary SEUs from disturbing next state calculation.**

Fail-safe Strategies for Single Event Upsets (SEUs)

- The following slides will demonstrate commonly used mitigation strategies for FPGA devices.
- What you should learn:
  - The differences between mitigation strategies.
  - Strengths and weaknesses of various strategies.
  - Questions to ask or considerations to make when evaluating mitigation schemes.
  - Which mitigation schemes are best for various types of FPGA devices.
- The scope of this presentation will cover fail-safe strategies for configuration and data-path SEUs

Fail-Safe Strategies for FPGA Critical Applications

Goal for critical applications: Limit the probability of system error propagation and/or provide detection-recovery mechanisms via fail-safe strategies.

Differentiating Fail-Safe Strategies:

- **Detection:**
  - **Watchdog (state or logic monitoring).**
  - **Can range from simplistic checking to complex Decoding.**
  - **Action (alerting, correction, or recovery).**
- **Masking (does not mean correction):**
  - **Preventing error propagation to other logic.**
  - **Requires redundancy + mitigation or detection.**
  - **Turn off faulty path.**
- **Correction (error may not be masked):**
  - **Error state (memory) is changed/fixed.**
  - **Need feedback or new data flush cycle.**
- **Recovery:**
  - **Bring system to a deterministic state.**
  - **Might include correction.**

Redundancy Is Not Enough

- **Simply adding redundancy to a system is not enough to assume that the system is well protected.**
- **Questions/Concerns that must be addressed for a critical system expecting redundancy to cure all (or most):**
  - **How is the redundancy implemented?**
  - **What portions of your system are protected? Does the protection comply with the results from radiation testing?**
  - **Is detection of malfunction required to switch to a redundant system or to recover?**
  - **If detection is necessary, how quickly can the detection be performed and responded to?**
  - **Is detection enough?... Does the system require correction?**

Listed are crucial concerns that should be addressed at design reviews and prior to design implementation

To be published on seemapld.org originally presented at 2016 Single Event Effects (SEE) Symposium and the Military and Aerospace Programmable Logic Devices (MAPLD) Workshop.

Mitigation

- **Error Masking vs. Error Correction… there's a difference.**
- **Mitigation can be:**
  - **User inserted: part of the actual design process.**
    - **User must verify mitigation… Complexity is a RISK!!!!!!!!**
  - **Embedded: built into the device library cells.**
    - **User does not verify the mitigation – manufacturer does.**
- **Mitigation should reduce error…**
  - **Generally through redundancy.**
  - **Incorrect implementation can increase error.**
  - **Overly complex mitigation cannot be verified and incurs too high of a risk to implement.**

To be published on seemapld.org originally presented at 2016 Single Event Effects (SEE) Symposium and the Military and Aerospace Programmable Logic Devices (MAPLD) Workshop.

Questions to Ask: Availability versus Correct Operation

- Requirements must be satisfied. Is there a metric for mitigation trade-offs and risks regarding system requirements?
- What is your expected up-time versus down-time (availability)?
- Is correct operation well defined? E.g., correct operation can be defined as working as expected through error states after an SEU strike… however, this must be clearly stated in requirements.
- Is system failure well defined?
- Can availability and correct operation be deterministic regardless of error signature?

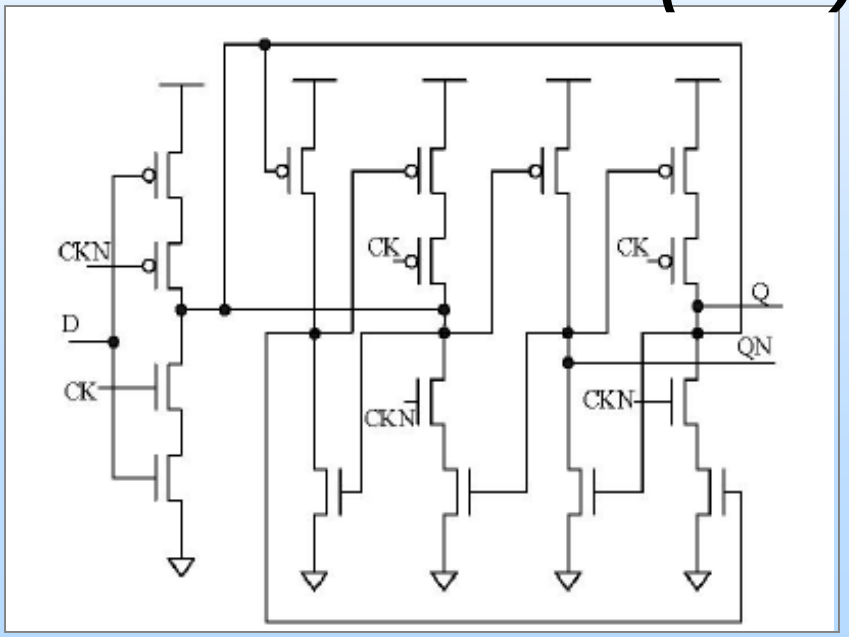
Detection and Recovery

- **Not all mitigation schemes require detection.**
- **Questions/Considerations – important review questions:**
  - **If your scheme requires detection:**
    - **Can the system detect all types of error signatures?**
    - **Can the system detect all error signatures fast enough?**
    - **Do different errors require different recovery schemes… can the system accommodate.**
  - **How are you going to verify the detection and recovery?**
  - **How much downtime will there be during recovery?**

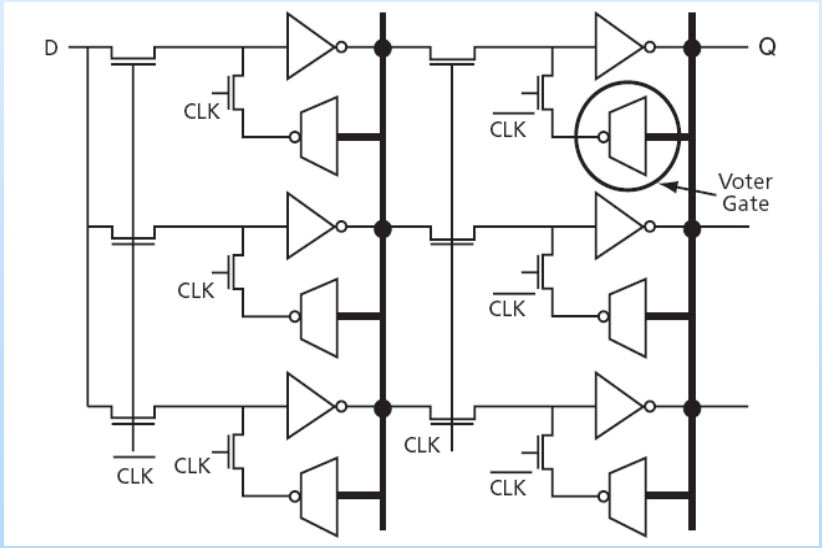
"We know it will work" are not good enough answers: Ask how and if the scheme has been verified!

Embedded Mitigation versus User Inserted Mitigation

Dual Interlocked Cell (DICE)



Localized Triple Modular Redundancy (LTMR)



To be published on seemapld.org originally presented at 2016 Single Event Effects (SEE) Symposium and the Military and Aerospace Programmable Logic Devices (MAPLD) Workshop.

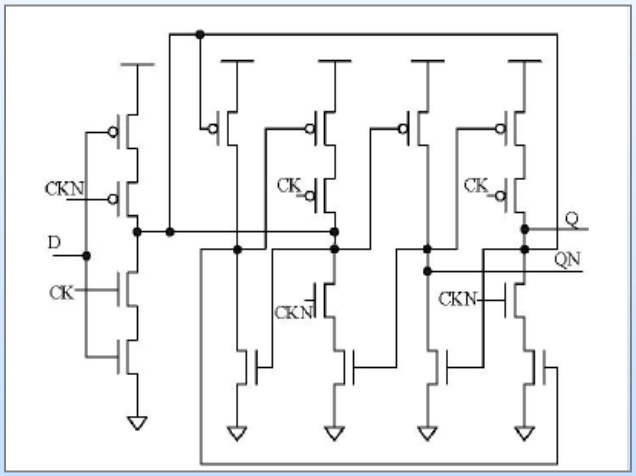
Radiation Hardened (per SEU) versus Commercial FPGA Devices

- **For this presentation, a radiation hardened (per SEU) device is a device that has embedded mitigation.**
- **Radiation hardened FPGA devices are available to users. They make the design cycle much easier!**
- **SEU mitigation is generally applied to the following:**
  - **Data-path elements:**
    - **Localized redundancy inserted into library cell flip-flops (DFFs).**
      - **Localized Triple Modular Redundancy (LTMR) or**
      - **Dual interlocked Cell (DICE)**
    - **SET filters inserted on the DFF data input pin.**
    - **SET filters inserted on the DFF clock input pin.**
  - **Global routes.**
  - **Memory cells.**

Localized Redundancy Embedded in the DFF Cells

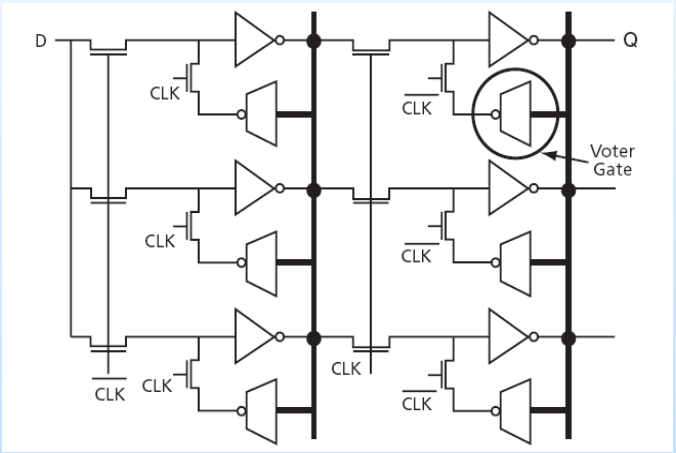
Warning! These figures are simplified schematics of the actual implementation.

Dual Interlocked Cell (DICE)



Xilinx

Localized Triple Modular Redundancy (LTMR)



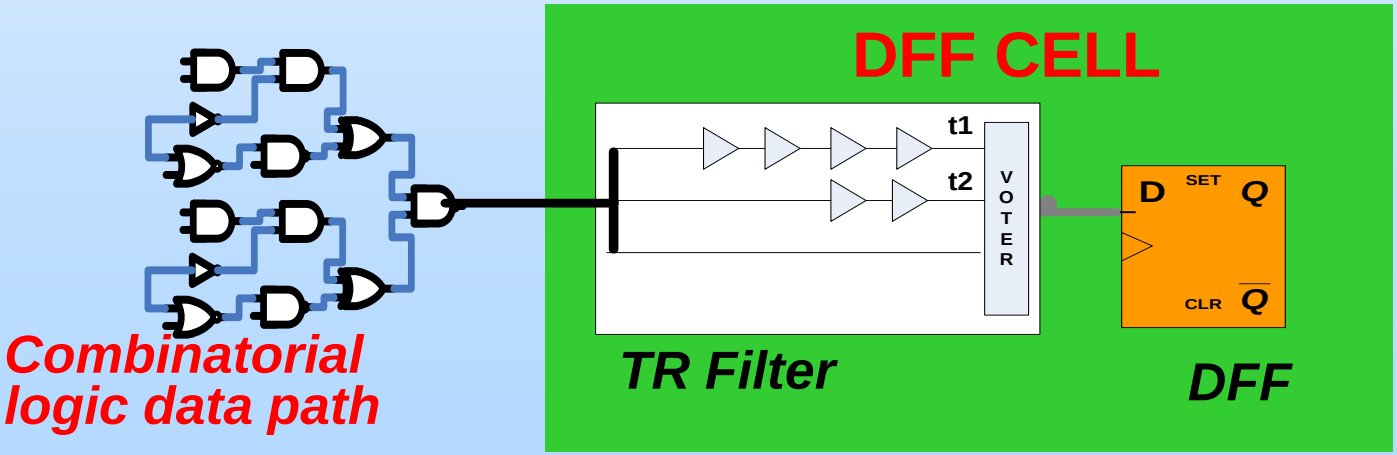
Microsemi

Problem! Although DFFs are protected, SETs from the combinatorial logic in the data path and SETs in the global routes can cause incorrect data to be captured by the DFF.

To be published on seemapld.org originally presented at 2016 Single Event Effects (SEE) Symposium and the Military and Aerospace Programmable Logic Devices (MAPLD) Workshop.

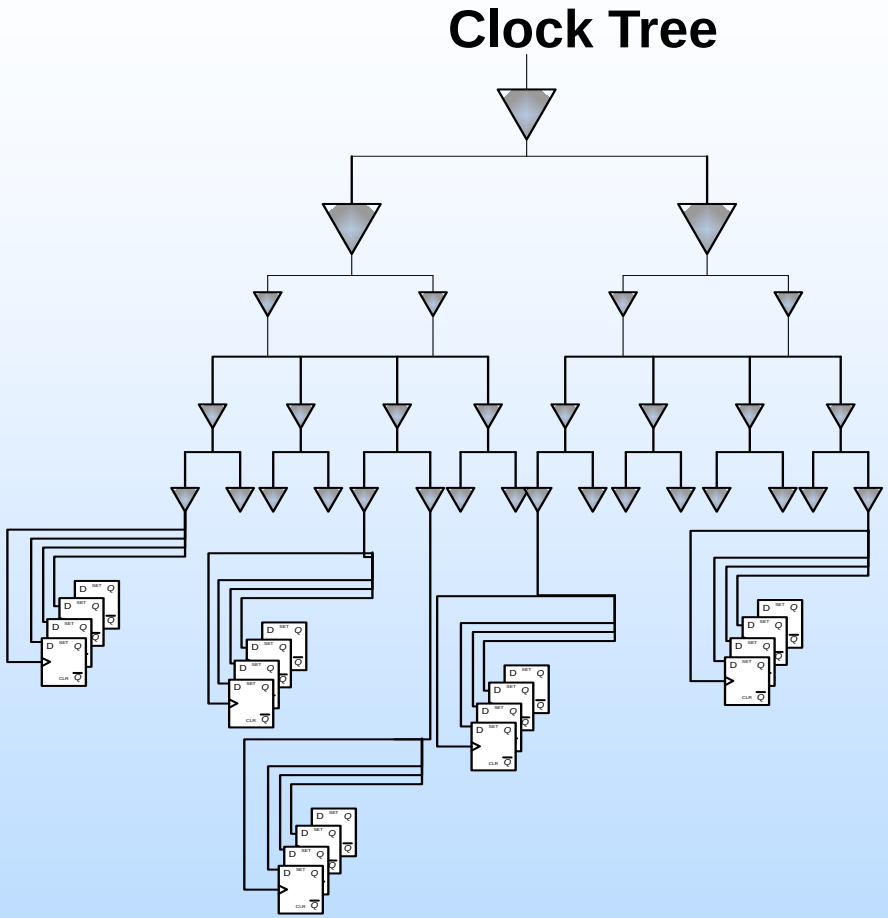
Embedded Temporal Redundancy (TR): SET Filtration in The Data Path

- Temporal Filter placed directly before DFF.
- Localized scheme that reduces SET capture in the data path.
- Delays must be well controlled.
  - Every delay path shall consistently have a predefined delay and must be verified.
- Do not implement TR as a user inserted mitigation scheme. Delay must be deterministic and it is too difficult to manage with place and route tools.
- Maximum Clock frequency is reduced by the amount of new delay.



