FPGA Mitigation Strategies for Critical Applications

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

Melanie.D.Berg@NASA.gov

Michael Campola NASA/GSFC

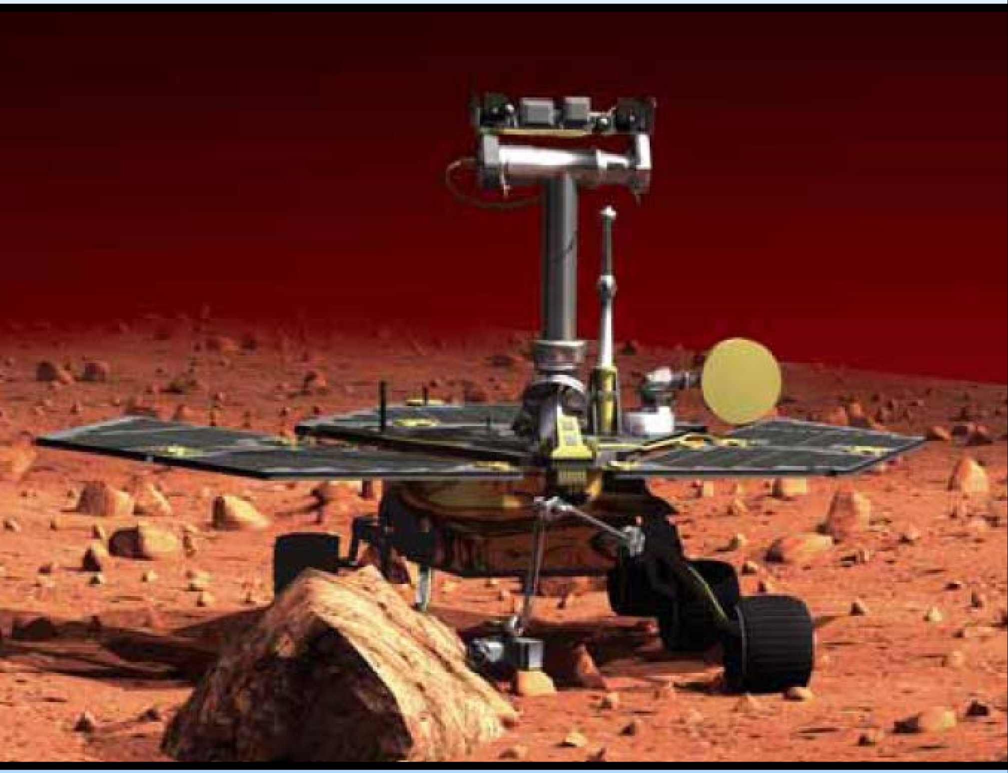
To be presented by Melanie Berg at the Radiation Effects on Components and Systems (RADECS) conference, Gothenburg, Sweden, September 16th – September 21st , 2018

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)
- Constant false alarm rate filter (CFAR)
- Device under test (DUT)
- Digital Signal Processing Block (DSP)
- Distributed triple modular redundancy (DTMR)
- Dual interlocked storage cell (DICE)
- Edge-triggered flip-flops (DFFs)
- Error detection and correction (EDAC)
- Error rate (dE/dt)
- Field programmable gate array (FPGA)
- Finite impulse response filter (FIR)
- Gate Level Netlist (EDF, EDIF, GLN)
- Global triple modular redundancy (GTMR)
- Input – output (I/O)
- INV (inverter)
- Linear energy transfer (LET)
- Local triple modular redundancy (LTMR)
- Look up table (LUT)
- Mean fluence to failure (MFTF)
- Mean Time to Failure (MTTF)
- Operational frequency (*fs*)
- Power on reset (POR)
- Place and Route (PR)
- Probability of flip-flop upset ($P_{DFFSEU \rightarrow SEU}$)
- Probability of logic masking (P_{logic})
- Probability of transient generation (P_{gen})
- Probability of transient capture ($P(fs)_{SET \rightarrow SEU}$)
- Probability of transient propagation (P_{prop})
- 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)
- Time delay (τ_{delay})
- Total Ionizing Dose (TID)
- Transient width (τ_{width})
- Universal Serial Bus (USB)
- Virtex-5QV (V5QV)
- Windowed Shift Register (WSR)

Agenda

- Field Programmable Gate Array (FPGA) Devices: Challenges for Critical Applications and Space Radiation Environments.
- Single Event Upsets (SEUs) and FPGA configuration
- Single Event Upsets (SEUs) and FPGA data paths
- Fail-Safe Strategies for Critical Applications.



FPGA Devices: Challenges for Critical Applications and Space Radiation Environments

Motivation: Concerns for using FPGA Devices in Critical Applications

Critical applications expect to avoid disaster when disaster is probable.

- **Safety**: can circuits or humans be damaged or hurt?
- **Reliability** : will the device operate as expected?
- **Availability**: Includes down-time… is it acceptable?
- **Recoverability**: if the device malfunctions, can the system come back to a working state?
- **Trust**: Will the insertion of the device compromise security?

How To Protect A System from Failure

- Always take into account mission requirements.
- Investigate failure modes – understand risk:
  - Reliability testing (temperature, voltage, mechanical, and logic switching stresses).
  - Radiation testing: Single event effects (SEE) and total ionizing dose (TID).
- Wisely add redundancy:
  - Replication with correction.
  - Replication with detection. Requires recovery:
    - Switch to another device,
    - Try to recover state,
    - Start over,
    - Alert,
    - Do nothing… die.
- Add filtration: e.g., Finite impulse response (FIR) filters or Constant false alarm rate filter (CFAR).
- Add masking: Protect system operation from failures.

Characterizing SEUs: Radiation Testing and SEU Cross Sections

SEU Cross Sections (σ_{seu}) characterize how many upsets will occur based on the number of ionizing particles to which the device is exposed.

$$\sigma_{seu} = \frac{\#errors}{fluence} \qquad MFTF = \frac{1}{\sigma_{seu}}$$

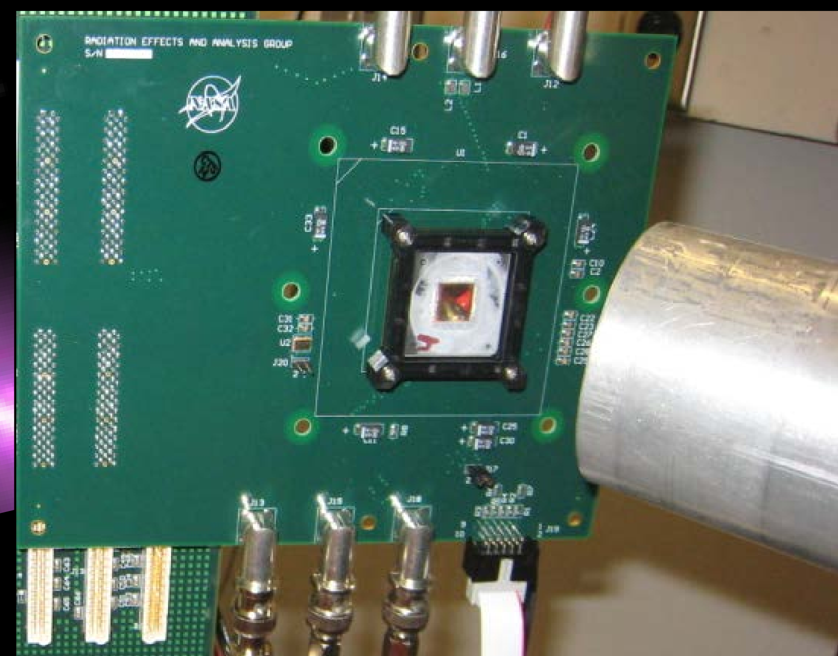
Terminology:

- **Flux: Particles/(sec-cm^2)**
- **Fluence: Particles/cm^2**

σ_{seu} is calculated at several LET values (particle spectrum).

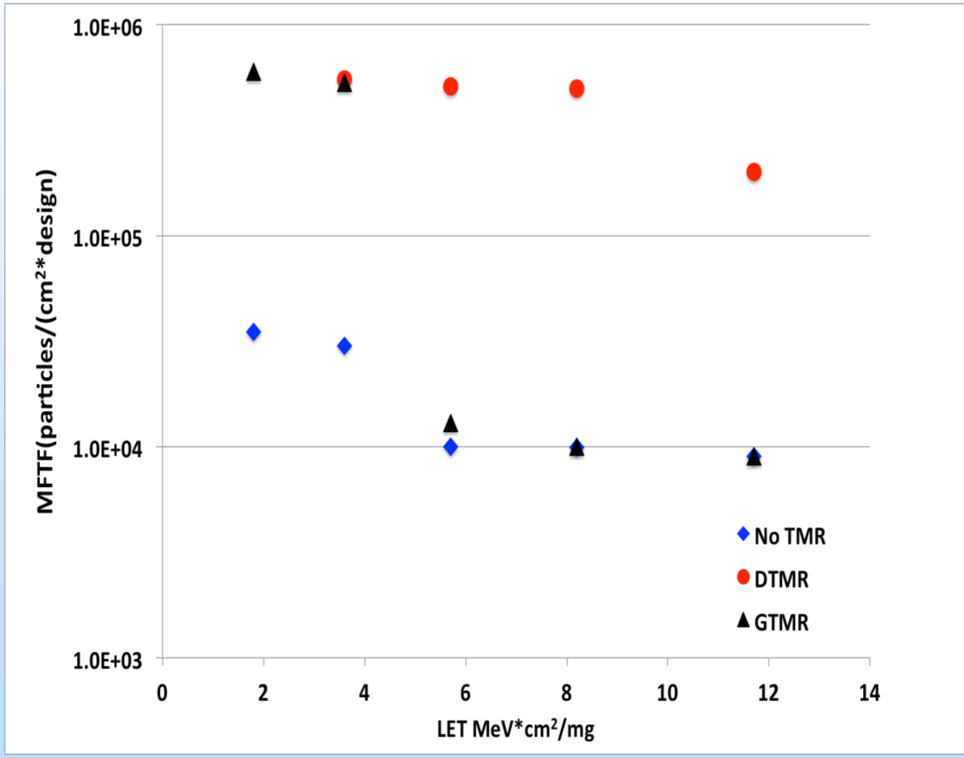
Mean fluence to failure (MFTF) is the inverse of σ_{seu}.

Heavy-ion testing at Texas A&M University

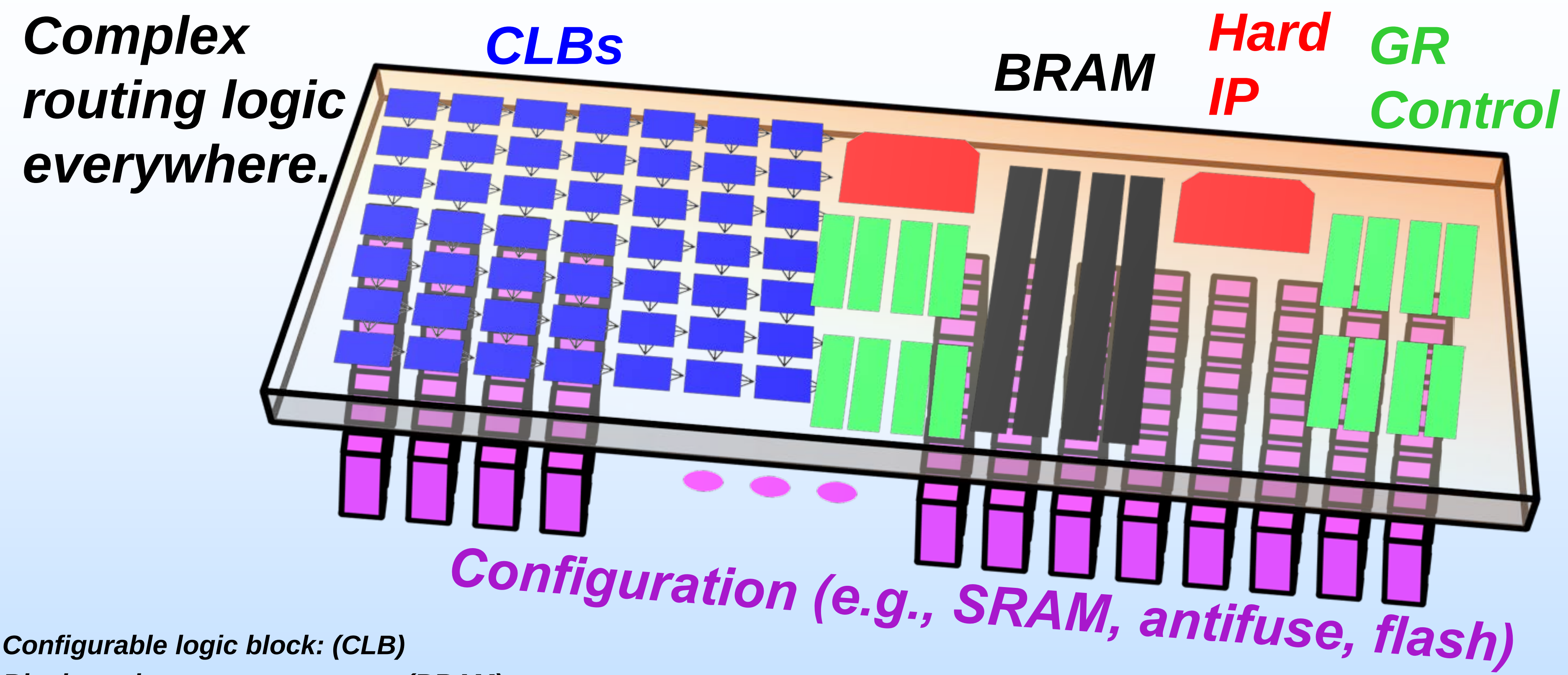


SEU Test Procedure and MFTF

- **Create design to be placed in FPGA device under test (DUT).**
- **Create tester. Provides stimuli and monitors DUT responses to beam exposure.**
- **Select energy, particle, and particle LET (if beam cocktail is heavy-ions).**
- **Expose DUT to beam.**
- **Monitor DUT response for unexpected behavior. Visibility is key and extremely challenging.**



FPGA SEU Cross Section Model



Configurable logic block: (CLB)
Block random access memory: (BRAM)
Intellectual property: (IP); e.g., micro processors, digital signal processor blocks (DSP), embedded state machines, etc,…
Global Routes: (GR)
Analog circuits

