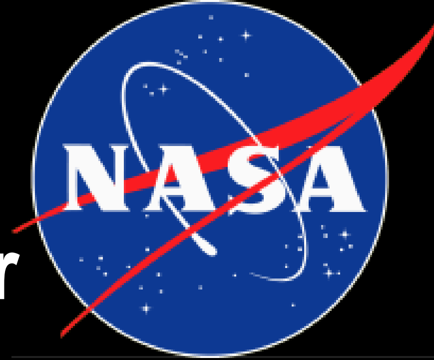
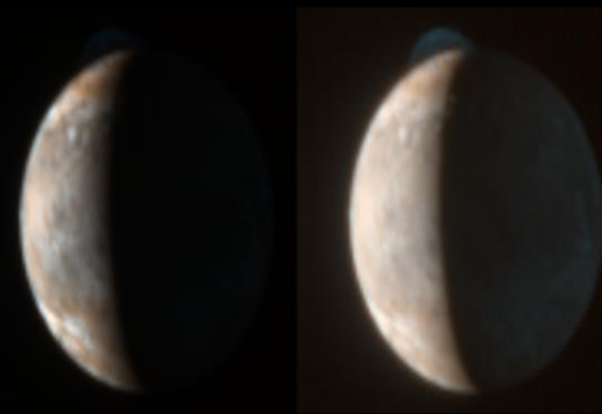
New Developments in FPGA SEUs and Fail-Safe Strategies from the NASA Goddard Space Flight Center Perspective





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

Melanie.D.Berg@NASA.gov

Deliverable to NASA Electronic Parts and Packaging (NEPP) Program to be published on nepp.nasa.gov.

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

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)
- 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)
- Gate Level Netlist (EDF, EDIF, GLN)
- Global triple modular redundancy (GTMR)
- Input – output (I/O)
- Linear energy transfer (LET)
- Local triple modular redundancy (LTMR)
- Look up table (LUT)
- Operational frequency (*fs*)
- Power on reset (POR)
- Place and Route (PR)
- Probability of flip-flop upset (P_{DFFSEU})
- Probability of logic masking (P_{logic})
- Probability of transient generation (P_{gen})
- 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)
- Transient width (τ_{width})
- Universal Serial Bus (USB)
- Virtex-5QV (V5QV)
- Windowed Shift Register (WSR)

NASA

Before we get started… here are some things to keep in mind during this VERY LONG presentation…

Single Event Effects (SEEs) and Common Terminology

- Single Event Latch Up (SEL): Device latches in high current state.
- Single Event Burnout (SEB): Device draws high current and burns out.
- Single Event Gate Rupture: (SEGR): Gate destroyed typically in power MOSFETs.
- Single Event Transient (SET): current spike due to ionization. Dissipates through bulk.
- Single Event Upset (SEU): transient is caught by a memory element. Causes an incorrect state. SETs are categorized under SEUs.
- Single Event Functional Interrupt (SEFI) - upset disrupts function.

SEUs and FPGAs

- Ionizing particles cause upsets (SEUs) in FPGAs.
- Each FPGA type has different SEU error signatures:
  - 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 FPGA and its surrounding circuitry susceptibility,
  - Implemented fail-safe strategies,
  - Reliable design practices,
  - Radiation environment, and
  - Trade space and decided risk

SEUs and FPGA Variations

- FPGA susceptibilities (error signatures) vary per FPGA type.
- How does a project manage and protect against FPGA susceptibilities? (mitigation schemes will change based on FPGA type).
- The most efficient solution will be based on understanding:
  - SEE theory,
  - FPGA SEE susceptibility (per FPGA type),
  - Proven mitigation strategies per FPGA type,
  - Validation and verification of implemented mitigation strategies, and
  - Limitations of tools and/or mitigation schemes.

Differentiating Fail-Safe Strategies:

- Detection:
  - Watchdog (state or logic monitoring).
  - Simplistic Checking … Complex Decoding.
  - Action (correction or recovery).
- Masking (does not mean correction):
  - Not letting an error propagate to other logic.
  - 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

- Just adding redundancy to a system is not enough to assume that the system is well protected.
- 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?

Agenda

- **Section I: General FPGA Description and Design Process.**
- **Section II: Single Event Effects (SEEs) in Digital Logic.**
- **Section III: Application of the NASA Goddard Radiation Effects and Analysis Group (REAG) FPGA SEU Model.**
- **Section IV: Reducing System Error: Common Mitigation Techniques.**
- **Section V: When Your Mitigation Fails.**
- **Section VI: Xilinx Virtex Series and Mitigation.**

FPGAs in Space Systems

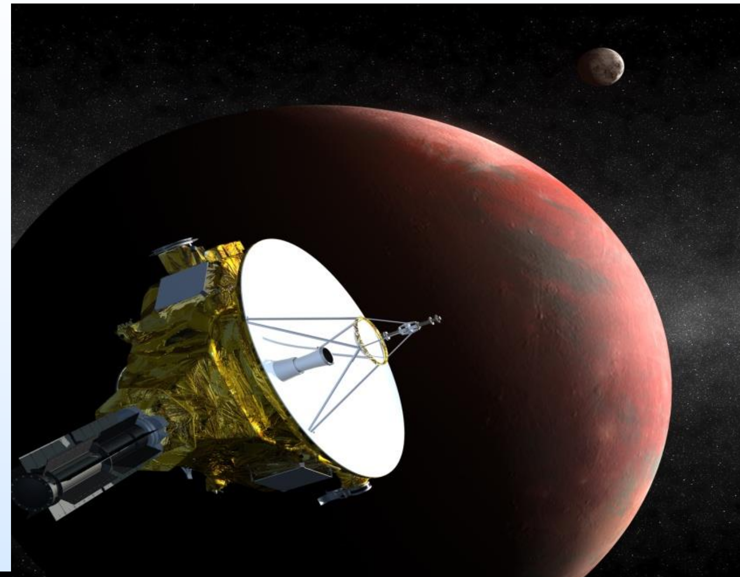
- FPGAs are used as:
  - Controllers,
  - Processors,
  - Interface Adaptation and Control.
- Based off of system requirements, the user implements a specified design (function) in an FPGA.

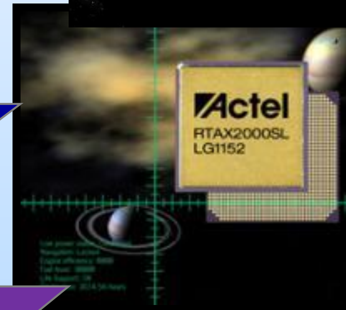
User obtains data sheets (FPGAs and peripherals)

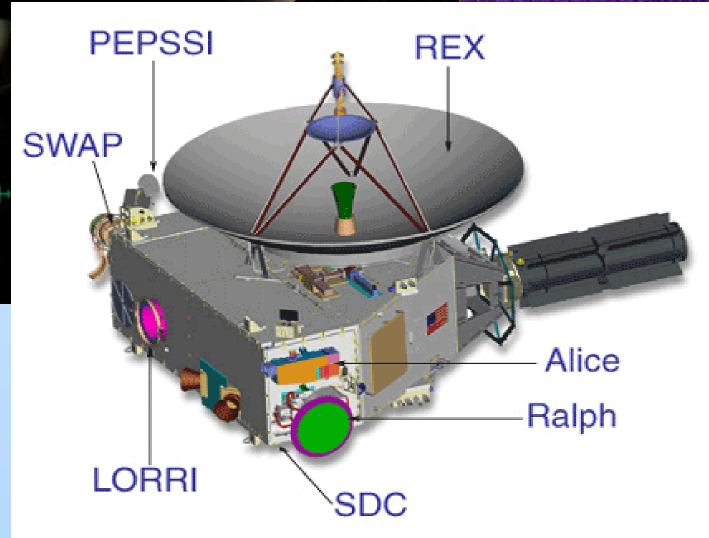
User decides speed(s) of operation

User describes hardware design (HDL)

User maps design logic gates to FPGA Fabric via manufacturer's configuration management tools







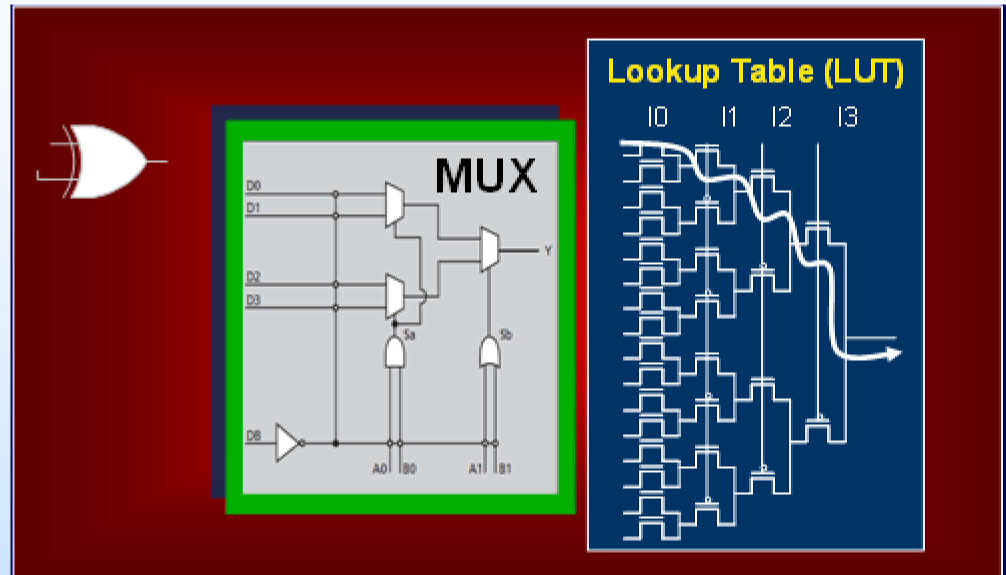
General FPGA Architecture: Fabric Containing Customizable Preexisting Logic…User Building Blocks

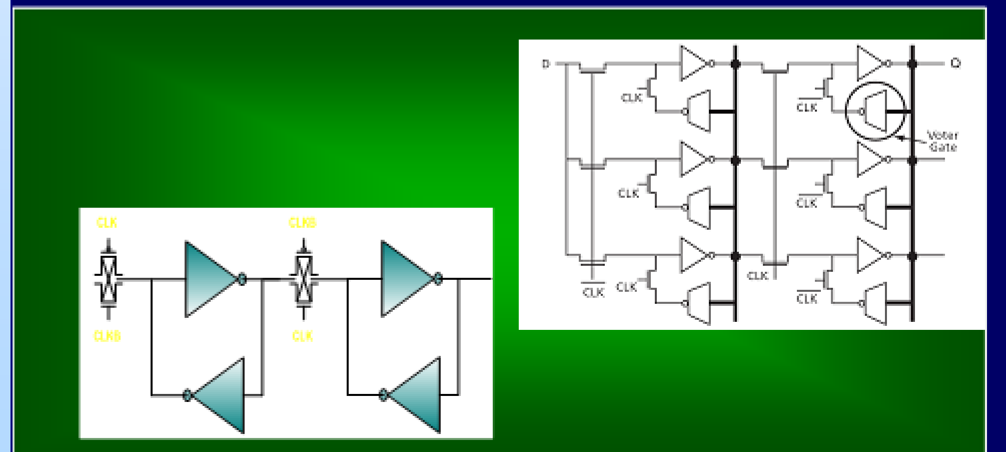
How Do FPGA's Differ?

- **Manufacturer Architecture (not all are listed):**
  - Configuration,
  - User building blocks (combinatorial logic cells, sequential logic cells),
  - Routing,
  - Clock structures,
  - Embedded mitigation, and
  - Embedded intellectual property (IP); e.g., memories and processors.
- **Manufacturer tool Environment:**
  - Synthesis,
  - Place and Route, and
  - Configuration management output.
- **SEU error signatures vary mostly because of manufacturer architecture implementation.**

FPGA Component Libraries: Basic Designer Building Blocks (They Differ per FPGA Type)

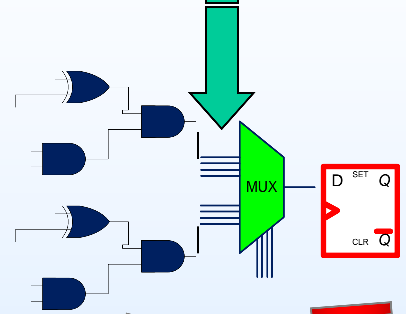
- **Combinatorial logic (CL) blocks**
  - **Vary in complexity.**
  - **Vary in I/O.**
- **Sequential Memory blocks (DFF)**
  - **Uses global Clocks.**
  - **Uses global Resets.**
  - **May have mitigation.**





User Maps the Design Logic into FPGA Preexisting Logic

Hardware design language (HDL)



HDL

Route

Place

LOGIC

LOGIC

LOGIC

LOGIC

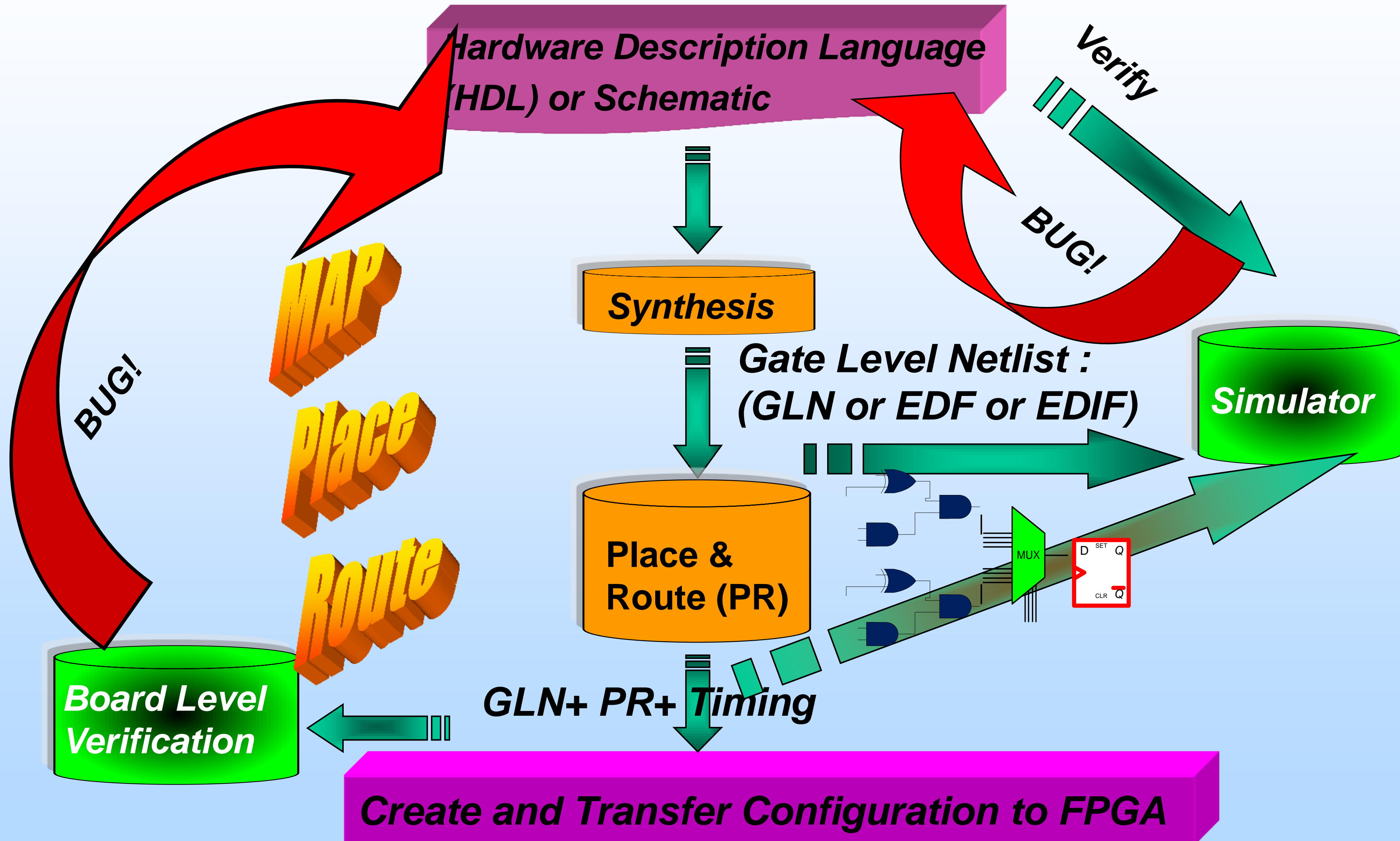
MAP

INTO FPGA LIBRARY

Combinatorial FPGA Equivalent Block

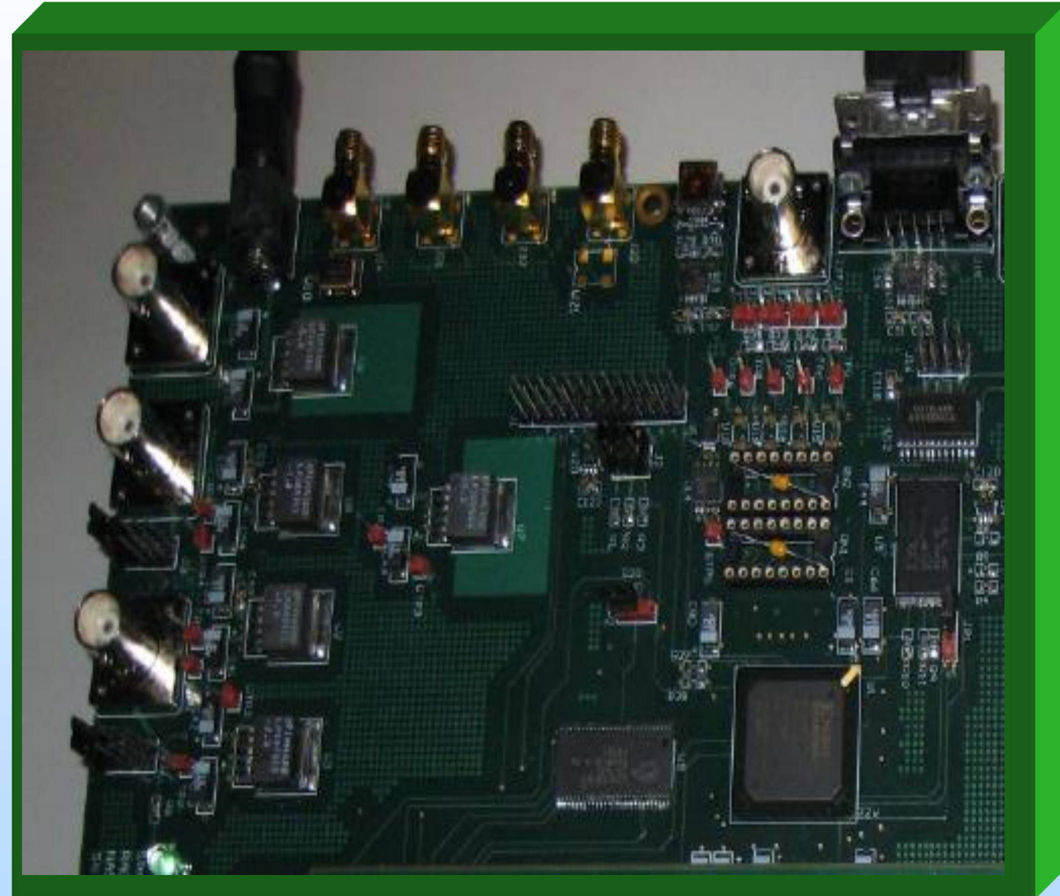
DFF FPGA Equivalent Block

A Closer Look at The FPGA Design Process from The User's Perspective



FPGA Inserted into a System…FPGA Inserted into a Board

- **Board contains:**
  - Oscillators (Clocks),
  - Resistors & Capacitors, for noise reduction, and
  - Peripheral devices.
- **FPGA must talk to other devices:**
  - Provide control signals or data, or
  - Perform data capture.
- **Designer uses peripheral device data sheets to design the control and capture logic.**

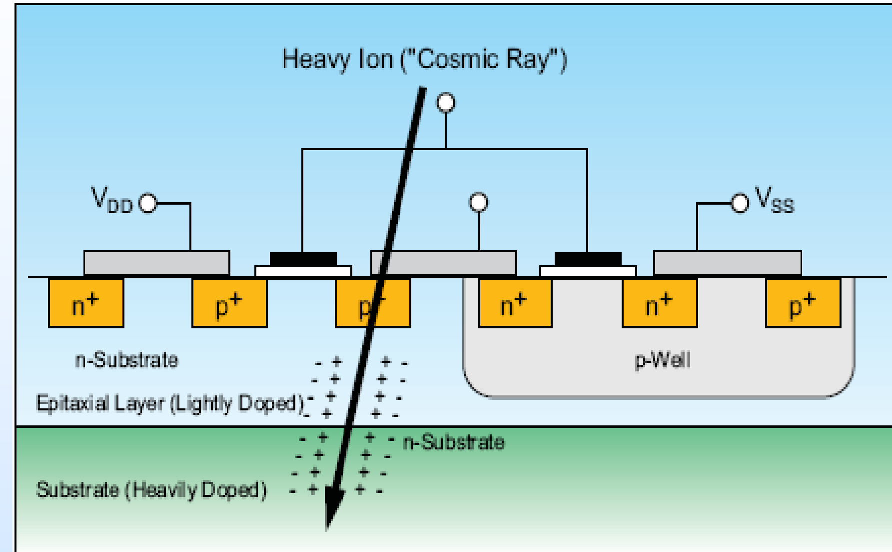


Agenda

- Section I: General FPGA Description and Design Process.
- **Section II: Single Event Effects (SEEs) in Digital Logic.**
- Section III: Application of the NASA Goddard Radiation Effects and Analysis Group (REAG) FPGA SEU Model.
- Section IV: Reducing System Error: Common Mitigation Techniques.
- Section V: When Your Mitigation Fails.
- Section VI: Xilinx Virtex Series and Mitigation.

Device Penetration of Heavy Ions and Linear Energy Transfer (LET)

- LET characterizes the deposition of charged particles.
- Based on Average energy loss per unit path length (stopping power).
- Mass is used to normalize LET to the target material.



Average energy deposited per unit path length

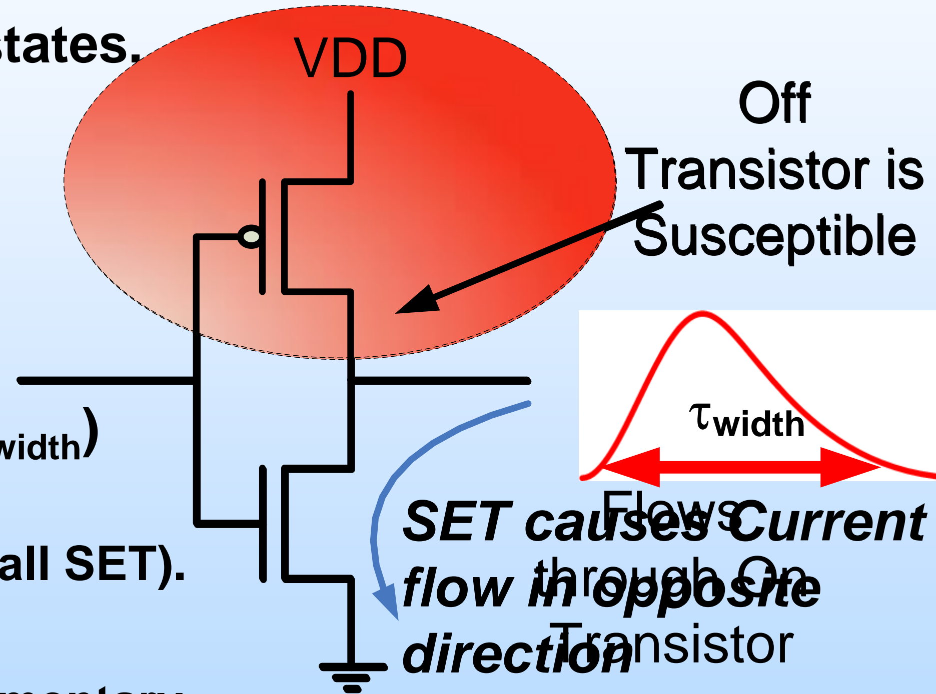
$$LET = \frac{1}{\rho}\frac{dE}{dx} \; ; \; MeV \frac{cm^2}{mg}$$

Density of target material

Units

SET Generation

- SET generation occurs due to an "off" gate turning "on".
- For a CMOS SET: there is a push-pull between the on gate and the off gate $Q_{coll.}$
- SETs have significant metastable states.



- SET has an amplitude and width (τ_{width}) based on:
  - Amount of Q_{coll} (i.e. small LET →small SET).
  - The capacitance of the gate's load.
  - The strength (current) of its complimentary "ON" gate.
  - The dissipation strength of the process.
- **Captured SET is an SEU.**

Characterizing SEUs: Radiation Testing and SEU Cross Sections

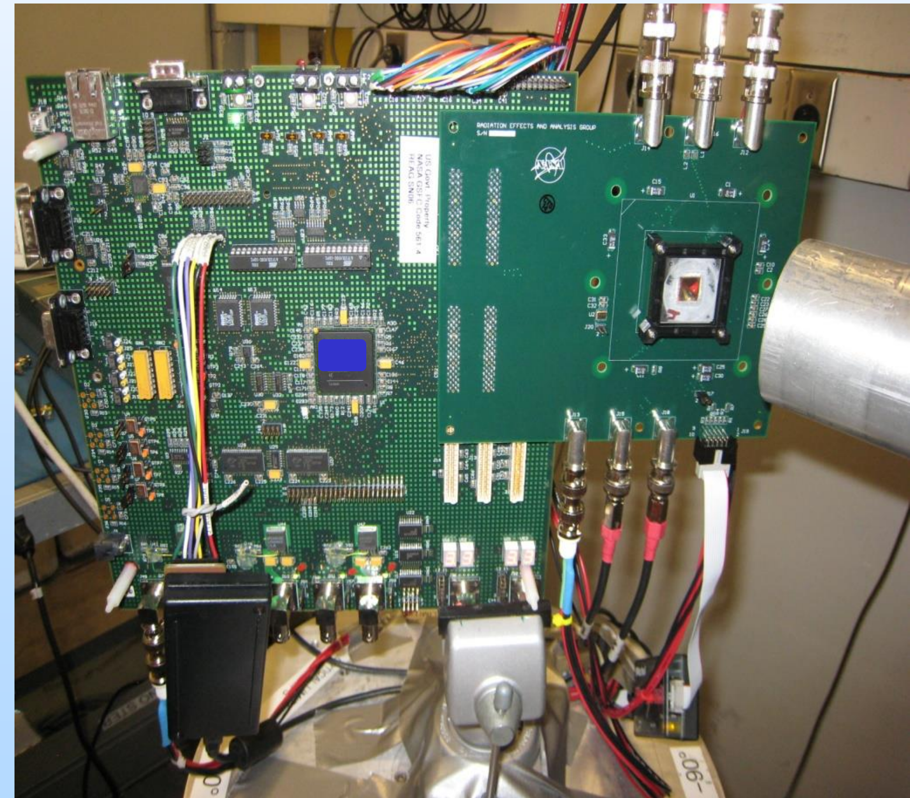
SEU Cross Sections (σ_{seu}) characterize how many upsets will occur based on ionizing particle exposure.

$$\sigma_{seu} = \frac{\#errors}{fluence}$$

Terminology:

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

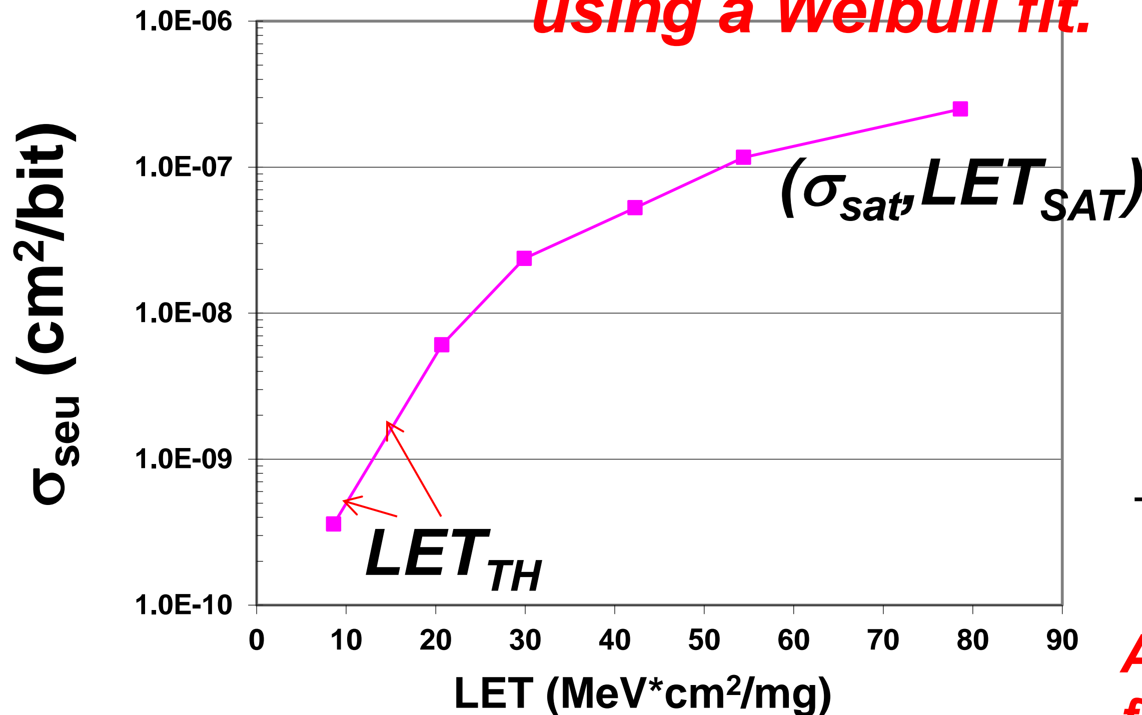
σ_{seu} is calculated at several LET values (particle spectrum).