Embedded Radiation Hardened Global Routes: SET Filtration in The Global Route Path

- Some FPGAs contain radiation-hardened clock trees and other global routes (Microsemi products only).
- Global structures are generally hardened by using larger buffers.
- TR has also been used on the lowest leaves of the clock trees… (Xilinx V5QV only).

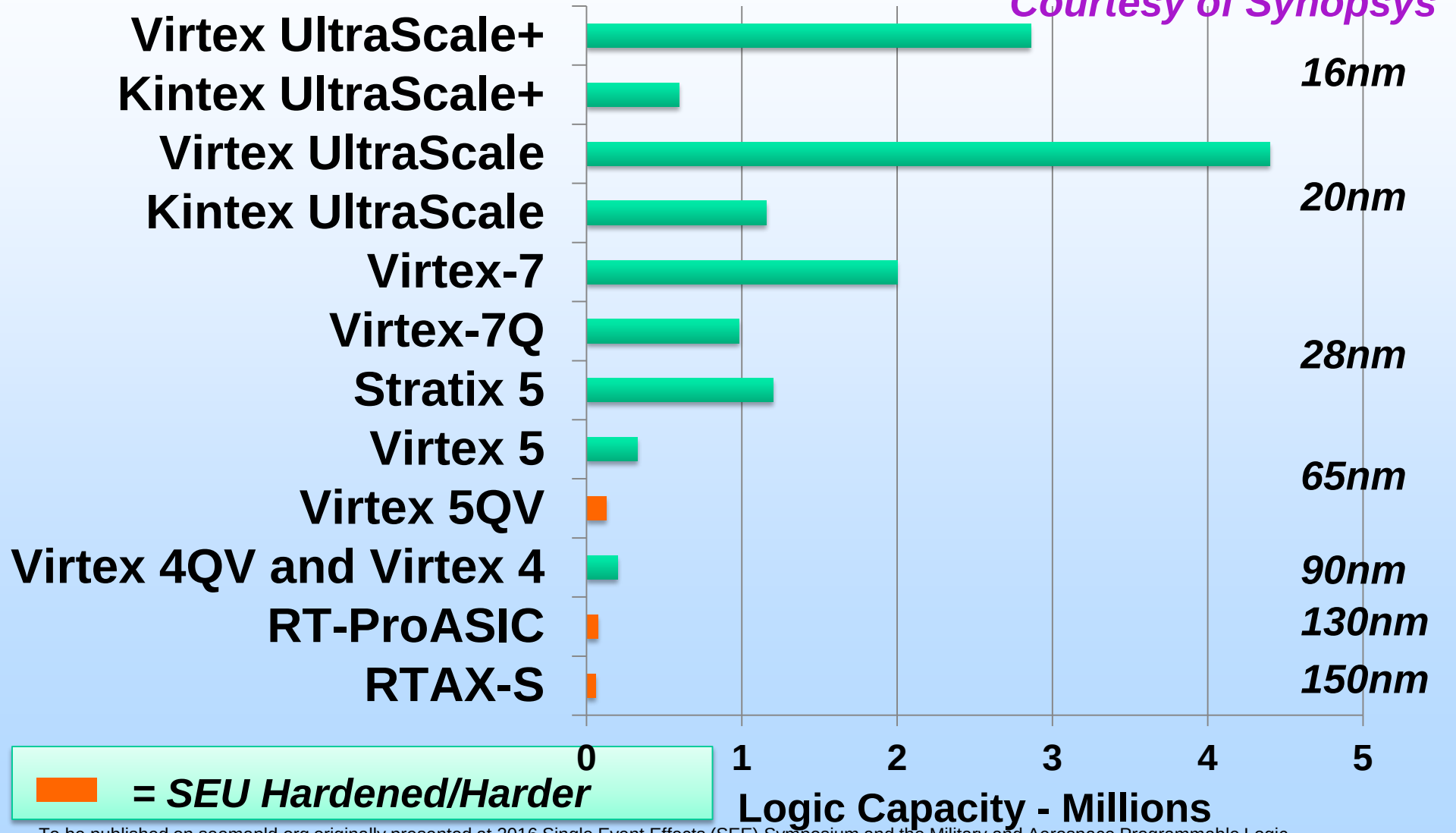


Global route susceptibility is often overlooked. Beware, many devices do not have hardened global routes.

Radiation Hardened versus Commercial FPGA Device Geometries And Gate Count

As Geometries Get Smaller, More Gates Are Available for Mitigation

Courtesy of Synopsys



To be published on seemapld.org originally presented at 2016 Single Event Effects (SEE) Symposium and the Military and Aerospace Programmable Logic Devices (MAPLD) Workshop.

FPGA Devices Listed by Configuration Type (Not All Are Included in The List): Susceptibility

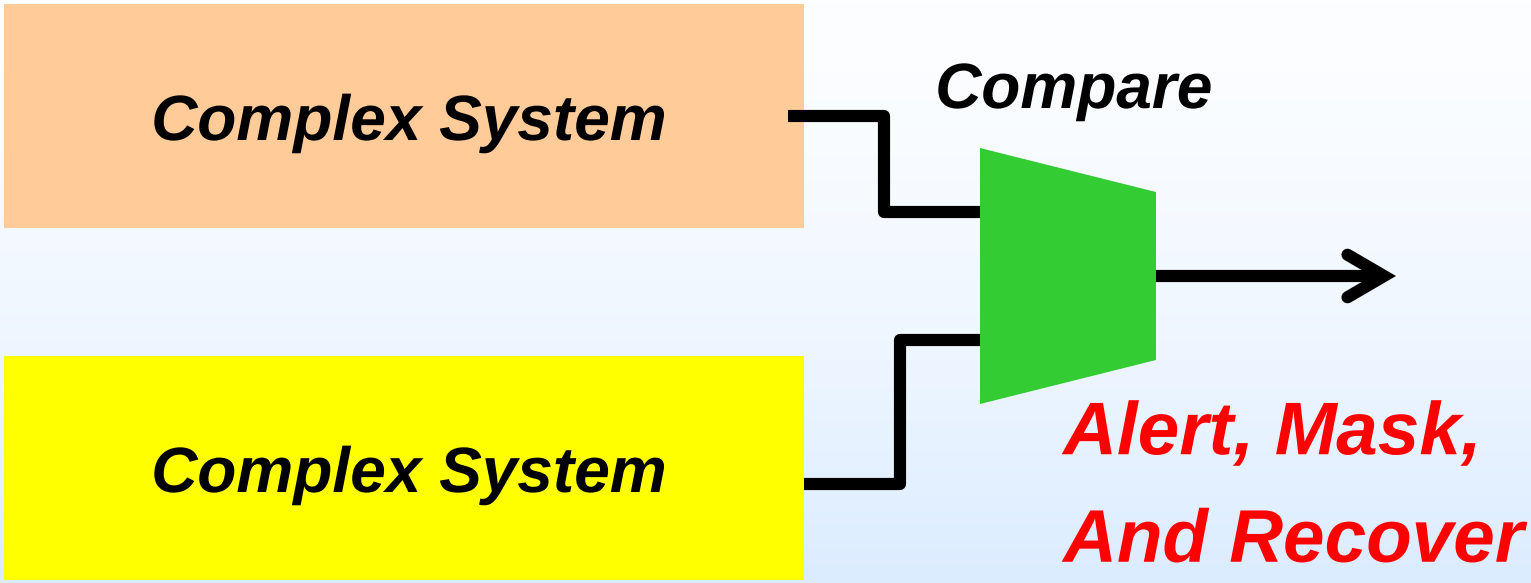
DFF: flip flop ***DICE: Dual interlocked Cell***

Configuration Type	Short List of Device Families	Embedded Mitigation	Most Susceptible Components
SRAM	Stratix, Virtex, Kintex	No	Configuration
Antifuse	RTAX, RTSXS	DFFs and clocks (configuration is already hardened by nature)	Combinatorial logic (however susceptibility considered low)
Flash	ProASIC3	Configuration is already hardened by nature.	DFFs and clocks
Hardened SRAM	Virtex V5QV	Configuration + DICE DFFs + SET filters	Clocks. In some cases additional mitigation may be necessary for configuration and DFFs

Go to http://radhome.gsfc.nasa.gov, manufacturer websites, and other space agency sites for more information on SEU data and total ionizing dose data.

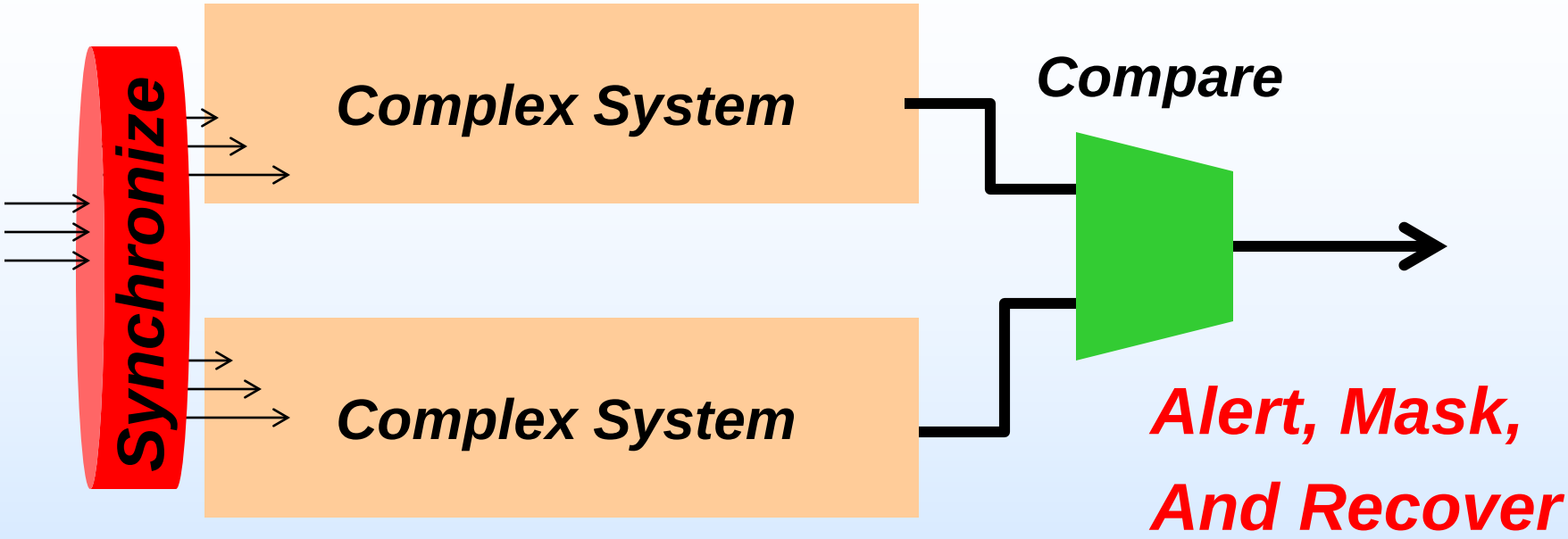
User Inserted Mitigation:
Dual Redundancy, Cold Sparing, and Triple Modular Redundancy (TMR)

Dual Redundancy



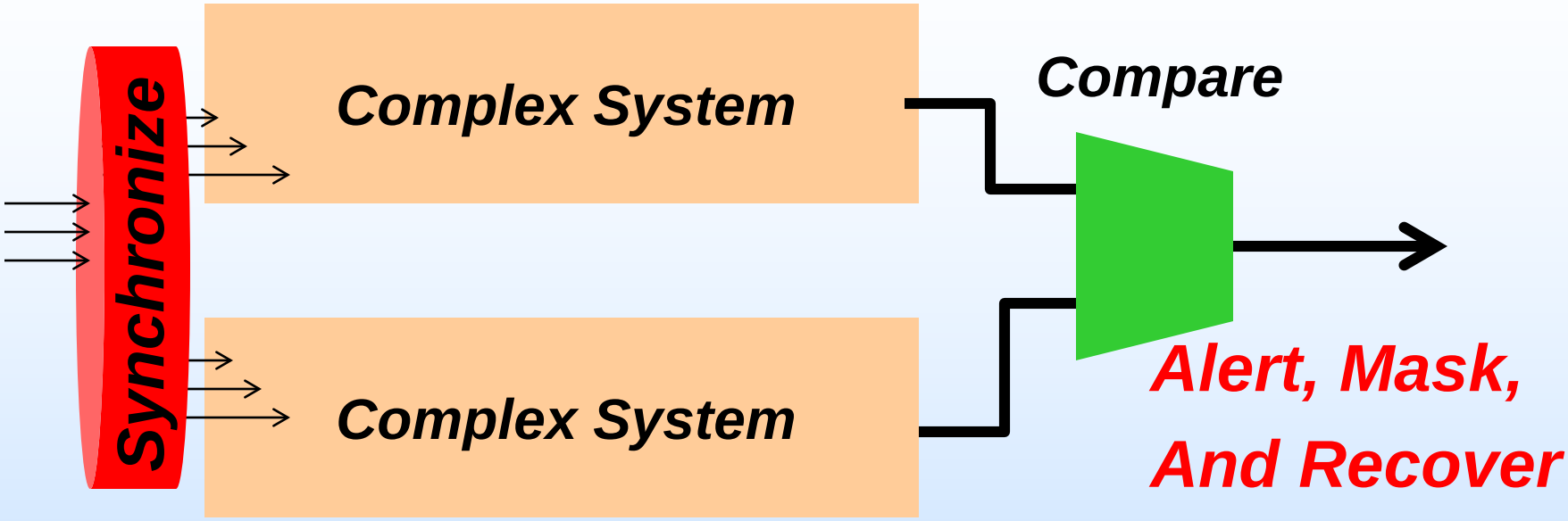
- Dual redundant systems cannot correct (roll-back is an exception); they can only detect.
- "Compare and Alert" systems must be highly reliable and verifiable.
- Generally not all I/O can be monitored or compared.
- Best used for data calculation and manipulation… easiest to place compares on data buses.