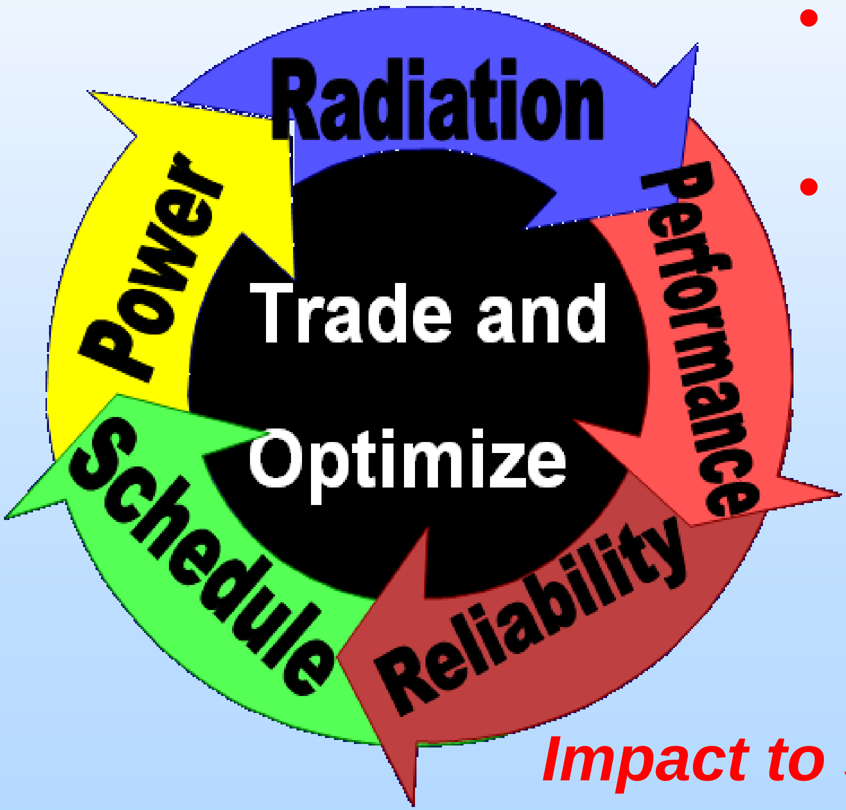
$$P(fs)_{error} \propto P(fs)_{Configuration} + P(fs)_{functionalLogic} + P(fs)_{SEFI}$$

Design σ_{SEU} | Configuration σ_{SEU} | Functional logic σ_{SEU} | SEFI σ_{SEU}

Preliminary Design Considerations for Mitigation And Trade Space

Determine Most Susceptible Components:

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



- **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.

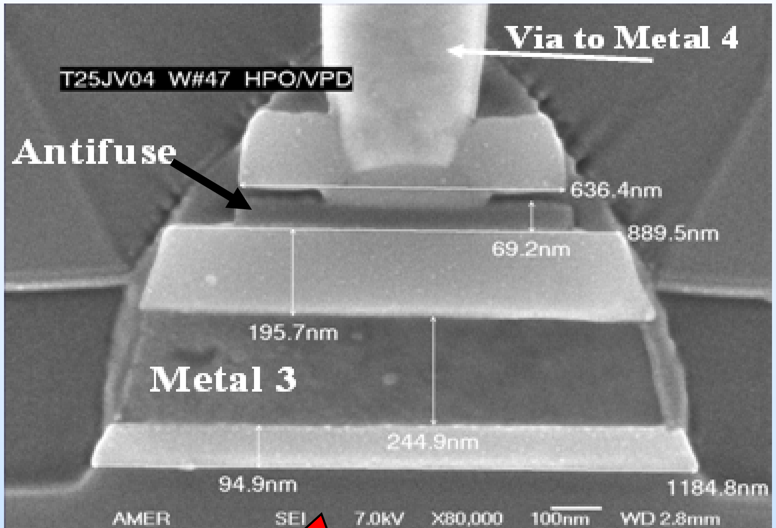
Single Event Upsets and FPGA Configuration

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

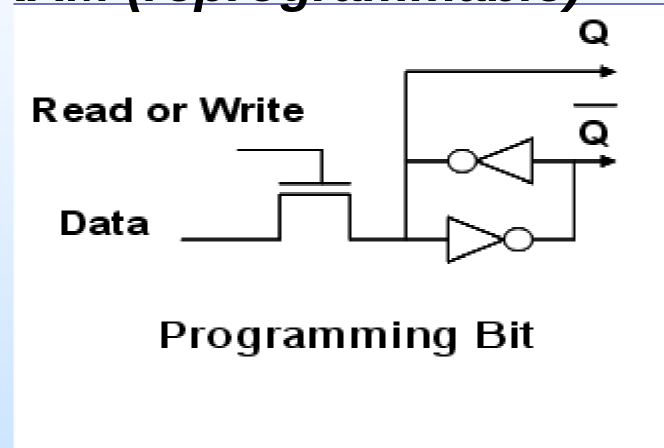
FPGA Configuration Implementation and SEU Susceptibility

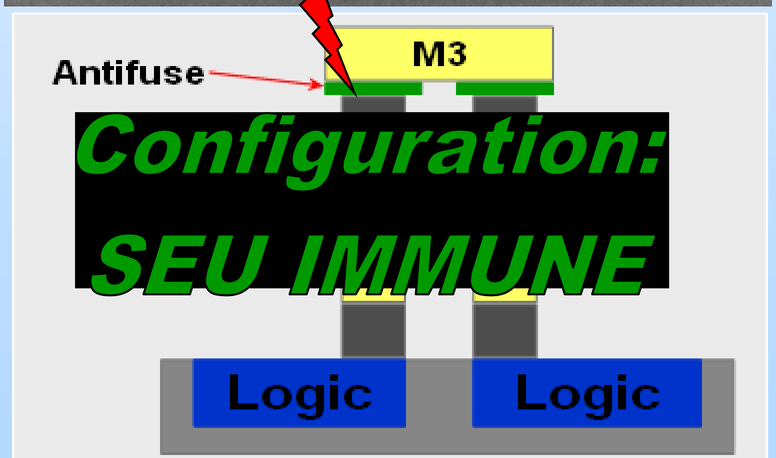
(There are a variety of FPGA configuration types)

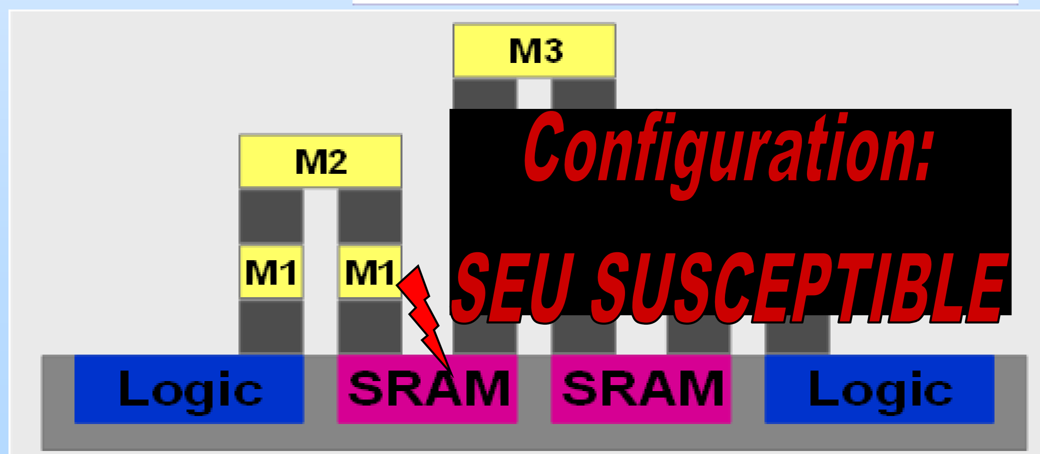
ANTIFUSE (one time programmable)



SRAM (reprogrammable)







Configuration SEU Test Results and the REAG FPGA SEU Model

Table shows the most significant SEE responses during accelerated radiation testing.

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

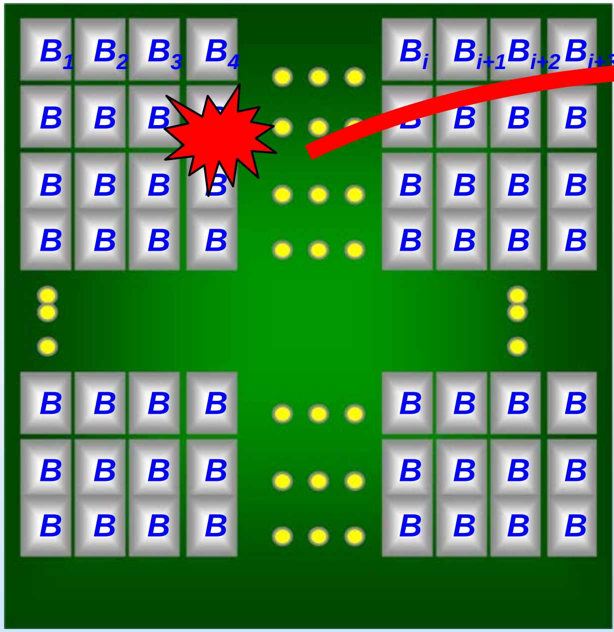
FPGA Configuration Type	REAG Model $P(fs)_{error}$
Antifuse	$P(fs)_{functionalLogic} + P(fs)_{SEFI}$
SRAM (non-mitigated)	$P(fs)_{Configuration} + P(fs)_{SEFI}$
Flash	$P(fs)_{functionalLogic} + P(fs)_{SEFI}$
Hardened SRAM	$P(fs)_{Configuration} + P(fs)_{functionalLogic} + P(fs)_{SEFI}$

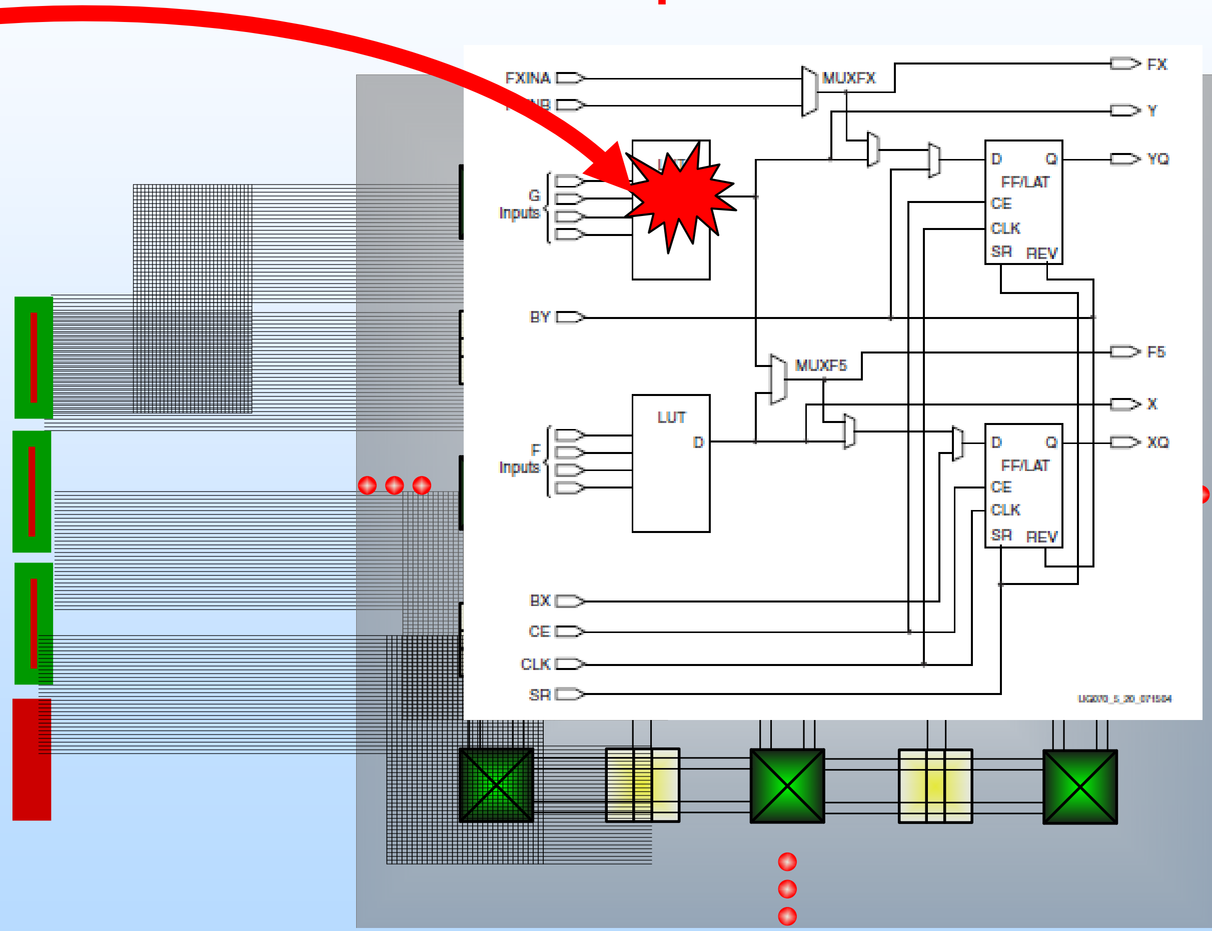
What Does The Last Slide Mean?

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).