Characterizing SEUs: LET vs. Error Cross Section Graph and How They Relate to Error Rates

$$\sigma_{seu} = \frac{\#errors}{fluence}$$

dE/dt (error rate) is calculated by integrating σ_{SEU} over the LET spectrum using a Weibull fit.



GEO Upset Rate:

$$\frac{dE}{dt} \approx \frac{400 * \sigma_{sat}}{LET_{th}^{\ 2}}$$

After Ed Petterson's figure of merit.

SEU Cross Sections and Error Rates – How We Apply Them to FPGAs

- A goal of SEU testing is to provide error rate (*dE(fs)/dt*) predictions to critical missions.
- $dE(f_s)/dt$ for FPGA and ASIC devices are calculated using :

System upset rate — SEU bit upset — Number of used flip-flops DFFs

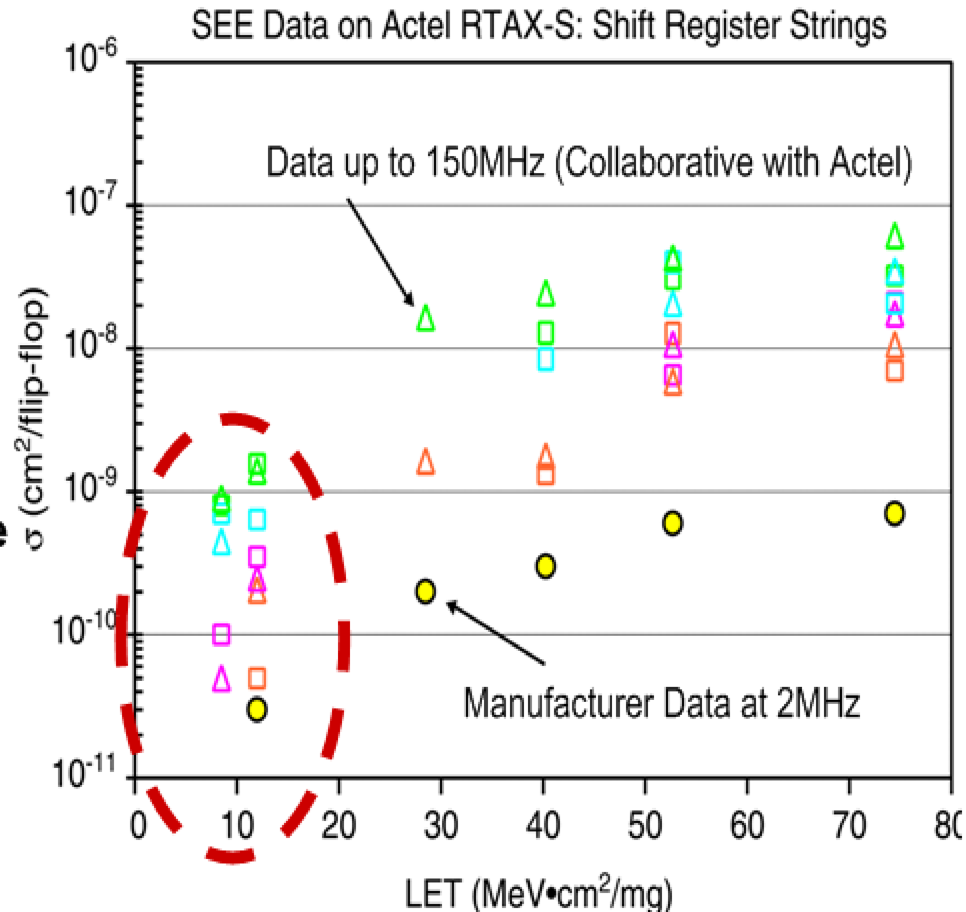
$$\frac{dE(fs)}{dt} < \frac{dE_{bit}(fs)}{dt} * (\#UsedDFFs)$$

- LET_{th} and LET_{sat} are significant factors for determining $dE_{bit}(f_s)/dt$. Hence, proper radiation testing is essential. Testing should take into account:
  - Frequency of operation,
  - Design Complexity,
  - Observability (upset visibility), and
  - Particle flux and fluence.

SEU Information: Manufacturer Datasheet Example

RTAX-S/SL RadTolerant FPGAs

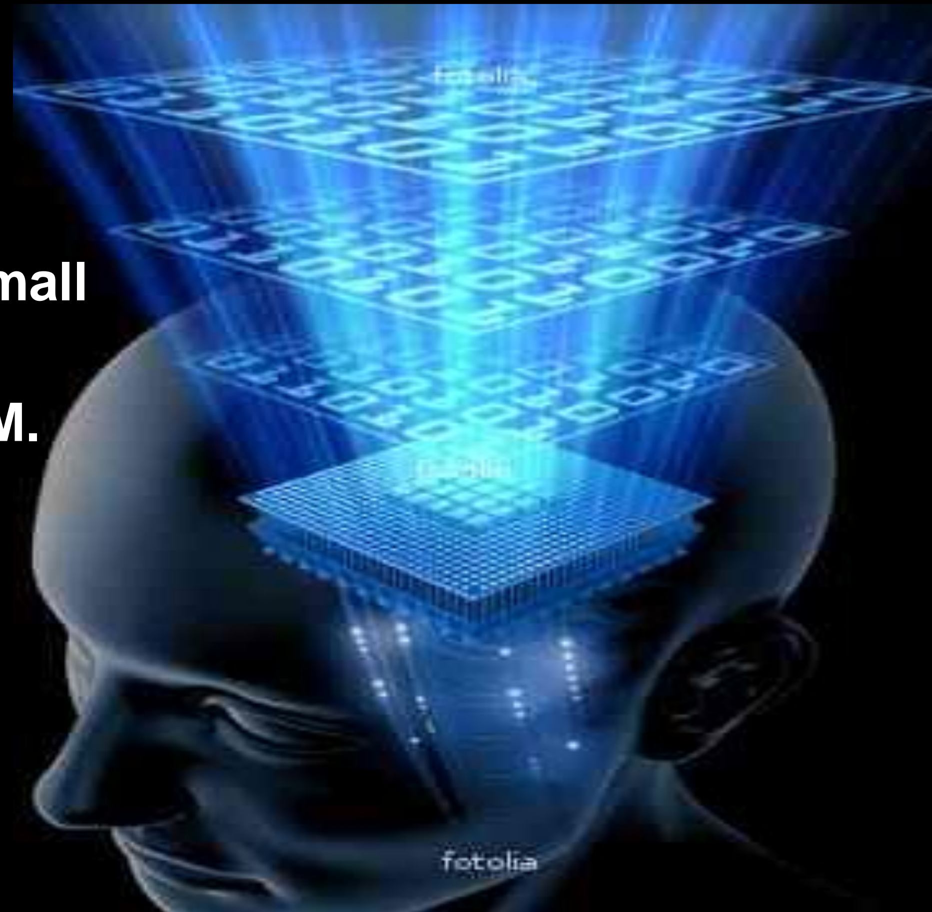
- SEU-Hardened Registers Eliminate the Need for Triple-Module Redundancy (TMR)
- Immune to Single-Event Upsets (SEU) to LET_{th} >37Mev-cm^2/mg
- SEU Rate < 10^{-10} Errors/bit-Day is Worst-Case



Radiation Data is always changing … best to keep yourself updated: http://radhome.gsfc.nasa.gov/

Key Points: SETs, SEUs and the Radiation Environment

- Heavy ions (HI) – Direct Ionization:
  - As LET increases, SET increases (τ_{width} and amplitude)… hence there is an increase in ionizing susceptibility.
  - HI SETs are significant upsets in data path CMOS (direct ionization).
- Protons – Indirect Ionization:
  - Secondary effects cause very small SETs in data path CMOS.
  - SEUs can be significant in SRAM.
- Keep up with ever changing radiation data.



The best designer in the world cannot mitigate against faults without understanding where, how, and when they are generated.

The following slides are applicable to FPGA synchronous designs:

FPGA SEU Susceptibility

- **FPGA SEUs or SETs can occur in:**
  - **Configuration,**
  - **Combinatorial Logic (CL)… including global routes or control,**
  - **Sequential Logic,**
  - **Memory Cells, or**
  - **Hidden logic (SEFI).**

Every Device has different Error Responses – We must understand the differences and design (or plan) accordingly

FPGA Structure Categorization as Defined by NASA Goddard REAG:

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

Design σ_{SEU}

Configuration σ_{SEU}

Functional logic σ_{SEU}

SEFI σ_{SEU}

Sequential and Combinatorial logic (CL) in data path

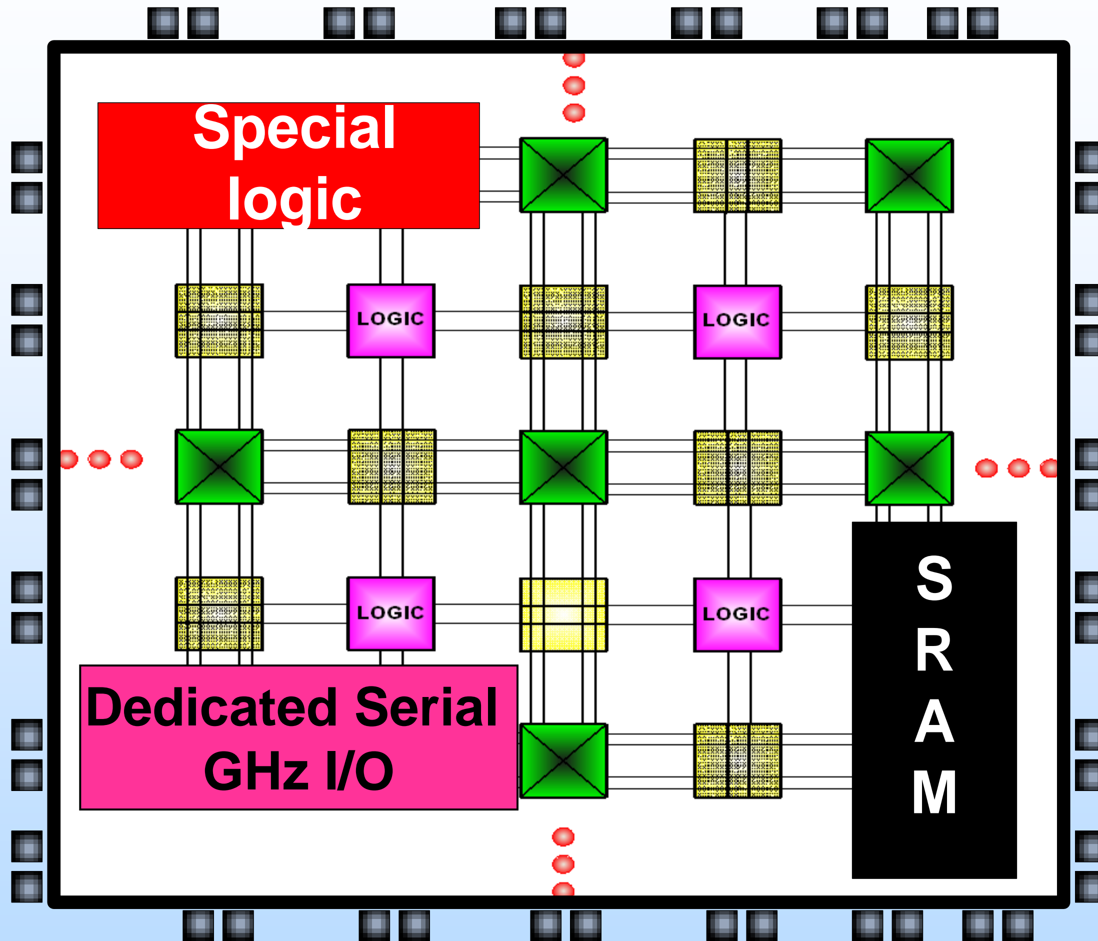
Global Routes and Hidden Logic

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

Configuration

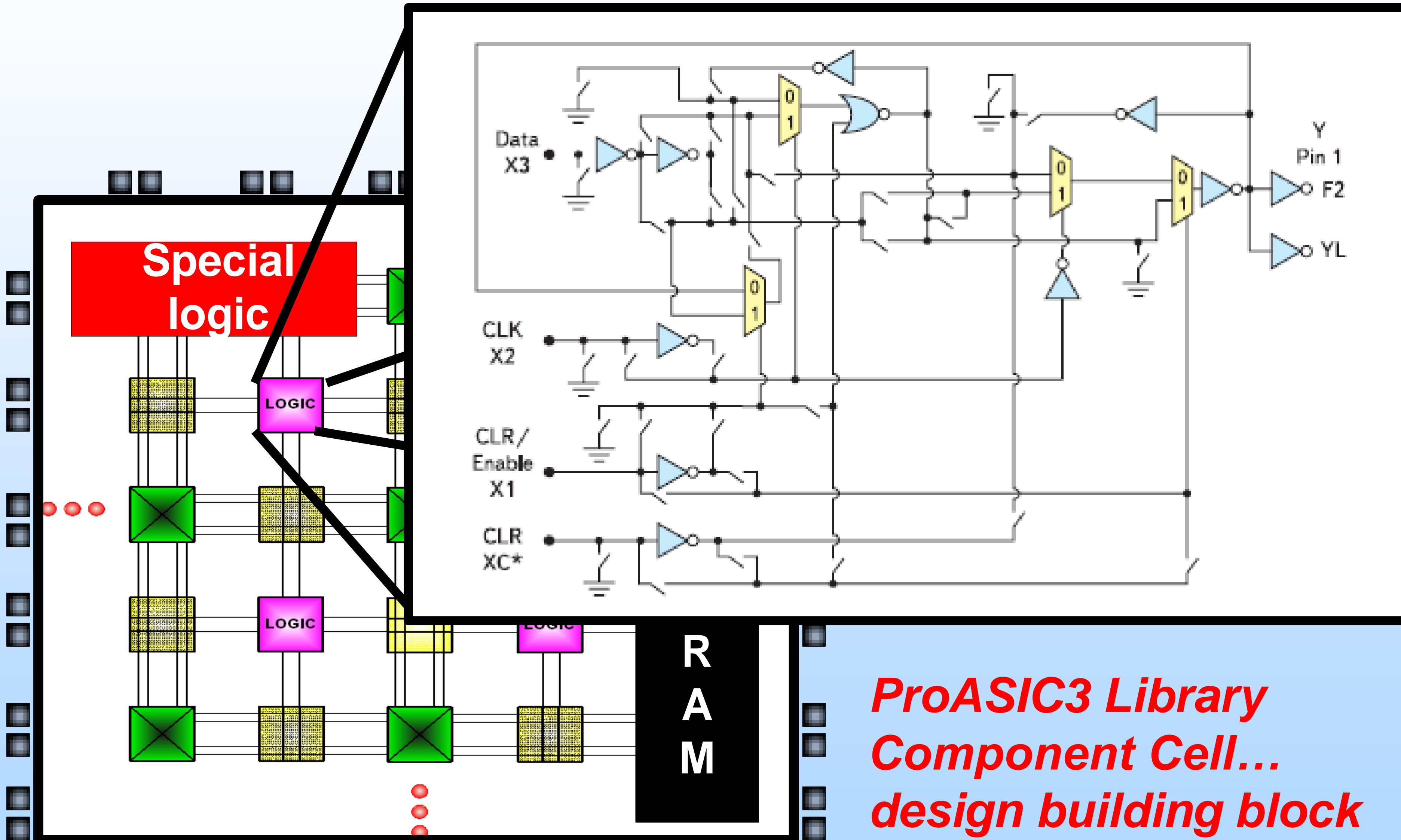
$$P(fs)_{error} \propto P_{Configuration} + P_{functionalLogic} + P_{SEFI}$$

Field Programmable Gate Array (FPGA) FABRIC – System On A Chip (SOC)



User creates a design by configuring pre-existing logic blocks and routes.

A Closer Look at an FPGA Logic Cell: Microsemi ProASIC3

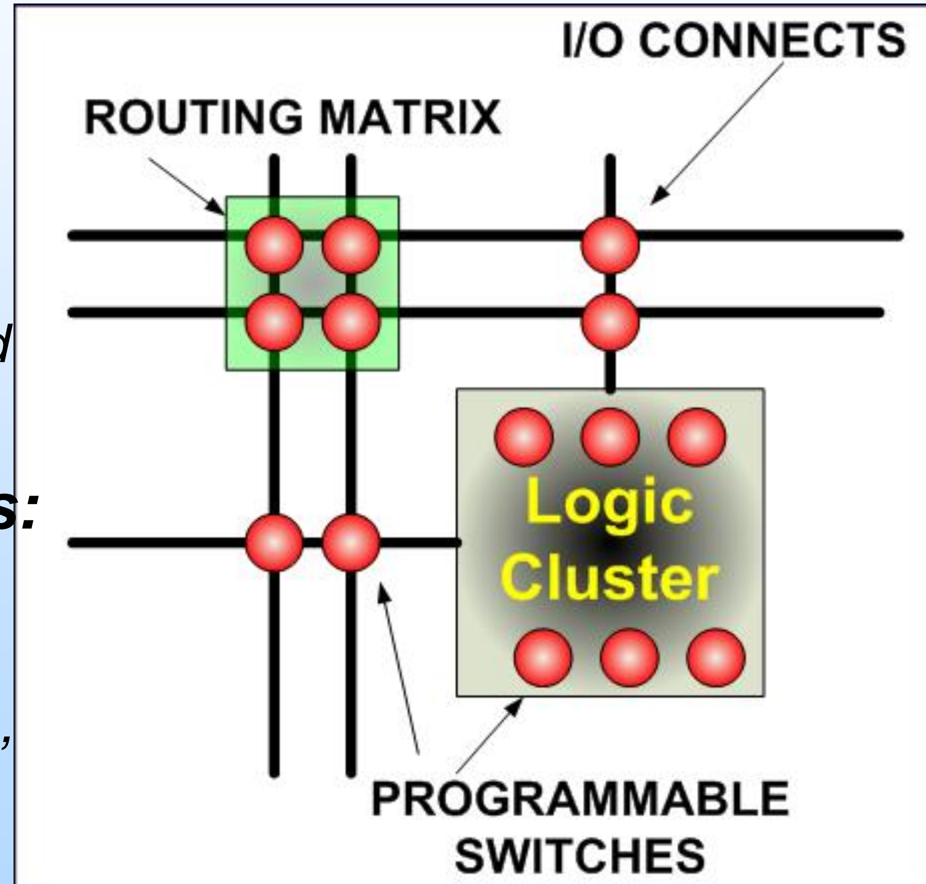


ProASIC3 Library Component Cell… design building block

FPGA Configuration

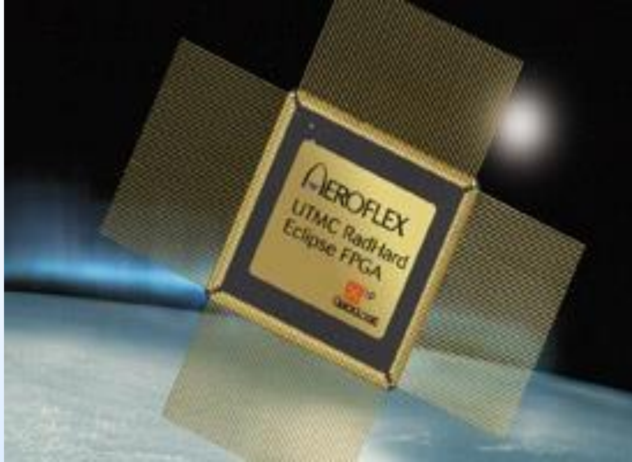
HDL → **FPGA MAPPING** → **Configuration**

- ***Configuration Defines:*** *Arrangement of pre-existing logic via programmable switches.*
  - *Functionality (logic cluster) and*
  - *Connectivity (routes)*
- ***Programming Switch Types:***
  - ***Antifuse:*** *One time Programmable (OTP),*
  - ***SRAM:*** *Reprogrammable (RP), or*
  - ***Flash:*** *Reprogrammable (RP).*



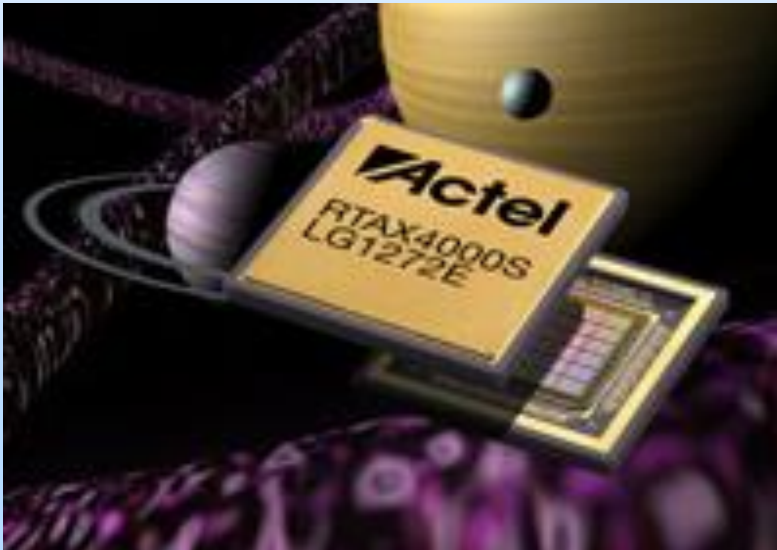
FPGA Categorized by Configuration Types

Antifuse



SRAM

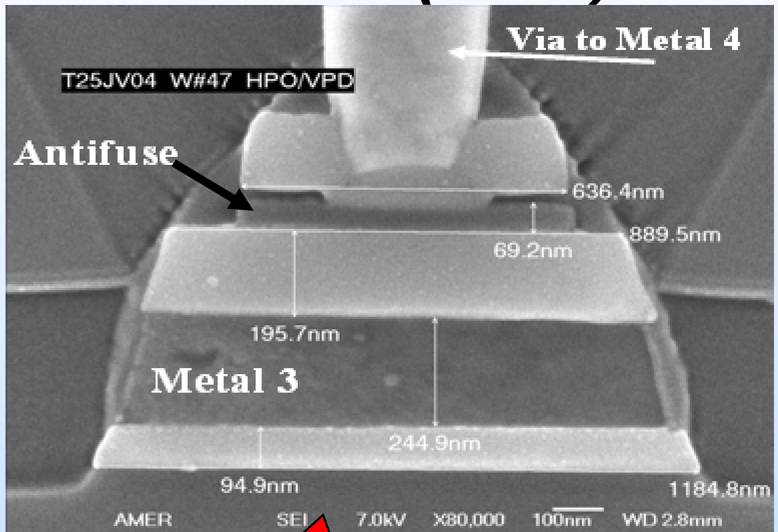
Antifuse

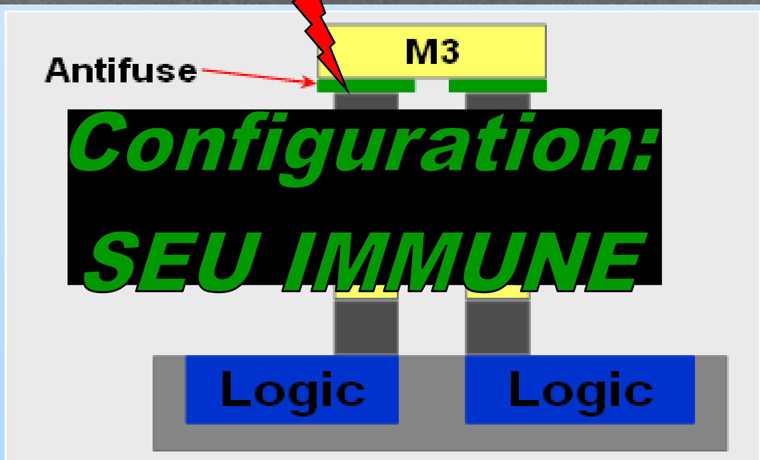


Flash

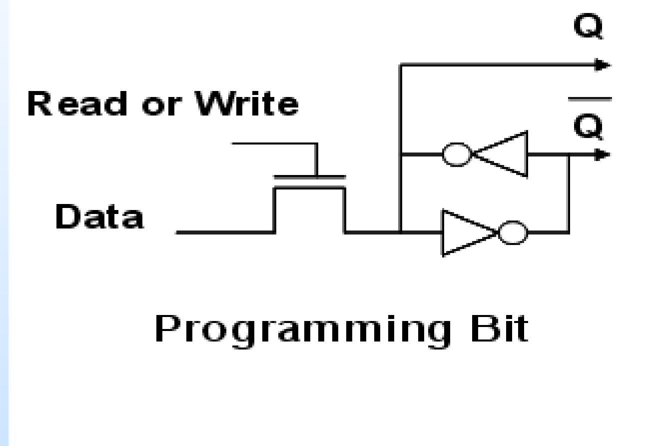
Programmable Switch Implementation and SEU Susceptibility

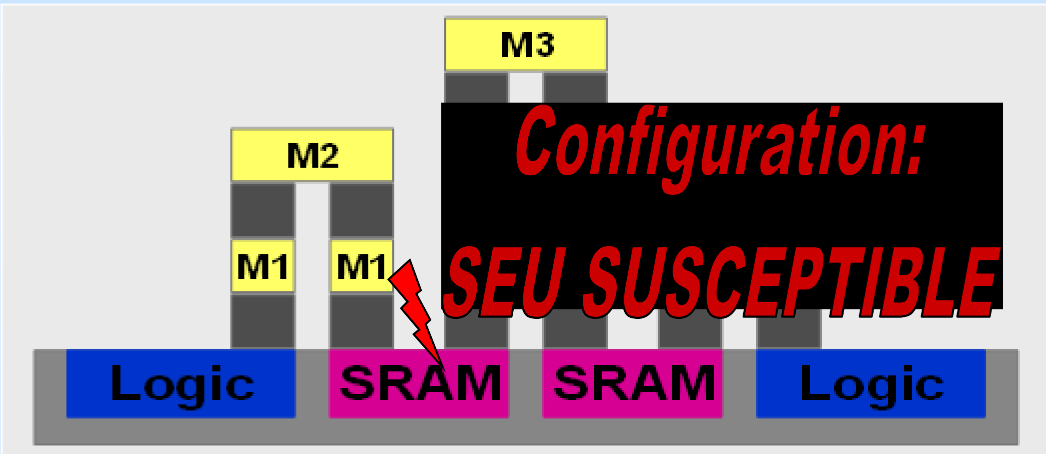
ANTIFUSE (OTP)





SRAM (RP)





Configuration SEU Test Results and the REAG FPGA SEU Model

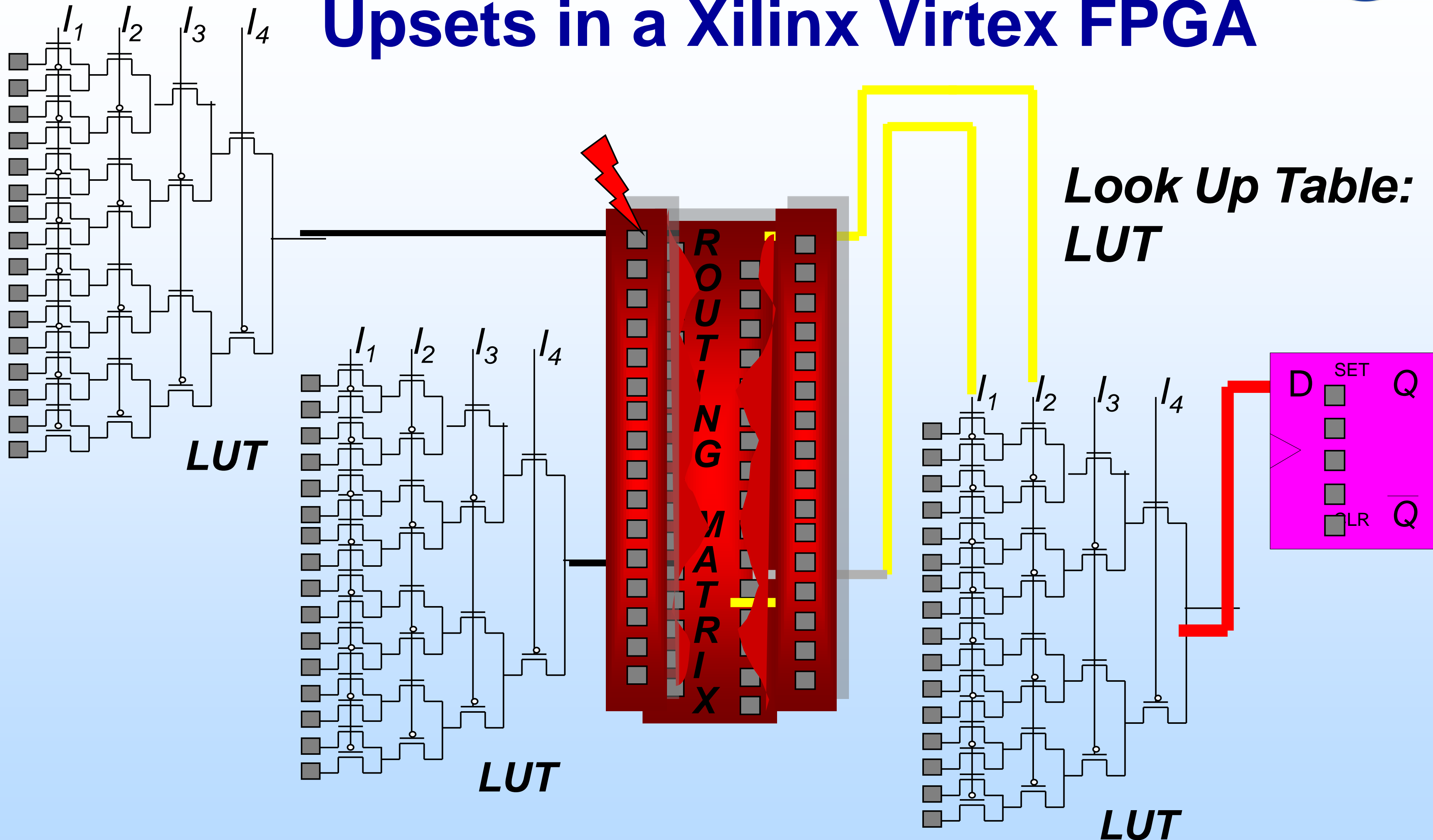
$$P(fs)_{error} \propto P_{Configuration} + P(fs)_{functionalLogic} + P_{SEFI}$$

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 hardened (but NOT hard) 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).