Lockstep Dual Redundancy



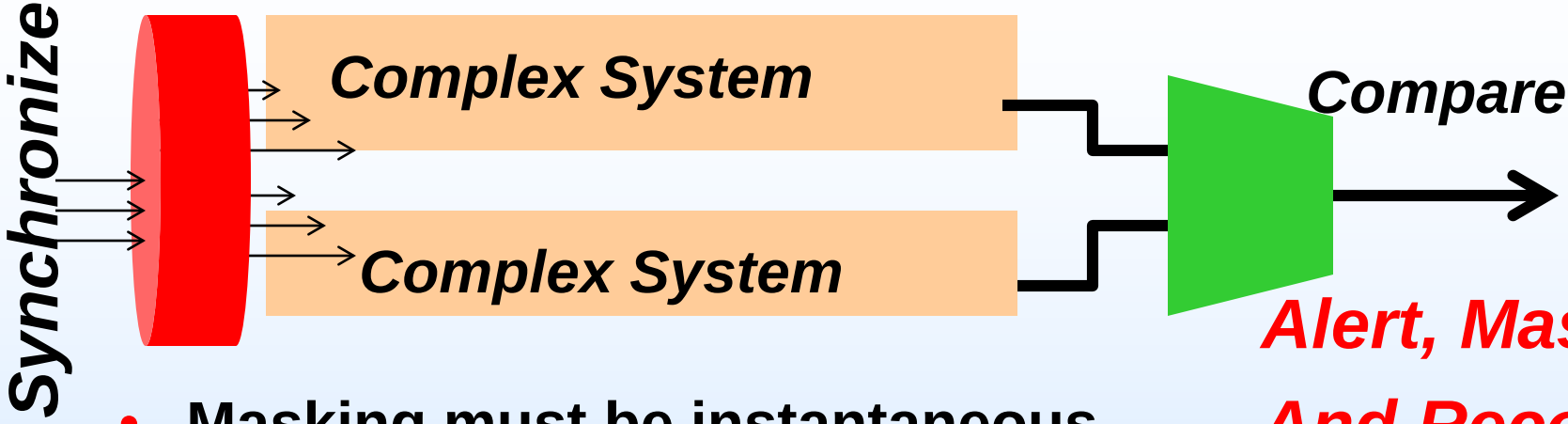
- For a lockstep, the dual complex systems are exact duplicates.
- Synchronization is necessary. It is challenging and sometimes unpredictable. If not well managed, availability is affected.
  - Cache misses.
  - Care must be taken to synchronize asynchronous inputs (e.g., interrupts).
  - Complex algorithms for pipelining or memory management must be controlled for data to be exact duplicates and in lockstep.

Lockstep Compare



- Best to use a " data valid signal" to indicate data are ready for compare.
- System performance can be affected because of data synchronization at the comparator.
- Halt control will be necessary in case of "out-of-sync" data.
- Should the comparator be hardened?

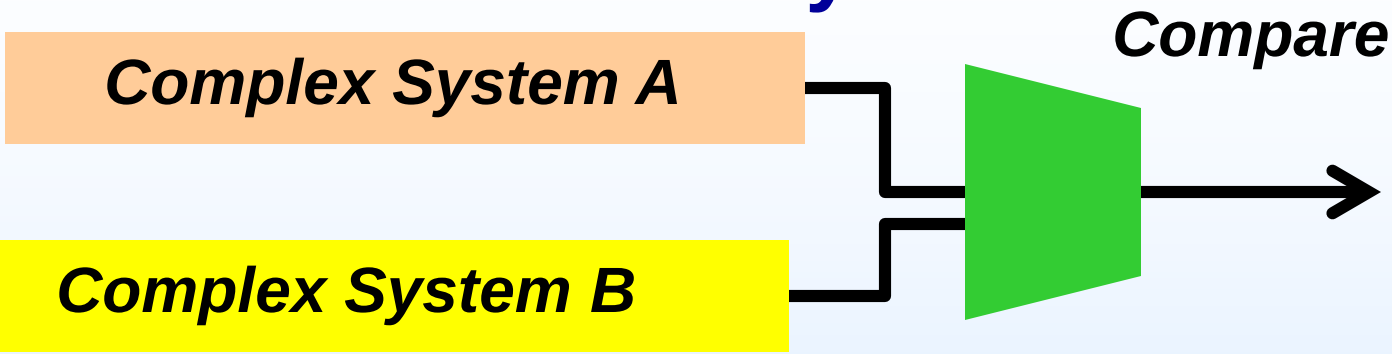
Lockstep Alert, Mask, and Recover



- **Masking must be instantaneous.**
- **Where will the alert go upon a miscompare?**
  - **Internal or external device (watchdog).**
- **What will be the system response to an alert?**
  - **Full reset versus partial resets.**
  - **FPGA (full reconfigure).**
  - **Power cycle.**
  - **Roll-back (correction) is an option. However, complex and unreliable.**

Usually system reset or power cycle is required upon SEU

Two Separate Systems Dual Redundancy



- Relaxes the system synchronization requirements of being in lockstep.
- Compares operate on valid signals and/or message passing. Compares are generally more complex.
- Two separate systems can be complex to manage:
  - Data reordering and cache misses must be managed.
  - System halts must be carefully managed.
- If not well managed, system performance and availability will be significantly affected; and SEUs will not be taken care of as expected.

Cold Sparing: Elongation of System Operation:

- One active system and alternate inactive systems.
- Upon active system failure, an inactive system is turned on.
- System operation is able to be elongated after failure.
- However:
  - Availability is not improved… there is downtime.
  - Can your system afford the downtime (critical application)?
  - How clean is the system switch over?
  - How long is the system switch over.
- Can the system ping-pong between active and inactive systems or is a system considered dead after failure?
  - Ping-ponging can be used for systems that have a low probability of destructive failures.
  - Ping-ponging can be complex and can affect availability.

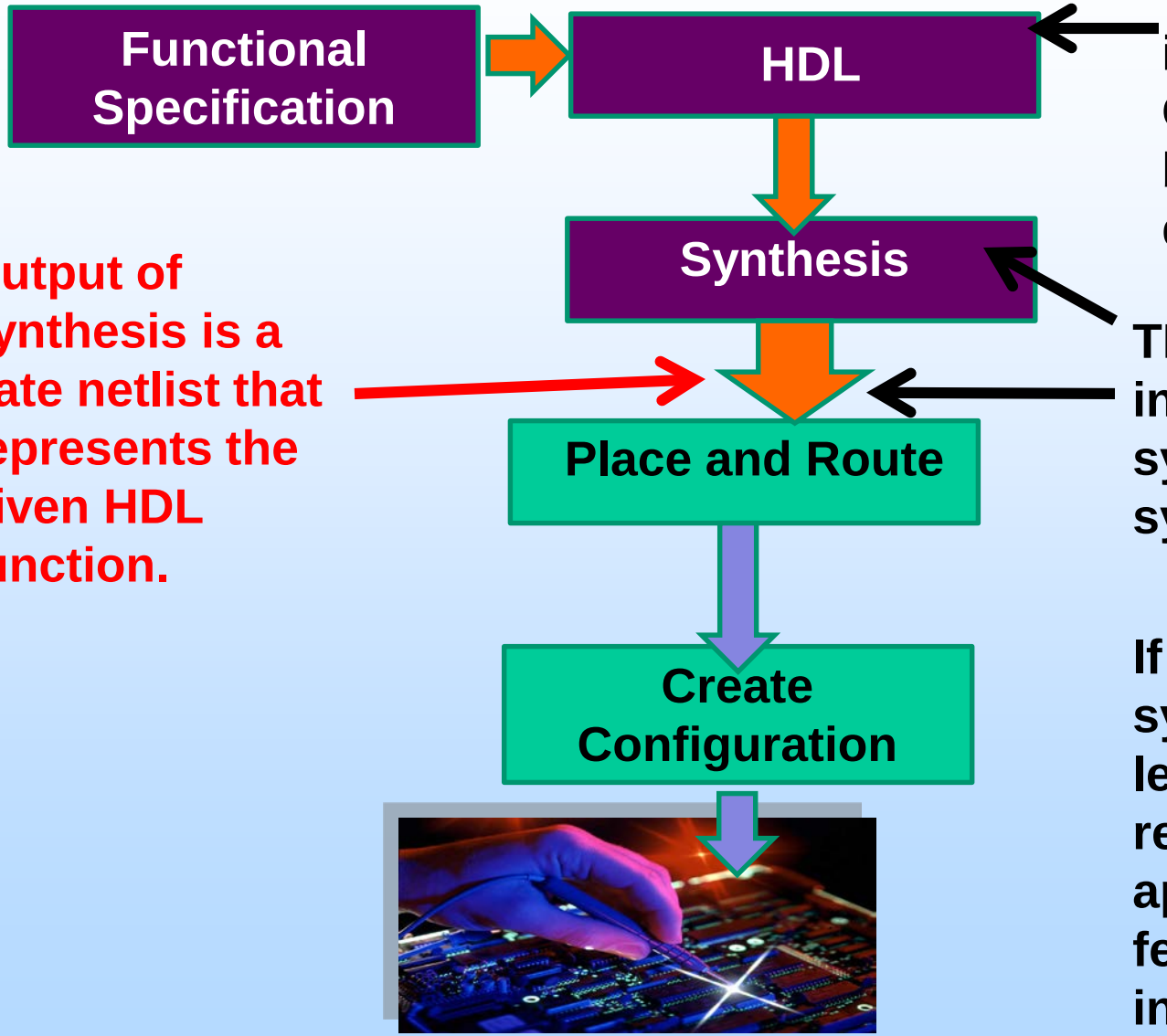
System versus Design Mitigation

- The previous slides were affiliated with system level mitigation.
- System level mitigation generally has:
  - Detection, masking, no correction, downtime, and recovery actions.
- The following slides will discuss triple modular redundancy (TMR) techniques that can be implemented as system or design-level mitigation.
- Most of the TMR techniques will incorporate masking and detection with no downtime (unless there is a single functional interrupt (SEFI)).
- Hence, TMR can improve system performance, availability, and elongate operation time.

Mitigation – Fail Safe Strategies That Do Not Require Fault Detection but Provide SEU Masking and/or Correction:
Triple Modular Redundancy (TMR)…
best two out of three.

How To Insert TMR into A Design:

FPGA User Design Flow



TMR can be written into the HDL. Generally not done because too difficult.

Output of synthesis is a gate netlist that represents the given HDL function.

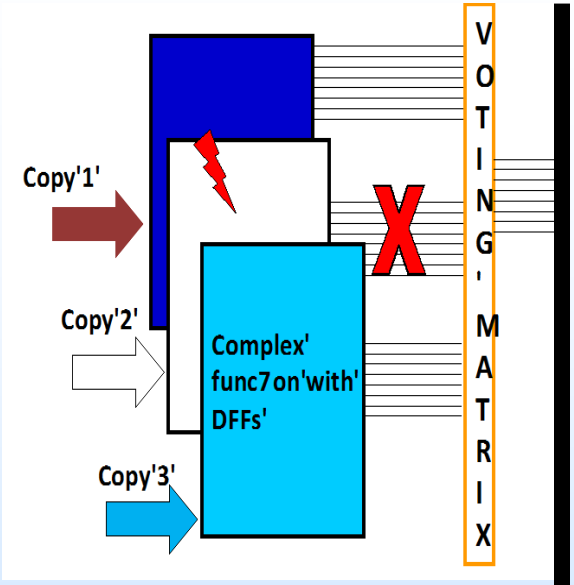
TMR can be inserted during synthesis or post synthesis.

If inserted post synthesis, the gate level netlist is replicated, ripped apart, and voters + feedback are inserted.

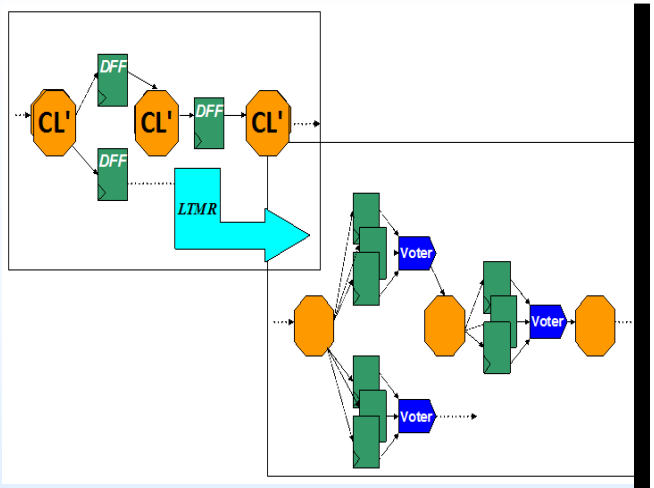
Local Mitigation versus Distributed or Global Mitigation

- **Local mitigation:**
  - **Only DFFs are mitigated.**
  - **Mitigation will include masking and potential correction at the DFF.**
  - **Used with systems where DFFs are the most susceptible component cells.**
- **Distributed or global mitigation:**
  - **The full design is mitigated with masking and correction.**
- **Depending on the target device, the clock tree and other global routes may also need hardening.**

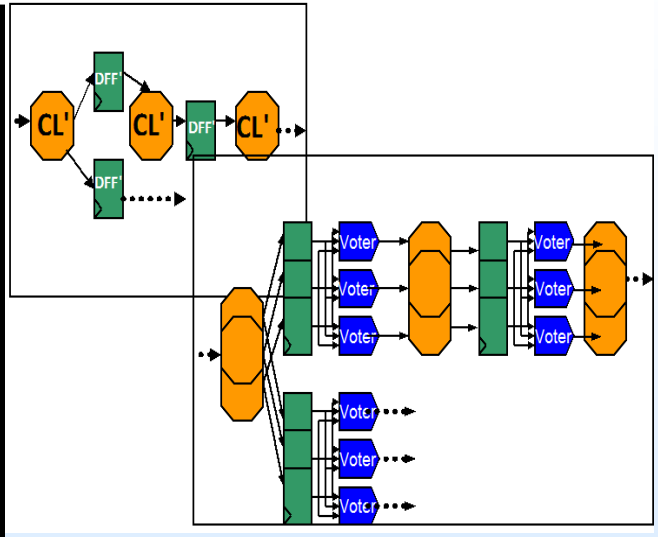
Various TMR Schemes: Different Topologies



Block diagram of block TMR (BTMR): a complex function containing combinatorial logic (CL) and flip-flops (DFFs) is triplicated as three black boxes; majority voters are placed at the outputs of the triplet.