Take Note: Configuration SRAM is NOT Utilized the Same Way as Traditional SRAM

Every active, used bit can instantaneously cause an unexpected effect

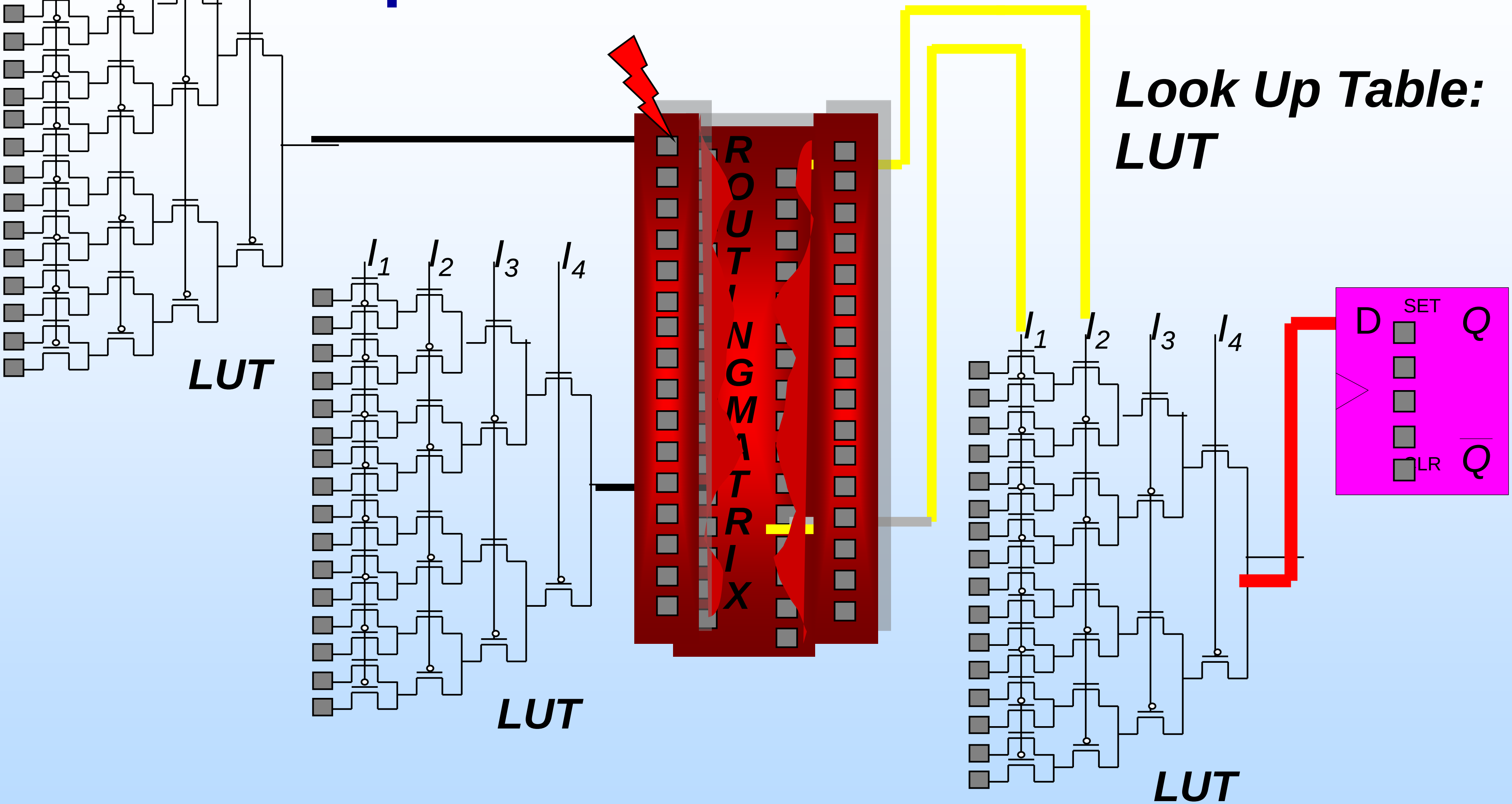




- **Direct connections from configuration to user logic.**
- **Upset occurs in an actively used configuration bit then, upset occurs in logic.**

No Read-Write cycle required!

Example: Routing Configuration Upsets in a Xilinx Virtex FPGA



One configuration bit flip can cause significant malfunction. Mitigate appropiately.

Fixing SRAM-based Configuration…Scrubbing Definition

- **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.

Scrubbers: Internal versus External

- **Internal and external scrubbers are implemented to correct configuration bit-flips:**
  - **Internal scrubber: is created out of hard cores that reside inside the FPGA device; or is created out of user fabric logic blocks located inside the FPGA device.**
  - **External scrubber is implemented in an separate device .**
- **Typically, external scrubbers are implemented in anti-fuse FPGA or flash-based FPGAs.**
- **Internal scrubbers are more susceptible than external scrubbers.**

Scrubbers: When Reality Defies Theory

- **Internal scrubbers are expected to provide satisfactory results in proton environments.**
  - **Clocks are not highly susceptible to protons because of their high drive strength.**
  - **Most of the logic used to implement the scrubber is embedded.**
  - **Scrubber should not require a large amount of circuitry.**
- **Note: Proton radiation testing of the Intel Cyclone 10 showed the device's internal scrubber does not work as expected.**
  - **Scrubber failed to remain operable with a fluence of 1×10^8 100MeV particles.**
  - **Results are unexpected.**
- **Implementation of the scrubber means everything!**
  - **Did Intel use a processor based internal scrubber?**
  - **Use of memory will cause the scrubber to be more susceptible than expected.**

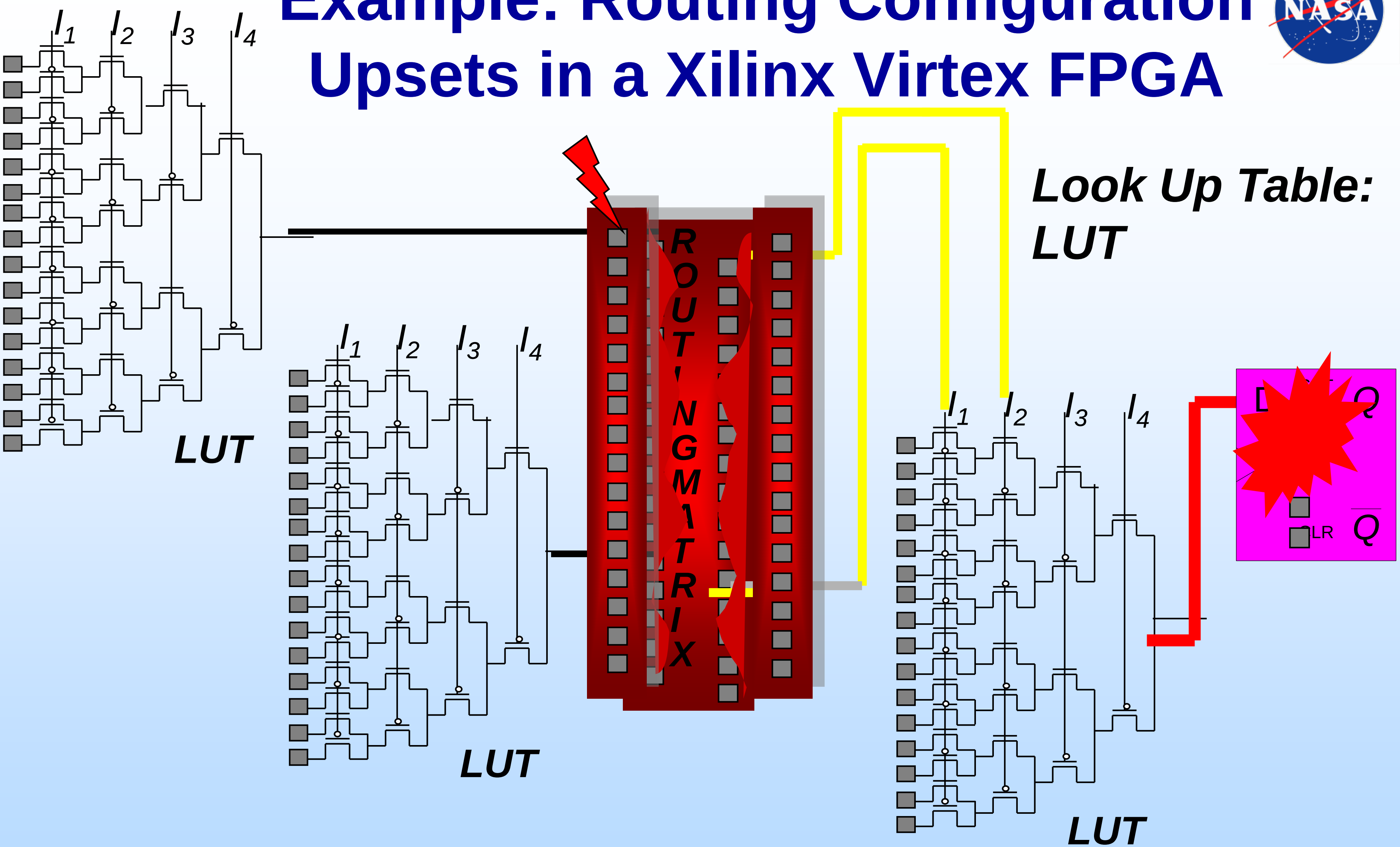
Warning!

- **Correcting 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.
- SEUs are asynchronous events that 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.
- The occurrence of an SET or SEU does not definitively cause system error.
- Probability of system malfunction is second order:
  - Probability that a transistor will unexpectedly change its state
    - Energy of particle
    - Type of particle
  - Probability that the changed state will cause the system to malfunction
    - Is the transistor in an active path?
    - Will its change of state be masked by other components?

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

- **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.**

Synchronous design was created because of the noise that is generated during transistor switching. This design topology also helps in de-rating SET capture.

Data-path SEU Susceptibility and Analysis : the NASA Electronics Parts and Packaging (NEPP) FPGA Model

Berg M.,” FPGA SEE Test Guidelines”, NASA Radiation Effects and Analysis Group Website: https://nepp.nasa.gov/files/23779/FPGA_Radiation_Test_Guidelines_2012.pdf, July 2012.

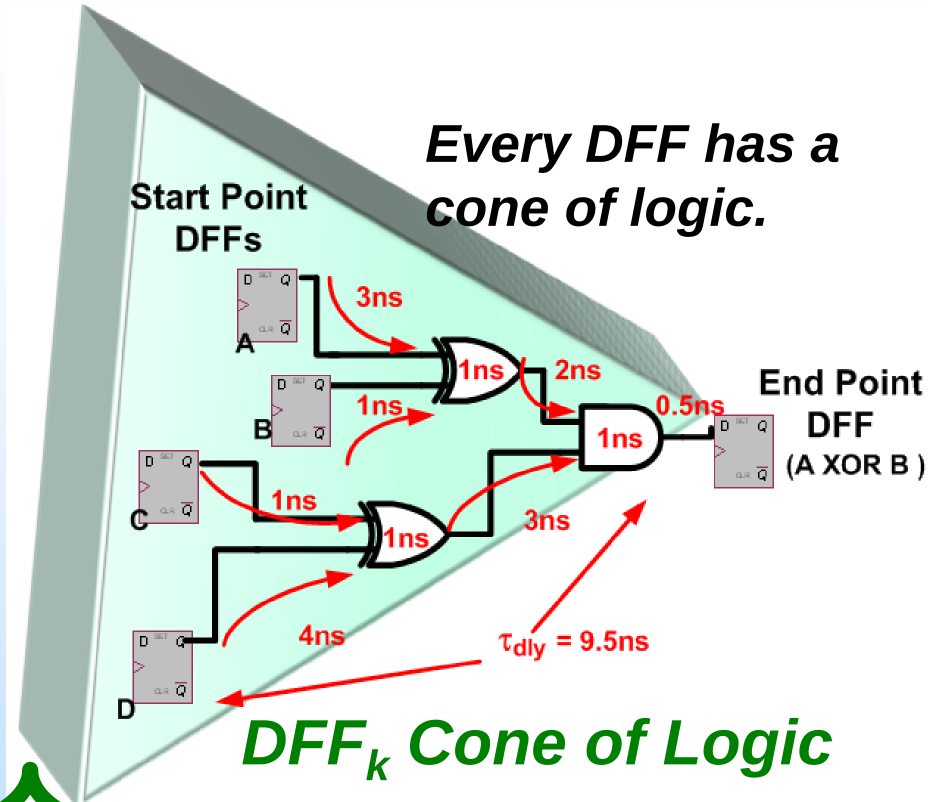
Incorrect DFF-States from SEUs

DFFs have various means of reaching a bad state due to SEUs.

System malfunction is not definitive with Wrong DFF State

Make a clear distinction between DFF SEUs based on Clock state and Capture.

Every DFF has a cone of logic.



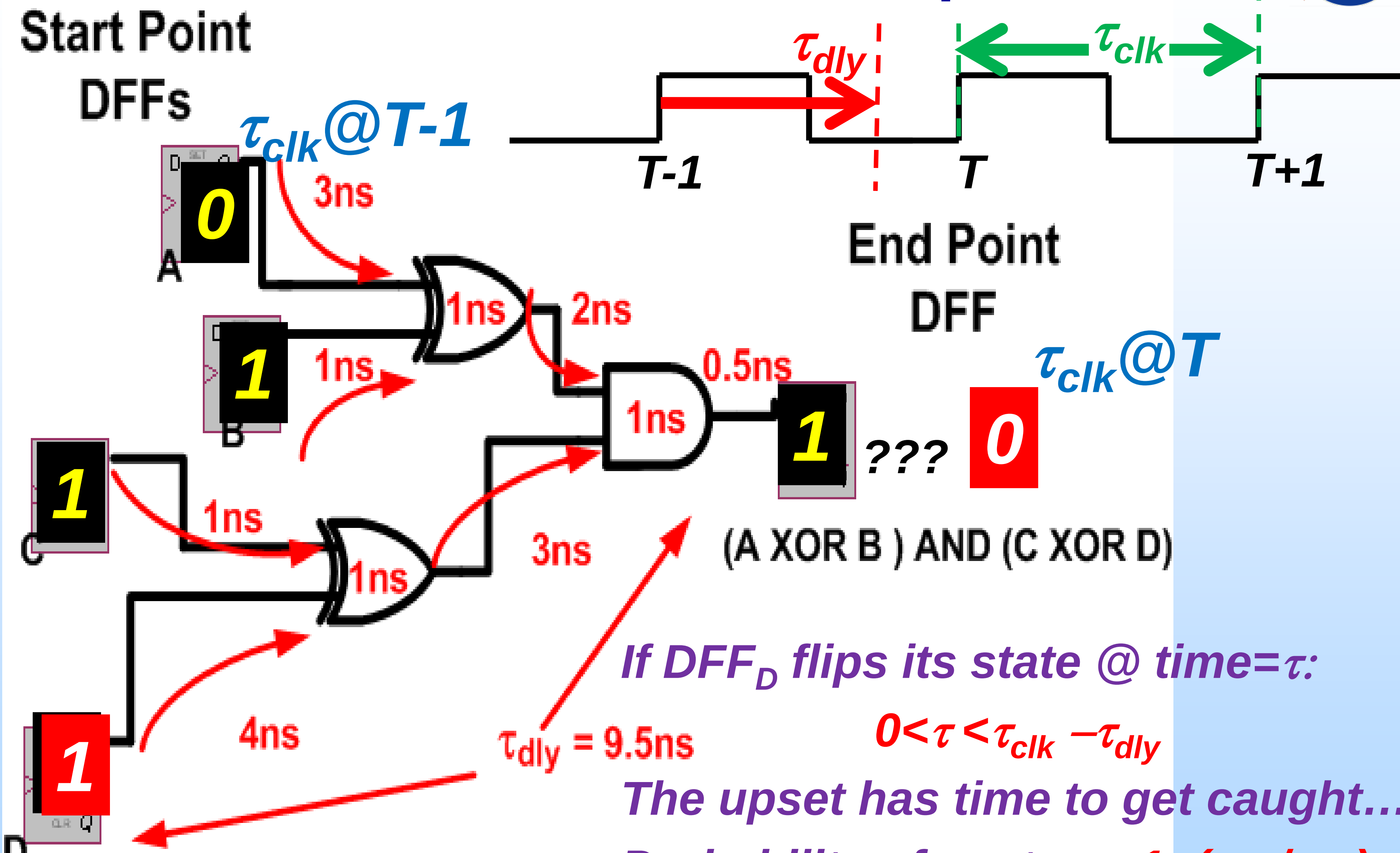
$\forall_{EndPoint\ DFF}$ ***EndPoint DFF SEUs + StartPoint DFF SEUs + CL SETs***

DFF upsets that occur at the clock edge.