Example: Routing Configuration Upsets in a Xilinx Virtex FPGA



Mitigation at the LUT or DFF level will not mitigate route breakage… this SEU can be more catastrophic than a DFF bit flip

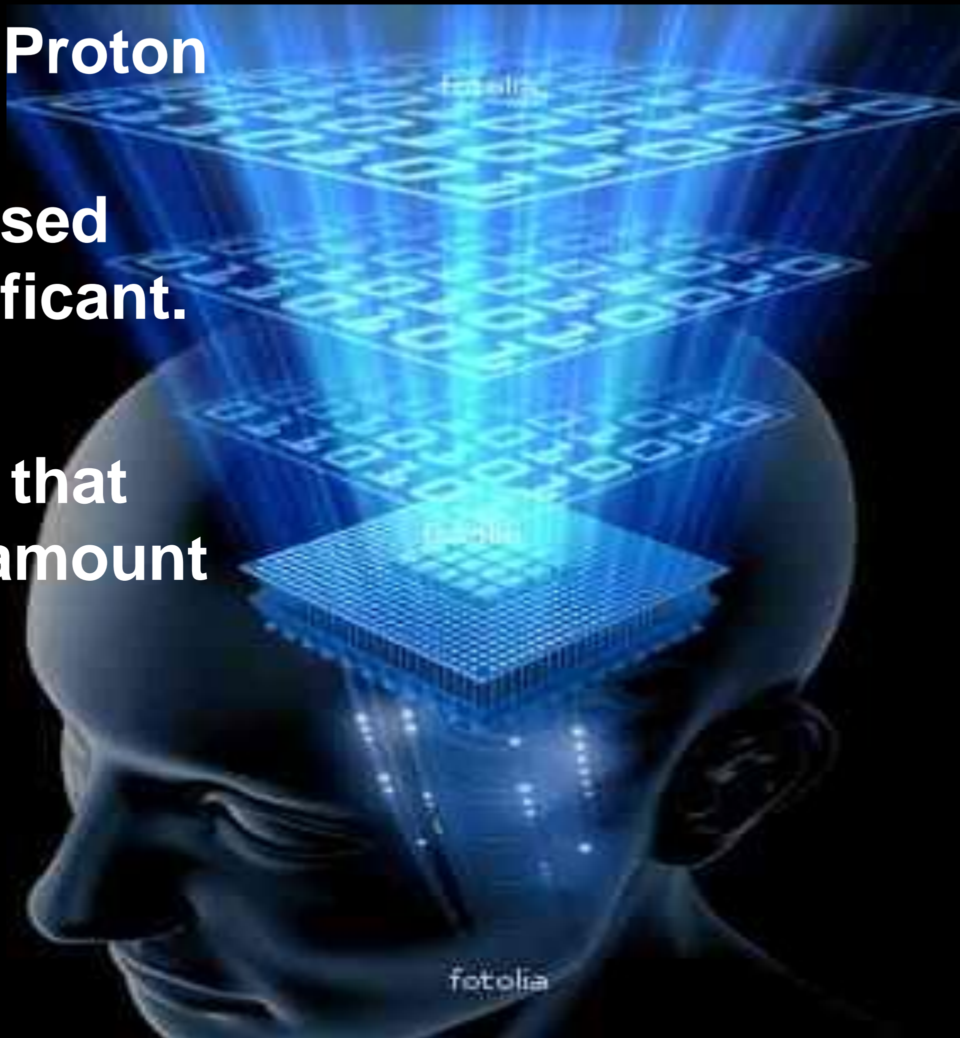
Based on FPGA Type and Project Requirements, Mitigation Strategy Will Vary

- **Hardened configuration FPGA's generally only need data-path mitigation strategies. However, if the global routes are not hardened, the application is critical (with strict requirements of availably and functionality), and the mission is targeted to a harsh radiation environment, a more complex (and costly) mitigation scheme may be required.**
- **Unhardened configuration FPGA's are generally not used for critical applications:**
  - **Uncritical application: don't use any mitigation scheme.**
  - **Critical application: complex and costly mitigation may be required.**
- **Cost, area, and power trade-offs are always considered.**

Mitigation will be discussed later in the presentation.

Configuration SEUs… Take-Away Points

- SRAM based configurations are highly susceptible to HI and Proton SEUs…
  - Other upsets in SRAM based FPGA devices are insignificant.
  - Mitigation will help.
  - Drastic upsets can occur that will require a significant amount of mitigation.
- Antifuse and Flash configurations are immune HI and proton SEUs… other types of upsets must be explored.



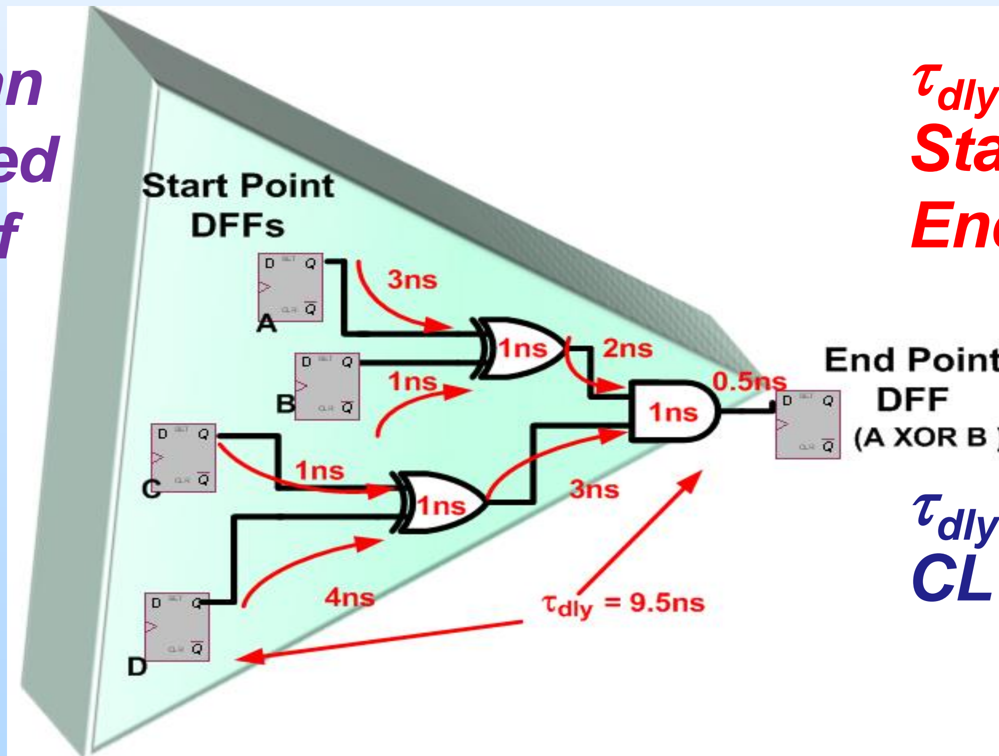
Functional Data Path SEU Modeling

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

Functional Data Path in A Synchronous Design

- **All synchronous design data paths contain:**
  - **Flip-flops(DFFs) and**
  - **Combinatorial Logic (CL).**
- **Data path susceptibility as it pertains to the component level and synchronous design will be discussed.**

Data paths can be modularized into "cones of logic"

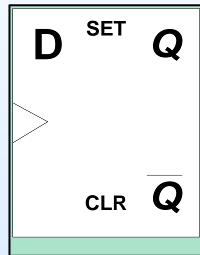


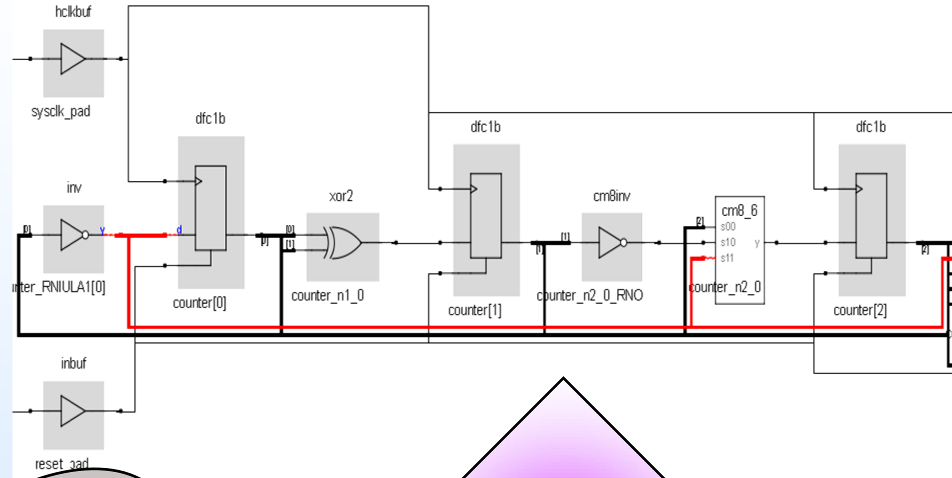
τ_{dly} = delay from StartPoint DFF to EndPoint DFF.

τ_{dly} is created by CL and routing.

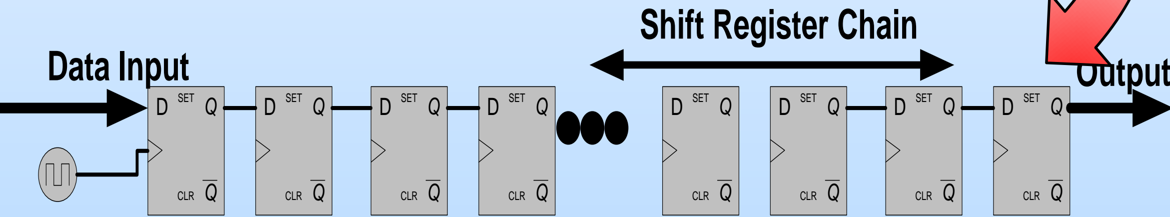
SEU Analysis and Trends: DFFs

DFF upset rate trends for this:





Are not the same for this:

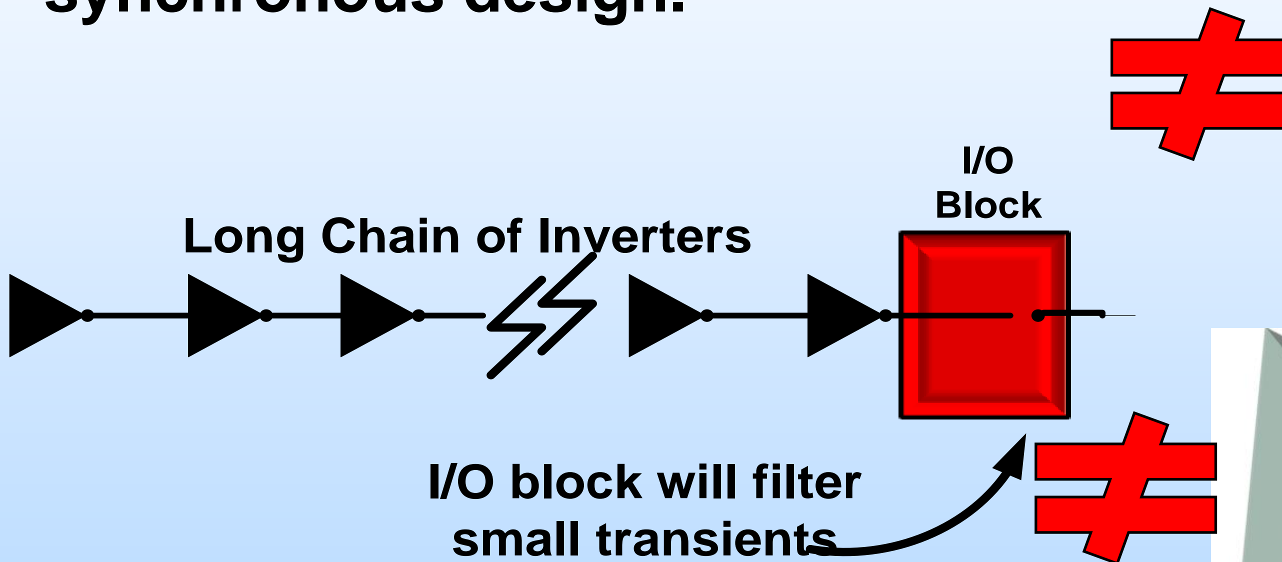


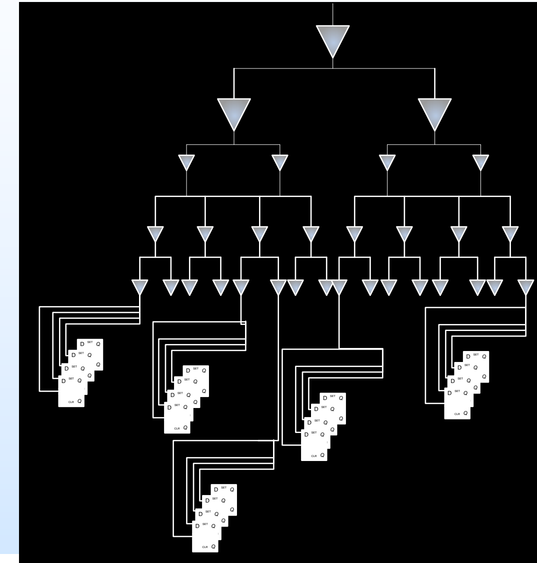
Are not the same for that:

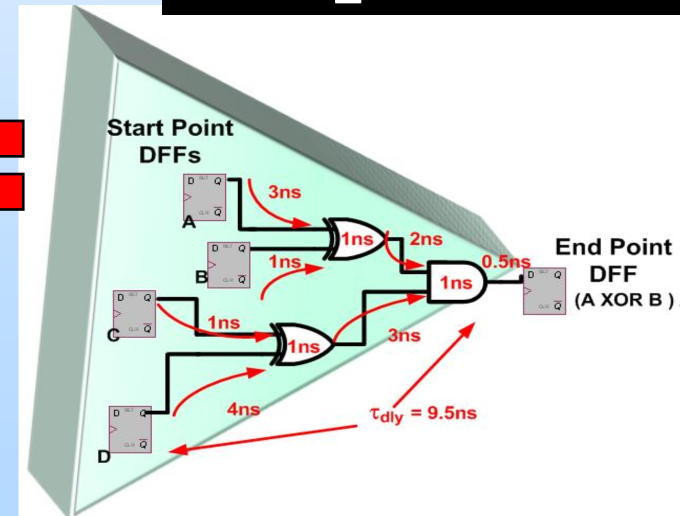
Frequency effects will differ based on DFF circuitry and design topology (feedback, fan-out, delay between DFF stages).

SEU Analysis and Trends: Combinatorial Logic (CL)

- **Studying a long chain of inverters is not the same as studying transient effects of combinatorial logic in synchronous design.**







SET propagation and probability of capture are different based on the design topology.

Point of The Last Two Slides

- Due to synchronous design topology and masking (electrical and logic), SEU data extrapolation from a test circuit may not be applicable to a real design across frequency or mitigation.
- We need to understand the target design topology and how components are used (or how often they are used):
  - Helps to construct better test circuits (synchronous, balanced clock trees, feedback, fan-out, masking, etc…).
  - Helps to correctly apply/extrapolate SEU data from test circuits to the design.

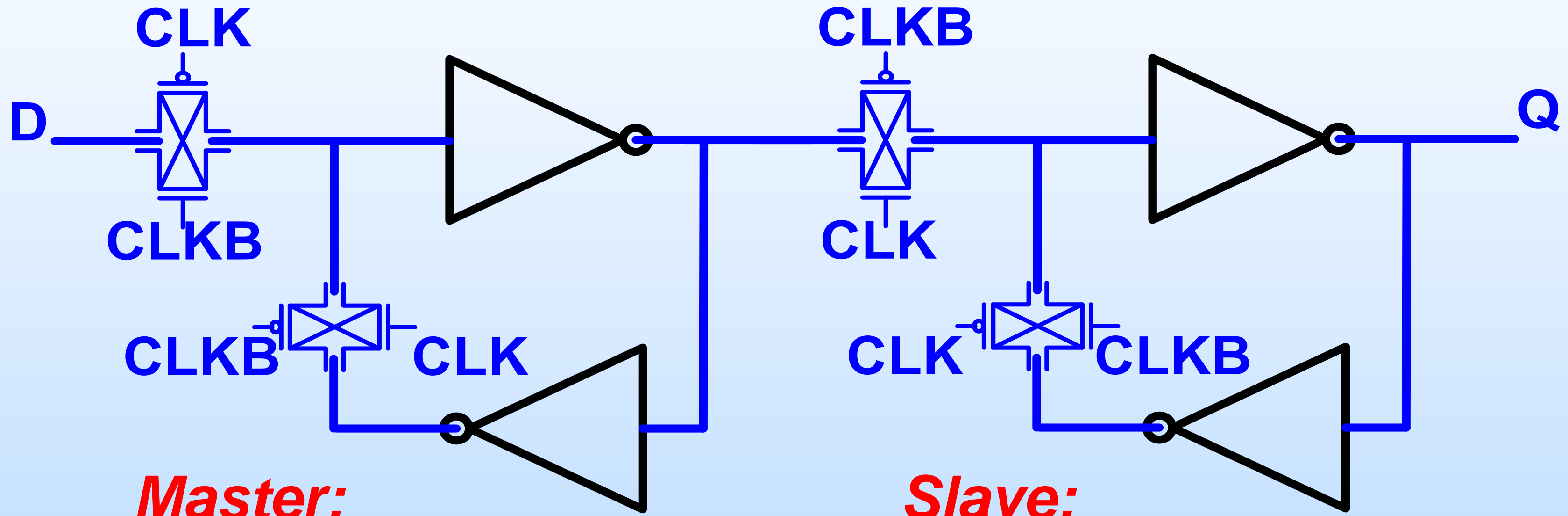
Common Misperceptions Regarding Test Circuit σ_{SEU} Data Extrapolation to Target Designs

- As frequency is increased, the upset rate (or σ_{SEU}) for a design implemented in an FPGA will increase.
  - Are the DFFs mitigated?
  - Are the clock trees or global routes mitigated?
- SET widening during propagation is a concern in FPGA devices. NOT True.
- Triple module redundancy (TMR) will mask all single bit upsets:
  - What type of TMR are you talking about?
  - There are various degrees of TMR application.

Edge Triggered Flip Flops... Creating Deterministic Boundary Points

D input must be settled by rising edge of clock

Output will only change at rising edge of clock



Master:

Clock Low: Transparent

Clock High: Hold

Slave:

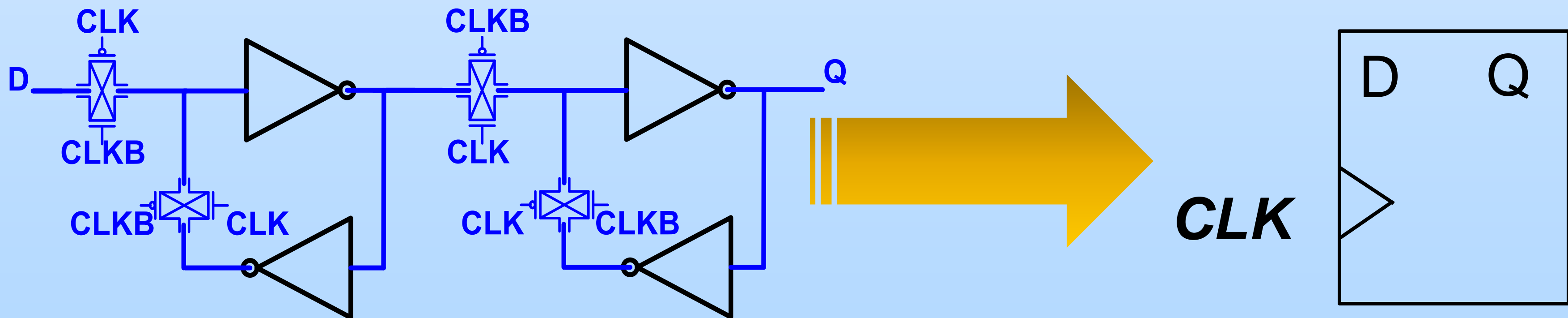
Clock Low: Hold

Clock High: Transparent

In order to create precise boundary points of state capture, Latches are NOT allowed in Synchronous designs!

DFF SEU Generation ($P(fs)_{DFFSEU}$)

- **DFF outputs define the state of the design.**
- **We will characterize DFF upset generation ($P(fs)_{DFFSEU}$) based on DFF output.**
- $P(fs)_{DFFSEU}$ **can affect the system differently depending on when and where the upset is actually generated.**



How is the DFF's Output Affected?... Clock Low

Master is transparent and Slave is in hold.

Master is cut off from slave and an SEU can occur in the slave.

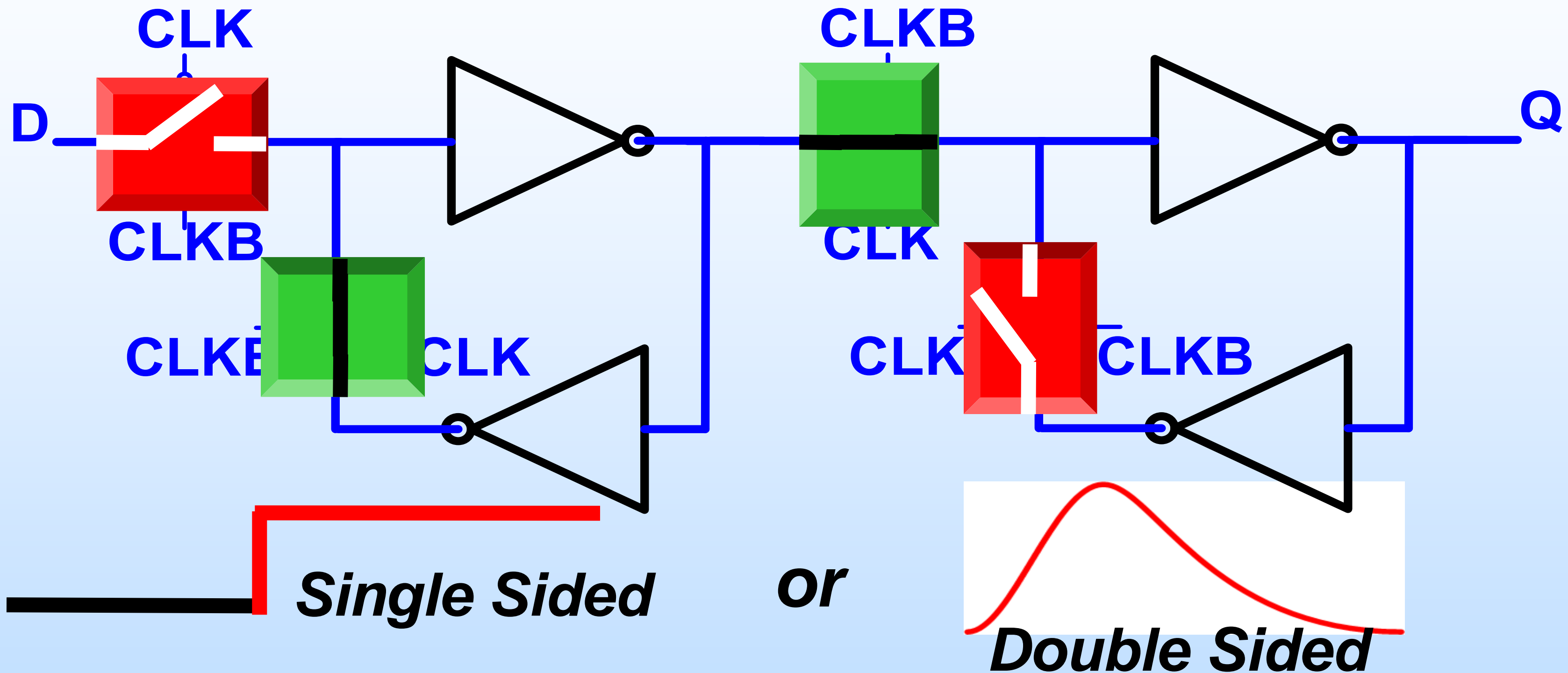
States are defined at the rising edge of clock.

If a single sided upset is generated while clock is low, generation occurs during an intermediate state.

Will the upset disrupt system behavior?

How are DFFs' Outputs Affected? Clock High

Master is in a hold state and Slave is transparent.