Block diagram of local TMR (LTMR): only flip-flops (DFFs) are triplicated and data-paths stay singular; voters are brought into the design and placed in front of the DFFs.



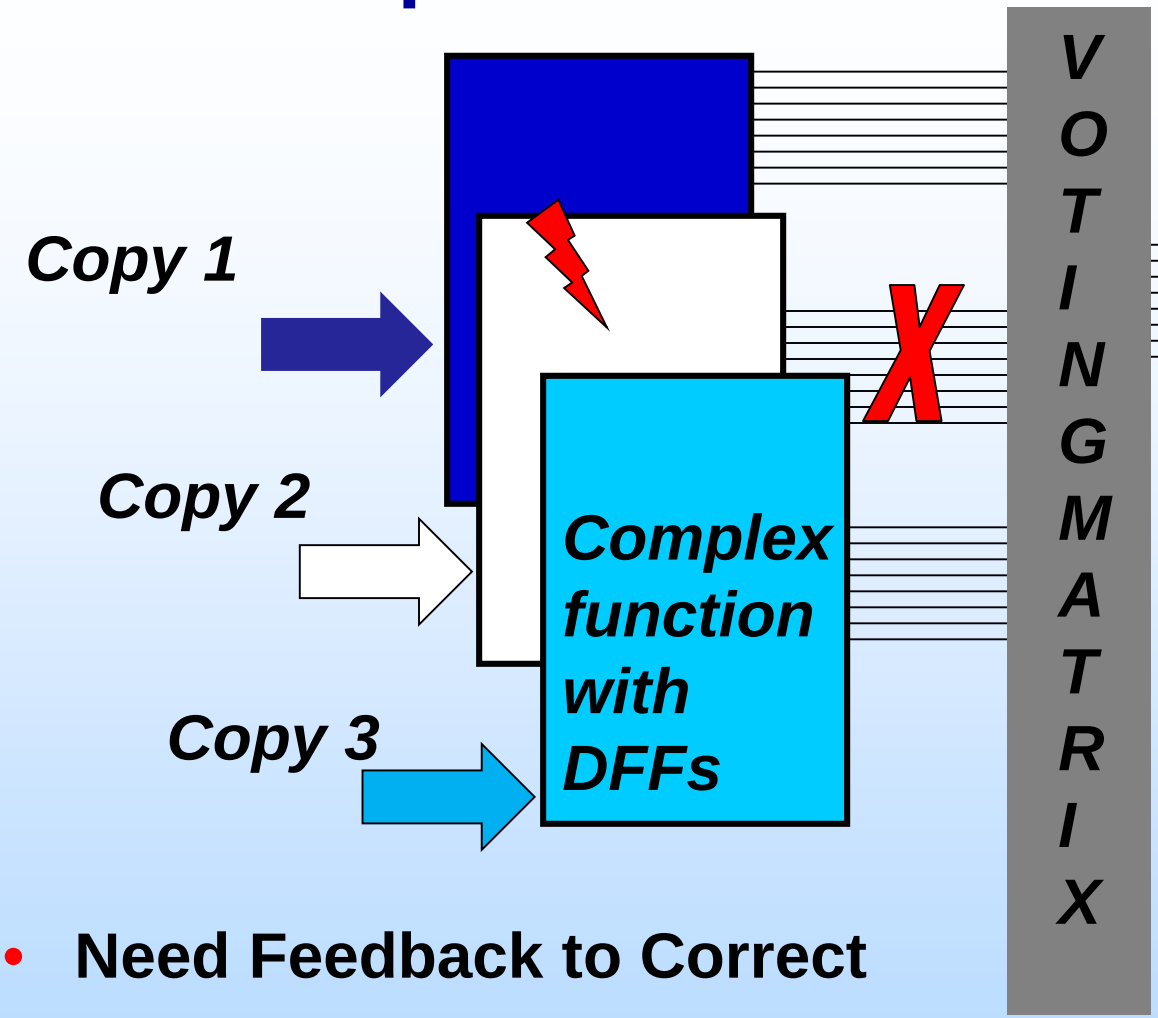
Block Diagram of distributed TMR (DTMR): the entire design is triplicated except for the global routes (e.g., clocks); voters are brought into the design and placed after the flip-flops (DFFs). DTMR masks and corrects most single event upsets (SEUs).

To be published on seemapld.org originally presented at 2016 Single Event Effects (SEE) Symposium and the Military and Aerospace Programmable Logic Devices (MAPLD) Workshop.

TMR Implementation

- As previously illustrated, TMR can be implemented in a variety of ways.
- The definition of TMR depends on what portion of the circuit is triplicated and where the voters are placed.
- The strongest TMR implementation will triplicate all data-paths and contain separate voters for each data-path.
  - However, this can be costly: area, power, and complexity.
  - Hence a trade is performed to determine the TMR scheme that requires the least amount of effort and circuitry that will meet project requirements.
- Presentation scope: Block TMR (BTMR), Localized TMR (LTMR), Distributed TMR (DTMR), Global TMR (GTMR).

Block Triple Modular Redundancy: BTMR



Can Only Mask Errors

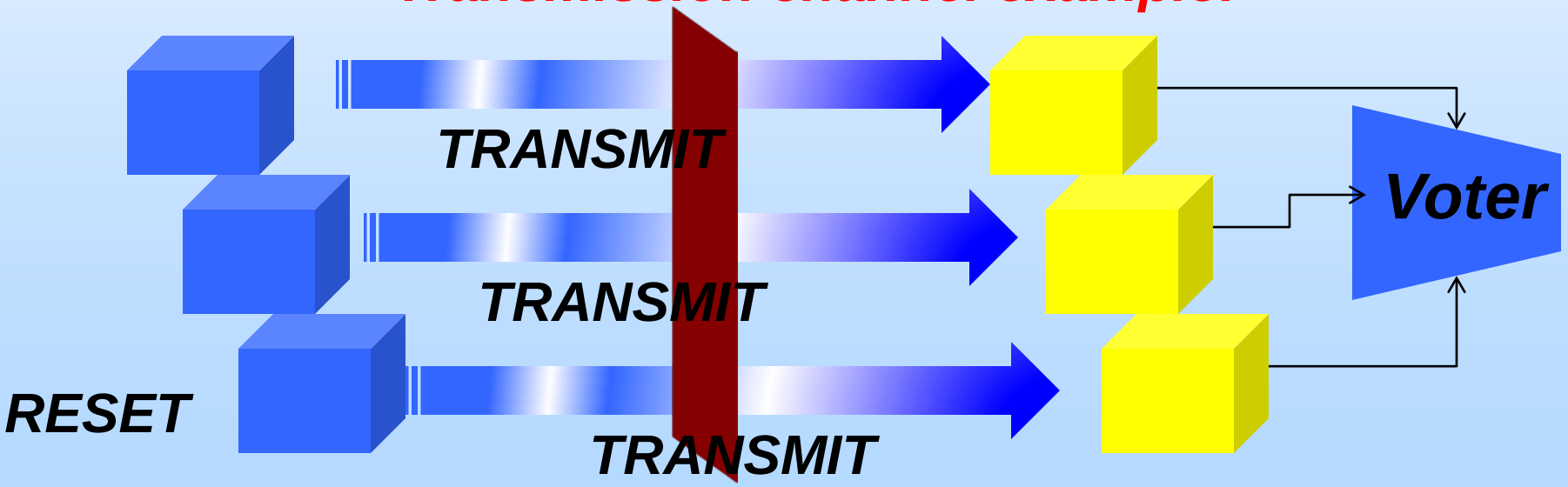
3x the error rate with triplication and no correction/flushing

- **Need Feedback to Correct**
- **Cannot apply internal correction from voted outputs**
- **If blocks are not regularly flushed (e.g. reset), Errors can accumulate – may not be an effective technique**

Examples of a Flushable BTMR Designs

- **Shift Registers.**
- **Transmission channels: It is typical for transmission channels to send and reset after every sent packet.**
- **Systems that can be reset (or power-cycled) every so-often.**

Transmission channel example:

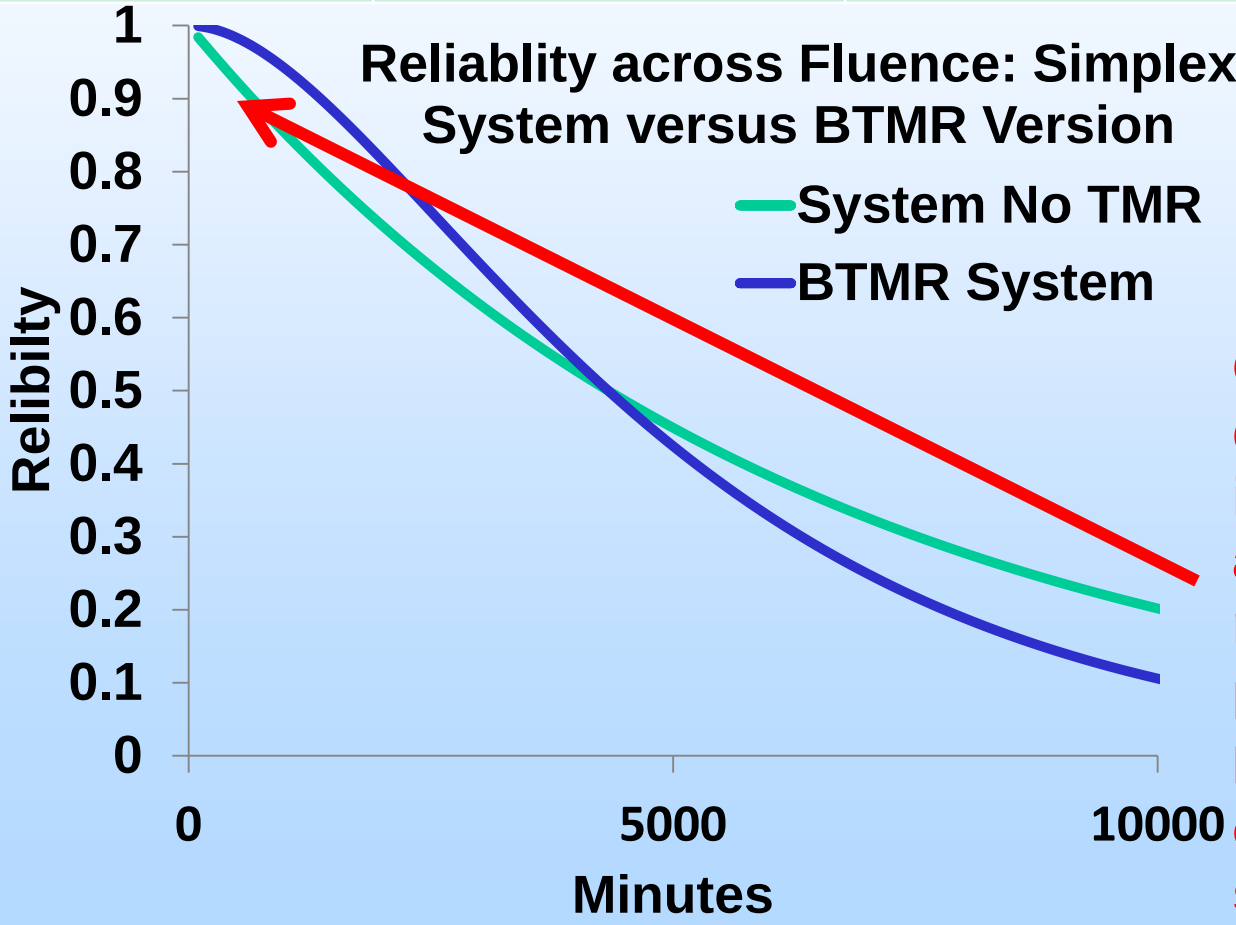


If The System Is Not Flushable, Then BTMR May Not Provide The Expected Level of Mitigation

- **BTMR can work well as a mitigation scheme if the expected MTTF for each module is much greater than the expected time-window of correct operation.**
- **But… If the expected time to failure for one block is close to the expected time-window of correct operation, then BTMR doesn't buy you anything.**
- **If not thought out well, BTMR can actually be a detriment – complexity, power, and area, and false sense of performance.**

Explanation of BTMR Strength and Weakness using Classical Reliability Models

Relibility for 1 block (R_{block})	Relibility for BTMR (R_{BTMR})	Mean Time to Failure for 1 block ($MTTF_{block}$)	Mean Time to Failure BTMR ($MTTF_{BTMR}$)
$e^{-\lambda t}$	$3\,e^{-2\lambda t}-2\,e^{-3\lambda t}$	$1/\lambda$	$(5/6\,\lambda)= 0.833/\lambda$



$$\lambda = \frac{Failures}{Time}$$

Operating a BTMR design in this time interval will provide an increase in reliability.

However, over time, BTMR reliability drops off faster than a system with No TMR.

BTMR Bottom Line

- **How long does your BTMR system need to operate relative to the MTTF for one of its unmitigated blocks?**
- **Overtime, a BTMR system has lower reliability than an unmitigated system.**
- **Adding more replicated blocks (e.g., N-out-of-M) system will only increase the reliability during the short window near start time. However, overtime, the reliability of an N-out-of-M system will fall faster as M (the number of replicated blocks) grows.**

What Should be Done If Availability Needs to be Increased?