DFF upsets that occur between clock edges and are captured by EndPoints.

Single Event Transients captured by EndPoints.

StartPoint DFF SEU Capture



If DFF_D flips its state @ time=τ:

$0<\tau<\tau_{clk}-\tau_{dly}$

The upset has time to get caught…

Probability of capture: $1-(\tau_{dly}/\tau_{clk})$

Percentage of Clock Cycle for SEU Capture:

$$\tau < \tau_{clk} - \tau_{dly}$$

Upset is caught within this timeframe.

$$\frac{\tau}{\tau_{clk}} < \frac{\tau_{clk} - \tau_{dly}}{\tau_{clk}} = 1 - \frac{\tau_{dly}}{\tau_{clk}}$$

Fraction of clock period for upset capture.

$$\tau \quad fs < 1 - \tau_{dly}\, fs$$

Upset capture with respect to frequency.

Details of Capturing StartPoint DFFs

$$\forall_{DFF}\left(\left(\sum_{j=1}^{\#StartPoint\,DFFs} \beta P(fs)_{DFFSEU(j)} \left(1-\tau_{dly(j)} fs\right) P_{logic(j)} \right) \right)$$

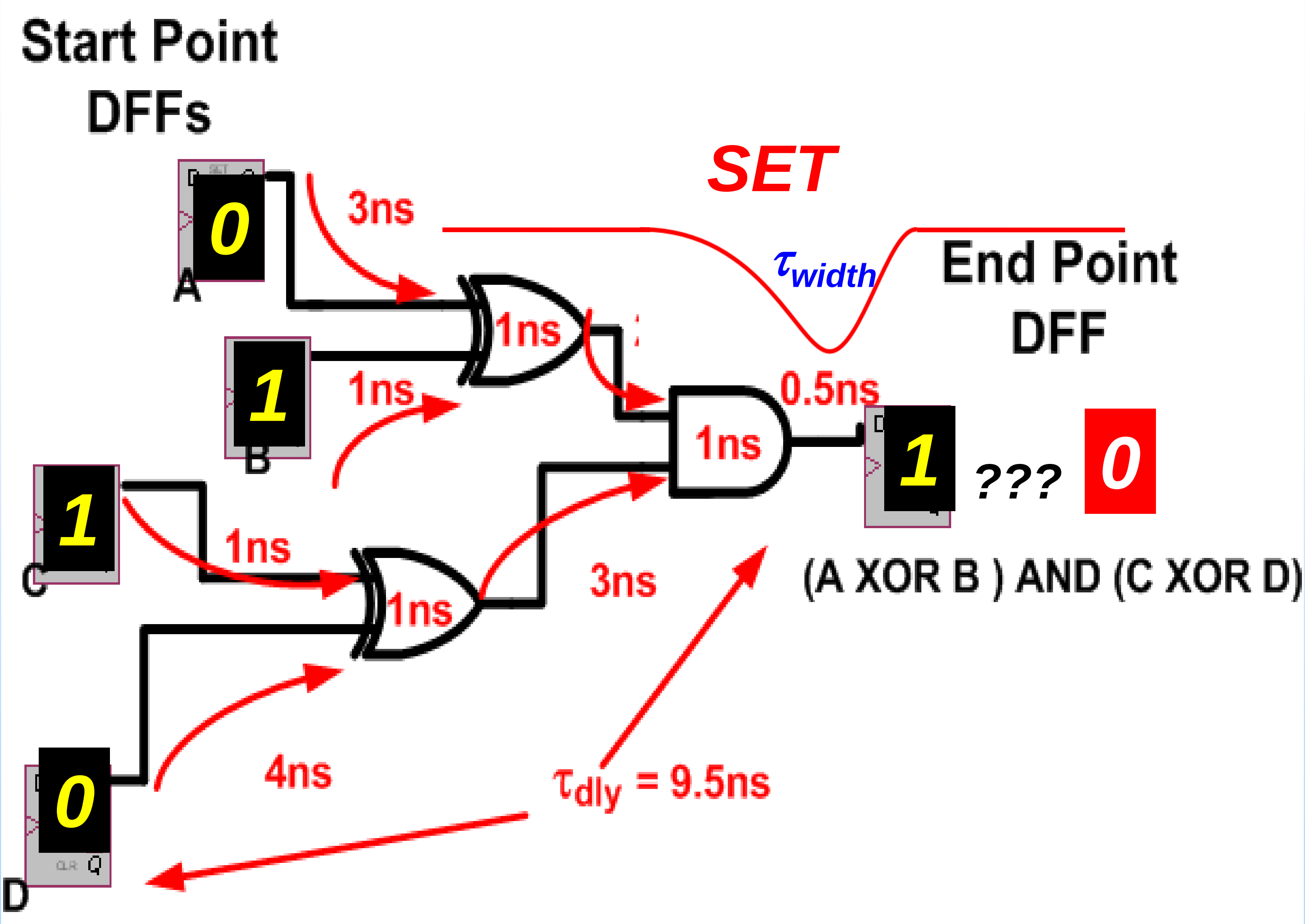
Upset generated internally to DFF between clock edges

Design Topology and Temporal Masking

SEU will not be logically masked

- **SEU generation occurs in a StartPoint between rising clock edges ($\beta P(fs)_{DFFSEU}$)**
- **StartPoint upsets can be logically masked by logic between the StartPoint and its EndPoint**
- **Design topology and temporal effects:**
  - **Increase path delay (# of gates) – decrease probability of capture**
  - **Increase frequency – decrease probability of capture**

Synchronous System: CL SET Capture



Details of CL SET Capture in a Synchronous System: P(fs)$_{DFF \rightarrow SET}$

$$\forall_{DFF}\left(\sum_{i=1}^{\#CombinatorialCells}\left(P_{gen(i)}P_{prop(i)}P_{logic}\tau_{width(i)}fs\right)\right)$$

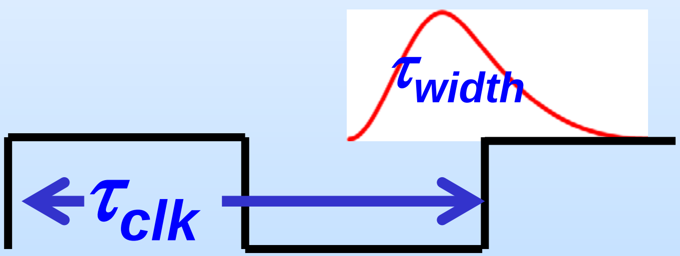
SET is Generated

Propagation:

SET can propagate through electrical medium (routes and gates) and reach the End-Point

SET will not be logically masked

Width of SET relative to clock period



- SET Generation (P_{gen}) occurs between clock edges
- EndPoint DFF captures the SET at a clock edge
  - Increase frequency – increase probability of capture.
  - Increase CL – increase probability of capture.
  - Increase LET – increase the width of the SET.

NEPP FPGA Model: Putting it All Together …

Analyzed Per Particle LET (as would be done for SEU Cross-sections)

$$\sum_{k=1}^{\#EndPoint\ DFFs} P_{logic(k)} * \left(\alpha P(fs)_{DFFSEU(k)} + \sum_{j=1}^{\#StartPoint\ DFFs} \left(\beta P(fs)_{DFFSEU(j)} (1-\tau_{dly(j)} fs) \right) * P_{logic(j)} \right) + \sum_{i=1}^{\#CL} \left(P_{gen_{(i)}} * P_{prop_{(i)}} * P_{logic_{(i)}} * \tau_{width_{(i)}} fs \right) \right)$$

EndPoint Logic Masking; EndPoint; StartPoints; CL

Table: Component Contribution to σ_{SEU} across Frequency and Gate Count

	Frequency	# of Gates in Path
EndPoint	Directly Proportional	N/A
StartPoint	Inversely Proportional	Inversely Proportional
CL	Directly Proportional	Directly Proportional

Clock Trees and SETs

- **Examples only considered data paths.**
- **However, clock and reset trees (global routes) are susceptible to SETs**
- **Clock trees in ASICs and FPGAs are the most overlooked mechanism of failure due to ionization.**
- **Global route susceptibilities must be taken into account when determining system risk.**
- **Global route susceptibilities are different for each FPGA device.**

There is not much a user can do to mitigate clock tree SETs. However, it is imperative to know susceptibilities – probability of occurrence and associated error signatures.

Fail-safe Strategies for Data-Path 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

Mitigation

- 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.
    - EXPENSIVE.
- Mitigation should reduce error…
  - Generally through redundancy and correction.
  - Incorrect implementation can increase error.
  - Overly complex mitigation cannot be verified and incurs too high of a risk to implement.

Embedded Mitigation versus User Inserted Mitigation

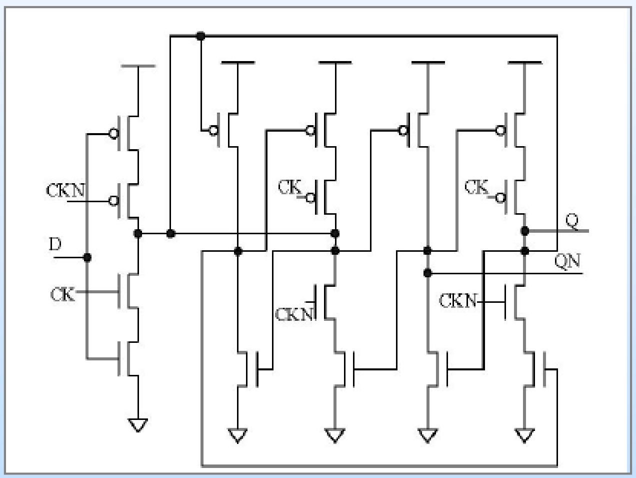
Radiation Hardened (per SEU) versus Commercial FPGA Devices

- 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 Manufacturer DFF Cells

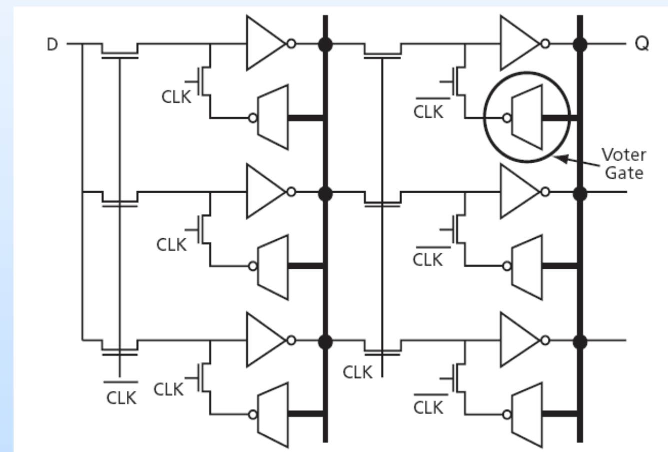
Warning! These figures are simplified schematics of the actual implementation.

Dual Interlocked Cell (DICE)



Xilinx

Localized Triple Modular Redundancy (LTMR)



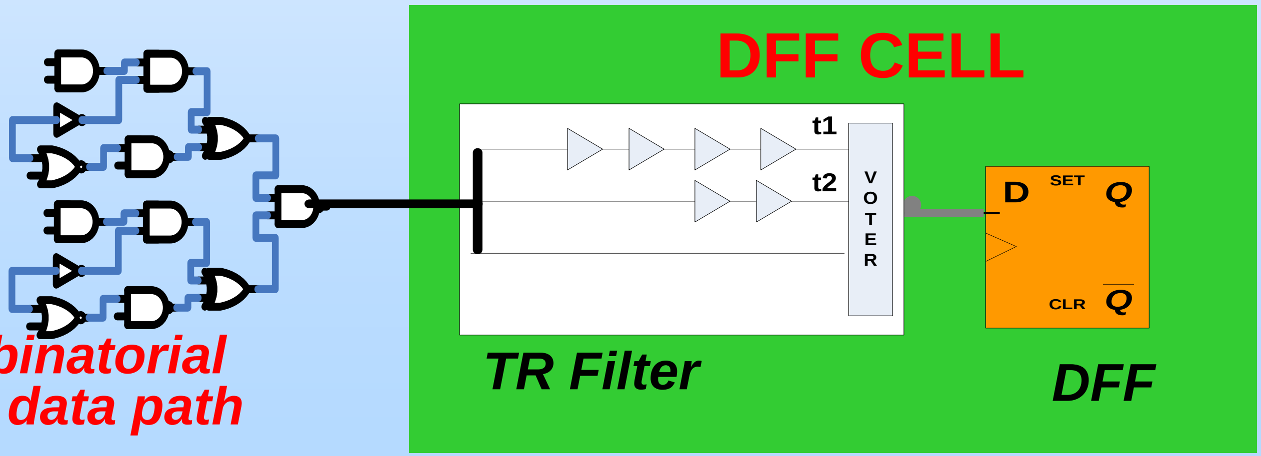
Microsemi

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.