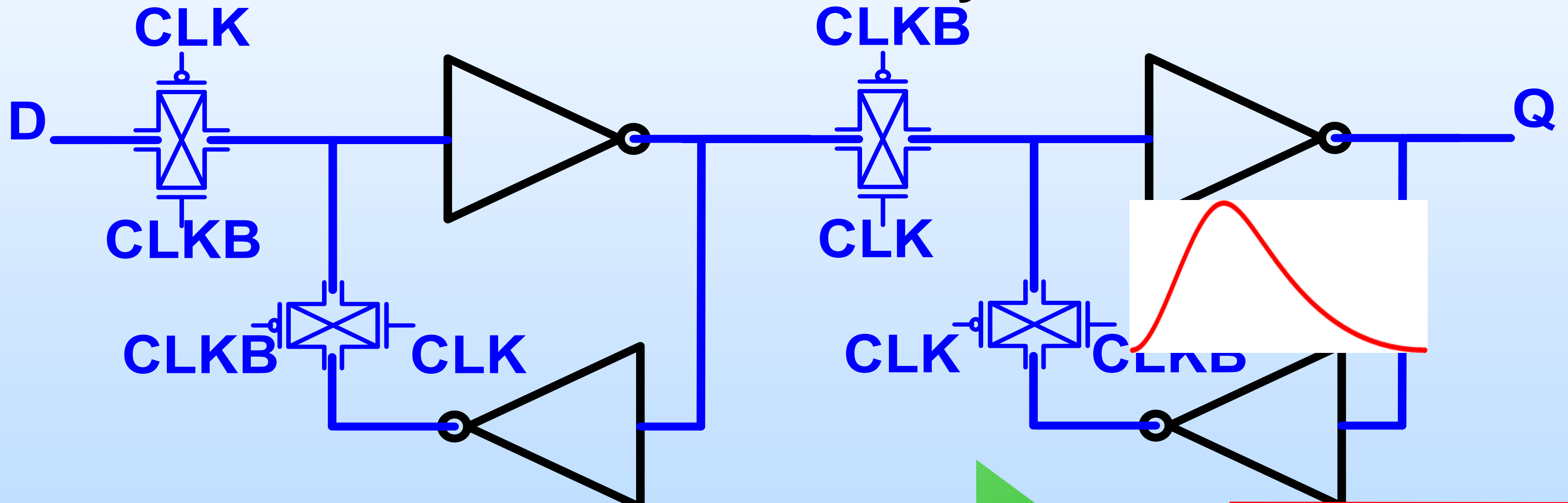
States are defined at the rising edge of clock.

If single sided or double sided upset is generated while clock is high, generation occurs during an intermediate state.

Will the upset disrupt system behavior?

How is the DFF's Output Affected? Falling Clock Edge… Clock High ➔ Clock Low

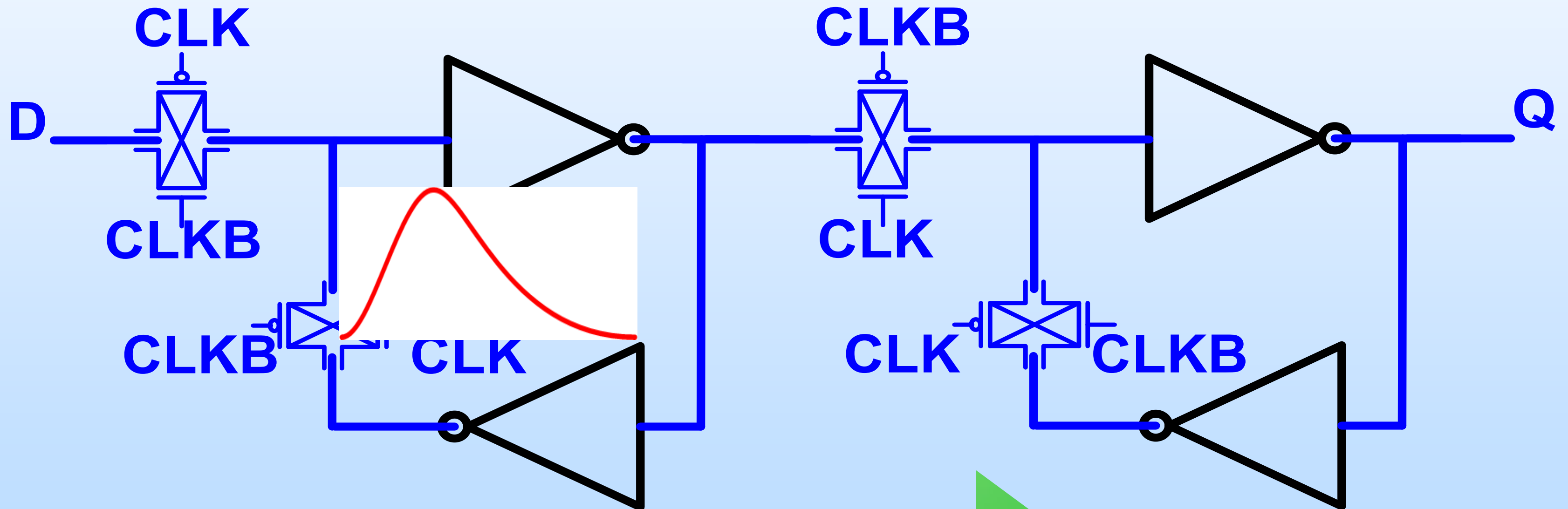
Slave can capture an SET from its transparent state. Single sided upset at output. Depends on the frequency, SET width, and the amount of circuitry in the Slave.



Output changes at falling clock edge, in between clock edges and may not affect the system.

How is the DFF's Output Affected? Rising Clock Edge… Clock Low ➔ Clock High

Master can capture an SET from its transparent state. Single sided upset at output. Depends on the frequency, SET width, and the amount of circuitry in Master.



Captured at rising clock edge

Single Sided

Output changes at clock edge, hence it will affect the system state.

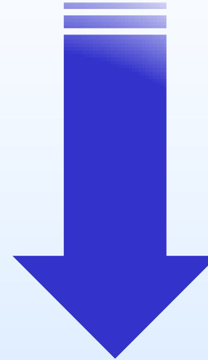
Summary of DFF SEU Susceptibility

Clock state	Upset Type	System Effect
High	Master: Single sided Slave: SET	Between rising clock edges
Low	Slave: Single sided	Between rising clock edges
High➔ Low	Slave latches its own transient: Single sided	Between rising clock edges
Low➔ high	Master latches its own transient: Single Sided	At rising clock edge

- ***Between rising clock edges: must be captured by the next clock edge to cause a system upset (if not later masked).***
- ***At rising clock edge: system upset (if not later masked).***

Single Sided Upset DFF Generation

$$P(fs)_{DFFSEU} = \alpha P(fs)_{DFFSEU} + \beta P(fs)_{DFFSEU} + P(fs)_{DFFSET}$$



SETs generated in slave when clock is high

- ***Frequency dependent.***

Percentage of single sided upsets that occur at clock edge (low to high)

- **Frequency dependent: Master SET gets trapped during transition from transparent to hold state (rising edge of clock).**
- **This is considered a state change.**

Percentage of single sided upsets that occur between clock edges

- ***Not Frequency dependent: Master or slave is in hold state.***
- ***Frequency dependent: Low➔High transition.***
- ***This is not considered a definitive state change.***

Agenda

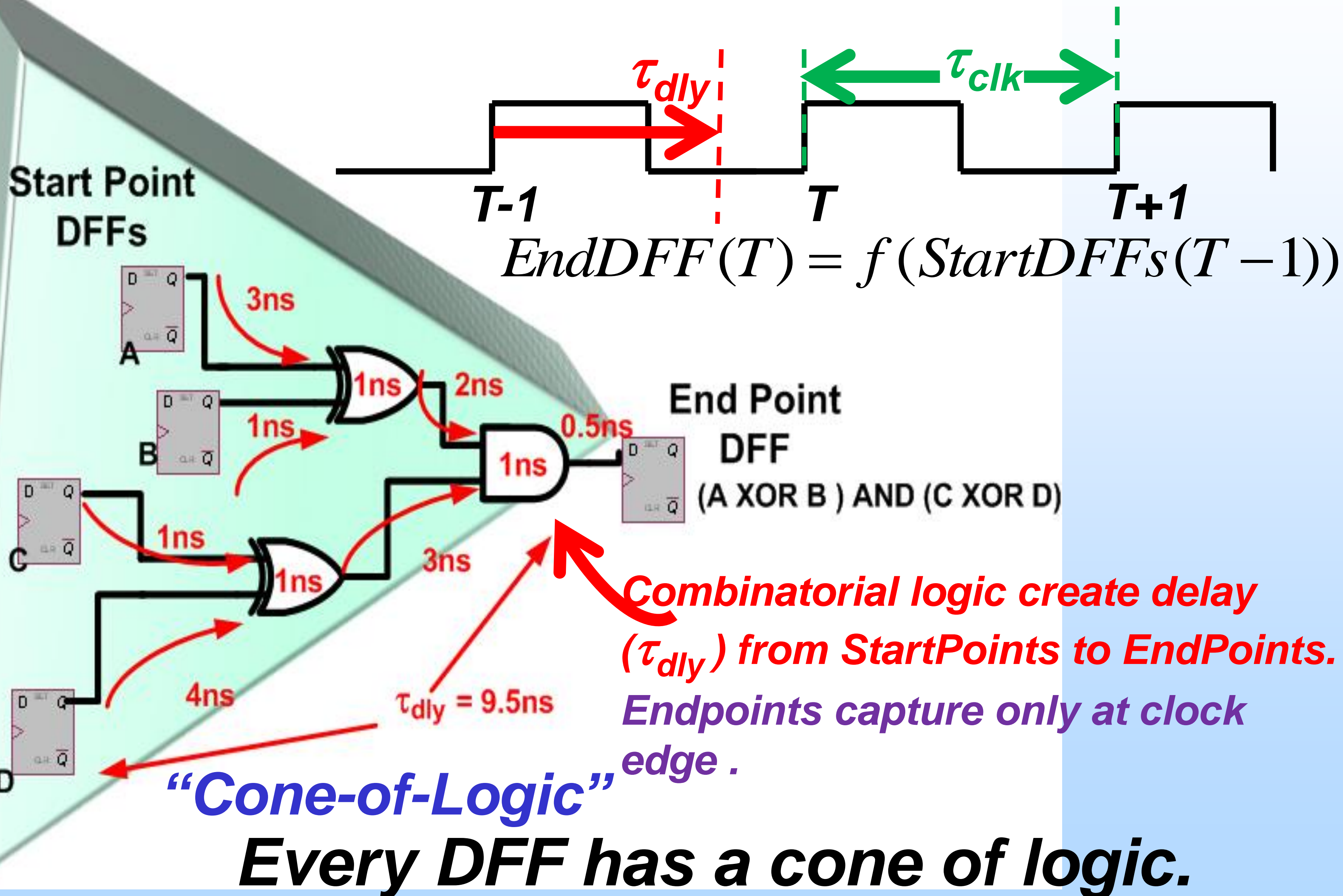
- Section I: General FPGA Description and Design Process.
- Section II: Single Event Effects (SEEs) in Digital Logic.
- **Section III: Application of the NASA Goddard Radiation Effects and Analysis Group (REAG) FPGA SEU Model.**
- Section IV: Reducing System Error: Common Mitigation Techniques.
- Section V: When Your Mitigation Fails.
- Section VI: Xilinx Virtex Series and Mitigation.

SEU Generation versus SEU Capture in Synchronous Systems

- **We discussed SET generation and SEU generation.**
- **It is not definitive that an SET or SEU will cause system upsets.**
- **It is essential to differentiate between SEE generation versus system upset.**
- **Clock edge upset:**
  - **Will be a system upset if not logically masked.**
- **Intermediate clock edge upset:**
  - **Will be a system upset if it affects a DFF at the next clock edge.**

The question becomes, will the upset be missed or will it manifest… depends on design topology.

Synchronous System Data Paths: StartPoint DFFs → EndPoint DFFs



$$EndDFF(T) = f(StartDFFs(T-1))$$

Combinatorial logic create delay (τ_{dly}) from StartPoints to EndPoints.

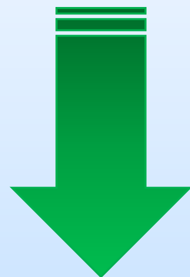
Endpoints capture only at clock edge .

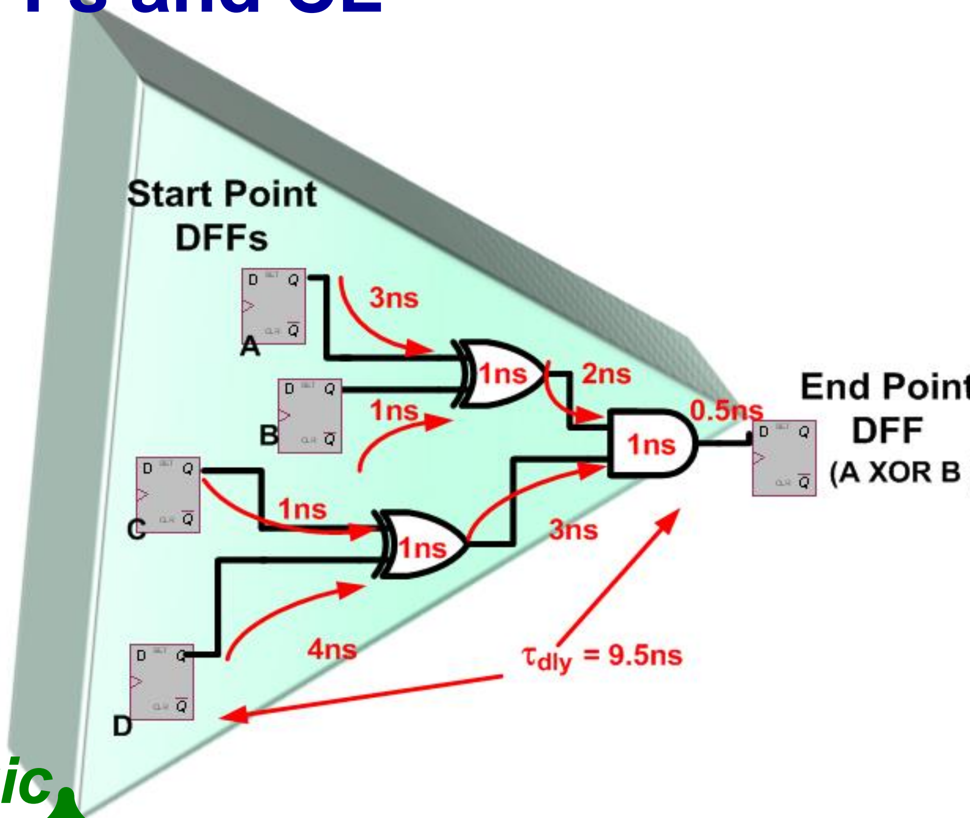
Every DFF has a cone of logic.

Data Path Model and DFF Logic Cones: Upsets Originate in DFFs and CL

$$P(fs)_{functionalLogic}$$

Evaluate Each DFF as an EndPoint





DFF_k Cone of Logic

$$\sum_{DFF} \textit{EndPoint DFF SEUs} + \textit{StartPoint DFF SEUs} + \textit{CL SETs}$$

$\alpha P(fs)_{DFFSEU}$

DFF upsets that occur at the clock edge.

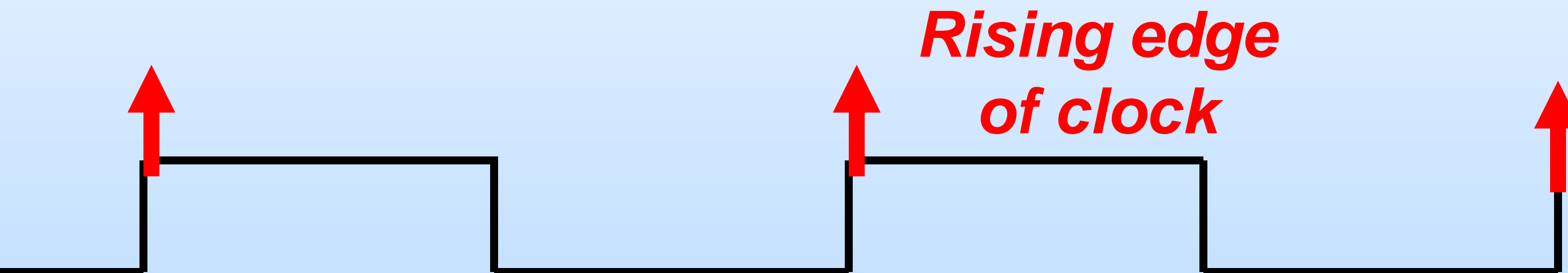
$\beta P(fs)_{DFFSEU}$

DFF upsets that occur between clock edges.

Characterizing EndPoint SEU System Effects

- EndPoint DFF SEU Capture:
  - SEU Generation in EndPoint DFF ($\alpha P(fs)_{DFFSEU}$) at rising edge of clock.
  - Logic Masking (P_{logic}) is after the EndPoint DFF.

Rising edge of clock



For this scenario remember:

upset generation is internal to the DFF… Master captures its own SET during rising edge clock transition

StartPoint DFF SEU Capture

Start Point DFFs τ_{clk}@T-1

T-1 T T+1

τ_{dly} τ_{clk}

A 0 3ns 1ns 2ns

B 1 1ns

C 1 1ns 1ns 3ns

D 1 4ns

1ns 0.5ns

End Point DFF τ_{clk}@T

1 ??? 0

(A XOR B) AND (C XOR D)

τ_{dly} = 9.5ns

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.

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

Upset capture with respect to to frequency.

Details of Capturing StartPoint DFFs

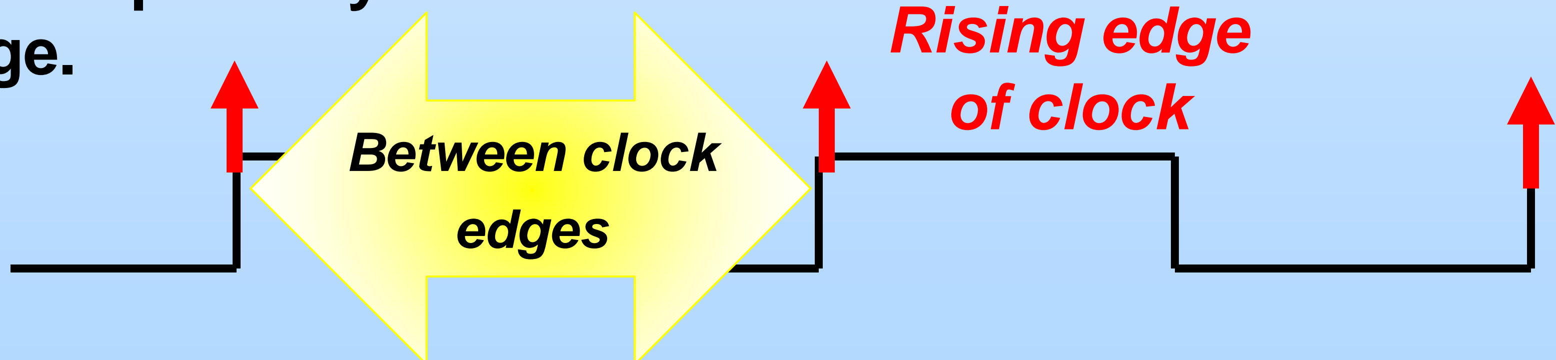
$$\lambda_{DFF} \approx \left(\sum_{j=1}^{\#StartPoint\,DFFs} \beta P(fs)_{DFFSEU(j)} (1 - t_{dly(j)} fs) P_{logic(j)} \right)$$

Generation

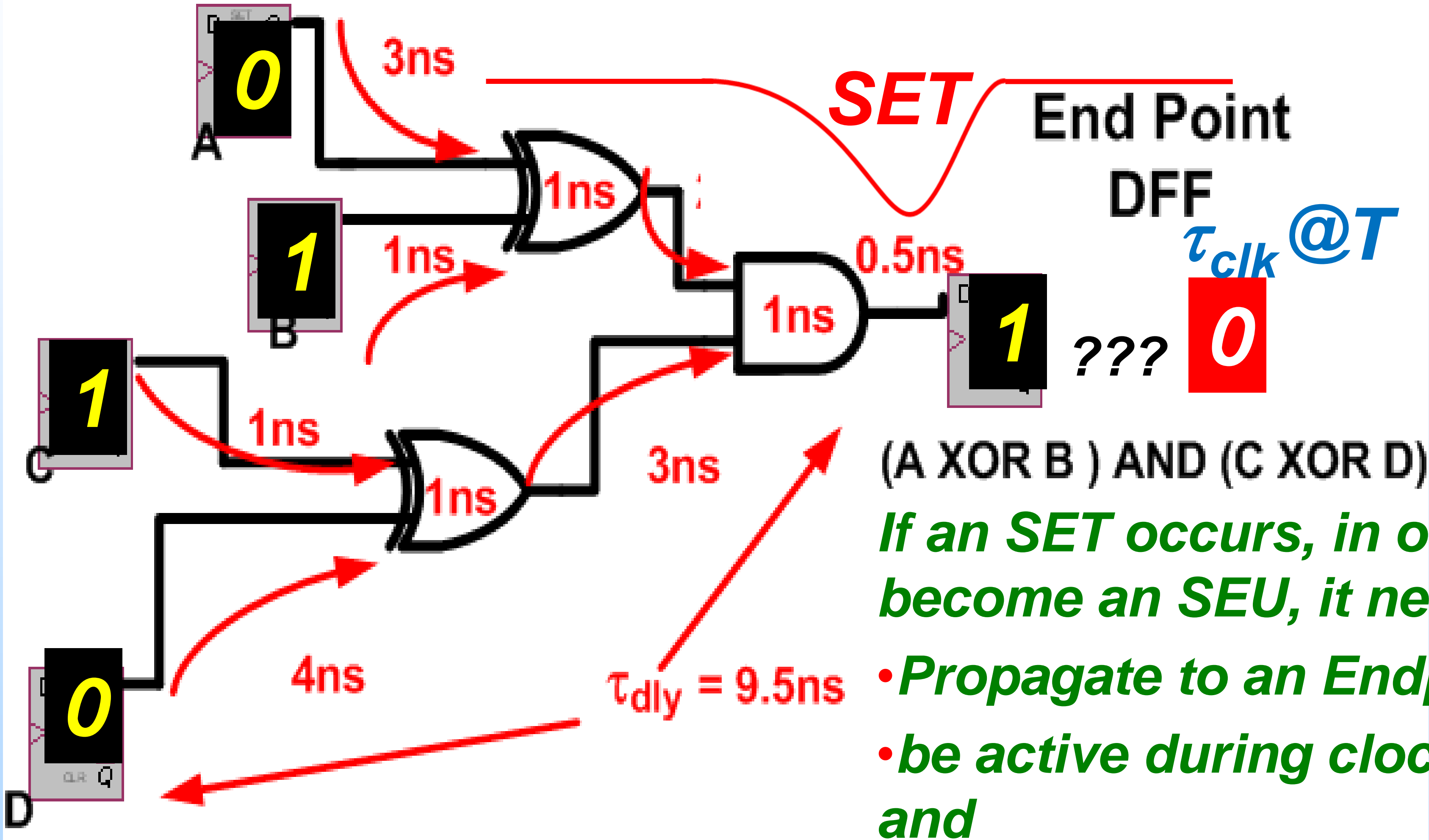
Capture

Logic Masking

- **SEU generation occurs in a StartPoint between clock edges ($\beta P(fs)_{DFFSEU}$).**
- **SEU capture by the EndPoint occurs at a clock edge.**



Synchronous System: CL SET Capture



(A XOR B) AND (C XOR D)

If an SET occurs, in order to become an SEU, it needs to:

- ***Propagate to an Endpoint,***
- ***be active during clock edge, and***
- ***be Captured τ_{width}/τ_{clk}.***

Details of Combinatorial Logic SET Capture

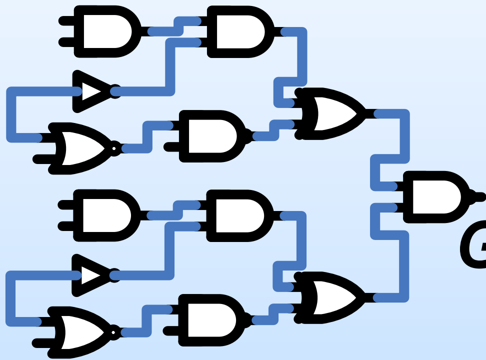
$$\$_{DFF} \left(\sum_{i=1}^{\#CombinatorialCells} \left(P_{gen(i)} P_{prop(i)} P_{logic} \, t_{width(i)} fs \right) \right)$$

Generation

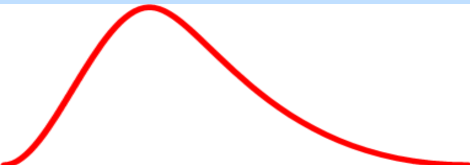
Propagation

Logic Masking

Capture



CL

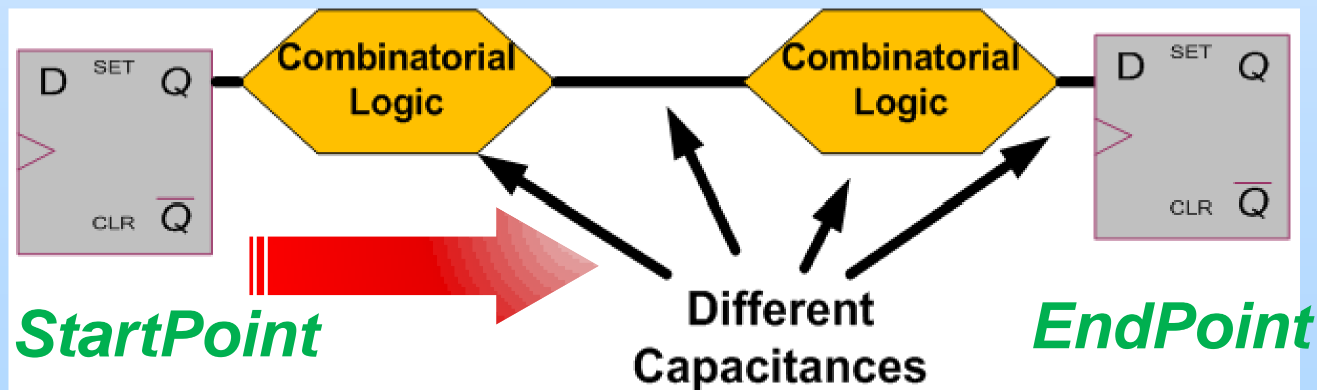


Double Sided

- SET Generation (P_{gen}) occurs between clock edges.
- SET Capture occurs at a clock edge.

SET Propagation to an EndPoint DFF: P_{prop}

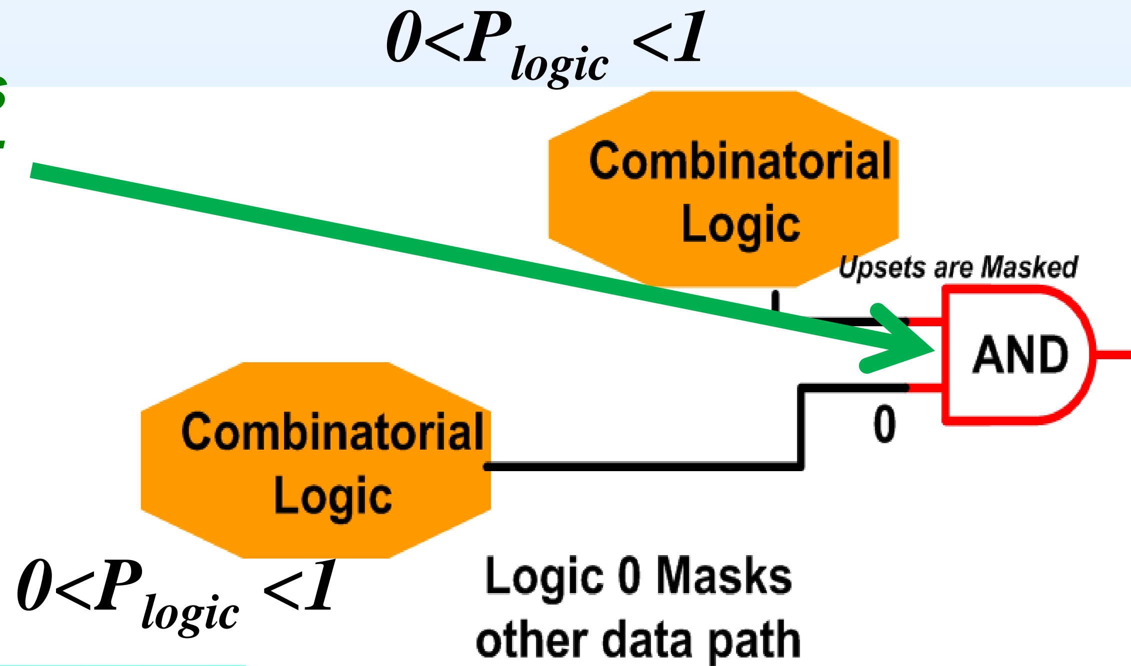
- **In order for the data path SET to become an upset, it must propagate and be captured by its Endpoint DFF.**
- **P_{prop} only pertains to electrical medium (capacitance of path… combinatorial logic and routing):**
  - **Capacitive SET amplitude reshaping.**
  - **Capacitive SET width reshaping.**
- **Small SETs or paths with high capacitance have low P_{prop}.**
- **P_{prop} contributes to the non-linearity of $P(fs)_{SET \rightarrow SEU}$ because of the variation in path capacitance.**



SET Logic Masking: P_{logic}

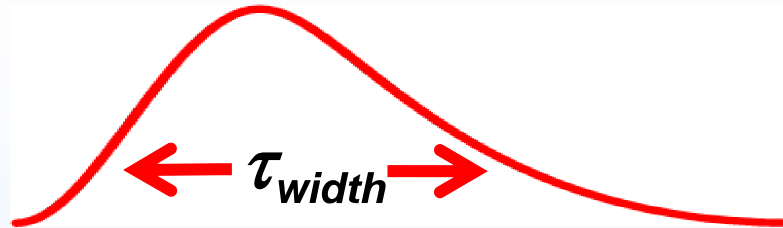
- P_{logic}: ***Probability that a SET can logically propagate through a cone of logic. Based on state of the combinatorial logic gates and their potential masking.***

"AND" gate reduces probability that SET will logically propagate.