- **If the blocks within the BTMR have a relatively high upset rate with respect to the availability window, then stronger mitigation must be implemented.**
- **Bring the voting/correcting inside of the modules… bring the voting to the module DFFs.**

The following slides illustrate the various forms of TMR that include voter insertion in the data-path.

TMR Nomenclature	Description *DFF: Edge triggered flip-flop; CL: Combinatorial Logic*	TMR Acronym
Local TMR	DFFs are triplicated	LTMR
Distributed TMR	DFFs and CL-data-paths are triplicated	DTMR
Global TMR	DFFs, CL-data-paths and global routes are triplicated	GTMR or XTMR

Describing Mitigation Effectiveness Using A Model

DFF: Edge triggered flip-flop

CL: Combinatorial Logic

$$P(fs)_{error} \propto P_{configuration} + P(fs)_{functionalLogic} + P_{SEFI}$$

$$P(fs)_{DFFSEU \rightarrow SEU} + P(fs)_{SET \rightarrow SEU}$$

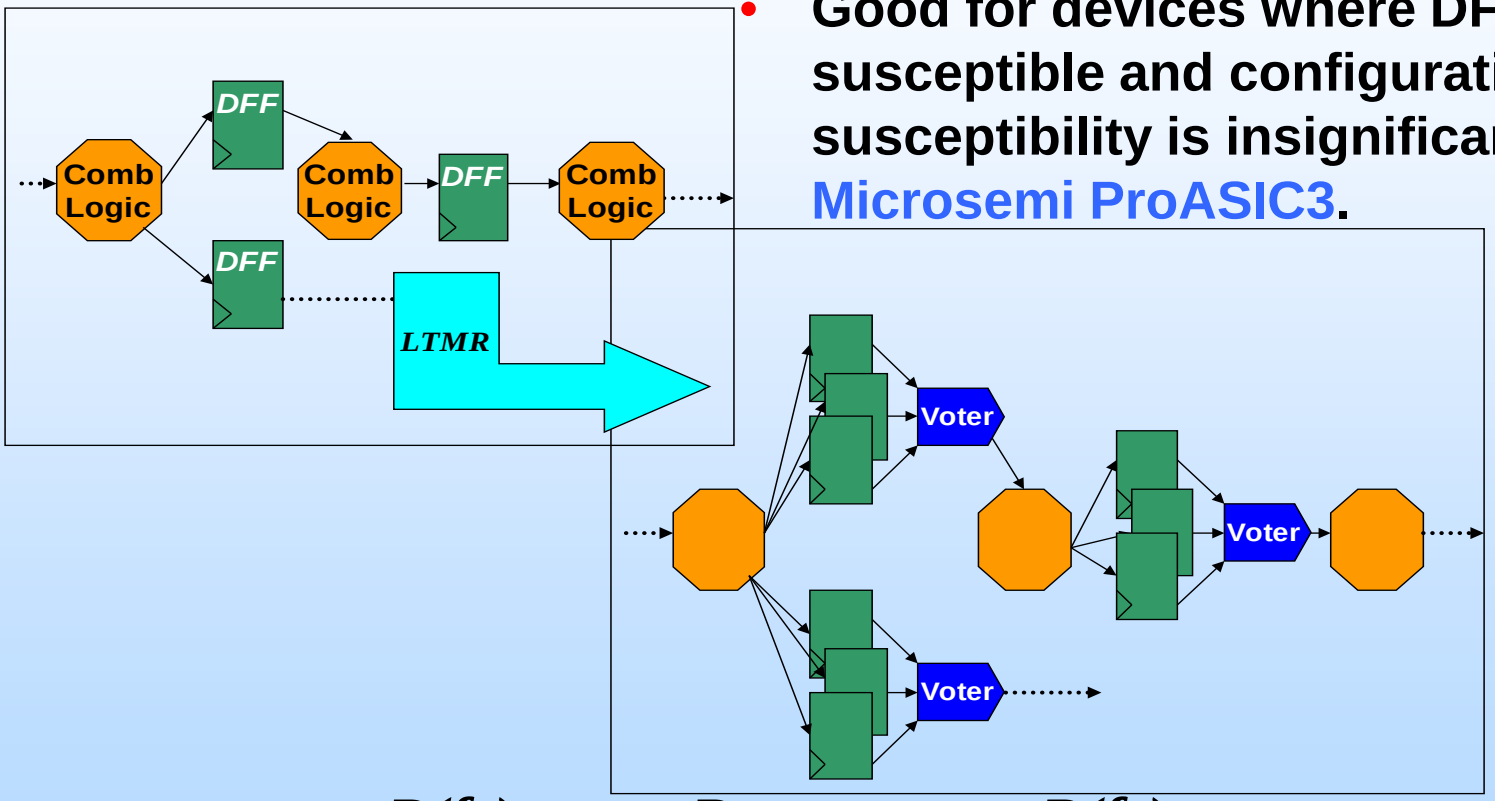
Probability that an SEU in a DFF will manifest as an error in the next system clock cycle

Probability that an SET in a CL gate will manifest as an error in the next system clock cycle

To be published on seemapld.org originally presented at 2016 Single Event Effects (SEE) Symposium and the Military and Aerospace Programmable Logic Devices (MAPLD) Workshop.

Local Triple Modular Redundancy (LTMR)

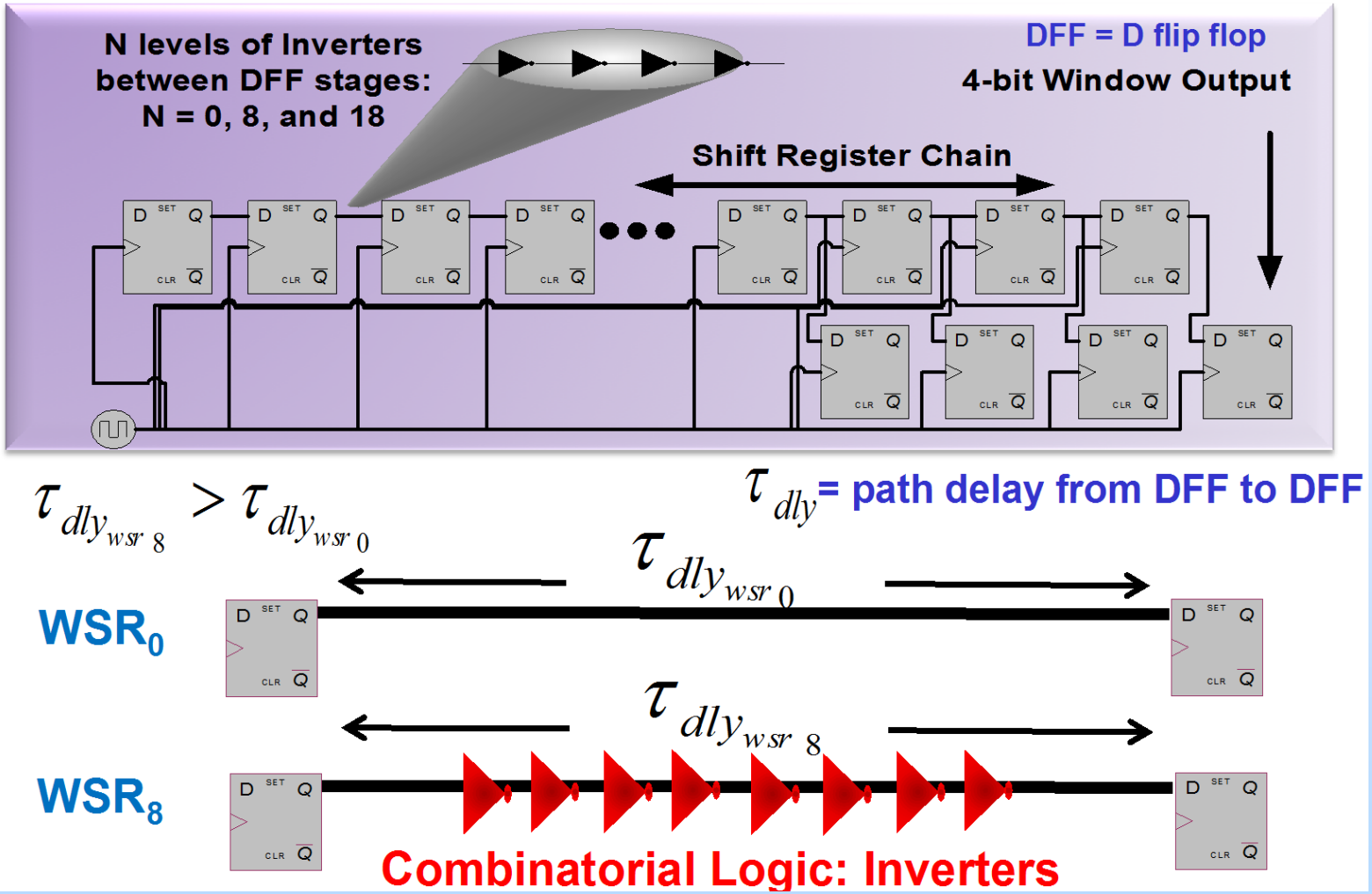
- Only DFFs are triplicated. Data-paths are kept singular.
- LTMR masks upsets from DFFs and corrects DFF upsets if feedback is used.
- Good for devices where DFFs are most susceptible and configuration and CL susceptibility is insignificant; e.g., Microsemi ProASIC3.



$$P(fs)_{error} \propto P_{configuration} + P(fs)_{functionalLogic} + P_{SEFI}$$

$$P(fs)_{DFFSEU \rightarrow SEU} \rightarrow 0 + P(fs)_{SET \rightarrow SEU}$$

Windowed Shift Registers (WSRs):
NEPP Test Structure



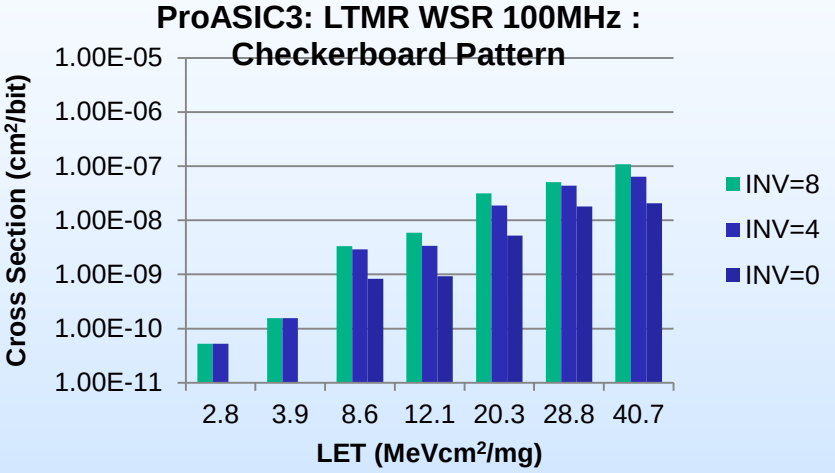
Adding LTMR to a Microsemi ProASIC3 Device versus RTAXs Embedded LTMR

- At lower LETs, applying LTMR to a ProASIC3 design, has similar (a little higher) SEU response to Microsemi RTAXs series.
- At higher LETs, clock tree upsets start to dominate and LTMR in the ProASIC3 is not as effective.
- Depending on your target radiation environment, for most critical applications, the ProASIC3 SEU responses will produce acceptable upset rates.

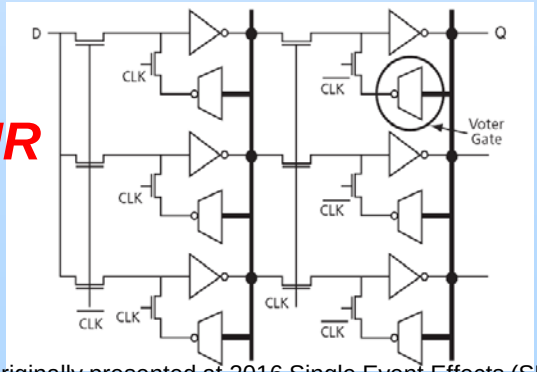
LET: linear energy transfer.

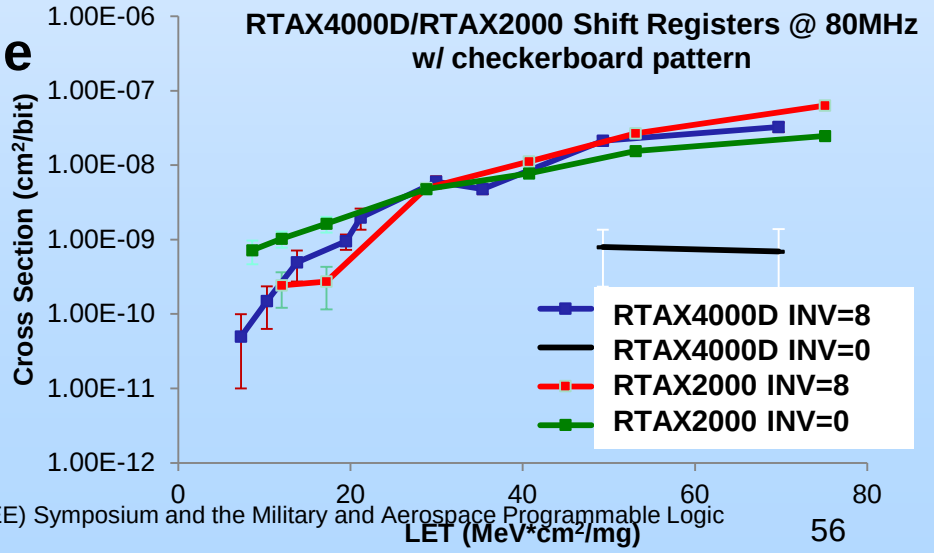
WSR: Test circuit…Windowed Shift Register.

INV: Inverters between WSR stages.