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.**

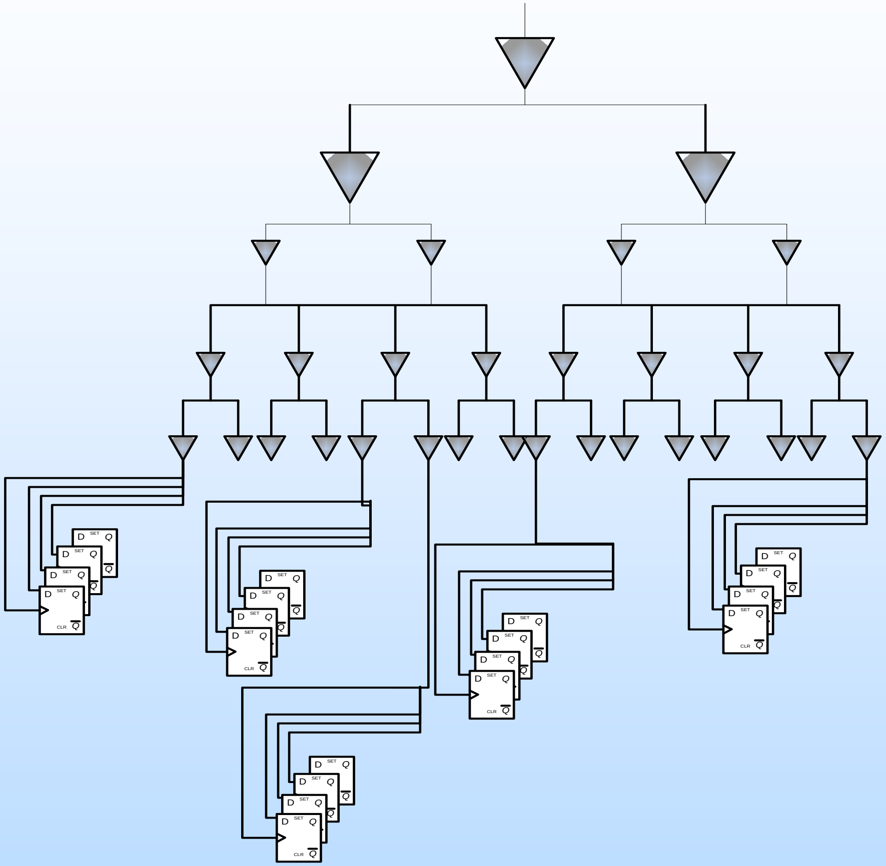
Crude example of TR implementation



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 DFF clock pin… (Xilinx V5QV only).**

Clock Tree



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

FPGA Devices and Manufacturer Embedded Mitigation

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 and clock trees
Antifuse	RTAX, RTSXS	DFFs and clocks (configuration is already hardened by nature)	Combinatorial logic (however susceptibility considered low)
Flash	ProASIC3,RTG4, SmartFusion(2)	Configuration is already hardened by nature.	ProASIC3 and SmartFusion: DFFs and clocks; RTG4: clocks and SETs
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:

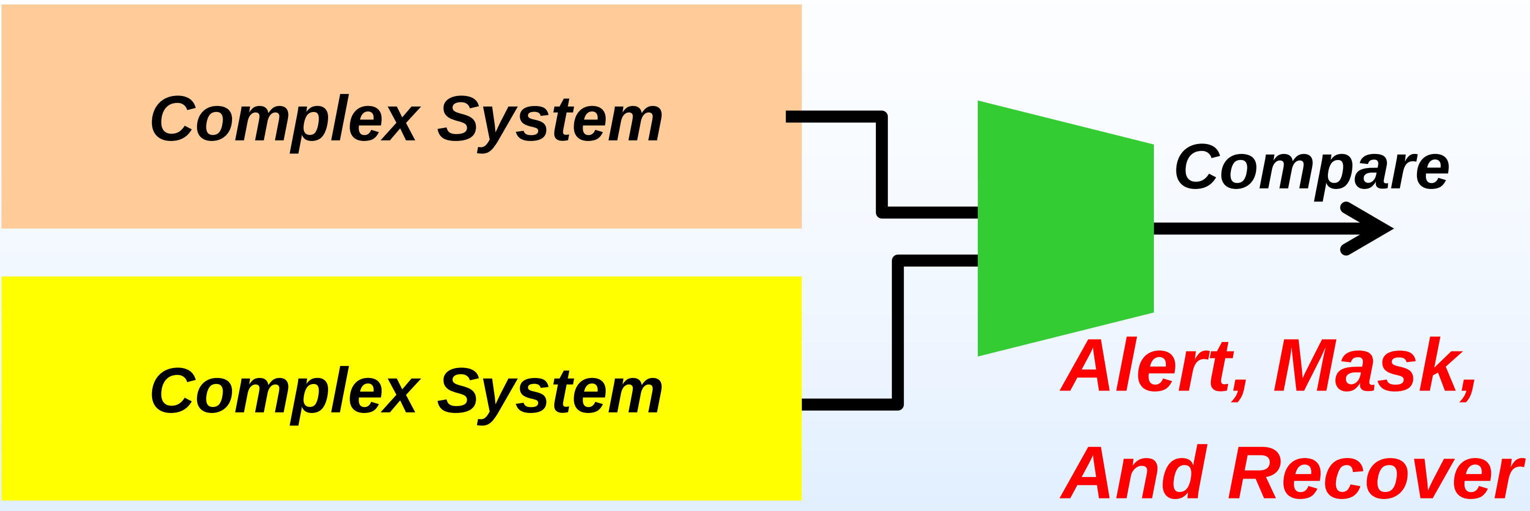
Flushing, Dual Redundancy, Cold Sparing, and Triple Modular Redundancy (TMR)

Most Commonly Implemented System Level Mitigation: Reset or Flush

- Critical applications require all registers (flip-flops) to be connected to a reset.
- A reset is used to force the system to a known (expected) state in a deterministic time period.
- All elements are expected to be able to operate from the reset state. However:
  - For some FPGAs, a reset is not enough. The configuration might also have to be flushed (reconfigure or scrub).
  - Availability is affected.
  - Next state information during event is most likely lost.
  - All must be taken into account when determining the effect of activating a reset in a system.

Warning: Resets are susceptible to SEEs

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.**
- **Can run in lockstep or free running.**

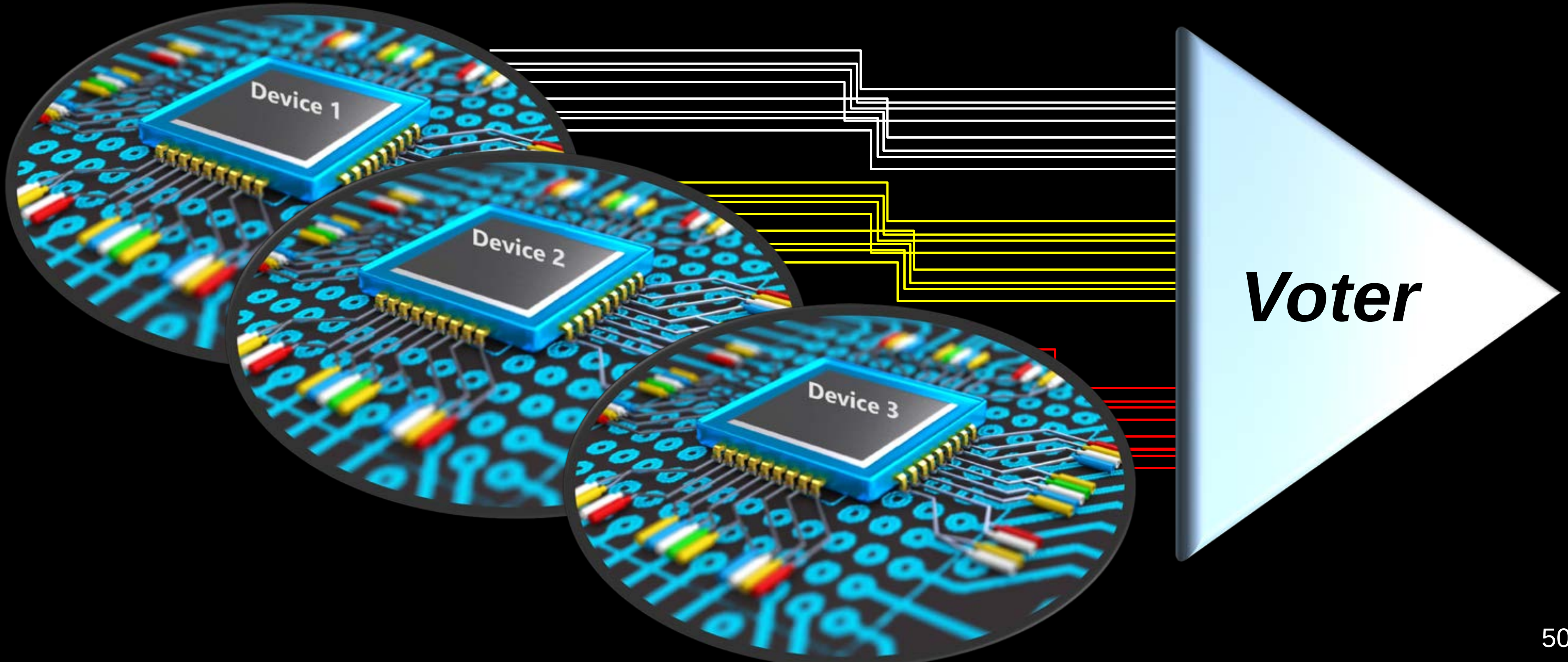
Cold Sparing: Elongation of System Operation

- 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 affected… 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

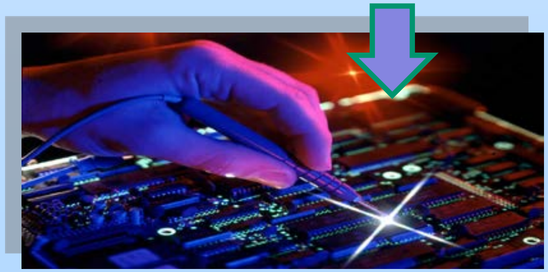
Functional Specification

HDL

Synthesis

Place and Route

Create Configuration



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.

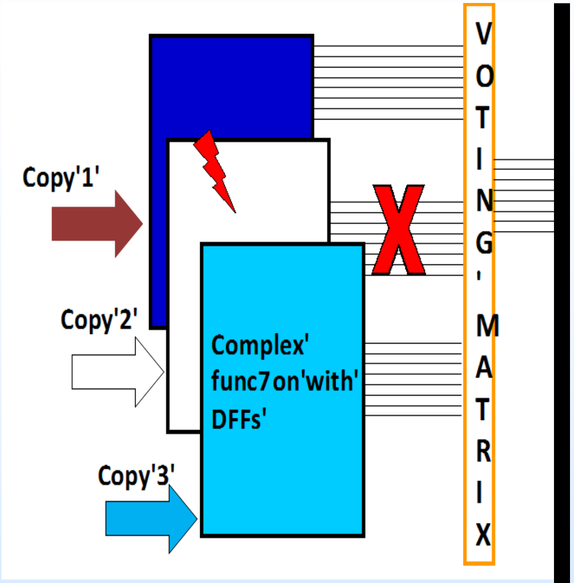
TMR can be inserted during synthesis or post synthesis.

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

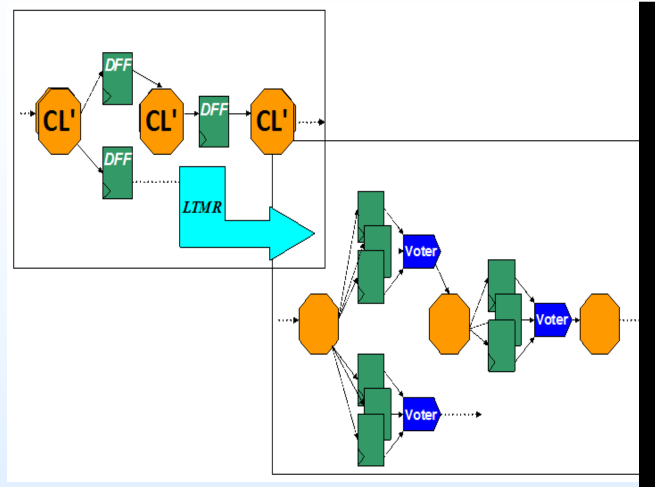
Local Mitigation versus Distributed or Global Mitigation

- **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.**

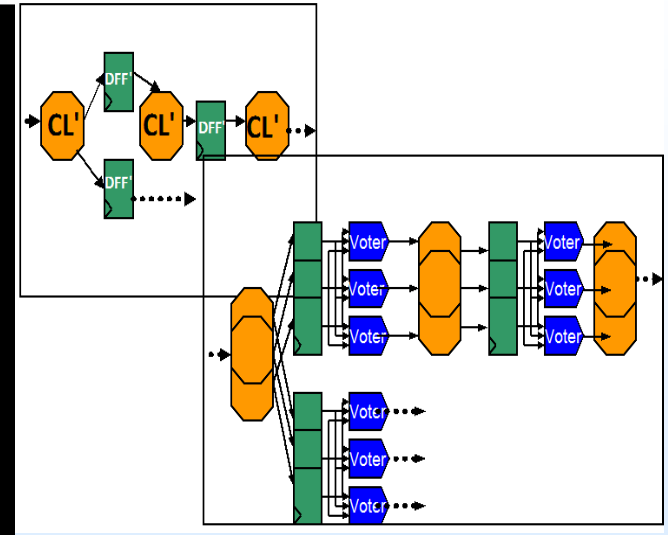
Various TMR Schemes: Different Topologies



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.