Determining P_{logic} for a complex system can be very difficult.

SET Capture at Destination DFF



The transient width (τ_{width}) will be a fraction of the clock period (τ_{clk}) for a synchronous design in a CMOS process.

$$P(\tau_{clk})_{SET \rightarrow SEU} \propto \frac{\tau_{width}}{\tau_{clk}}$$

$$P(fs)_{SET \rightarrow SEU} \propto \tau_{width} fs$$

Probability of capture is proportional to the width of the transient as seen from the destination DFF.

Putting it All Together: NASA REAG FPGA Data Path Susceptibility Model

$$P(fs)_{functionalLogic}$$

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

EndPoint Logic Masking (under $P_{logic(k)}$)

EndPoint (above $\alpha P(fs)_{DFFSEU(k)}$)

StartPoints (above the j sum)

Combinatorial Logic (below the i sum)

Note:

$P_{logic(k)}$ ➔ ***EndPoint Logic Masking***

$P_{logic(j)}$ ➔ ***StartPoint Logic Masking***

$P_{logic(i)}$ ➔ ***Combinatorial Logic Masking***

Take Away Points: Functional Data Path Upsets

$P(fs)_{functionalLogic}$ upsets occur due to DFFs and CL.

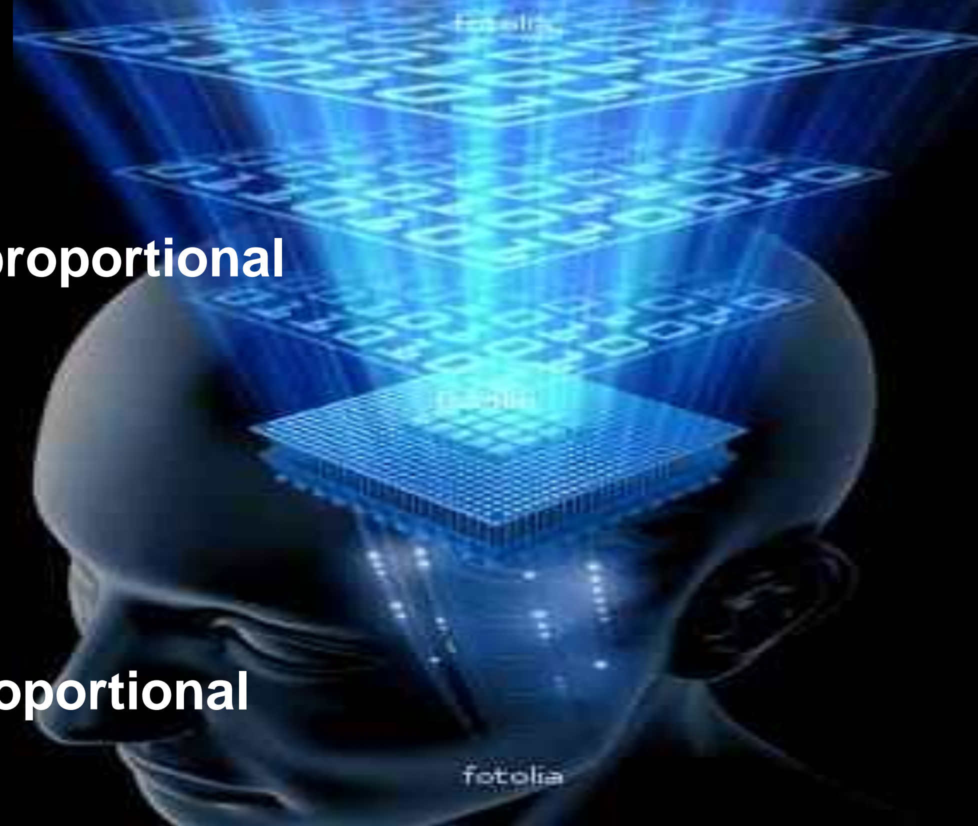
$$\exists_{DFF}\left(\left(\sum_{j=1}^{\#StartPointDFFs} P_{DFFSEU(j)}(1-\tau_{dly(j)} fs) P_{logic(j)} \right) + \sum_{i=1}^{\#CombinatorialCells} (P_{gen(i)} P_{prop(i)} P_{logic} \tau_{width(i)} fs) \right)$$

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

- DFFs have flipped states (SEUs) .
- System Susceptibility is Inversely proportional to frequency and CL between DFFs.

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

- CL have glitches (SETs).
- System Susceptibility is Directly proportional to frequency and CL between DFFs.



Agenda

- Section I: General FPGA Description and Design Process.
- Section II: Single Event Effects (SEEs) in Digital Logic.
- Section III: Application of the NASA Goddard Radiation Effects and Analysis Group (REAG) FPGA SEU Model.
- **Section IV: Reducing System Error: Common Mitigation Techniques.**
- Section V: When Your Mitigation Fails.
- Section VI: Xilinx Virtex Series and Mitigation.

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.

Want to reduce as many terms as possible:

$$P(fs)_{error} \propto P_{Configuration} + P(fs)_{DFFSEU \to SEU} + P(fs)_{SET \to SEU} + P_{SEFI}$$

TMR Schemes Use Majority Voting

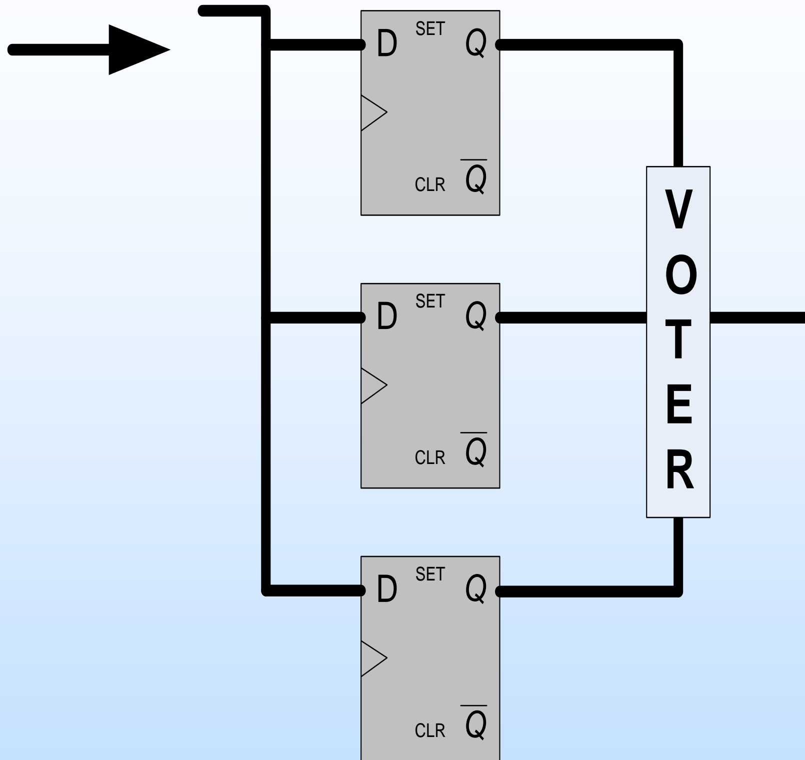
$$MajorityVoter = I1 \wedge I2 + I0 \wedge I2 + I0 \wedge I1$$

I0	I1	I2	Majority Voter
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
[illegible]	[illegible]	[illegible]	1

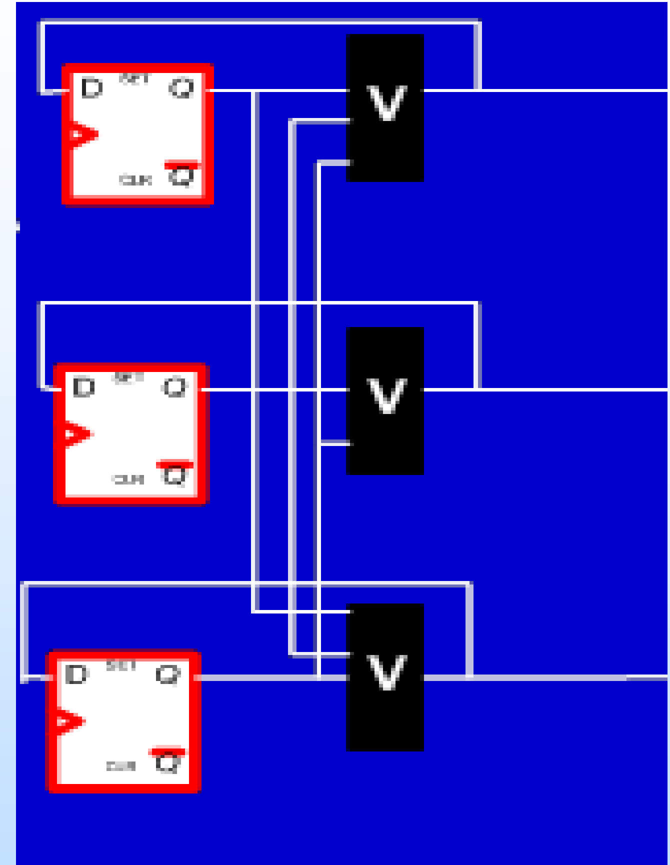
Best 2 out of 3

Triplicate and Vote

Triplicate and Vote



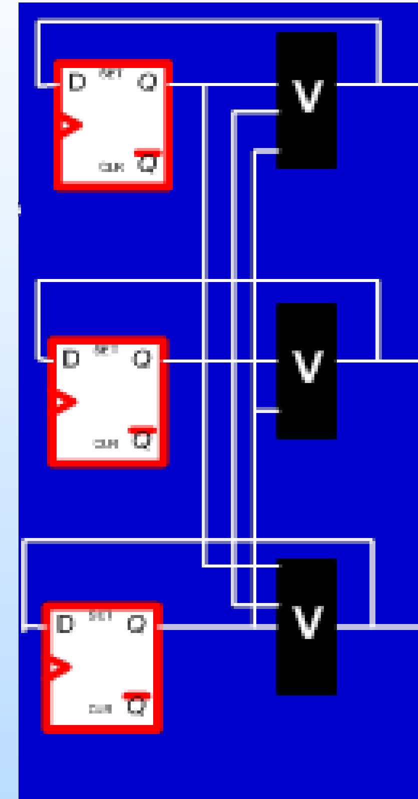
Singular Data Path



Redundant Data Path

TMR: Correction versus Masking Captured SEUs

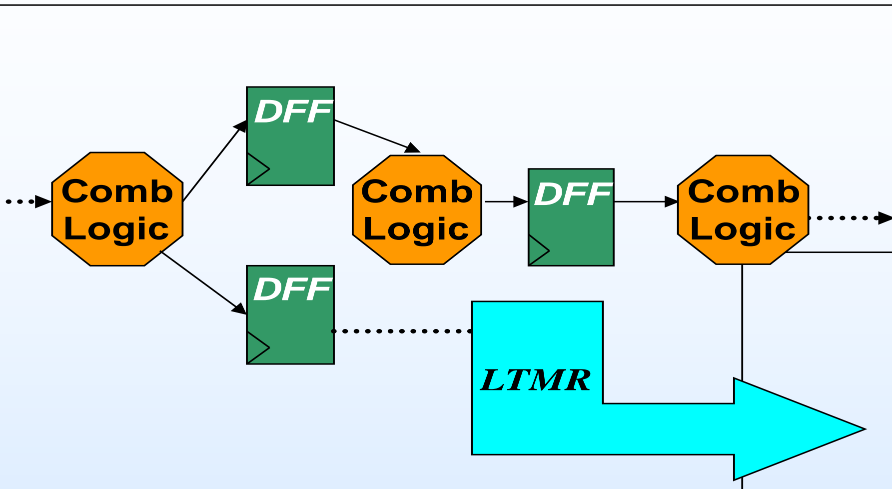
- TMR with feedback+Voter will mask and correct an error in a DFF
  - Mask upset on current clock cycle
  - Correct error on following clock cycle
- TMR with no feedback will only mask an error when using a Voter… however new enabled data cycle will flush
- DFFs that capture new data every cycle cannot use feedback from the Voter!...
  - Data path presides!!!! And will have new data to feed to DFF …. Upset is wiped out
  - However, the error will not propagate if a Voter is placed in front of the DFF



Block Triple Modular Redundancy: BTMR

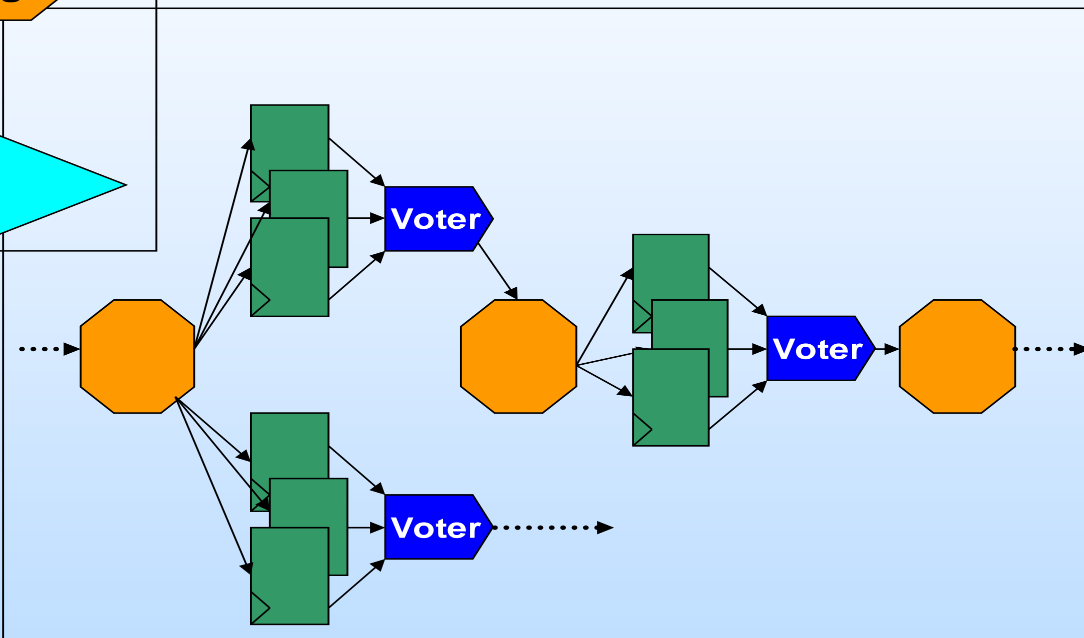
- Need Feedback to Correct
- Cannot apply internal correction from voted outputs
- If blocks are not regularly flushed (e.g. reset), Errors can accumulate – may not be an effective technique

Local Triple Modular Redundancy (LTMR)



Masks upsets from DFFs

Corrects DFF upsets if feedback is used

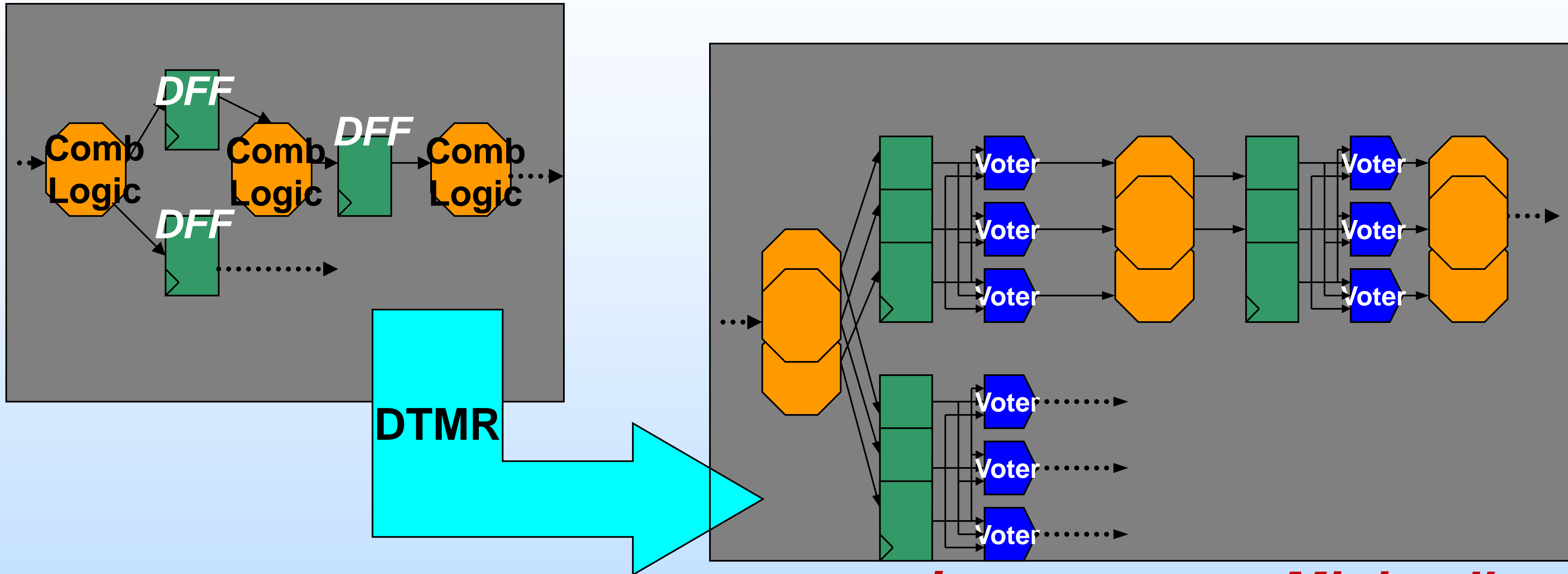


Only the DFFs are triplicated and mitigated

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

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

Distributed Triple Modular Redundancy (DTMR): DFFs + Data Paths
All DFFs with Feedback Have Voters



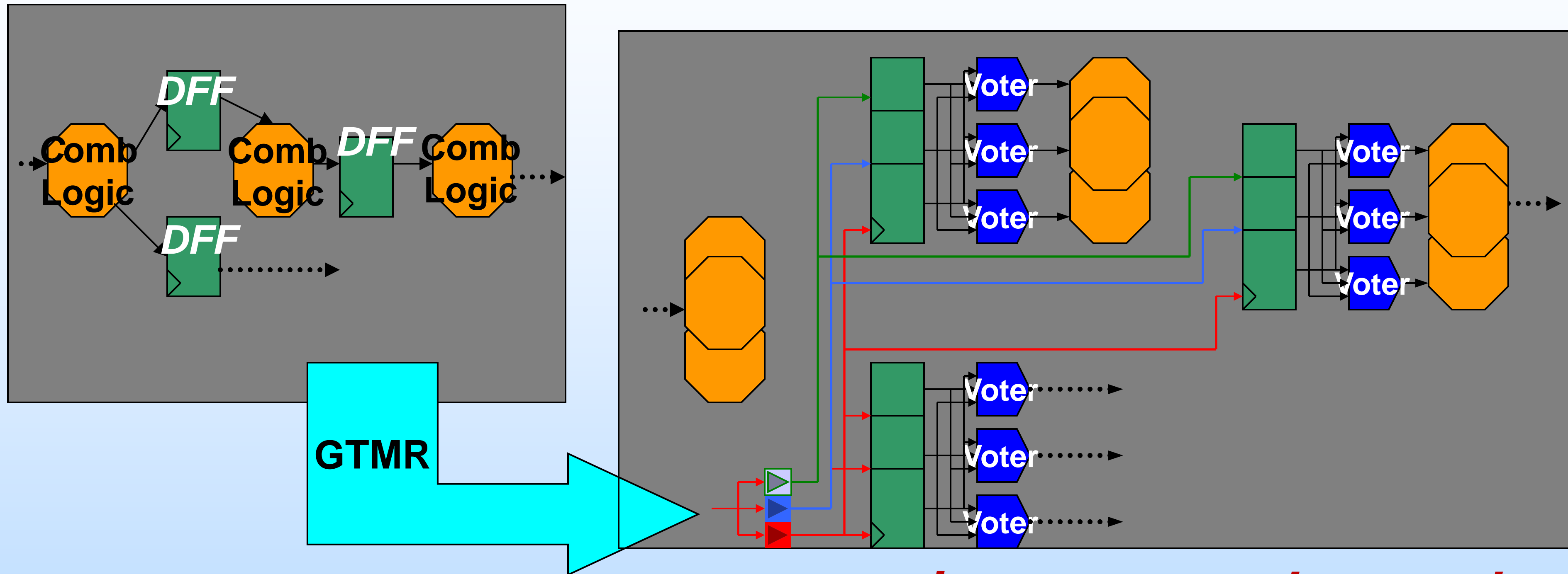
$$P(f_s)_{error} \propto P_{configuration} + P(f_s)_{functionalLogic} + P_{SEFI}$$

Low — Minimally Lowered

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

0 — Low

Global Triple Modular Redundancy (GTMR):DFFs + Data Paths + Global Routes All DFFs with Feedback Have Voters



Low | Lowered

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

Low | Low

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

GTMR Proves (via Accelerated Heavy Ion and Proton Testing) To be A Great 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 feedback of a voter to its associated DFF:
  - 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 Voter Placement Considerations

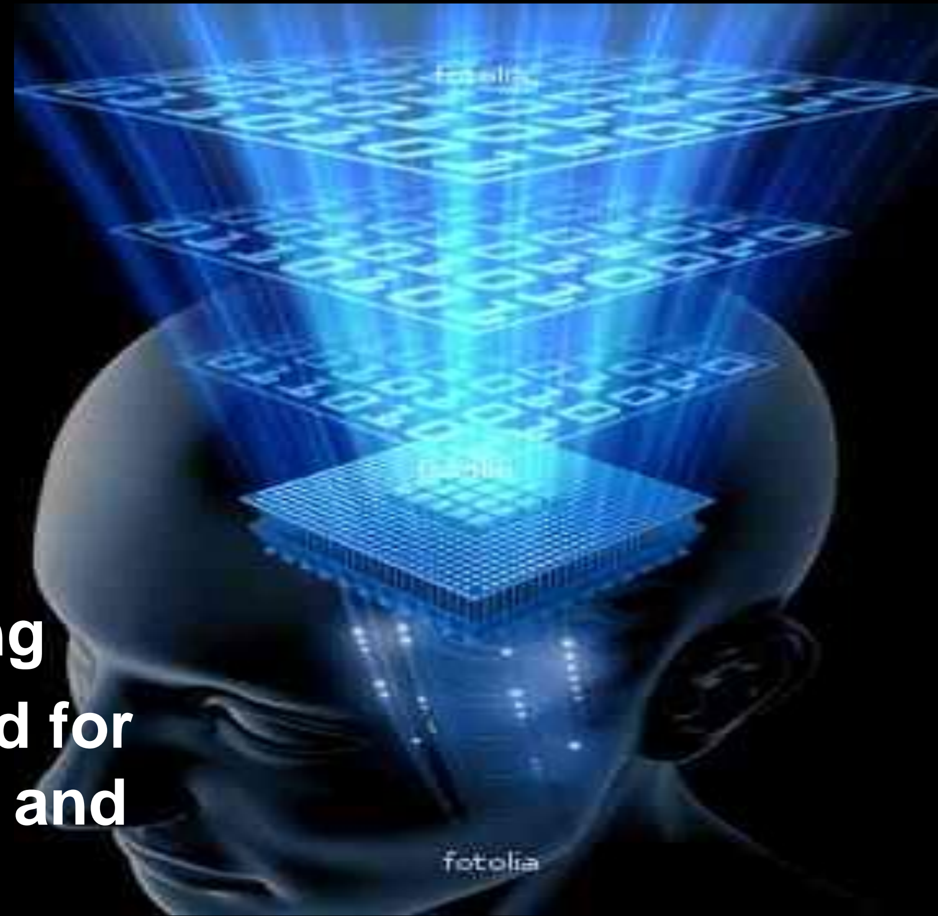
- Due to area concerns, in some schemes, voters are not placed after every DFF:
  - GTMR, or
  - DTMR.
- DFFs with feedback:
  - If the path is enabled often, may not need a Voter (game of probability).
  - If the path is enabled infrequently, then a Voter is probably needed.
  - If the path feeds a significant amount of complex circuitry a inserting a voter is a good option.
- DFFs that always capture each cycle from their data path (no feedback) do not necessarily need a Voter.

Summary of A Variety of TMR Effects

	No-TMR	LTMR	DTMR	GTMR
Dominant upsets	StartPoint DFF SEU capture	Combinatorial: SET capture	Clocks or resets (globals)	SEFIs
Dominant Model component	$P_{DFFSEU}(1-\tau_{dly}fs)$	$P_{gen}P_{prop}\tau_{width}fs$	P_{SEFI}	P_{SEFI}
Upset type	One sided function	Two-sided function		
Relationship to Frequency	Inversely proportional	Directly proportional	Design dependent	
Relationship to CL in datapath	Inversely proportional	Directly proportional	Design dependent	

The strength of GTMR and DTMR depends on voter insertion and scrubber efficiency (scrubbing will be discussed later).

Take Away Points: Mitigation and Design Considerations



- TMR is a commonly used mitigation strategy.
- Various types of TMR:
  - LTMR,
  - DTMR, or
  - GTMR.
- Voter placement is challenging
- Verification of TMR is required for user inserted implementation and can be difficult.
- There are many nuances regarding TMR insertion. They are very complex and should be handled by an automated tool.