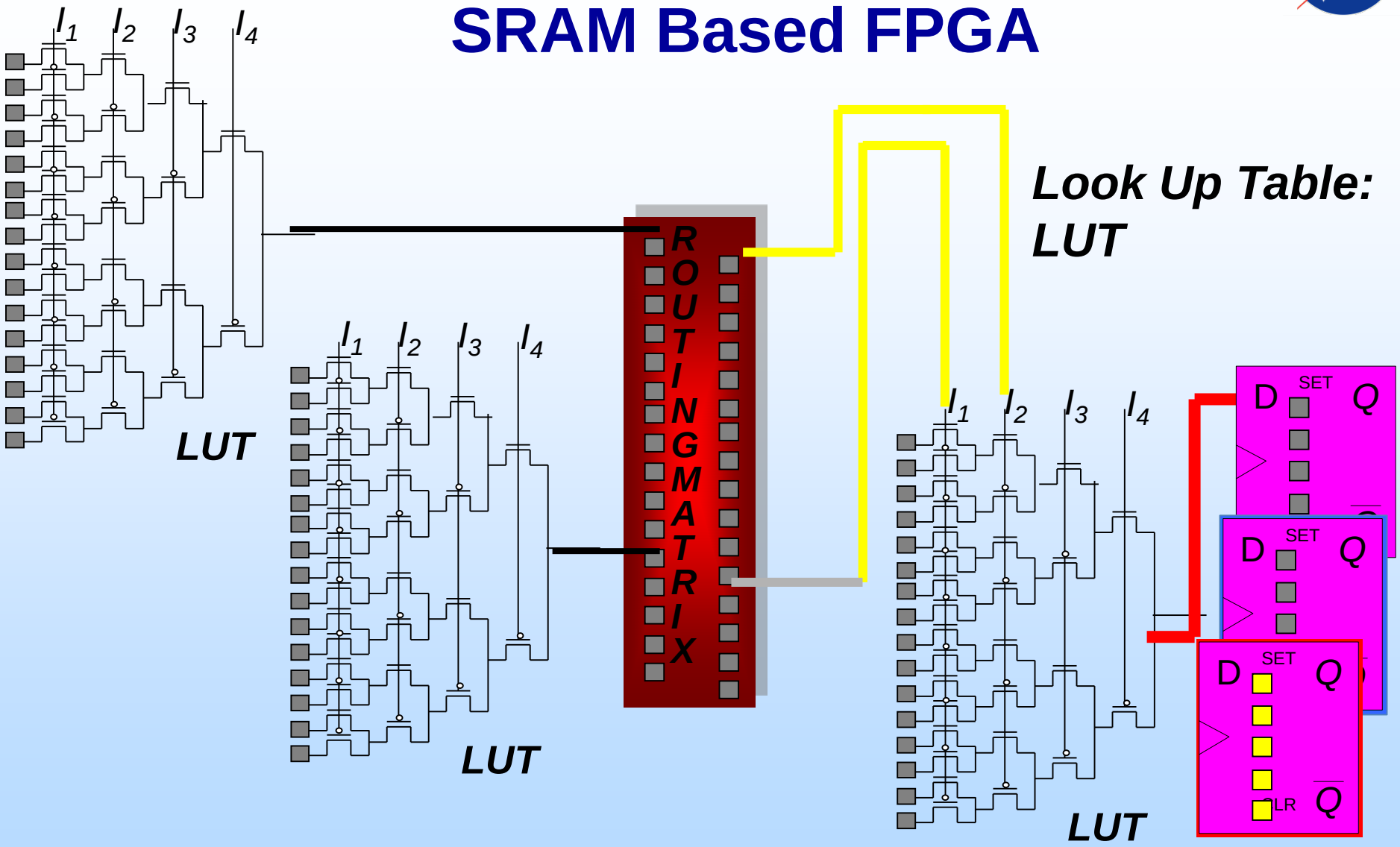
Embedded LTMR in a RTAXs DFF cell.





To be published on seemapld.org originally presented at 2016 Single Event Effects (SEE) Symposium and the Military and Aerospace Programmable Logic Devices (MAPLD) Workshop.

LTMR Should Not Be Used in An SRAM Based FPGA

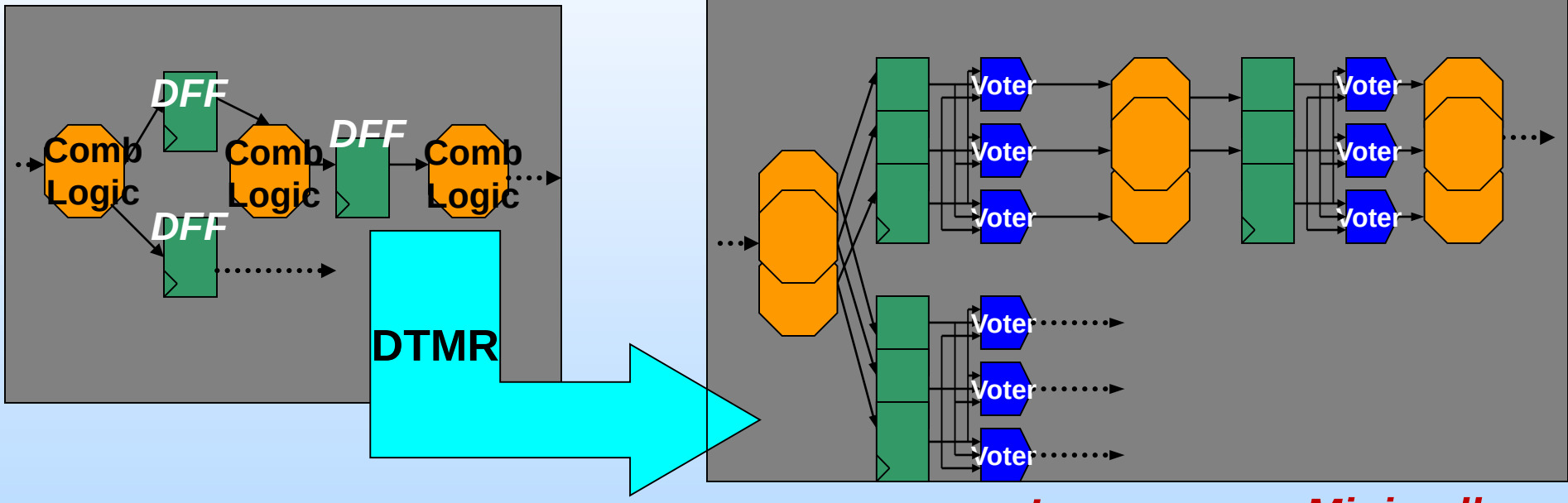


Proven via NEPP experiments: SEU data for LTMR implemented in Xilinx FPGA devices are similar or worse than no added mitigation.

To be published on seemapld.org originally presented at 2016 Single Event Effects (SEE) Symposium and the Military and Aerospace Programmable Logic Devices (MAPLD) Workshop.

Distributed Triple Modular Redundancy (DTMR)

- Triple all data-paths and add voters after DFFs.
- DTMR masks upsets from configuration + DFFs + CL and corrects captured upsets if feedback is used.
- Good for devices where configuration or DFFs + CL are more susceptible than project requirements; e.g., Xilinx and Altera commercial FPGAs.



$$P(f_s)_{error} \propto P_{configuration} + P(f_s)_{functionalLogic} + P_{SUFI}$$

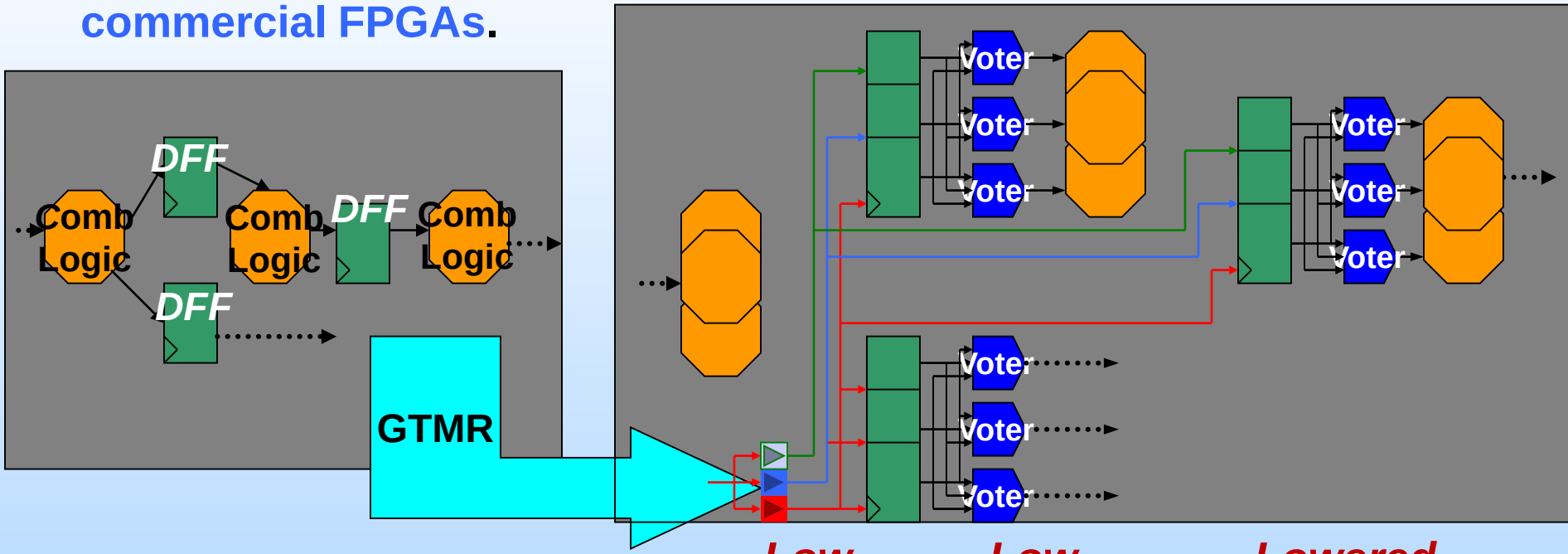
Low → Minimally Lowered

$$P(f_s)_{DFFSEU \rightarrow SEU} + P(f_s)_{SET \rightarrow SEU}$$

0 → Low

Global Triple Modular Redundancy (GTMR)

- Triple all clocks, data-paths and add voters after DFFs.
- GTMR has the same level of protection as DTMR; however, it also protects clock domains.
- Good for devices where configuration or DFFs + CL are more susceptible than project requirements; e.g., Xilinx and Altera commercial FPGAs.



$$P(f_s)_{error} \propto P_{configuration} + P(f_s)_{functionalLogic} + P_{SEFI}$$

Low Low Lowered

$$P(fs)_{DFFSEU \rightarrow SEU} + P(f_s)_{SET \rightarrow SEU}$$

Low Low

Theoretically, GTMR Is The Strongest Mitigation Strategy… BUT…

- Triplicating a design and its global routes takes up a lot of power and area.
- Generally performed after synthesis by a tool– not part of RTL.
- Skew between clock domains must be minimized such that it is less than the shortest routing delay from DFF to DFF (hold time violation or race condition):
  - Does the FPGA contain enough low skew clock trees? (each clock + its synchronized reset)x3.
  - Limit skew of clocks coming into the FPGA.
  - Limit skew of clocks from their input pin to their clock tree.
- Difficult to verify.

TMR and Verification

- **If a system is required to be protected using triple modular redundancy (TMR), improper insertion can jeopardize the reliability and security of the system.**
- **Due to the complexity of the verification process and the complexity of digital designs, there are currently no available techniques that can provide complete and reliable confirmation of TMR insertion.**
- **Can you trust that TMR has been inserted as expected (correct topological scheme) and has not broken existing logic during the insertion process?**

We are working on it!

Currently, What Are The Biggest Challenges Regarding Mitigation Insertion?

- Tool availability… Synopsys is not quite ready for DTMR or GTMR.
- User's are not selecting the correct mitigation scheme for their target FPGA.
- Mitigation is too complex to fully verify.

FPGA Type	LTMR	DTMR	GTMR
Antifuse+LTMR: Microsemi RTAX or RTSX family		?????	
Commercial SRAM: Xilinx and Altera devices			
Commercial Flash: Microsemi ProASIC family		?????	
Hardened SRAM: Xilinx V5QV		?????	

General Recommendation
Not Recommended but may be a solution for some situations
Will not be a good solution

To be published on seemapld.org originally presented at 2016 Single Event Effects (SEE) Symposium and the Military and Aerospace Programmable Logic Devices (MAPLD) Workshop.

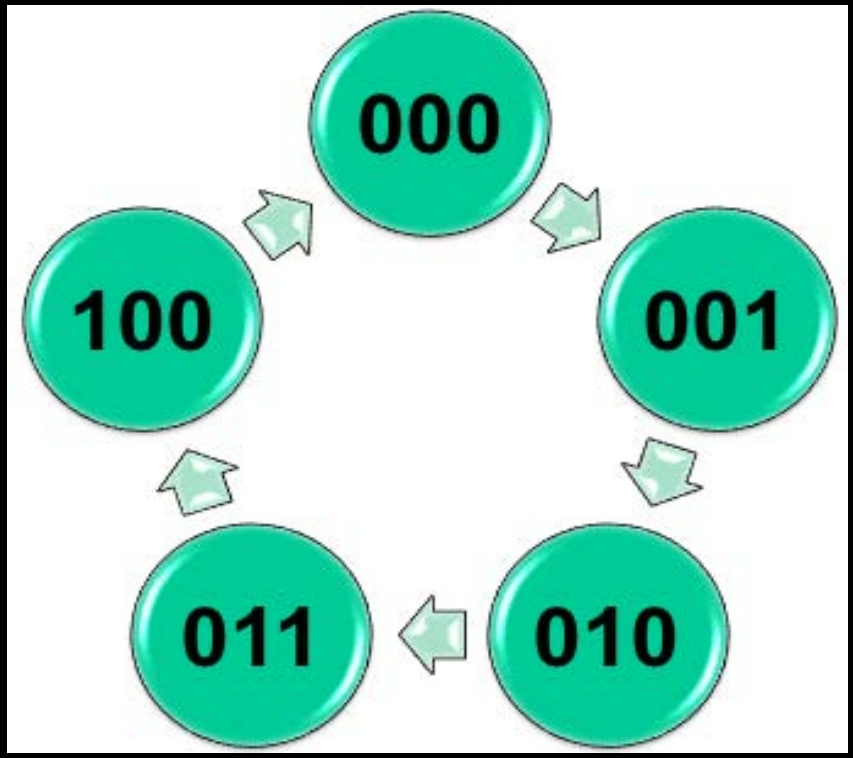
User versus Embedded Mitigation

- **A subset of user inserted mitigation strategies have been presented.**
- **None of the strategies are 100% fail-safe.**
- **Depending on the project requirements, and the target device's SEU susceptibility, the most efficient mitigation strategy should be selected.**
- **In most cases, devices with embedded mitigation do not require additional (user inserted) mitigation.**

Beware of unhardened global routes. They do cause system upsets.