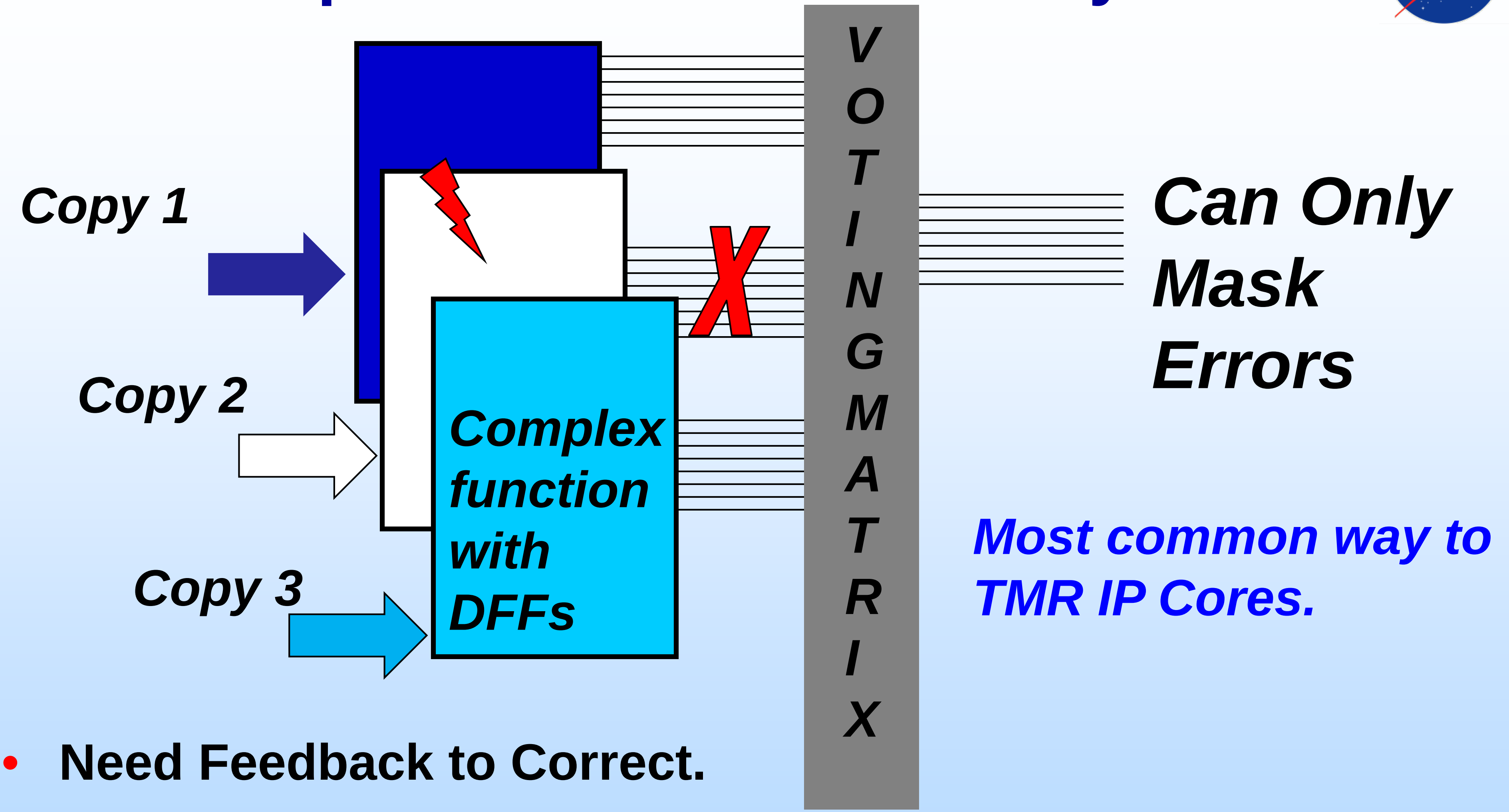


Block Diagram of distributed TMR (DTMR): the entire design is triplicated except for the global routes (e.g., clocks); voters are brought into the design and placed after the flip-flops (DFFs). DTMR masks and corrects most single event upsets (SEUs).

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



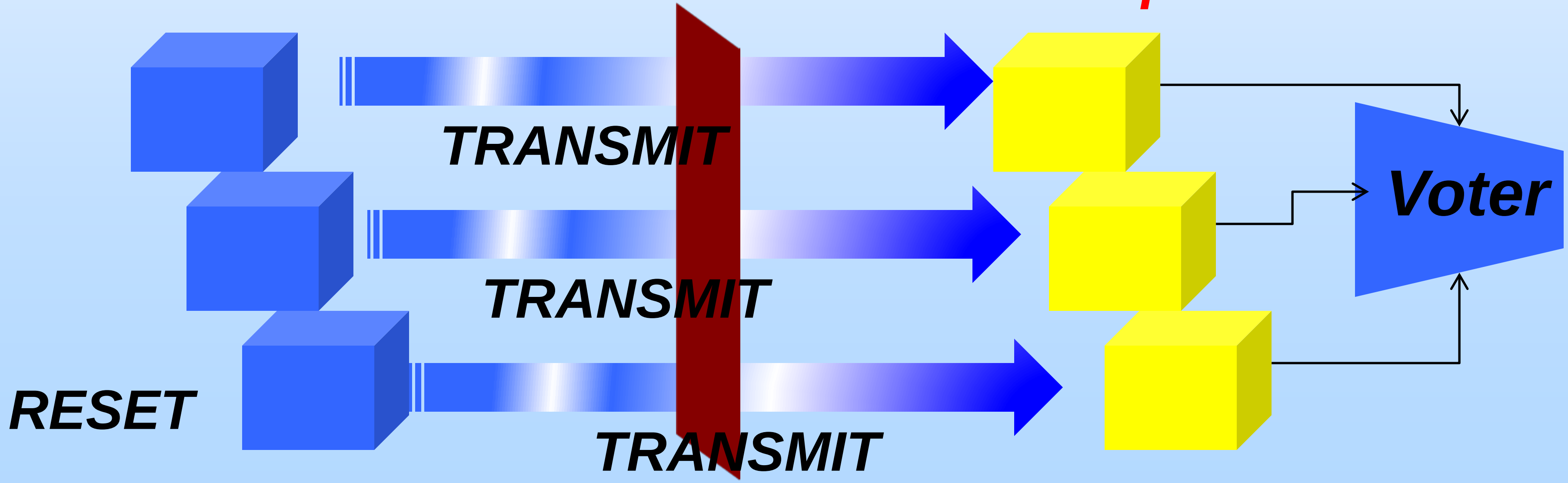
Most common way to TMR IP Cores.

- **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

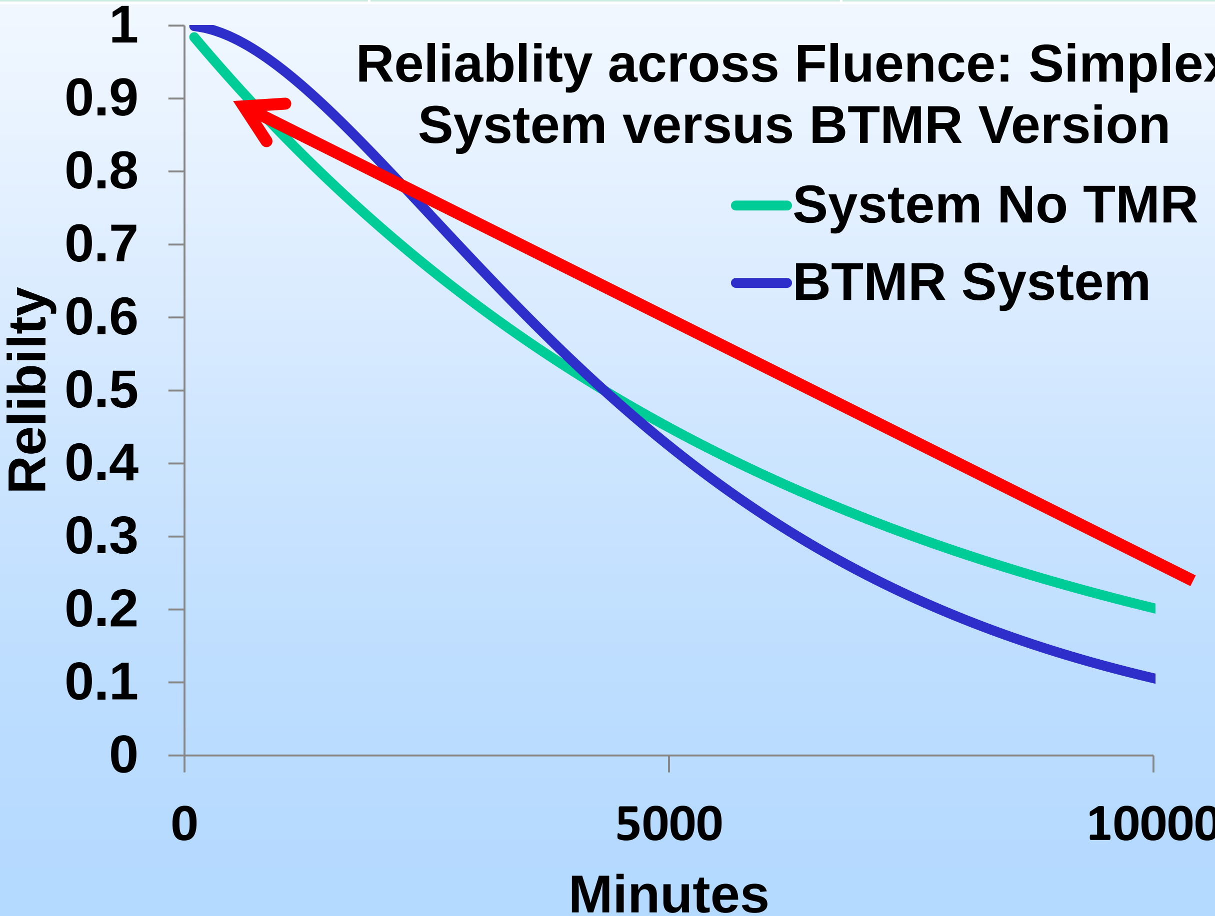
- **Shift Registers.**
- **Transmission channels: It is typical for transmission channels to send and reset after every sent packet.**
- **Systems that can be reset (or power-cycled) every so-often.**

Transmission channel example:



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 ($MTTF_{BTMR}$)
$e^{-\lambda t}$	$3e^{-2\lambda t} - 2e^{-3\lambda t}$	$1/\lambda$	$(5/6\lambda) = 0.833/\lambda$



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

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.
- Benefit!!!! BTMR can block an error from propagating to other areas of the system.

Warning

- **With BTMR, not all I/O can be monitored.**
- **Usually need an additional detection signal to know when one of the systems are in failure.**
- **Need stop upon first system failure and correct system state.**

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

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

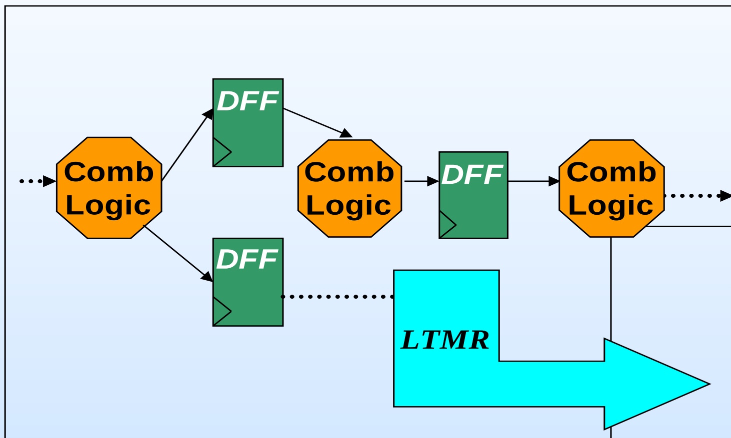
$$P(fs)_{functionalLogic} = P(fs)_{DFFSEU \rightarrow SEU} + P(fs)_{SET \rightarrow SEU}$$

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

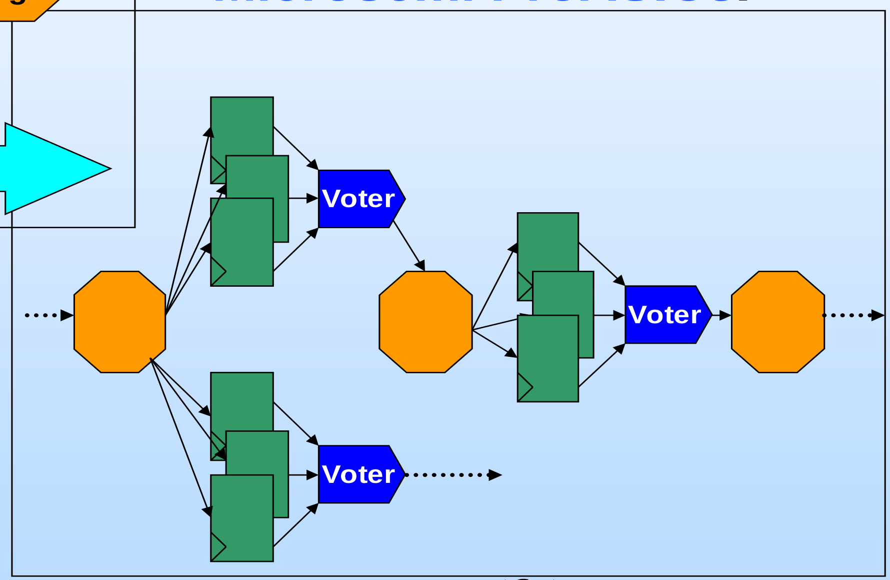
Probability that an SET in a CL gate will manifest as an error in the next system clock cycle