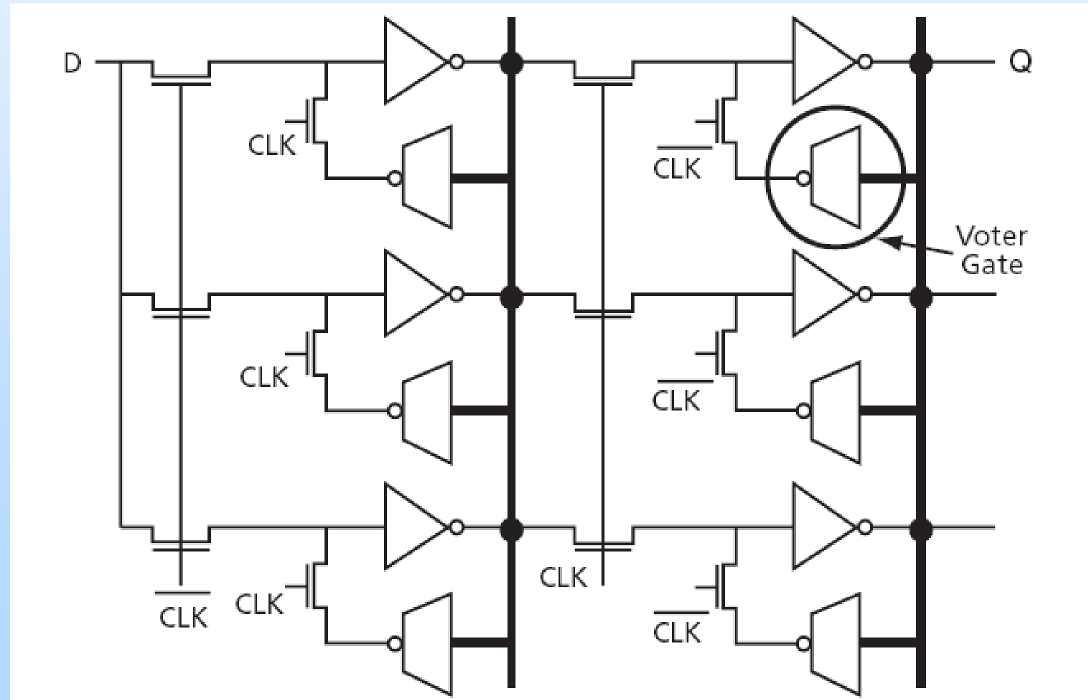
Agenda

- **Section I: General FPGA Description and Design Process.**
- **Section II: Single Event Effects (SEEs) in Digital Logic**
- **Section III: Application of the NASA Goddard Radiation Effects and Analysis Group (REAG) FPGA SEU Model**
- **Section IV: Reducing System Error: Common Mitigation Techniques**
  - **Triple Modular Redundancy (TMR)**
  - **Embedded Radiation Hardened by Design (RHBD)**
- **Section V: When Your Mitigation Fails**
- **Section VI: Xilinx Virtex Series and Mitigation**

Know your device… does it already have mitigation and if so, does it need more

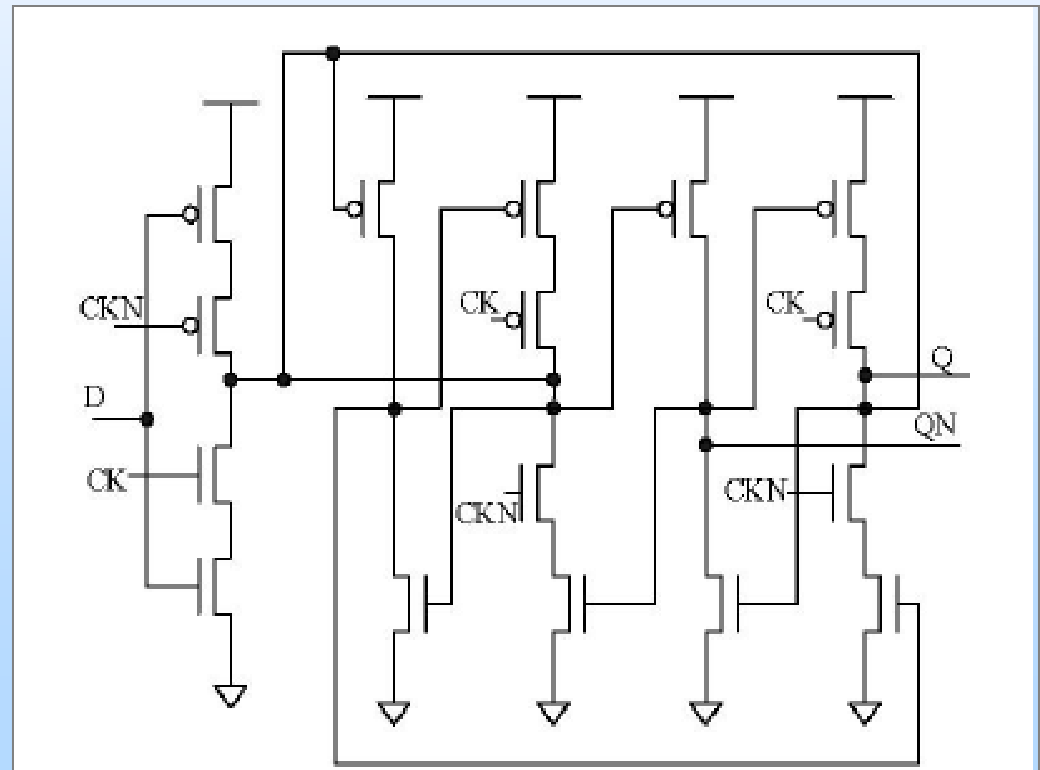
DFF with Embedded LTMR: Microsemi (Actel) RTAXs Family of FPGA

- **Localized (only at DFF).**
- **Microsemi uses Wired "OR" approach to voting – no SETs on voters.**
- **Correction doesn't require a clock (asynchronous internal voter feedback).**



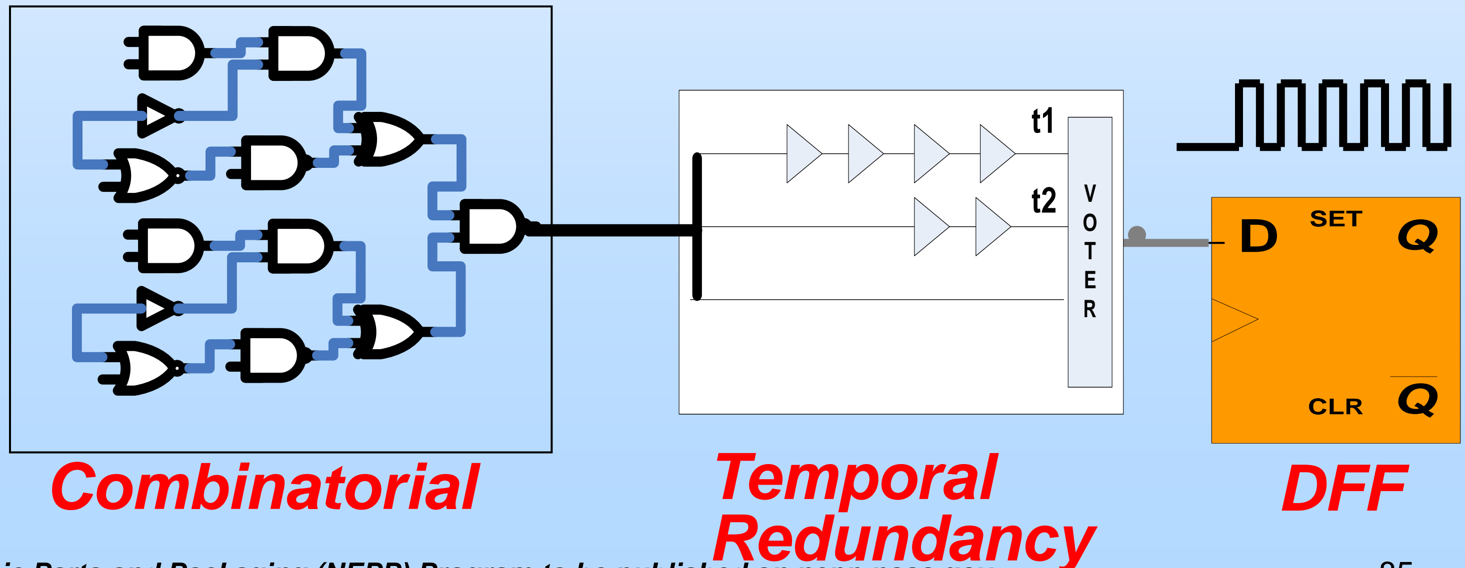
DFF with Embedded Dual Interlock Cell (DICE): Aeroflex Eclipse FPGA

- Localize mitigation for DFFs.
- Uses a Dual Redundancy Scheme instead of LTMR.
- Single nodes can become upset but their partner node will pull the output in the correct direction.



Embedded Temporal Redundancy (TR): SET Filtration

- Temporal Filter placed directly before DFF.
- Localized scheme that reduces SET capture.
- Delays must be well controlled.
  - Every delay path shall consistently have a predefined delay and must be verified… you say you have… you better have it.
  - Recommended that FPGA designers do not implement – too difficult to manage with place and route tool best if embedded.
- Maximum Clock frequency is reduced by the amount of new delay

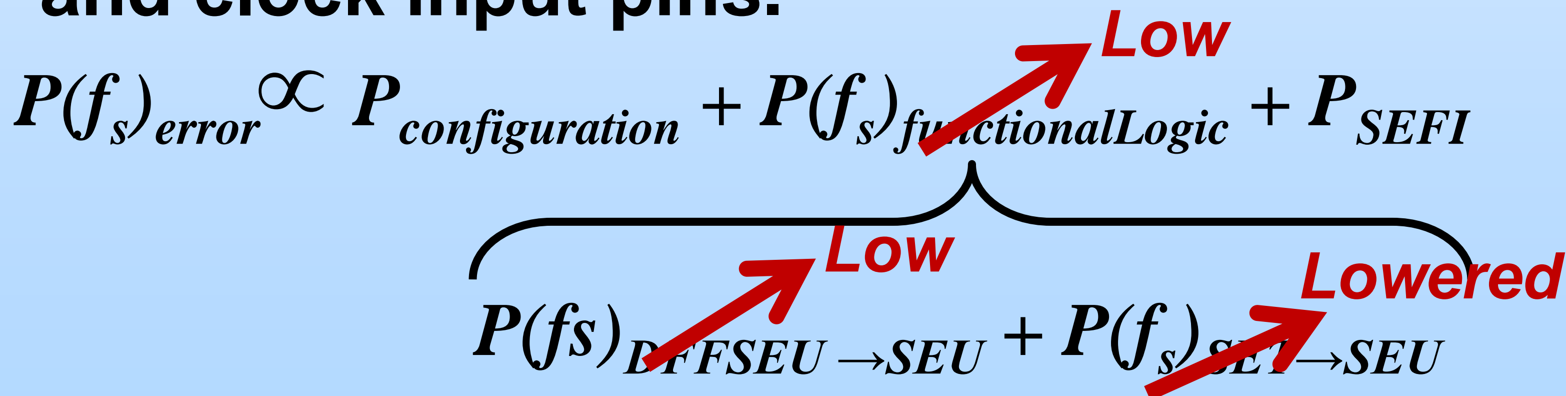


Reiterating…TR Is Only Efficient As An Embedded Scheme

- User should not attempt to implement TR in the fabric of an FPGA.
- The manufacturer place&route tool will create non-deterministic filtration when TR is implemented in the FPGA user fabric.
- This also makes the mitigation very difficult to verify.
- However, when TR is embedded (in a cell), it’s routing is internal to a cell and hence is deterministic.
- In addition, this scheme (when implemented in user fabric) will drastically affect critical path timing.

Combining Embedded Schemes

- **Some Radiation Hardened by Design (RHBD) schemes combine embedded temporal redundancy with localized redundant latches:**
  - **TR+LTMR, or**
  - **TR+DICE.**
- **New Xilinx RHBD FPGA (Virtex 5QV) has embedded TR+DICE. They refer to TR as SET filters. The SET filters are placed at the DFF data and clock input pins.**

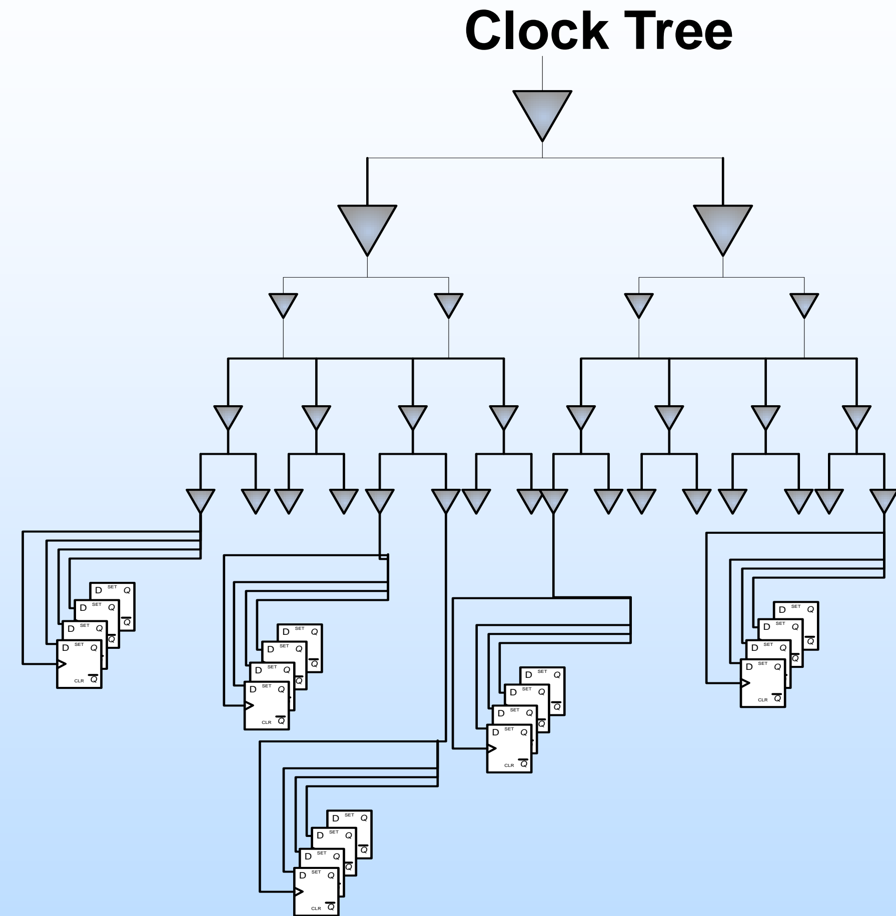


$$P(f_s)_{error} \propto P_{configuration} + P(f_s)_{functionalLogic} + P_{SEFI}$$

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

RHBD for Global Routes

- Some RHBD FPGAs contain hardened clock trees and other global routes.
- Global structures are generally hardened by using larger buffers.

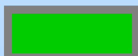


Helps to reduce P_{SEFI}

Recommeded TMR Mitigation Strategies per FPGA Type

- It is up to the designer to understand which type of TMR to implement based on the target FPGA and the target space environment.
- Tools are available but should not be blindly used as a push button solution.

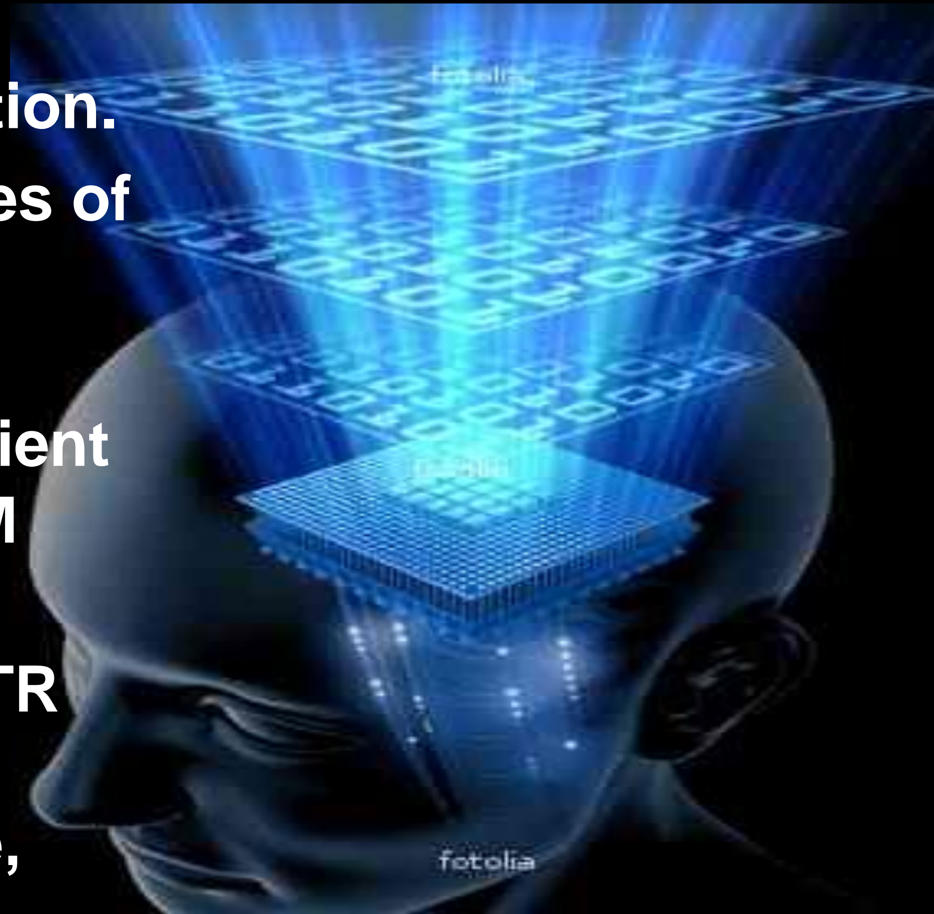
FPGA	LTMR	DTMR	GTMR
Antifuse			
Antifuse+LTMR			
Commercial SRAM			
Flash			
Xilinx V5QV			

 General Recommendation

Not Recommended but may be a solution for some situations

Will not be a good solution

Take Away Points: User Insertion of Mitigation



- Some FPGA devices contain embedded RHBD SEU mitigation.
- Understand the susceptibilities of the selected FPGA prior to inserting TMR.
- Only DTMR or GTMR is sufficient enough for commercial SRAM devices.
- Users should not insert TR. TR should only be used as an embedded mitigation scheme,
- Flash FPGA devices can benefit from LTMR with error-cross sections (at low LETs) that will approach an RTAXs device. However, at higher LETs, global upsets will dominate.

Agenda

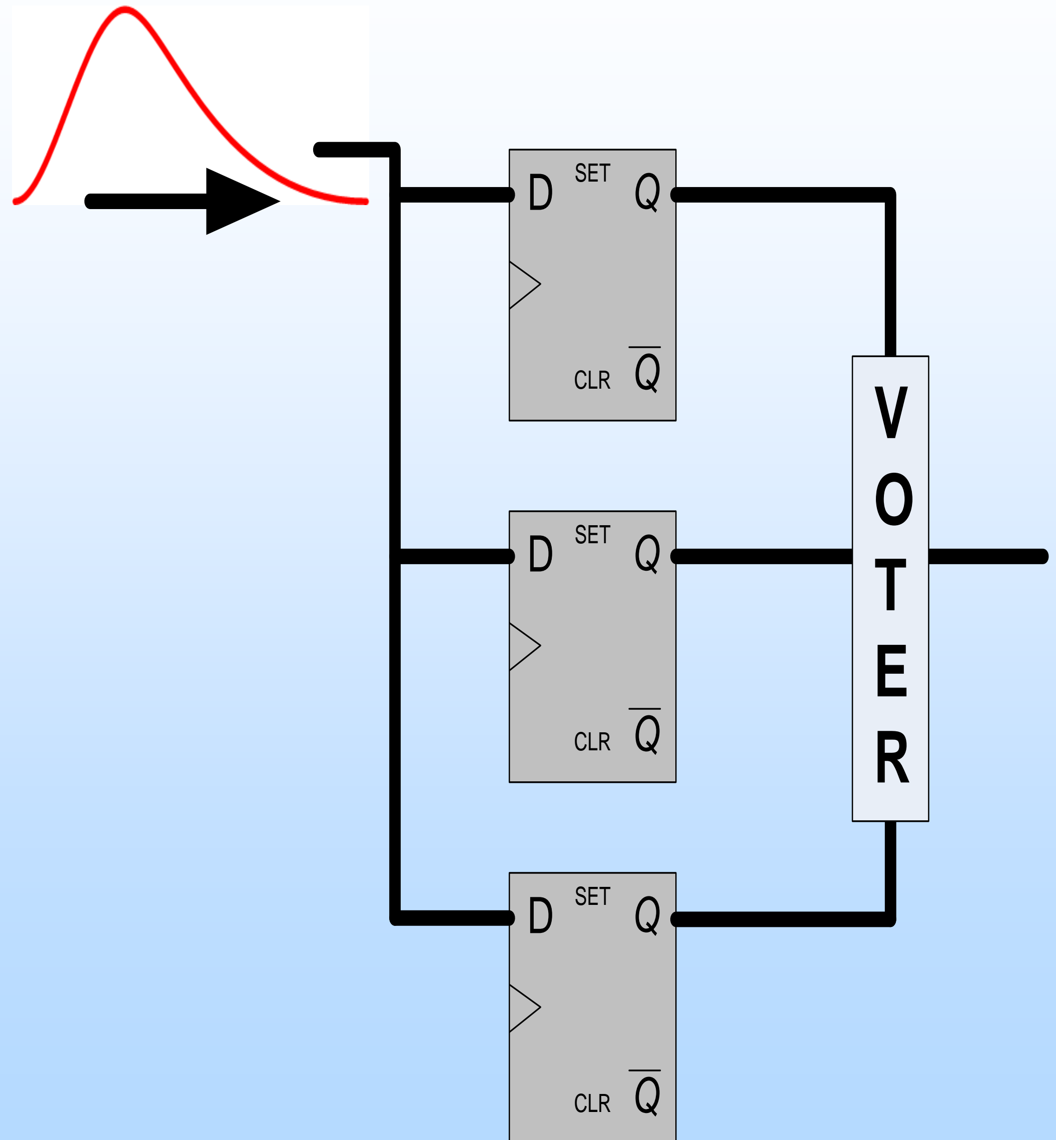
- Section I: General FPGA Description and Design Process.
- Section II: Single Event Effects (SEEs) in Digital Logic.
- Section III: Application of the NASA Goddard Radiation Effects and Analysis Group (REAG) FPGA SEU Model.
- Section IV: Reducing System Error: Common Mitigation Techniques.
- **Section V: When Your Mitigation Fails:**
  - **Types of mitigation failures.**
  - Measuring Mitigation Strength.
- Section VI: Xilinx Virtex Series and Mitigation.

Types of Mitigation Failures

All of the mitigation strategies presented reduce upset rates, however they all have modes of failure. These modes of failure must be recognized to accommodate for mitigation limitations.

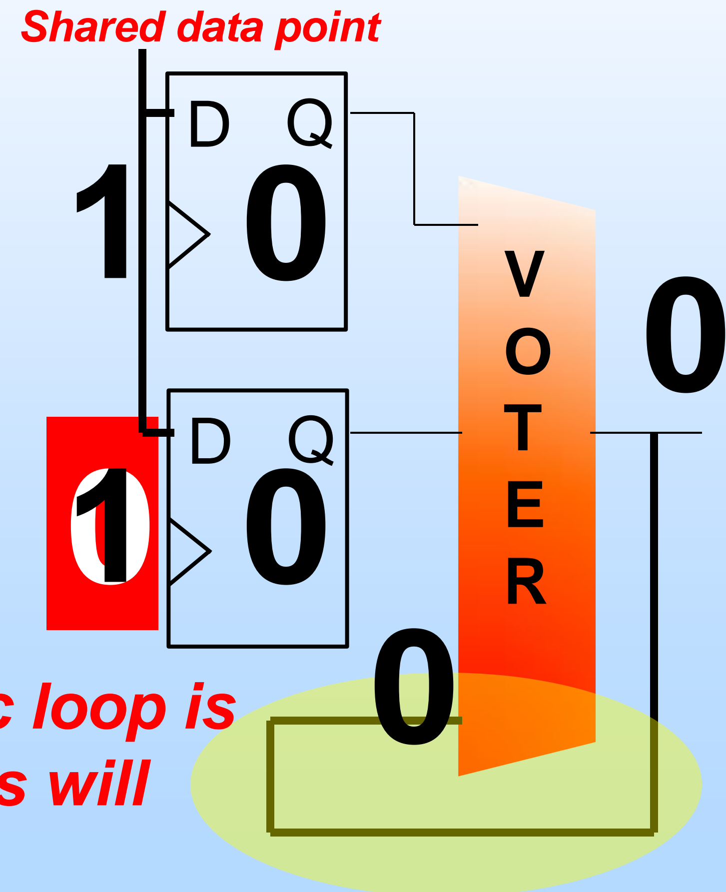
LTMR Failure

- Shared Data Path into DFFS.
- Voters can upset.
- Global routes.



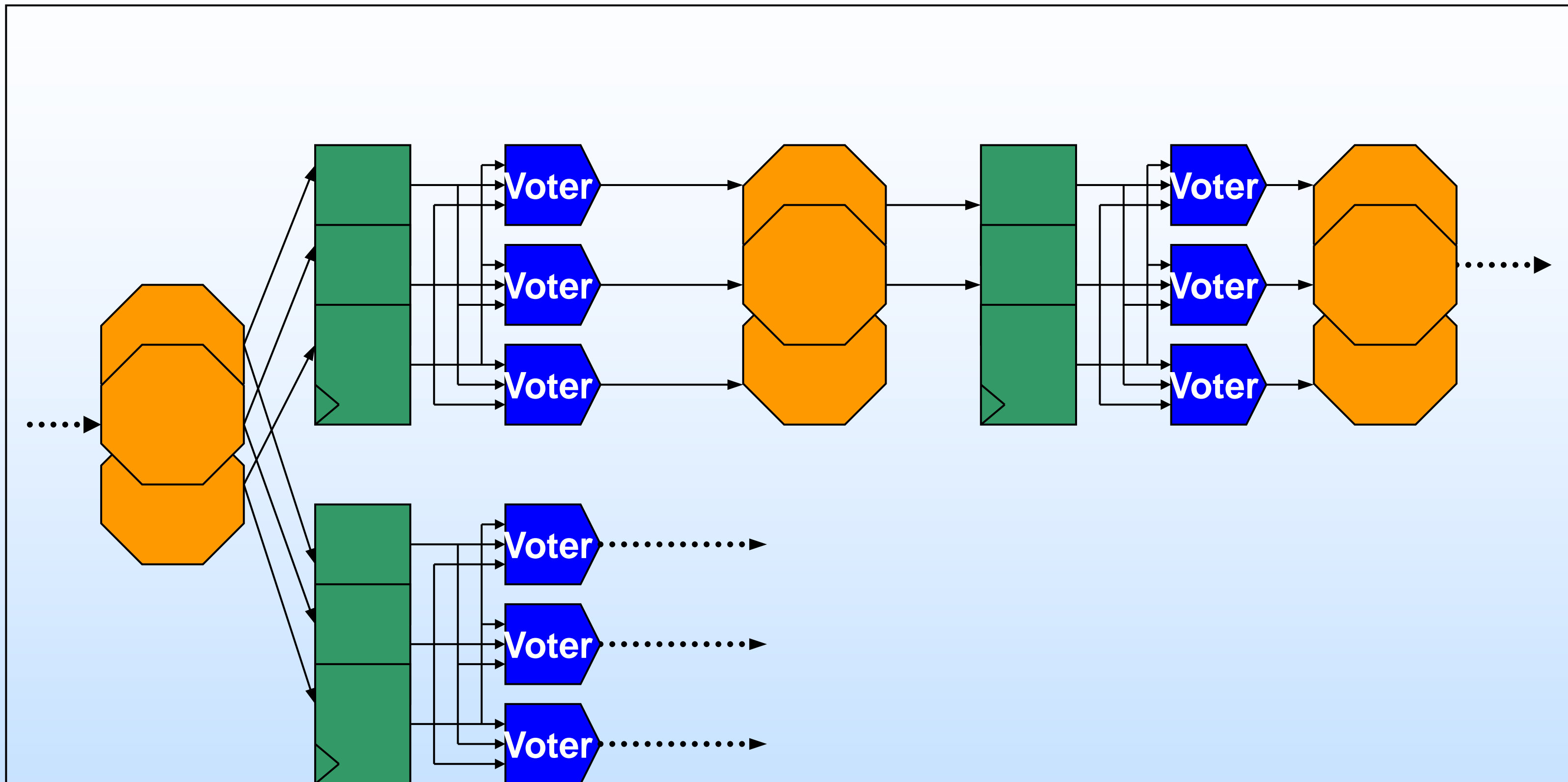
Localized Dual Modular Redundancy (LDMR)

- Not previously listed because it is in violation of synchronous design.
- However, its been considered by some organizations (outside of USA).
- Susceptibilities:
  - Shared data path (as with LTMR).
  - Transient from Master can get caught by Slave… this will not get voted out… susceptibility is increased from LTMR



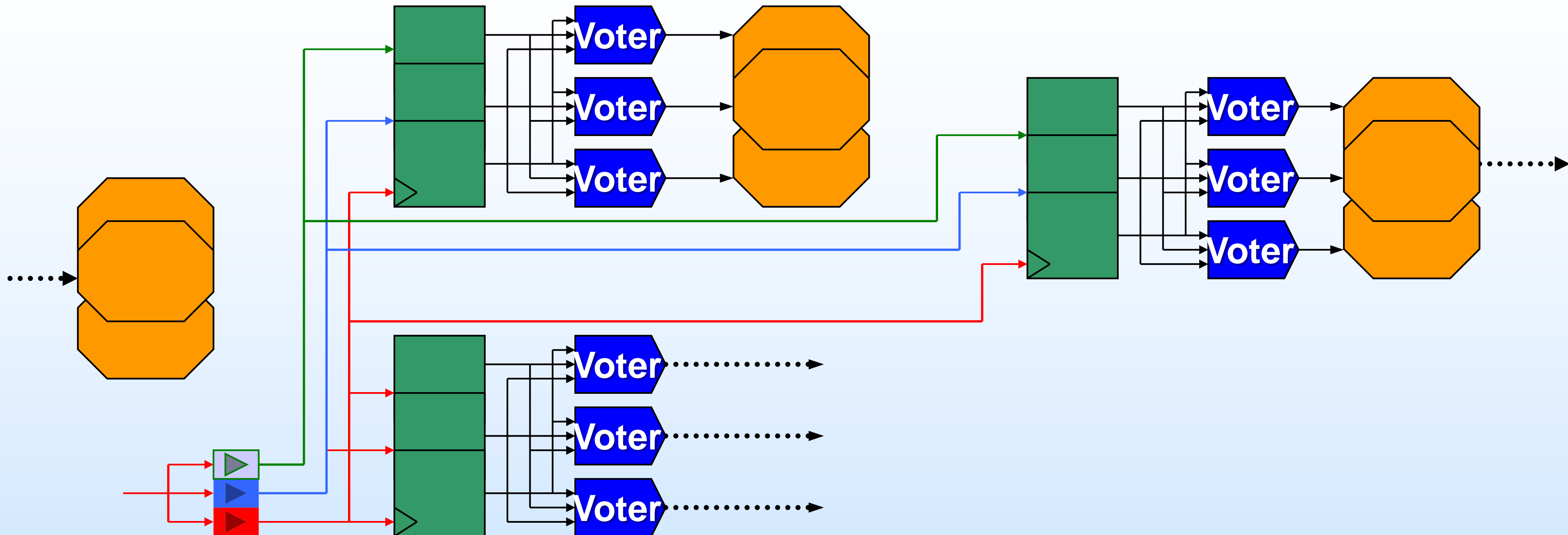
Most importantly, Combinatorial logic loop is illegal for synchronous design… tools will have difficulty.