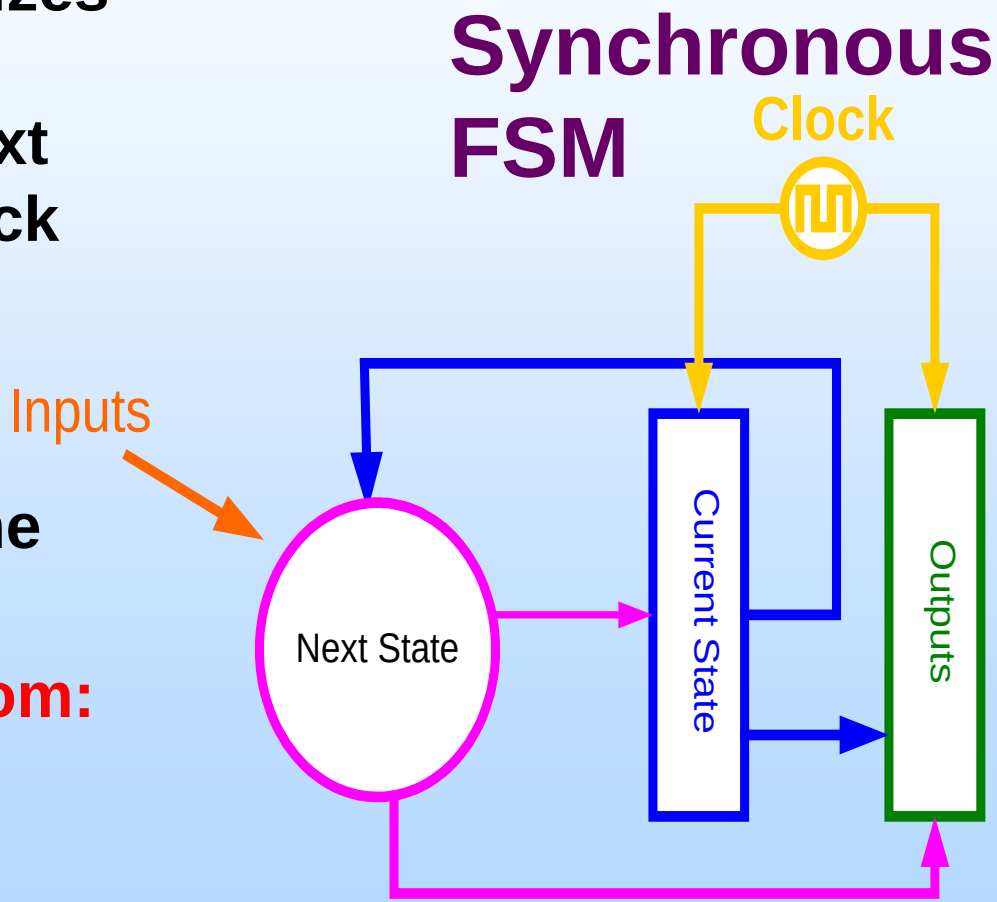
V5QV (SIRF) and ProASIC3 have radiation hardened configuration but do not have hardened global routes.

Fail-Safe State Machines

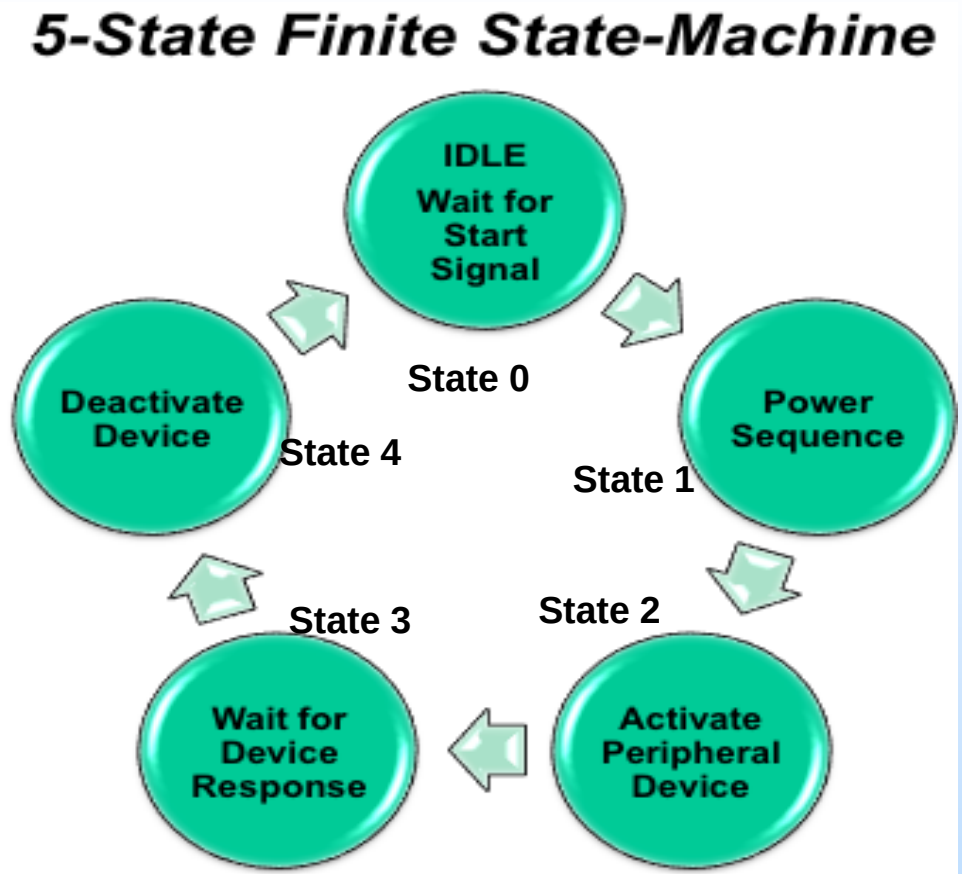


Synchronous FSMs and SEUs

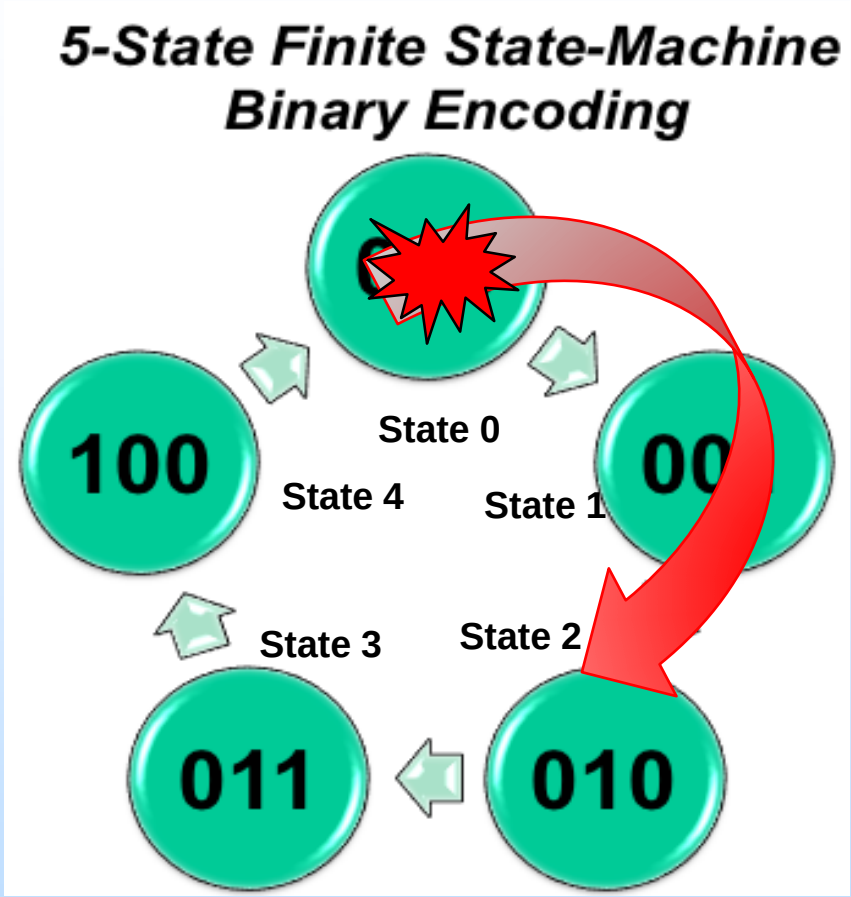
- A synchronous FSM is designed to deterministically transition through a pattern of defined states
- A synchronous FSM utilizes DFFs to hold its current state, transitions to a next state controlled by a clock edge and combinatorial logic, and only accepts inputs that have been synchronized to the same clock
- FSM SEUs can occur from:
  - Caught data-path SETs
  - DFF SEUs
  - Clock/Reset SETs



5-State FSM Binary Encoding Example



Example of an FSM used to control a peripheral device



5-State FSM with each state encoded as binary numbers.

An SEU can change current state and cause a catastrophic event

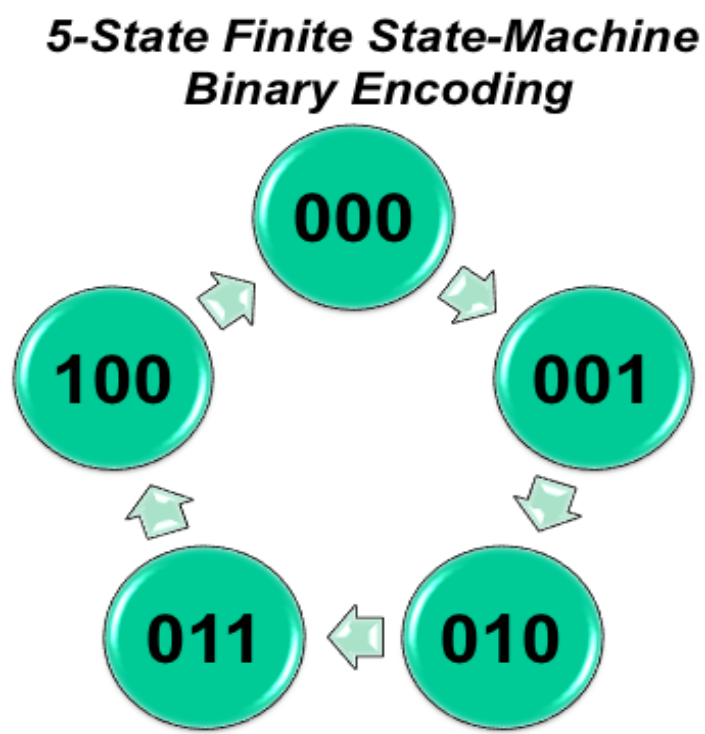
To be published on seemapld.org originally presented at 2016 Single Event Effects (SEE) Symposium and the Military and Aerospace Programmable Logic Devices (MAPLD) Workshop.

How Do We Implement Fail-Safe FSMs?

- **Question: A designer states that all FSMs have been implemented as “safe”, what do you expect?**
- **Correction? Detection? Masking?**
  - **What does correction mean?**
  - **All mitigation shall be defined unambiguously by the requirements and by the designer.**

Safe State Machines

- As currently defined by design tools and by some designers, the term “safe” state machine is a misnomer.
- Auto transitioning (“safe state-machine”) is a reaction to a small subset of incorrect transitions (unmapped states). They do not correct or mask (protect) against incorrect transitioning.

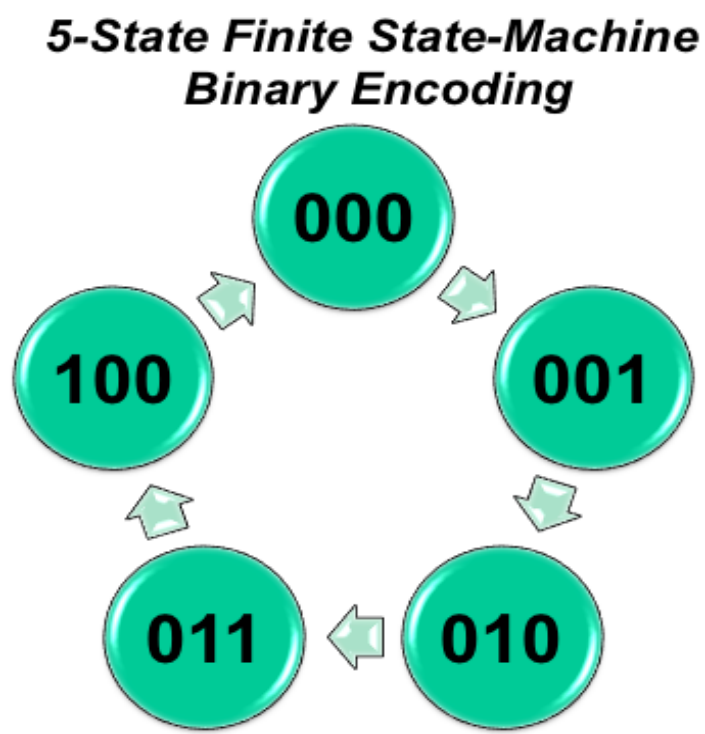


What happens if an SEU causes a transition from “001” to “101” ?

State	Mapped or Unmapped
000	Yes
001	Yes
010	Yes
011	Yes
100	Yes
101	No
110	No
111	No

Safe State Machines: What happens if an SEU causes a transition from “001” to “101” ?