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.**

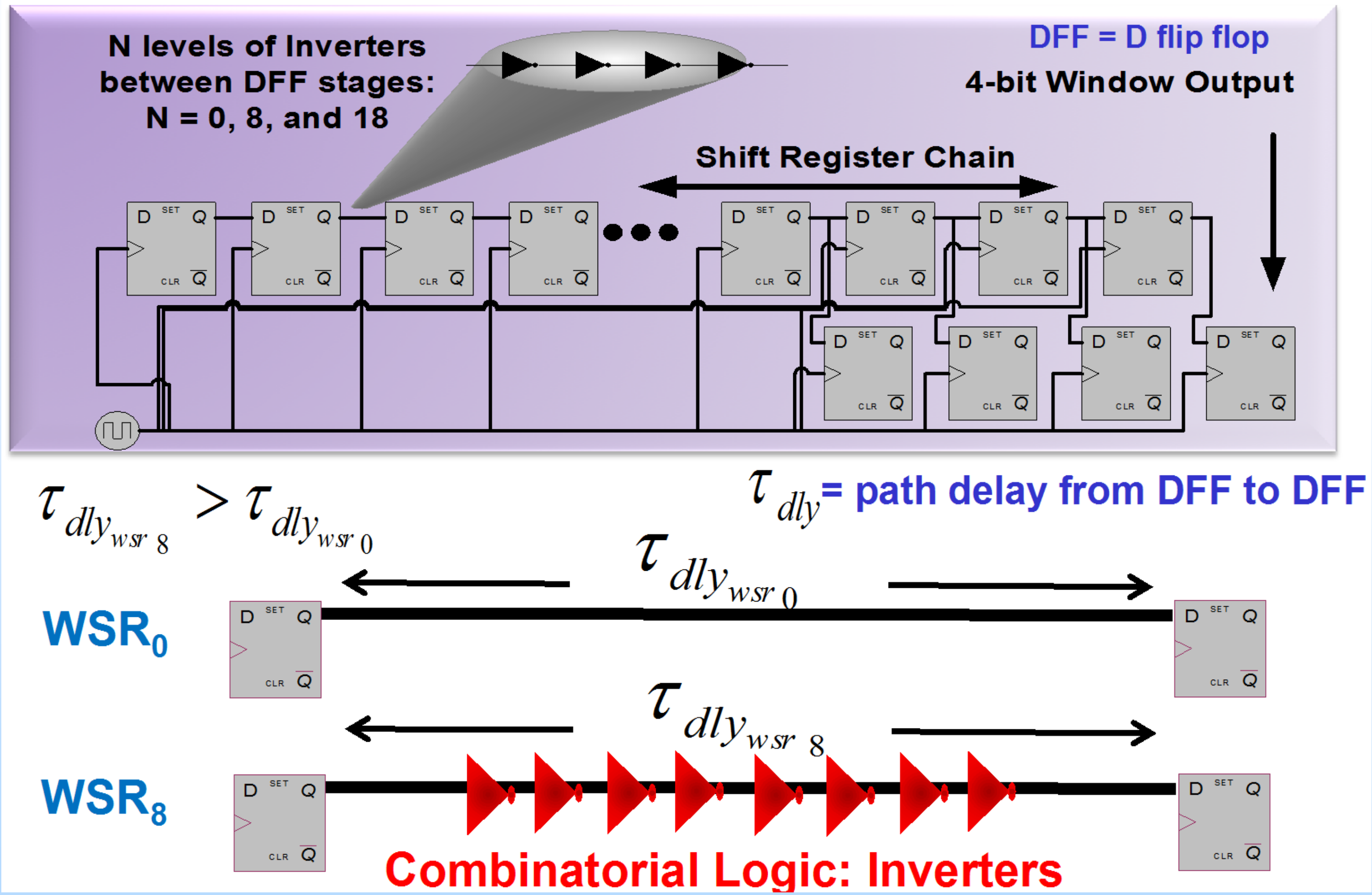


$$P(fs)_{error} \propto P_{configuration} + P(fs)_{functionalLogic} + P_{SEFI}$$

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

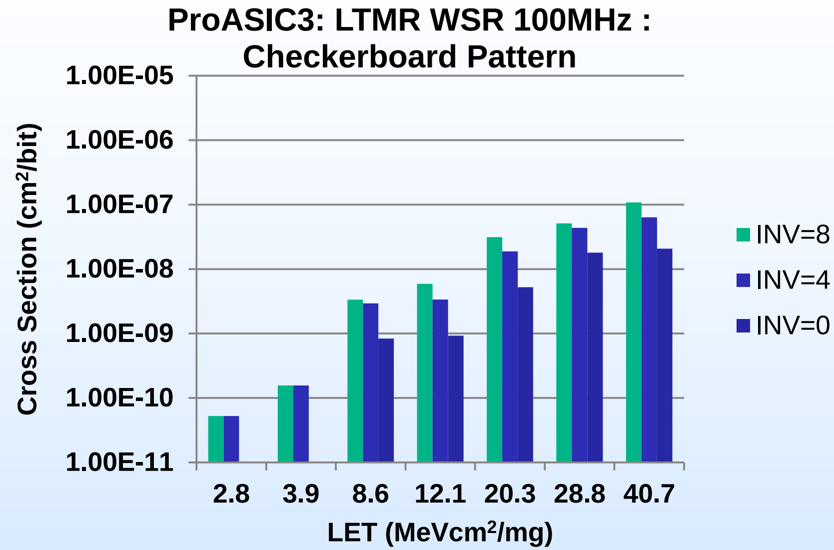
0

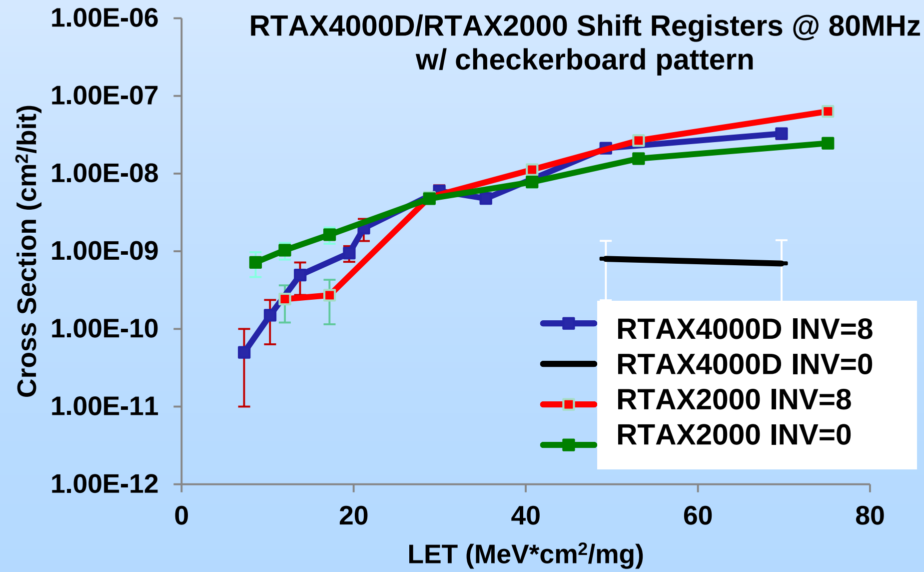
Windowed Shift Registers (WSRs):
NEPP Test Structure



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.



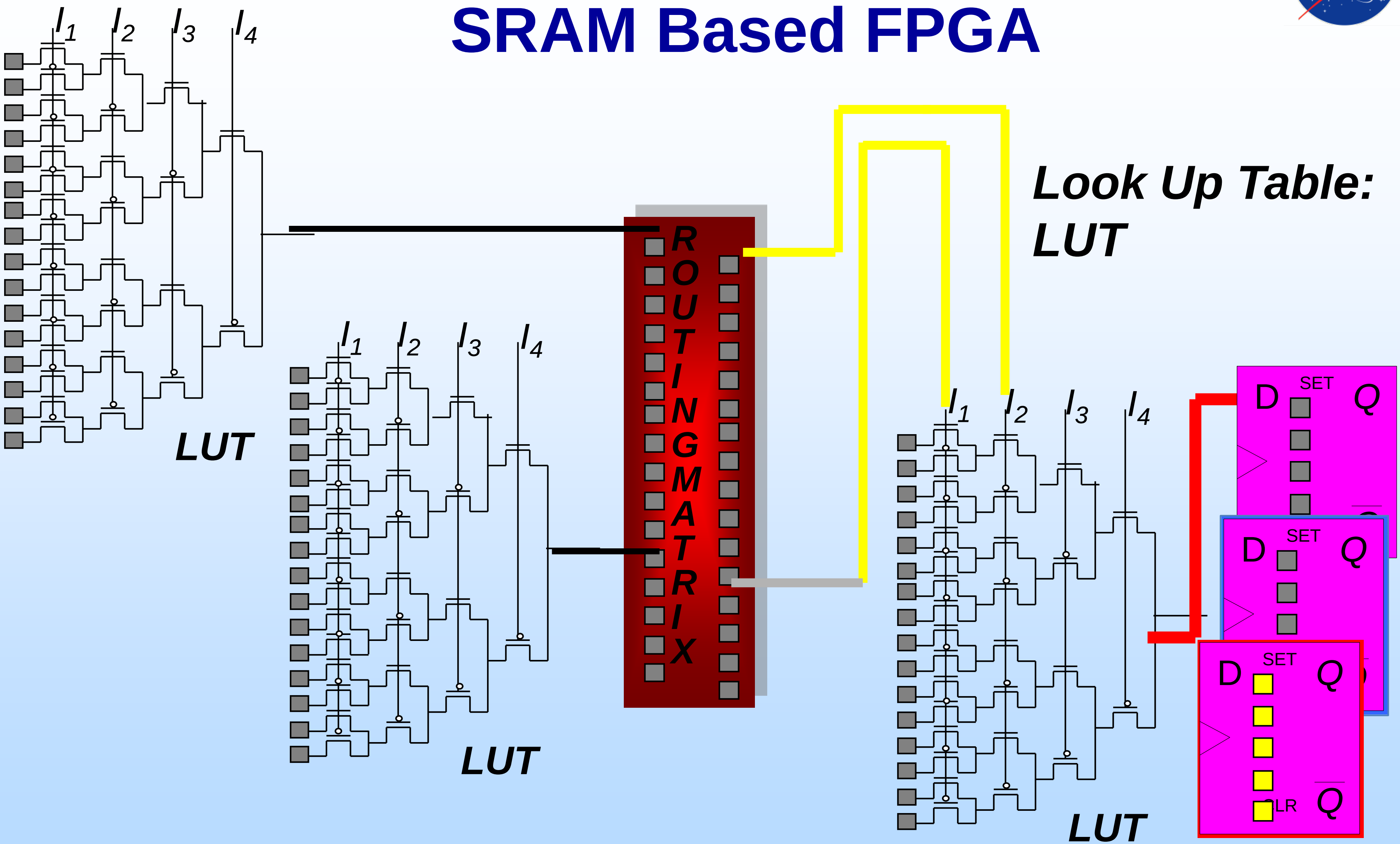


LET: linear energy transfer.

WSR: Test circuit…Windowed Shift Register.

INV: Inverters between WSR stages.

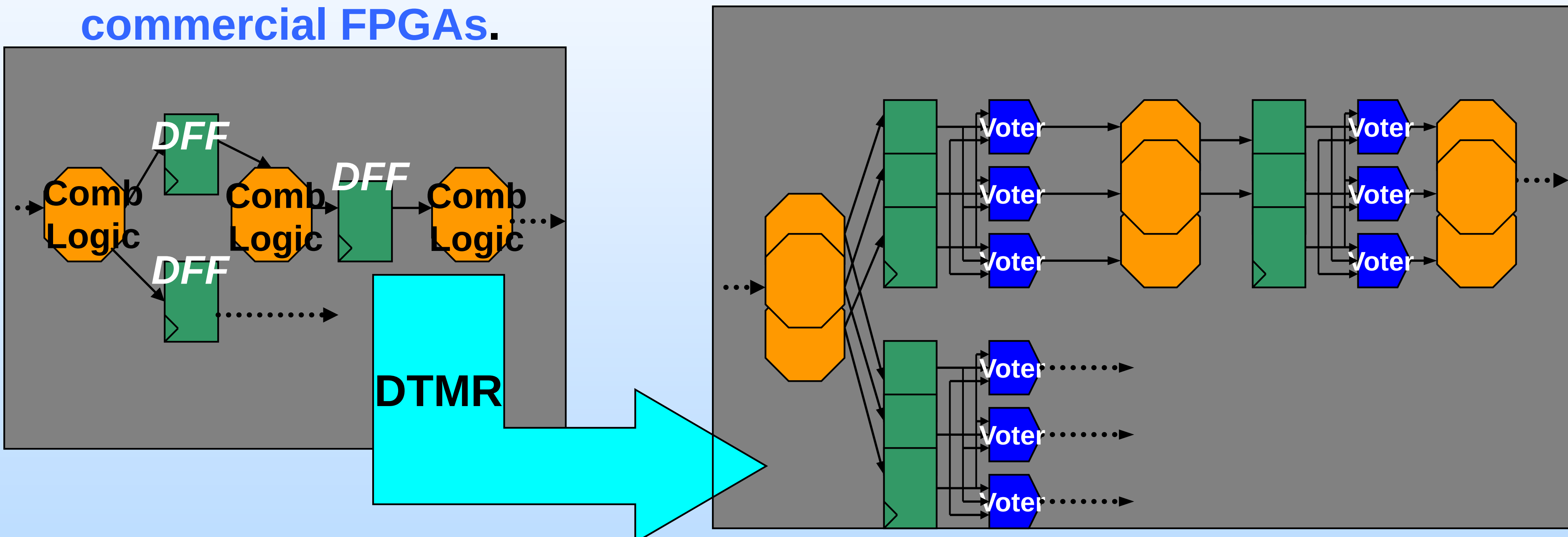
LTMR Should Not Be Used in An SRAM Based FPGA



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

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.



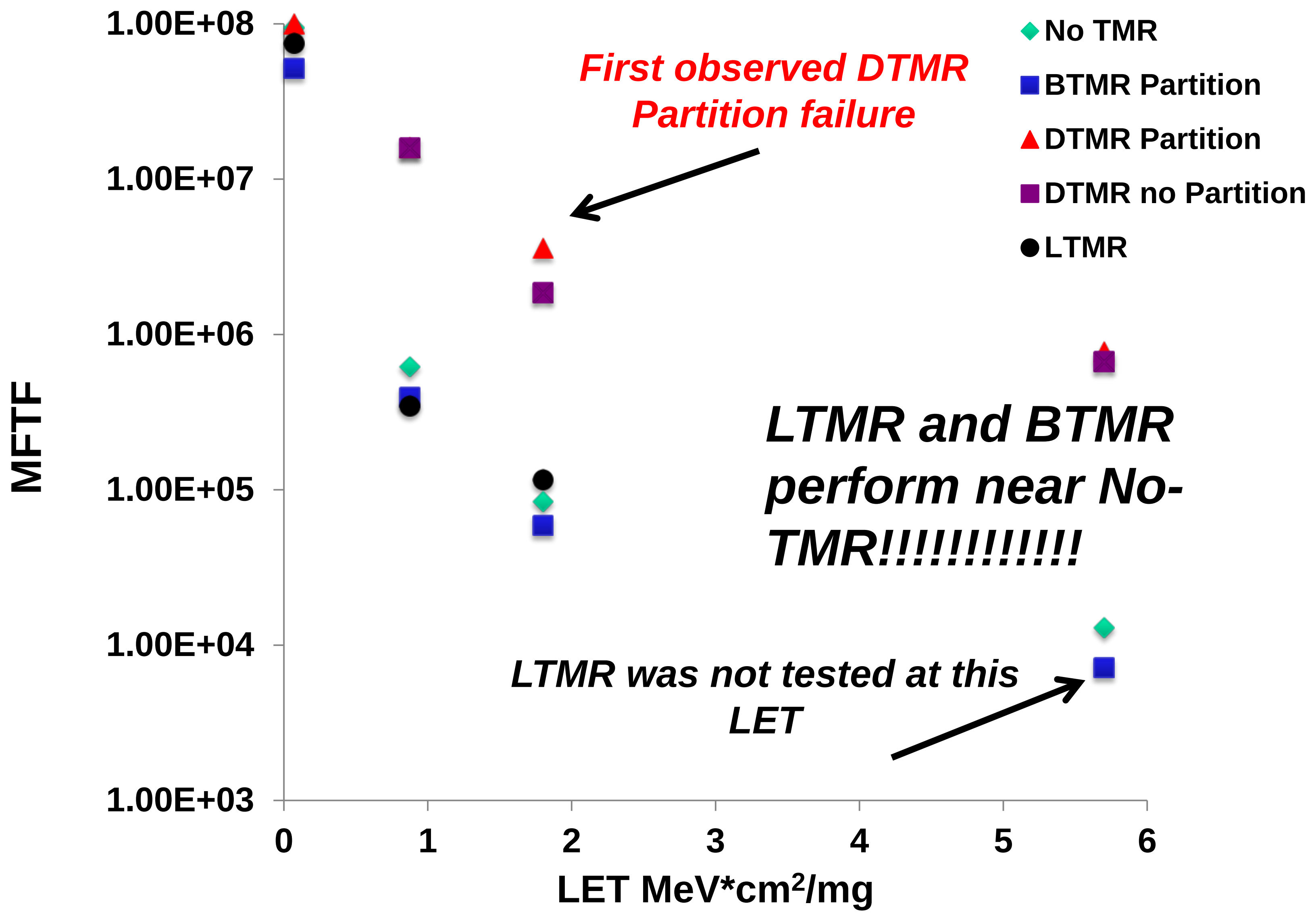
$$P(f_s)_{error} \propto P_{configuration} + P(f_s)_{functionalLogic} + P_{SEFI}$$