DTMR Failures



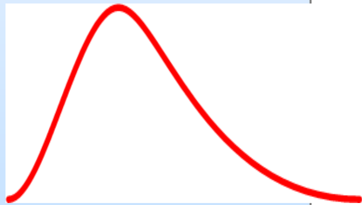
- Global routes.
- Domain placement:
  - Possible for domains to share common routing matrix.
  - Hit to shared routing matrix can take out two domains.

GTMR Failures



- **Domain placement:**
  - **Possible for domains to use shared resources.**
  - **Hit to shared routing matrix can take out two domains.**
- **Clock Skew (one path can be out of sync… lose correction capability).**
- **Asynchronous clock domain crossings need additional voter insertion – tools do not auto handle.**

DICE Susceptibility: Same as LTMR… SET on Data Input Can Get Caught at Clock Edge



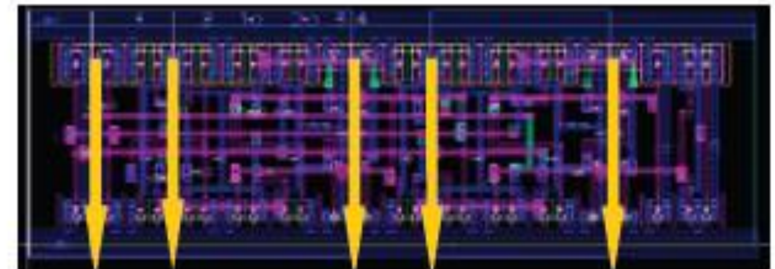
CKN
D
CK
CK
CKN
CK
CKN
Q
QN

Deliverable to NASA Electronic Parts and Packaging (NEPP) Program to be published on nepp.nasa.gov.

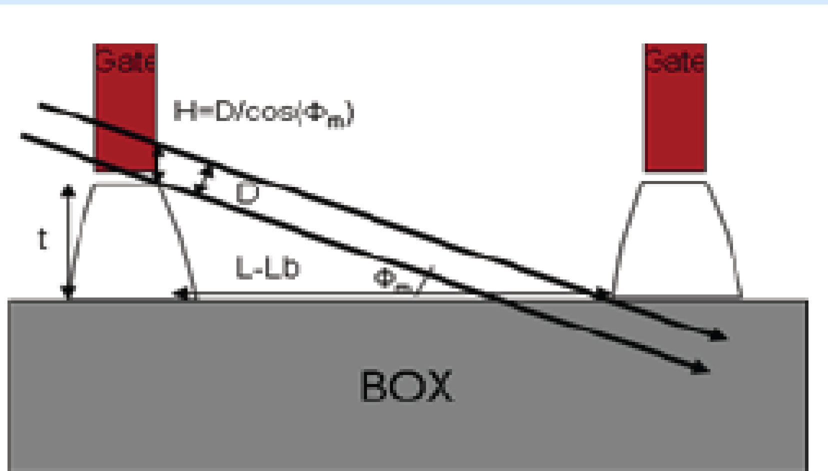
DICE Susceptibility

- One particle strike can take out 2 nodes and break Dice.
- Requires either angular particle strikes or charge sharing between DICE nodes.

Source: "Radiation Hard by Design at 90nm" ; Warren Snapp et. al, MRQW December 2008



Minimum spaced DICE flip flop
(lines show critical node)



Multiple bit upset ion strike simplified geometric model

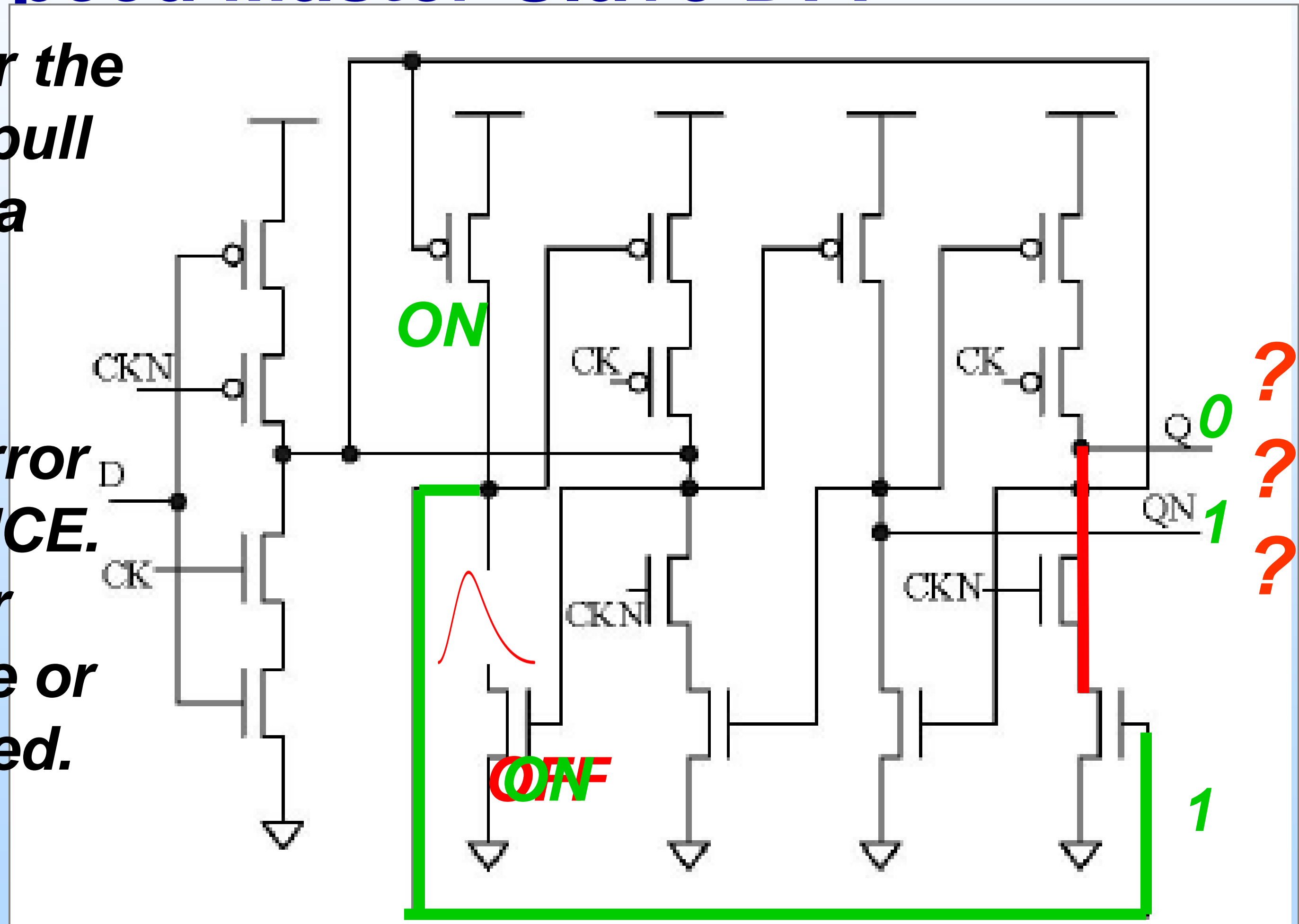
However, There Exists Another Problem with DICE Master-Slave DFFs

- Many radiation tests are performed on DICE latches or DICE DFFs that are not in a fully synchronous topology.
- Results from these tests will not expose the following problem: SET's generated in the Master stage that are caught by the Slave stage:
  - Probability of capture increases with frequency.
  - Probability of capture increases with node stability during SET generation; i.e., during an active SET, how quickly can the sister DICE node correct the SET.
- This is NOT angular dependent and does NOT depend on charge sharing… i.e., this is not a problem of two DICE nodes being affected by one SET.
- This scenario of DICE susceptibility concerns time to correction and Slave capture.

DICE Susceptibility: Works for a SRAM Cell – However, Can Cause Metastability Problems or Capture SETs in a High Speed Master-Slave DFF

Takes time for the dual node to pull the output to a correct state.

There is no error masking in DICE. Either wait for upset to settle or it gets captured.



DICE Susceptibility: Works for a SRAM Cell – However, Can Cause Metastability Problems or Capture SETs in a High Speed Master-Slave DFF (Continued)

Takes time for the dual node to pull the output to a correct state.

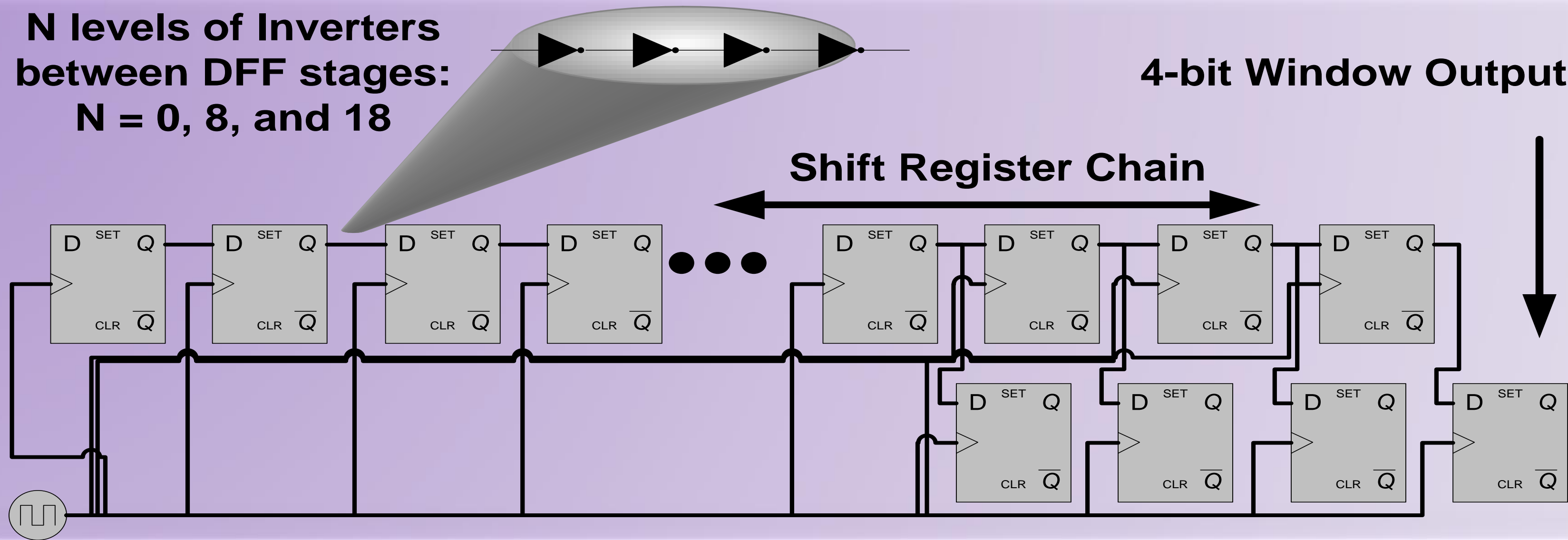
There is no error masking in DICE. Either wait for upset to settle or it gets captured .

First reported by:

Weizhong Wang and Haiyan Gong, "Edge Triggered Pulse Latch Design With Delayed Latching Edge for Radiation Hardened Application", 3626 IEEE TRANSACTIONS ON NUCLEAR SCIENCE, VOL. 51, NO. 6, DECEMBER 2004.

NASA Goddard Radiation Effects and Analysis Approach to Measuring Mitigation Strength: Is your mitigation working as expected… or at all?

Radiation Test Structures: Measuring CL Contributions to σ_{SEU} using Shift Registers



$$\tau_{dly_{wsr\ 8}} > \tau_{dly_{wsr\ 0}}$$

$\tau_{dly_{wsr\ 0}}$

WSR$_0$

$\tau_{dly_{wsr\ 8}}$

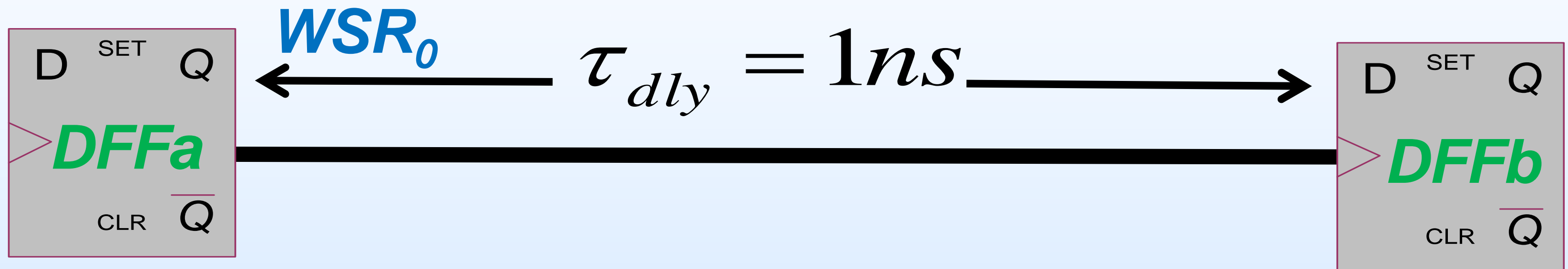
WSR$_8$

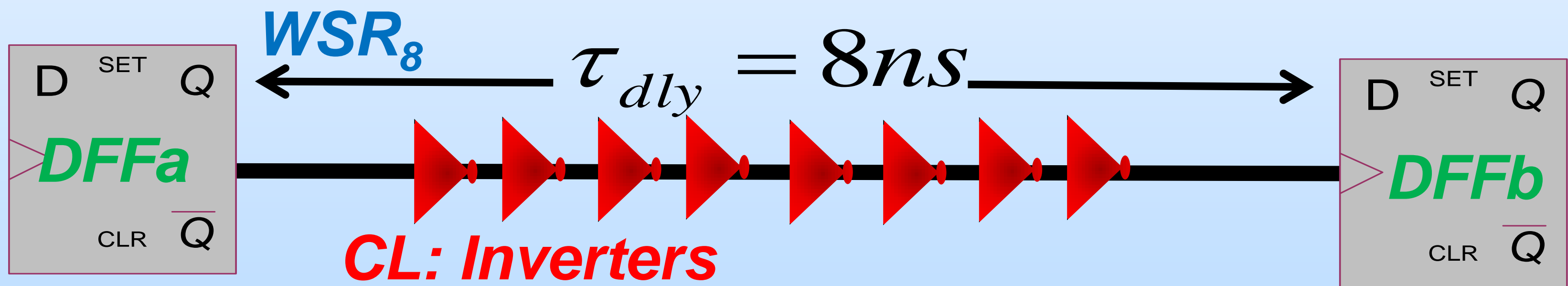
CL: Inverters

Which String Would You Expect to Have a Higher SEU Cross Section? WSR_0 or WSR_8

Startpoint

Endpoint



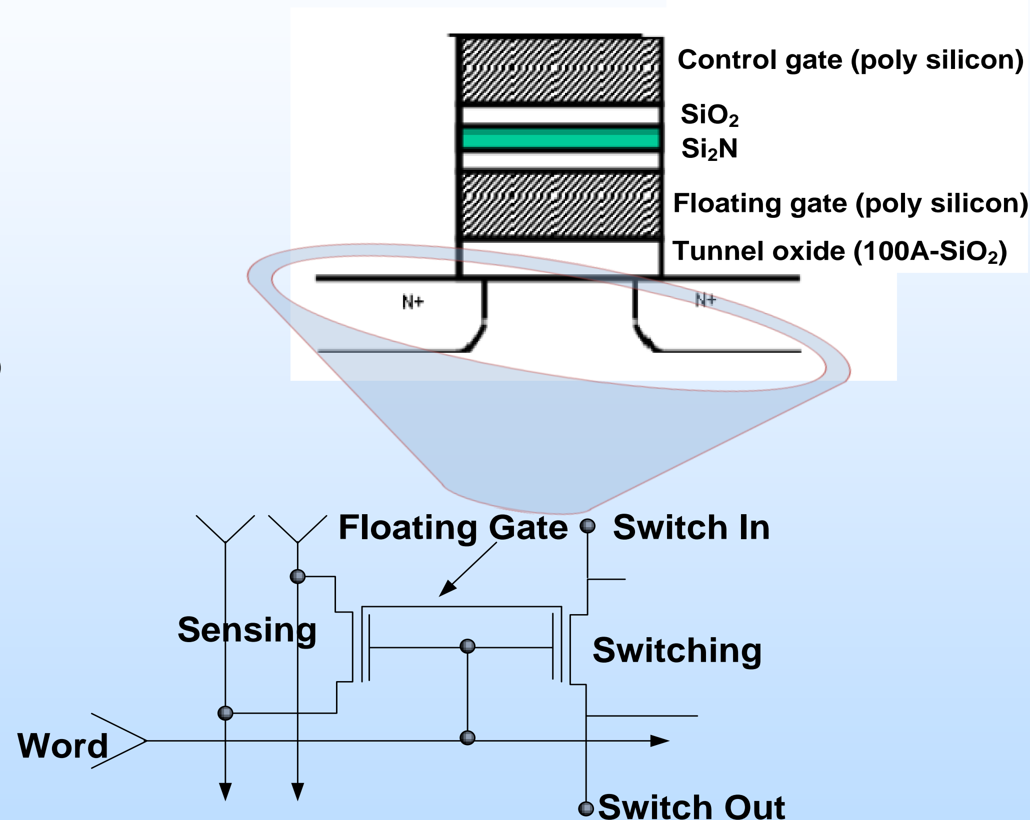


You can't answer the question until you understand the relative σ_{SEU} contribution of DFFs to CL

Is there Logic Mitigation?

Example: Micro-Semi (Actel) ProASIC3 Flash Based FPGA

- Originally a commercial device.
- Configuration is flash based and has proven to be almost immune to SEUs.
- No embedded mitigation in device.
- User must insert mitigation if σ_{SEU} reduction is required.
- DFFs are more susceptible than CL.



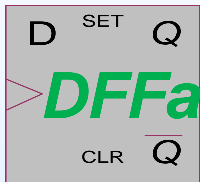
No-TMR ProASIC3: Which String Would You Expect to Have a Higher SEU Cross Section? WSR_0 or WSR_8

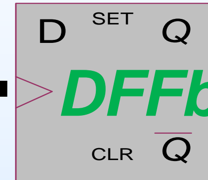
StartPoint

WSR_0

$\tau_{dly} = 1ns$

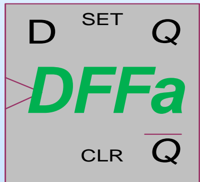
EndPoint

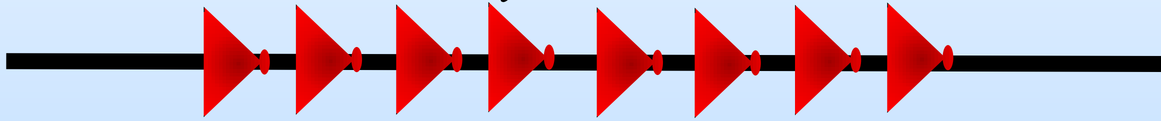


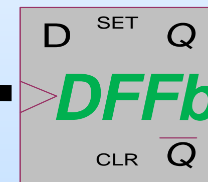


WSR_8

$\tau_{dly} = 8ns$







$$P(fs)_{functionalLogic} \propto \underset{DFF}{\exists}\left(\left(\sum_{j=1}^{1} P_{DFFSEU(j)}(1 - \tau_{dly(j)} fs) \right) \right)$$

No-TMR: DFFs are most susceptible

σ_{SEU} is inversely proportional to τ_{dly}

σ_{SEU} WSR_0 > σ_{SEU} WSR_8

ProAsic3 σ_{SEU} Test Results: Windowed Shift Registers (WSRs) No-TMR

- **No-TMR:** $\sigma_{SEU}\ WSR_0 > \sigma_{SEU}\ WSR_8$ **For every LET**
- **No-TMR: Increasing CL in a data path does not increase σ_{SEU} because increase in τ_{dly}**

No-TMR100MHz Checkerboard LET Versus SEU Cross Section

$P_{DFFSEU}(1-\tau_{dly}fs)$

Legend: WSR N=8; WSR N=0

Y-axis: σ_{SEU} (cm²/bit) — 0.0E+00, 5.0E-08, 1.0E-07, 1.5E-07, 2.0E-07, 2.5E-07

X-axis: LET MeV*cm²/mg — 2.80, 3.96, 8.60, 12.16, 20.30, 28.71

What Does This Mean?

- The system topology is controlling the σ_{SEU} trend:
  - SEUs in DFFs (P_{DFFSEU}) are masked by the delay between StartPoint to EndPoint.
  - Increase in CL increases Delay in the path.
  - The trend is opposite of what we expect… one would believe if you add more logic – you should increase the σ_{SEU}.
- CL trend was consistent across all LETs for all WSR strings as we increased CL.
- DFF upsets are dominant in non-mitigated synchronous design paths.
- Accordingly, we should also see the σ_{SEU} decrease as frequency increases.

$$P(fs)_{functionalLogic} \propto \underset{DFF}{\exists}\left(\left(\sum_{j=1}^{1} P_{DFFSEU(j)} \left(1 - \tau_{dly(j)} fs\right)\right)\right)$$

ProASIC3 User Inserted LTMR with WSR Chains

WSR_0

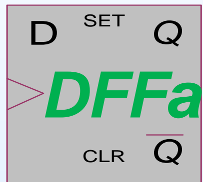
WSR_8

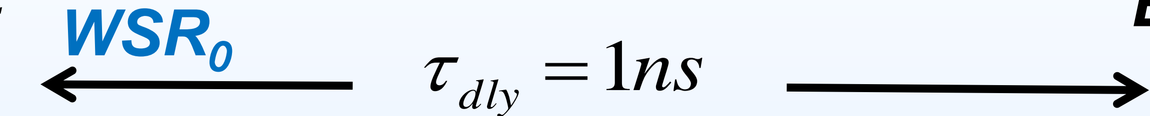
Automation is the best approach for insertion.

LTMR ProASIC3: Which String Would You Expect to Have a Higher SEU Cross Section? WSR_0 or WSR_8

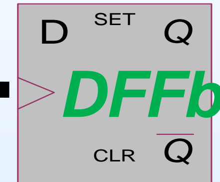
StartPoint

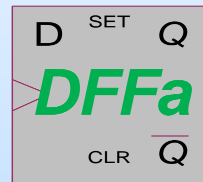
EndPoint

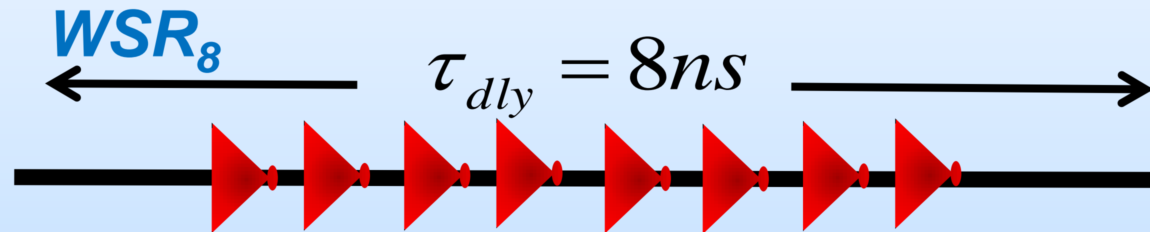




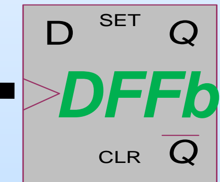
$\tau_{dly} = 1ns$







$\tau_{dly} = 8ns$



$$P(fs)_{funtionalLogic} \propto \sum_{i=1}^{\#CombinatorialLogicGates} P(fs)_{SET \rightarrow SEU(i)} \propto P_{gen(i)} P_{prop(i)} P_{\text{logic}} \tau_{width(i)} fs$$

As we increase #combinatorial logic gates we increase σ_{SEU}

Hence for LTMR (disregarding P_{prop}),

$\sigma_{SEU}\ WSR_8 > \sigma_{SEU}\ WSR_0$

ProASIC3 σ_{SEU} Test Results: Trend Reverses with LTMR – Directly Proportional to Amount of CL

- **LTMR is effective and has mitigated** $P(fs)_{DFFSEU \rightarrow SEU}$.
- **LTMR:** σ_{SEU} $WSR_0 < \sigma_{SEU}$ WSR_8 **For every LET.**
- **Increasing CL in the data path increases** σ_{SEU} **.**

No-TMR and LTMR 100MHz Checkerboard LET Versus SEU Cross Section

$P_{gen} P_{prop} \tau_{width} fs$

WSR N=8 No-TMR
WSR N=0 No-TMR
WSR N=8 LTMR
WSR N=0 LTMR

σ_{SEU}(cm²/bit)

1.0E-06
1.0E-07
1.0E-08
1.0E-09
1.0E-10

2.80
3.96
8.60
12.16
20.30
28.71

LET MeV*cm²/mg

Another Look at No-TMR versus LTMR with the ProASIC3… Regard the Frequency Trends

LET = 20.3 NoTMR versus LTMR- checker pattern

Cross Section (cm²/bit)

1.6E-07
1.4E-07
1.2E-07
1.0E-07
8.0E-08
6.0E-08
4.0E-08
2.0E-08
0.0E+00

0.00E+00
5.00E+07
1.00E+08
1.50E+08
2.00E+08
2.50E+08

Frequency (Hz)

NoTMR WSR_0

$P_{DFFSEU}(1-\tau_{dly}fs)$

NoTMR WSR_8

No TMR WSR_{16}

LTMR WSR_{16}

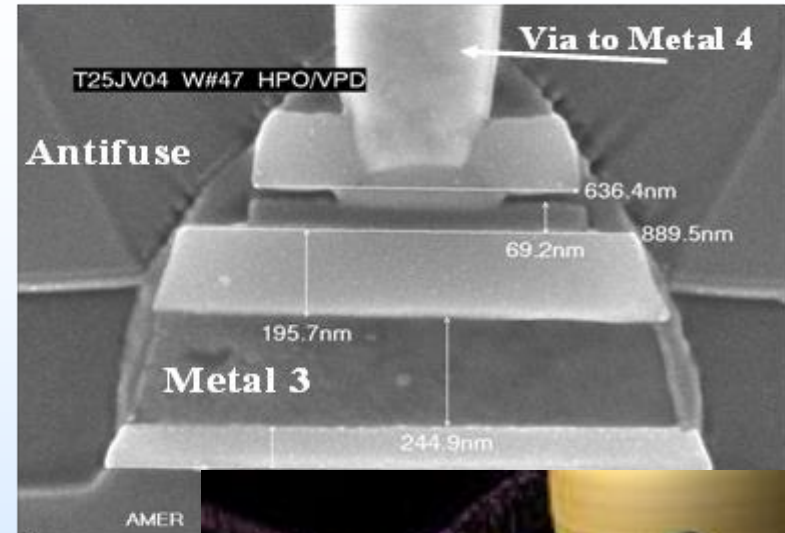
LTMR WSR_8

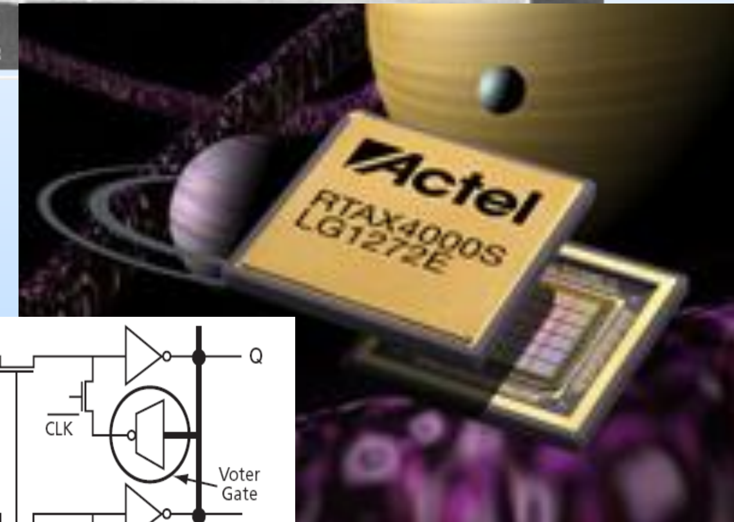
$P_{gen}P_{prop}\tau_{width}fs$

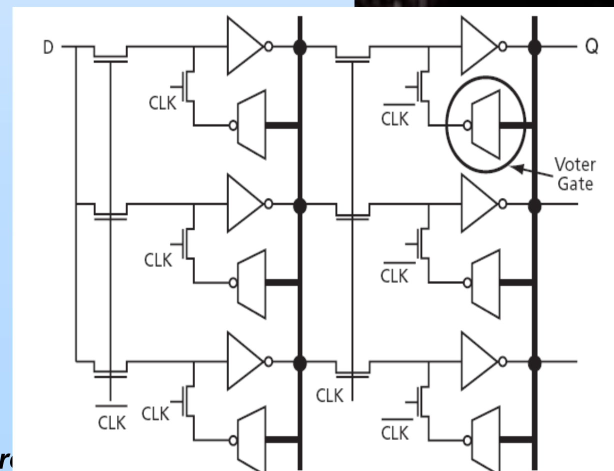
LTMR WSR_0

Microsemi RTAXs

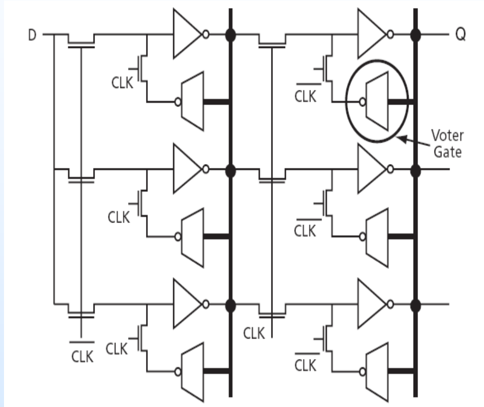
- **Made for Space!**
  - **Antifuse Configuration:** $P_{configuration} \rightarrow 0$.
  - **Embedded LTMR at each DFF.**
- **Due to LTMR mitigation, CL is the most susceptible.**
- **New RTAX4000D:**
  - **Has major processing power.**
  - **Hardened DSP Blocks.**
  - **Less susceptible than RTAX2000s.**