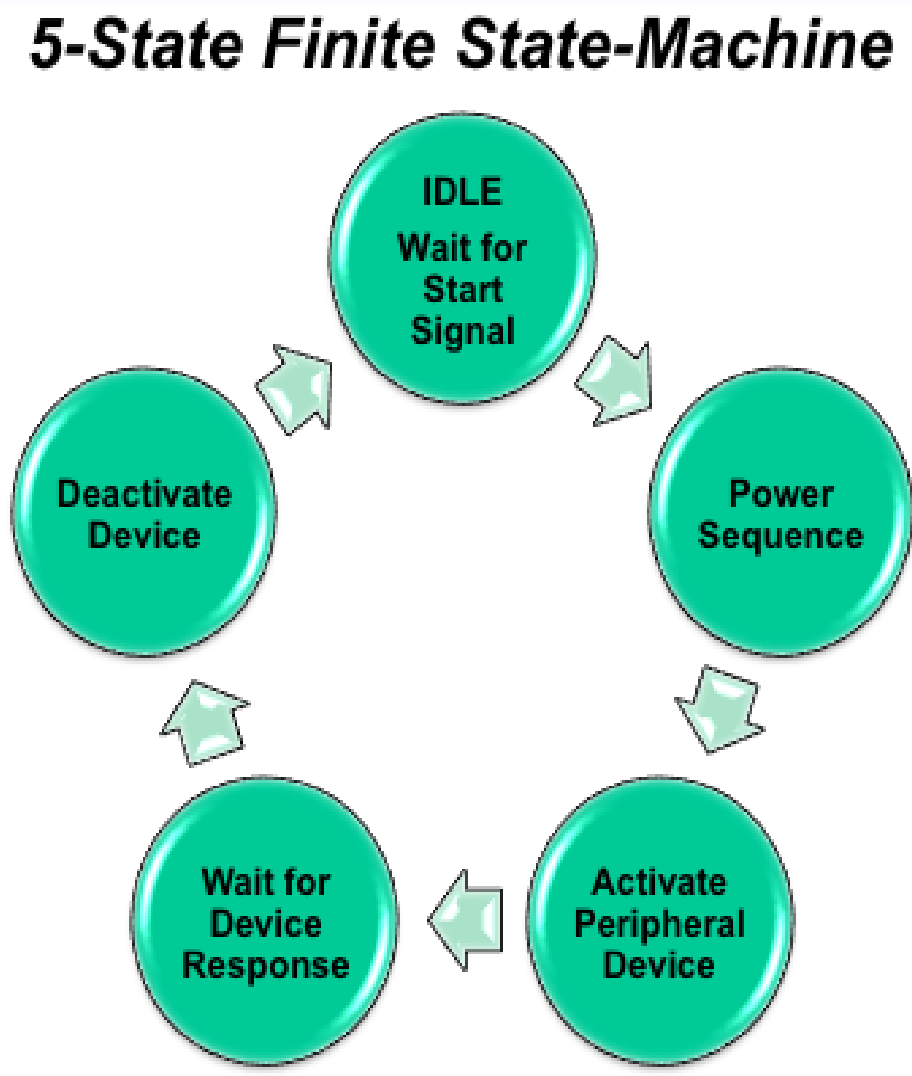
- As currently implemented, a “safe” state machine will automatically transition to a reset (or “safe” state).
- Problem: this could be detrimental to your system



State	Mapped or Unmapped
000	Yes
001	Yes
010	Yes
011	Yes
100	Yes
101	No
110	No
111	No

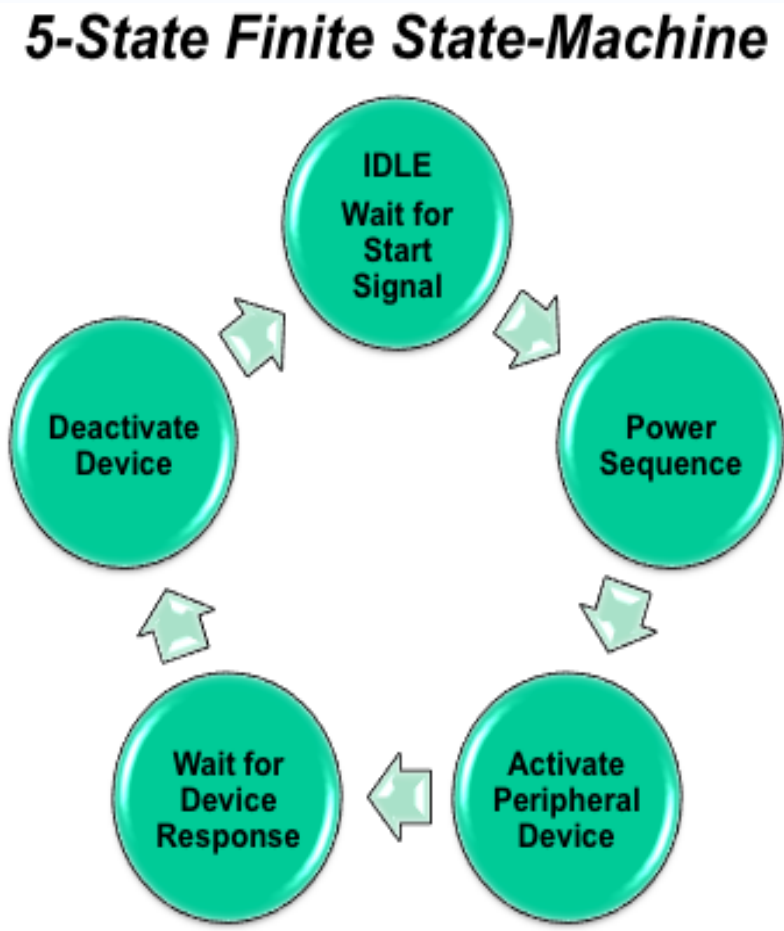
Problems with Current "Safe" FSM Definition

- Sounds more safe than what it really is.
- Does not do anything for incorrect transitions into mapped states.
- Does not correct the state:
  - Something that is supposed to be on will abruptly shut off.
  - Other FSMs or control logic can become unsynchronized with the bad FSM; with or without the automated jump to a "safe" state.



Can Auto-transitioning Work for Your Mission?

- **Auto-transitioning can work if incorrect sequencing of your FSM will not cause system failure; e.g. mathematical logic control.**
- **Auto-transitioning can be acceptable if it is used in conjunction with a detection flag. The detection flag must propagate to all necessary logic.**
- **But remember, there is no protection or detection with auto-transitioning when incorrectly transitioning to a mapped state.**



Auto-transitioning + detection is available with computer aided design (CAD) tools.

To be published on seemapld.org originally presented at 2016 Single Event Effects (SEE) Symposium and the Military and Aerospace Programmable Logic Devices (MAPLD) Workshop.

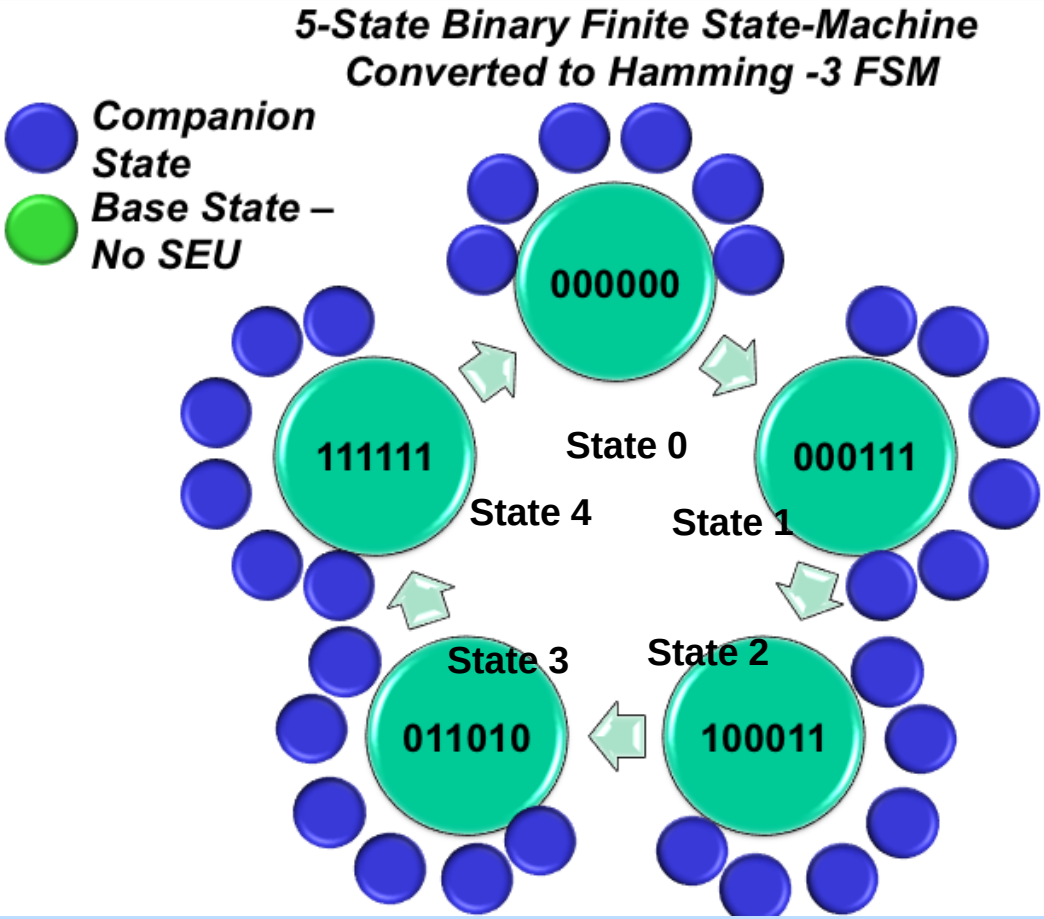
Implementing Corrective Logic for FSMs

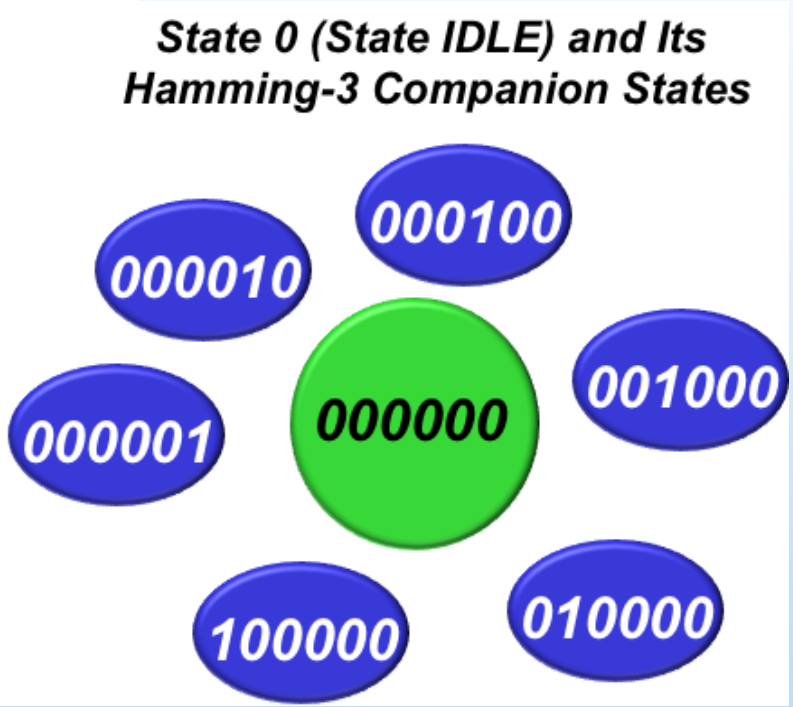
- **ASICs or FPGAs with hardened configuration:**
  - **LTMR: Triplicate each DFF and use a majority voter.**
    - **The triplication + voter is treated as one DFF**
    - **Encoding doesn't change**
    - **Resultant FSM has 3 times the number of DFFs than the original encoding scheme.**
    - **Combinatorial logic (not including the voters) does not change**
  - **Hamming Code-3: requires a new encoding scheme.**
- **FPGAs with commercial SRAM configuration: DTMR is suggested.**

There are computer aided design tools (CAD) that can assist in adding all of the above mitigation strategies.

Be careful regarding unhardened (per SEU) global routes.

FSM Fault Tolerance:
5-State Conversion to a Hamming Code-3 FSM





Hamming Code-3 FSM Diagram for a 5 Base-State FSM: Would need 5*7=35 FSM states to be represented… 6 DFFs

A closer look at a base-state (state 0) and its companion-states

Some Thoughts

Concerns and Challenges of Today and Tomorrow for Mitigation Insertion

- **User insertion of mitigation strategies in most FPGA and ASIC devices has proven to be a challenging task because of reliability, performance, area, and power constraints.**
  - **Difficult to synchronize across triplicated systems,**
  - **Mitigation insertion slows down the system.**
  - **Can't fit a triplicated version of a design into one device.**
  - **Power and thermal hot-spots are increased.**
- **The newer devices have a significant increase in gate count and lower power. This helps to accommodate for area and power constraints while triplicating a design. However, this increases the challenge of module synchronization.**
- **Embedded mitigation has helped in the design process. However, it is proving to be an ever-increasing challenge for manufacturers.**
  - **We (users) want embedded systems: cheaper, faster, and less power hungry.**
  - **However, heritage has proven that for critical applications, embedded systems have provided excellent performance and reliability.**

To be published on seemapld.org originally presented at 2016 Single Event Effects (SEE) Symposium and the Military and Aerospace Programmable Logic Devices (MAPLD) Workshop.

Summary

- For critical applications, mitigation may be required.
- Determine the correct mitigation scheme for your mission while incorporating given requirements:
  - Understand the susceptibility of the target FPGA and potential necessity of other devices.
  - Investigate if the selected mitigation strategy is compatible to the target FPGA device.
  - Calculate the reliability of the mitigation strategy to determine if the final system will satisfy requirements.
  - Ask the right questions regarding functional expectation, mitigation, requirement satisfaction, and verification of expectations.
- Although it is desirable from a user's perspective to have embedded mitigation, cost seems to be driving the market towards unmitigated commercial FPGA devices. Hence, it will be necessary for user's to familiarize themselves with optimal mitigation insertion and usage.

Acknowledgements

- ***Some of this work has been sponsored by the NASA Electronic Parts and Packaging (NEPP) Program and the Defense Threat Reduction Agency (DTRA).***
- ***Thanks is given to the NASA Goddard Radiation Effects and Analysis Group (REAG) for their technical assistance and support. REAG is led by Kenneth LaBel and Jonathan Pellish.***

Contact Information:

Melanie Berg: NASA Goddard REAG FPGA Principal Investigator:

Melanie.D.Berg@NASA.GOV