Low; Minimally Lowered

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

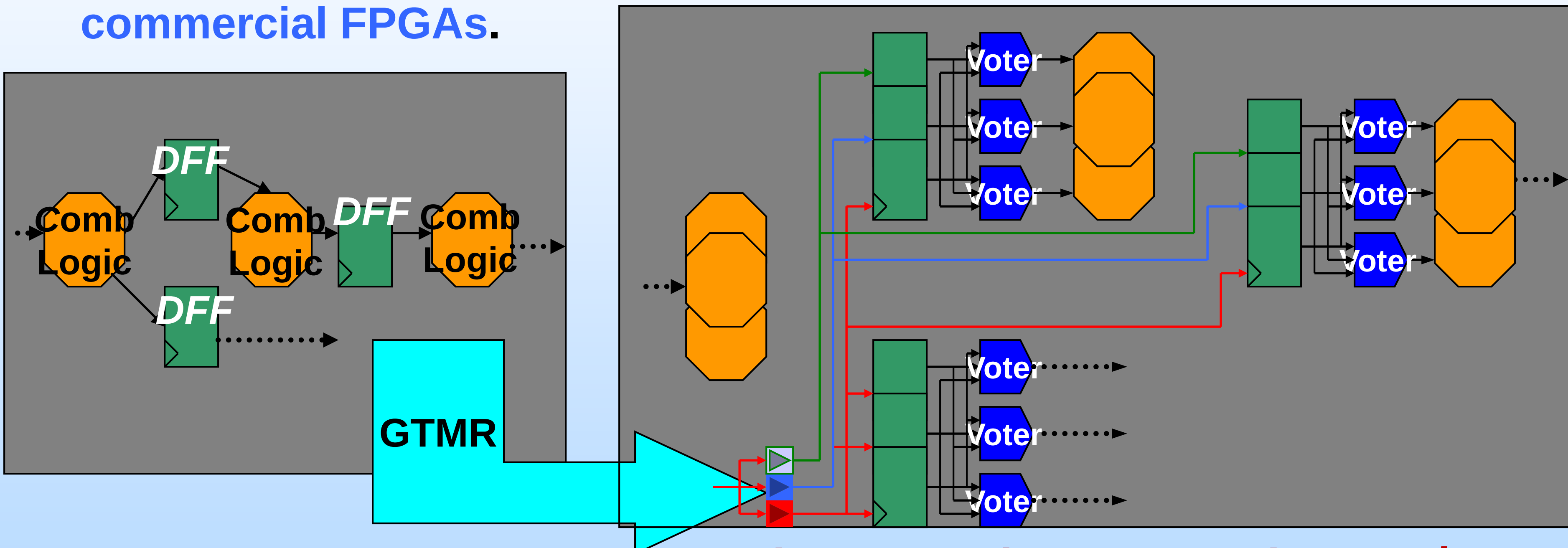
0; Low

Xilinx Kintex UltraScale Mitigation Study: 8-bit Counters



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.



Low Low Lowered

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

Low Low

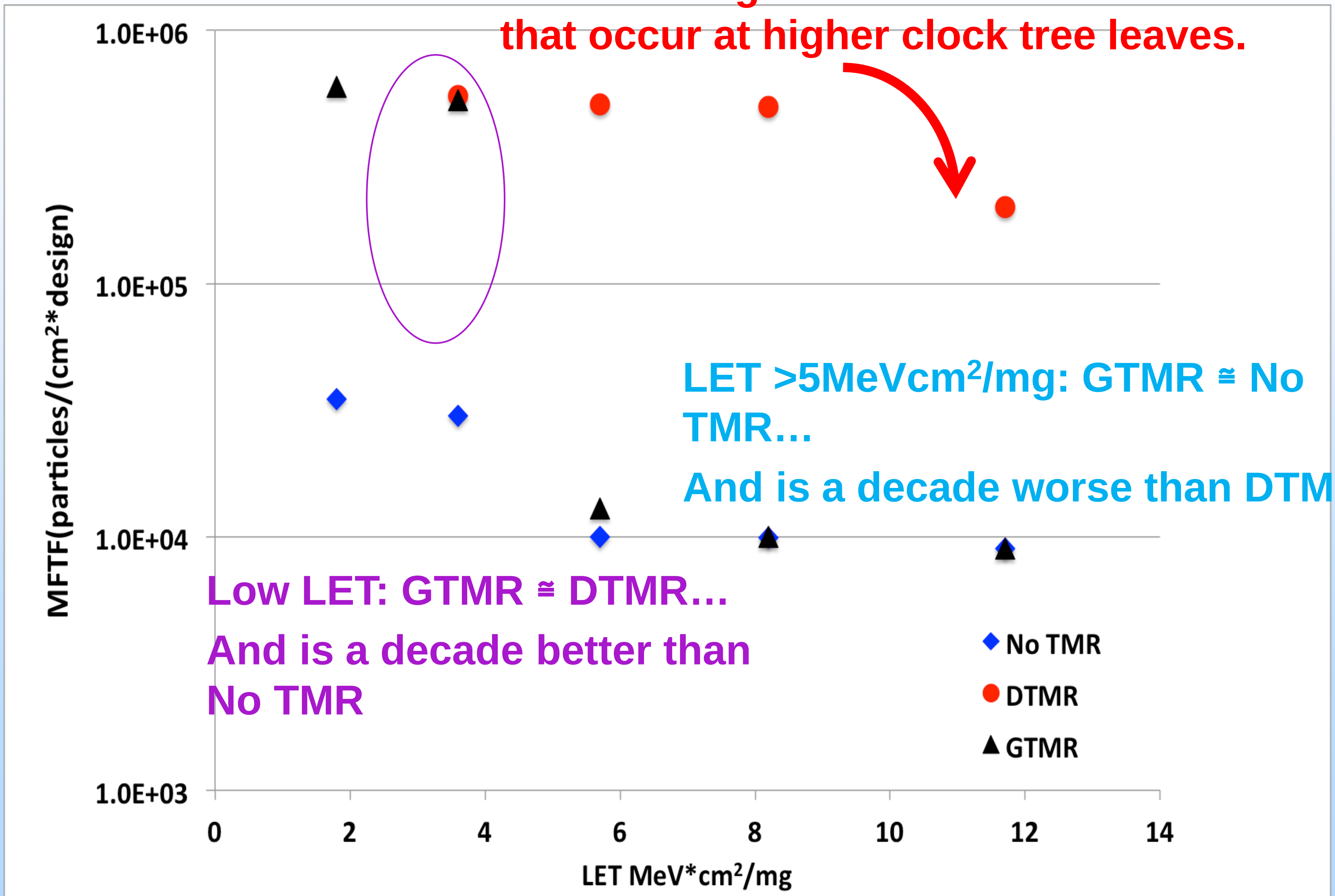
$$P(fs)_{DFFSEU \to SEU} + P(f_s)_{SET \to SEU}$$

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.**

Kintex-7 Counter Heavy-Ion Results: GTMR Does Not Perform Well – Clock Skew

DTMR strength decreased due to clock SETs that occur at higher clock tree leaves.



LET >5MeVcm^2/mg: GTMR ≅ No TMR…

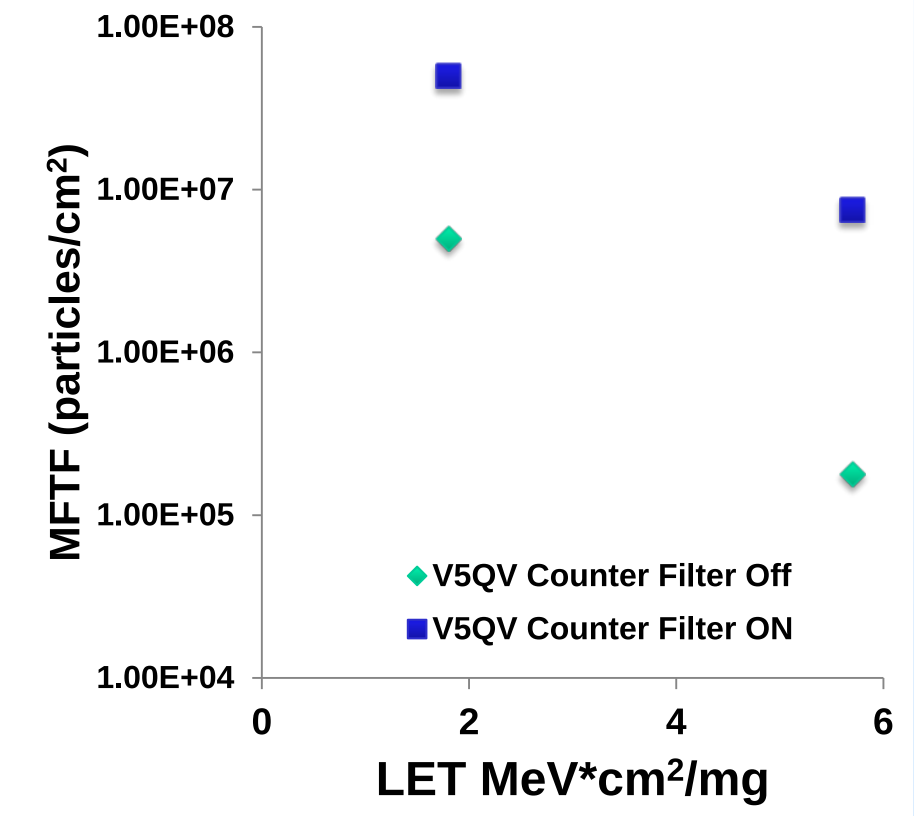
And is a decade worse than DTMR

Low LET: GTMR ≅ DTMR…

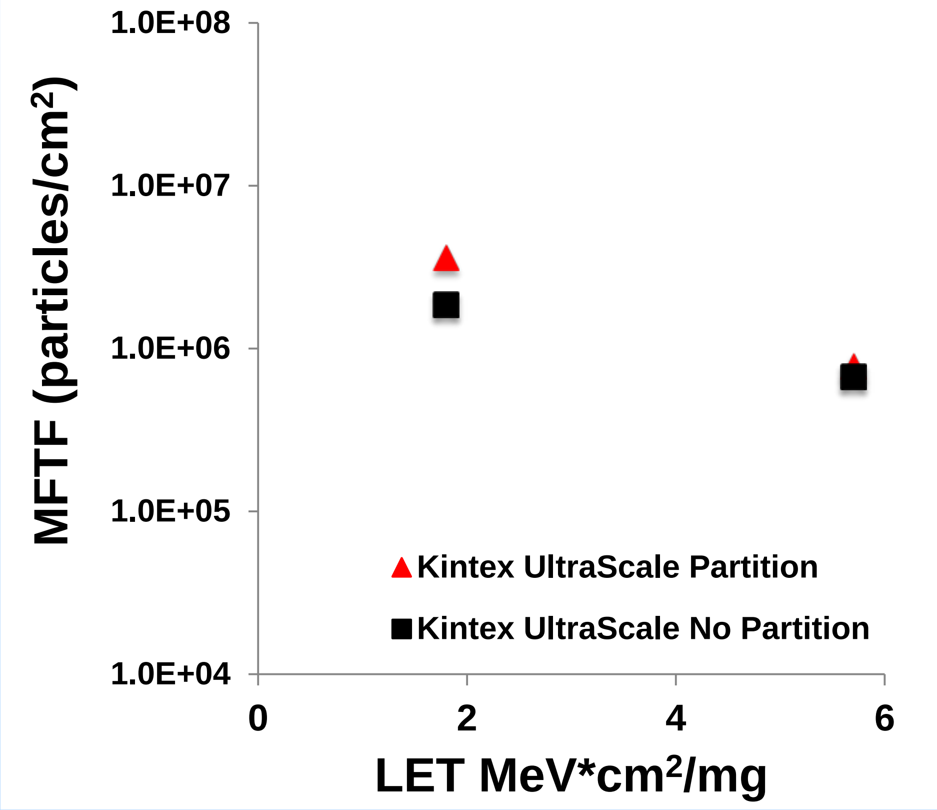
And is a decade better than No TMR

Comparison of V5QV and Kintex UltraScale with Mitigation

V5QV Counters



Kintex UltraScale DTMR Counters



DTMR inserted with Synopsys synthesis tool

Warning

- There are significant differences between TMR schemes. Select the correct type for your application and requirements.
- Do not use LTMR in a Xilinx Device!
- BTMR is a sufficient mitigation strategy if the required reliability window is relatively small as compared to MTTF of a non-redundant (non-mitigated) system.
- Clock skew with GTMR can reduce mitigation strength. Best to stay away.

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!

TMR Rules of Thumb

- **FPGAs with embedded mitigation do not usually require additional (user inserted) TMR.**
- **FPGAs with soft configuration will only benefit from DTMR or BTMR (in appropriate situations).**
- **FPGAs with hard configuration and no other embedded mitigation will benefit from local mitigation strategies.**
- **Most FPGAs cannot accommodate the clock skew between clock trees to properly implement GTMR.**

Some Thoughts

Concerns and Challenges of Today and Tomorrow for Mitigation Insertion (1)

- **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 commercial 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.**

Concerns and Challenges of Today and Tomorrow for Mitigation Insertion (2)

- **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.**
- **Tool availability… Getting better… IP Cores are still problematic.**
- **User's are not selecting the correct mitigation scheme for their target FPGA.**
- **Mitigation is too complex to fully verify.**

Warning

- **You should not mitigate failure mechanisms that have insignificant contribution to the overall failure rate:**
  - **This adds risk.**
  - **Slows down system.**
  - **Can provide a false sense of protection.**
  - **Gain is not significant.**

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

Summary

- For critical applications, mitigation might 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.