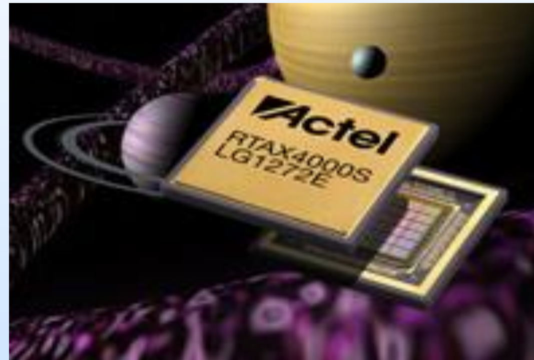


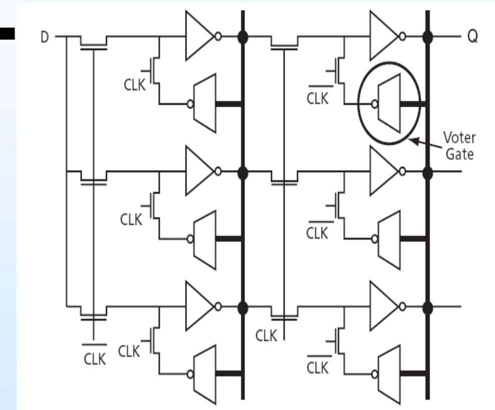


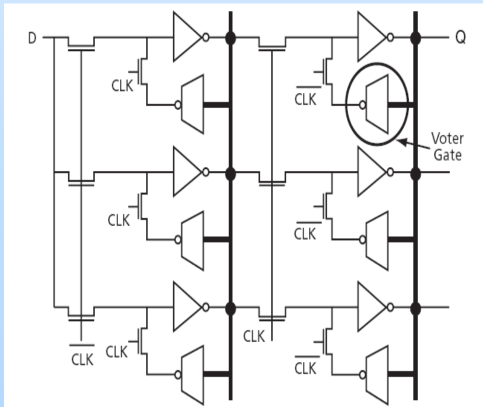


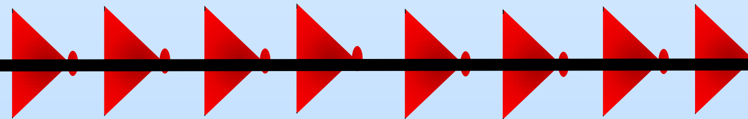
RTAX Shift Register Test Structures

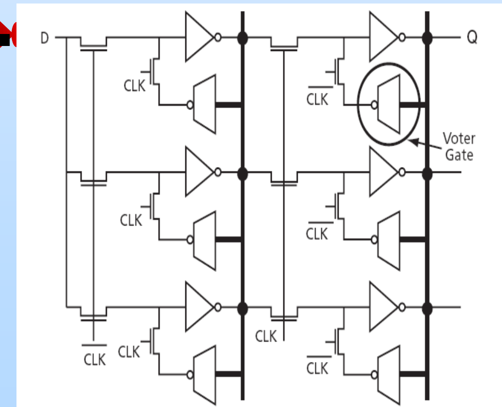






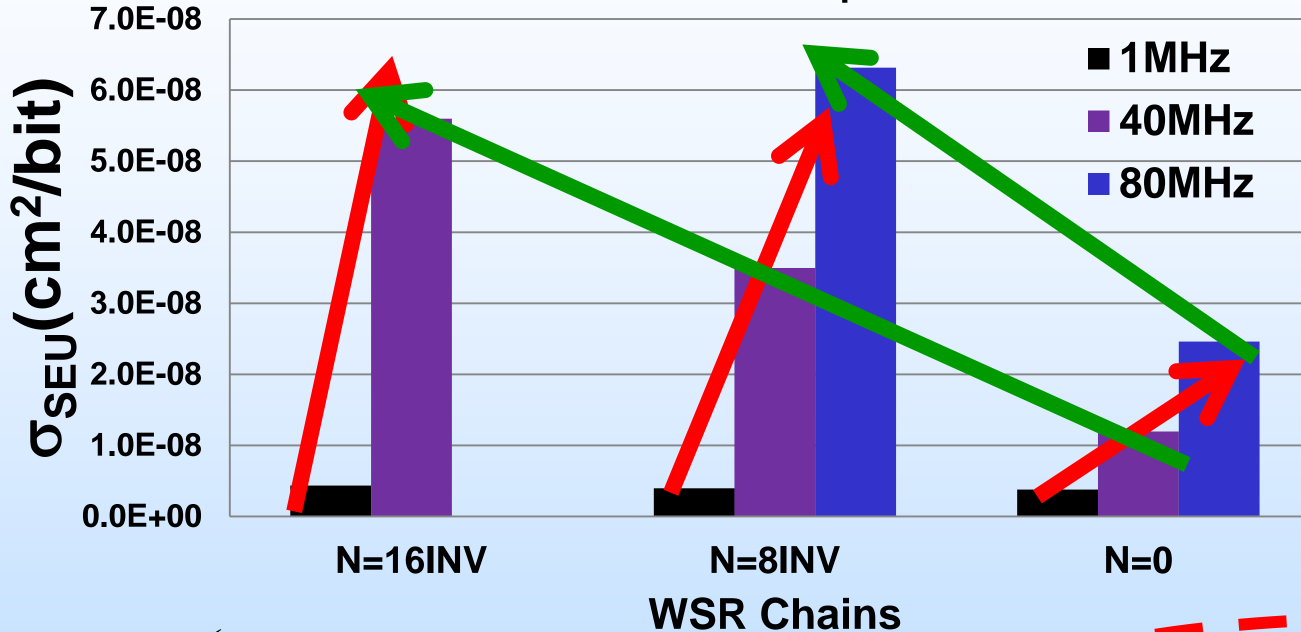






Inverters are mapped into CCELLs (RTAXs combinatorial logic cells).

Microsemi RTAX2000s: As Frequency Increases, σ_{SEU} Increases



LTMR DFFs: 150nm CMOS

$$\exists_{DFF}\left(\left(\sum_{j=1}^{\#StartPointDFFs} P_{DFFSEU(j)}(1-\tau_{dly(j)} fs) P_{logic(j)} \right) + \sum_{i=1}^{\#CombinatorialCells} (P_{gen(i)} P_{prop(i)} P_{logic} \tau_{width(i)} fs) \right)$$

Increase Frequency→ Increase Cross section

Increase CL→ Increase Cross section

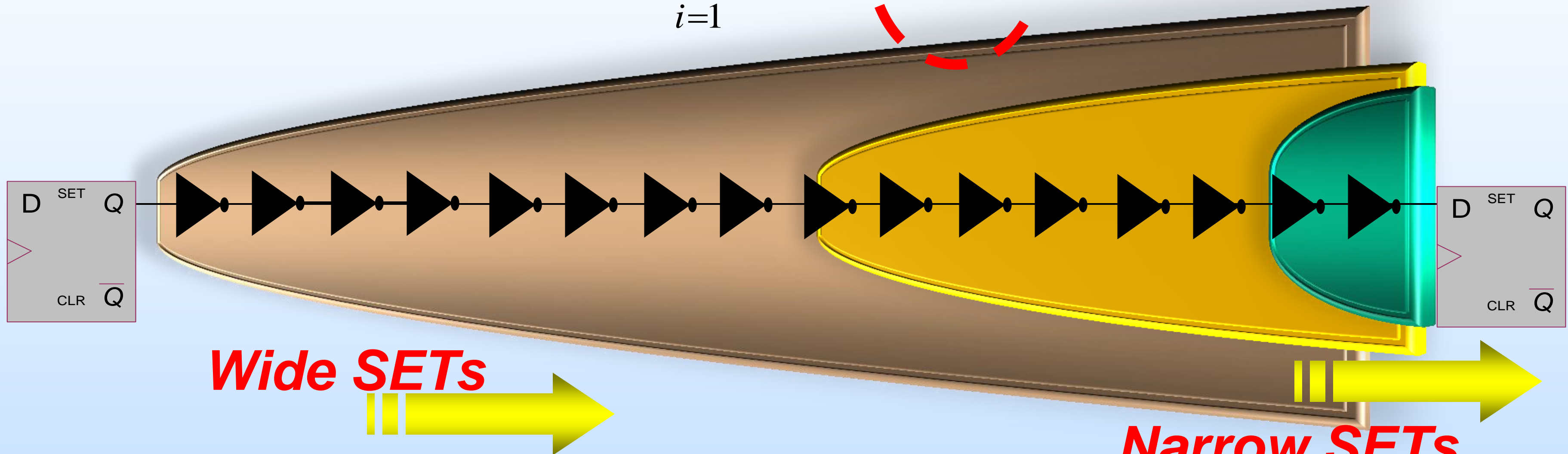
Upsets are dominated by SETs (due to LTMR).

Conclusion: LTMR DFFs have strong mitigation.

Deliverable to NASA Electronic Parts and Packaging (NEPP) Program to be published on nepp.nasa.gov.

P(fs)$_{SET \rightarrow SEU}$ is Non-Linear with respect to Serially Adding CCELL logic … P_{prop}

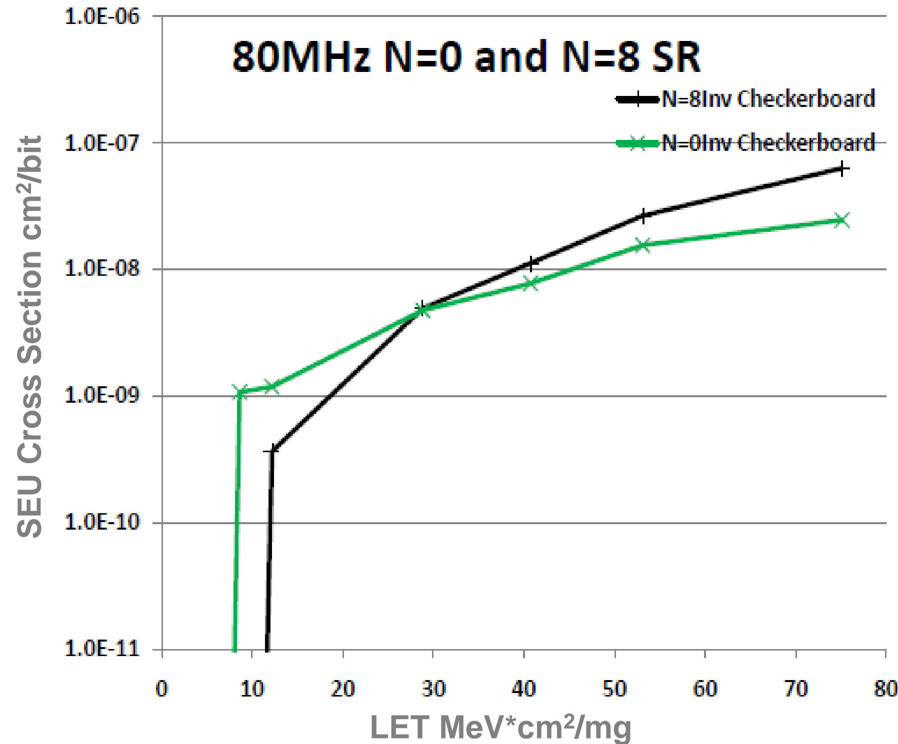
$$P(fs)_{SET \rightarrow SEU} \propto \sum_{i=1}^{NumberofCCells} P_{gen(i)} P_{prop(i)} \tau_{width} fs$$

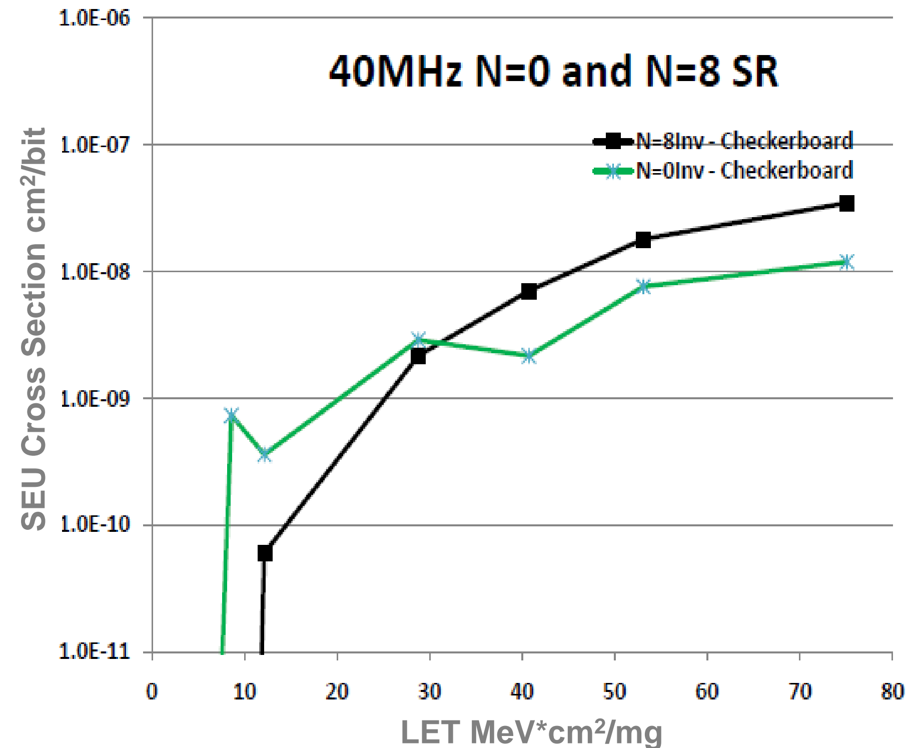


- **Higher LETs** → **Wider SETs Propagate .with sufficient amplitude→ More CCELLS contribute to upsets.**
- **Low LETs→ some SETs get attenuated by CCELLs, hence SRs with a small number of CCELLS can have a slightly larger cross section at low LET.**
- **Non-Linear: Depending on the CCELL configuration and the CCELL output load, LET attenuation can result in varying SET shapes.**

LETs Affect SET Propagation and Consequently Affect SEU Cross Section

SR = Shift Register

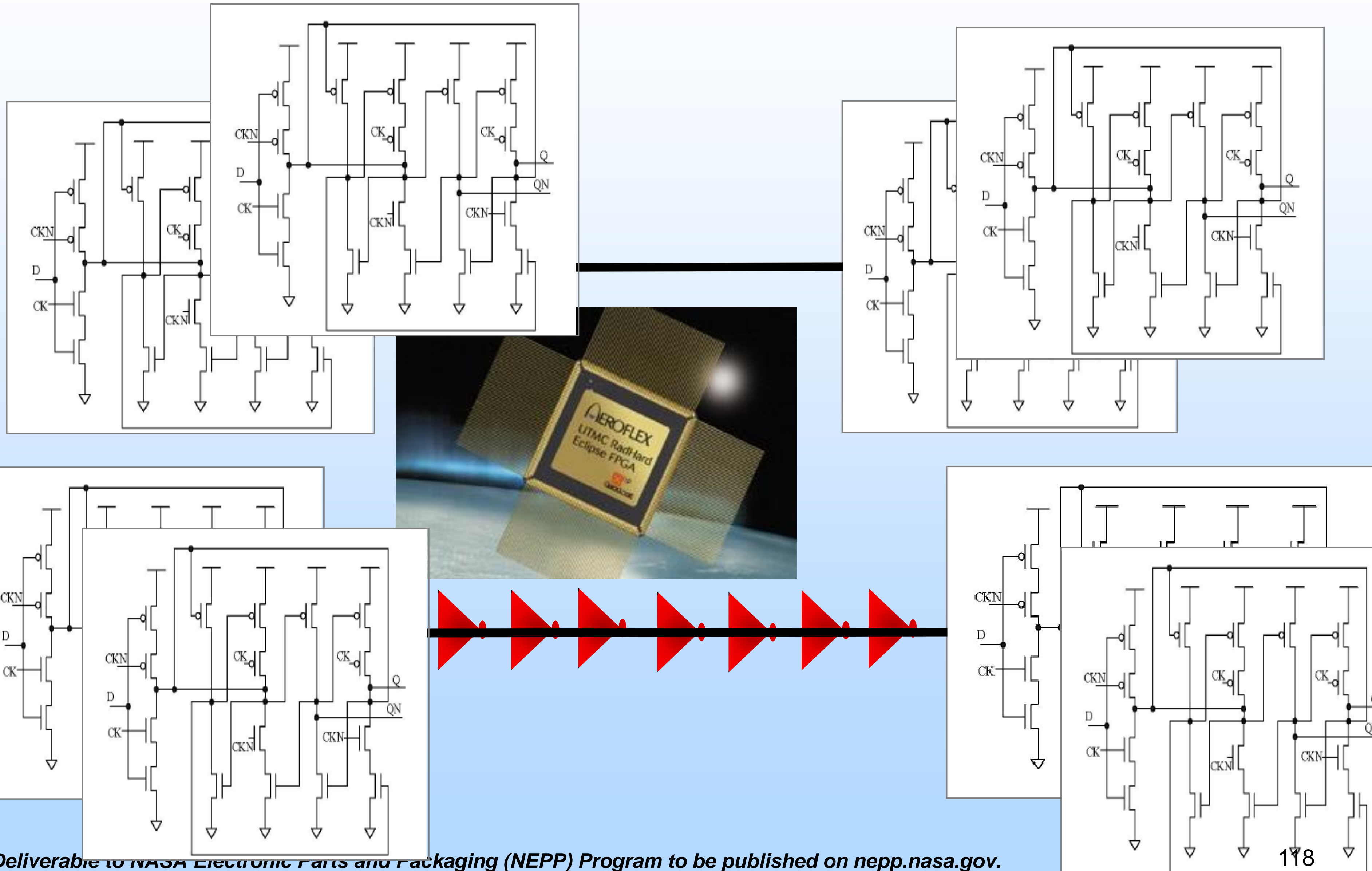




$$P(fs)_{SET \rightarrow SEU} \propto \sum_{i=1}^{NumberofCells} P_{gen(i)} P_{prop(i)} \tau_{width} fs$$

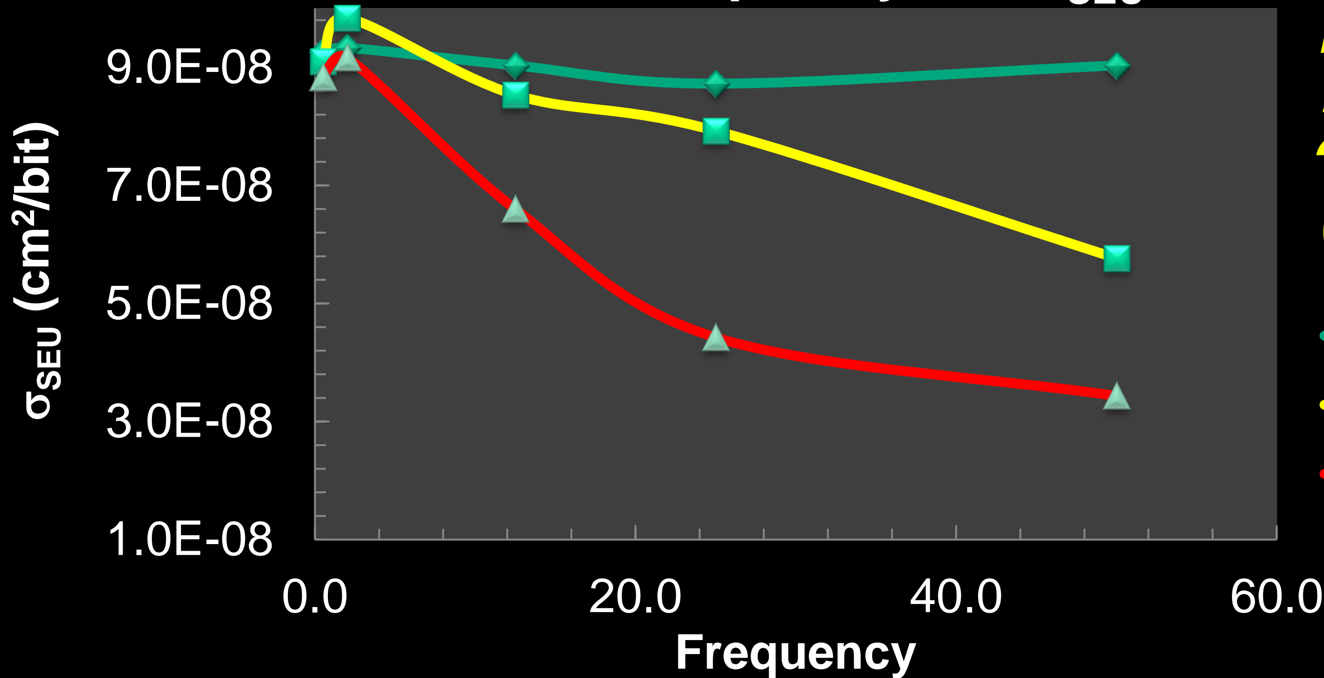
Lower LET values→ SR N=0 has higher SEU cross section than SR N= 8. Demonstrates SET filtration effects.

Aeroflex Eclipse FPGA uses DICE DFFs



Aeroflex Eclipse: As τ_{dly} Increases and Frequency Increases, σ_{SEU} Decreases

$$\underset{DFF}{\exists}\left(\left(\sum_{j=1}^{\#StartPointDFFs} P(fs)_{DFFSEU(j)} (1-\tau_{dly(j)} fs) P_{logic(j)} \right) + \sum_{i=1}^{\#CombinatorialCells} (P_{gen(i)} P_{prop(i)} P_{logic} \tau_{width(i)} fs) \right)$$

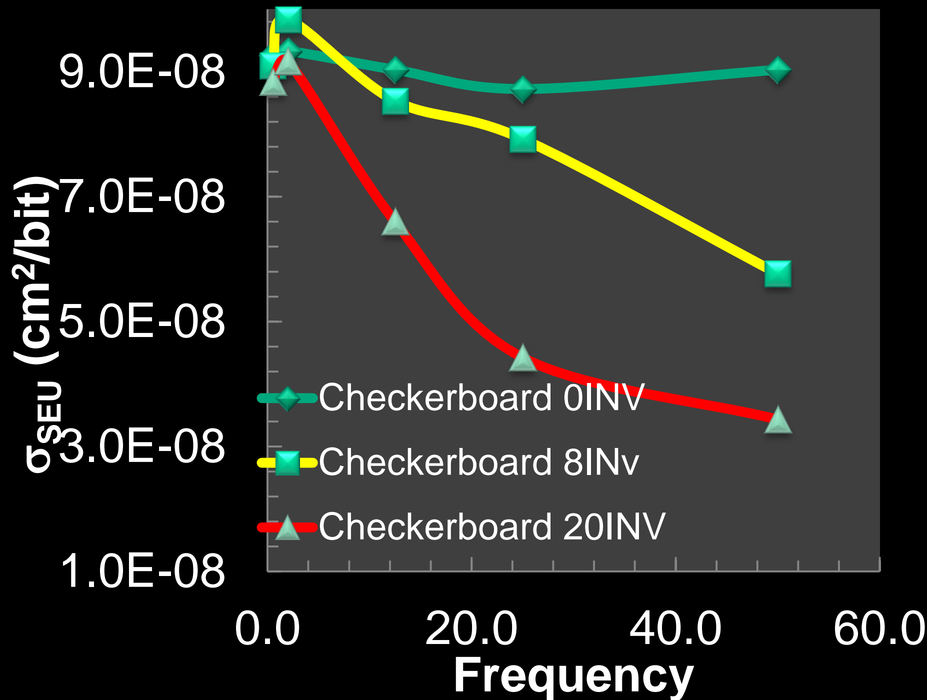


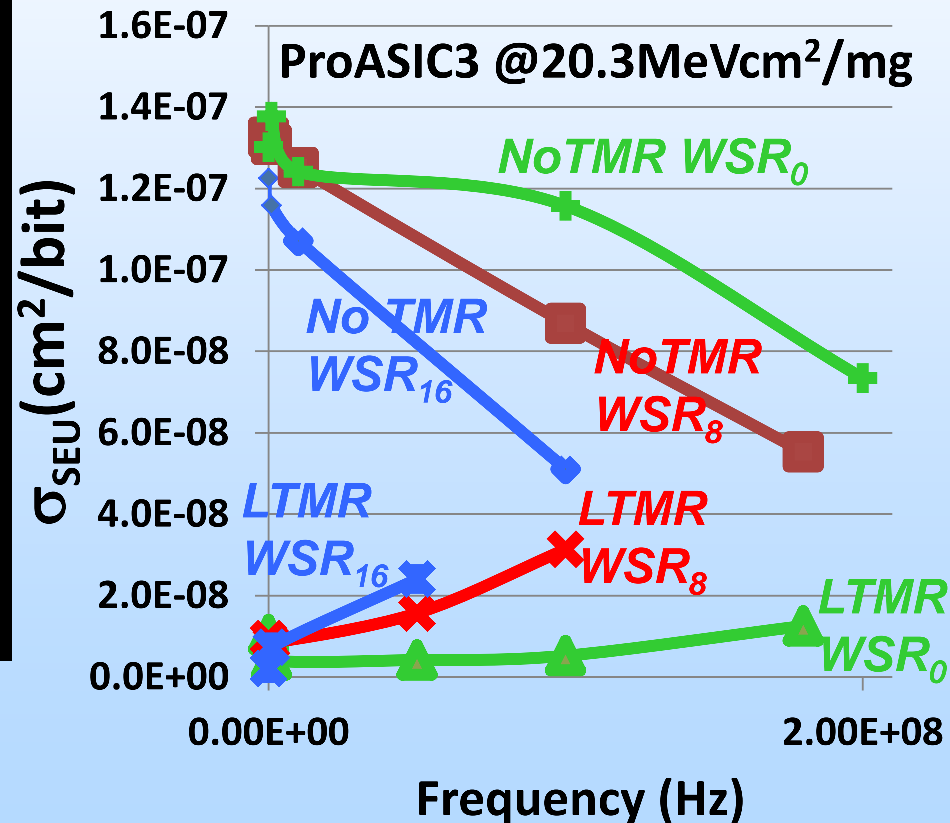
DICE DFFs: 250nm CMOS

ALL LETS are incident angle. Upsets are dominated by DFFs.

Conclusion: DICE DFFs do not have strong mitigation.

Comparing DICE to ProASIC3… Notice that DICE has the same trend as the No-TMR Strings

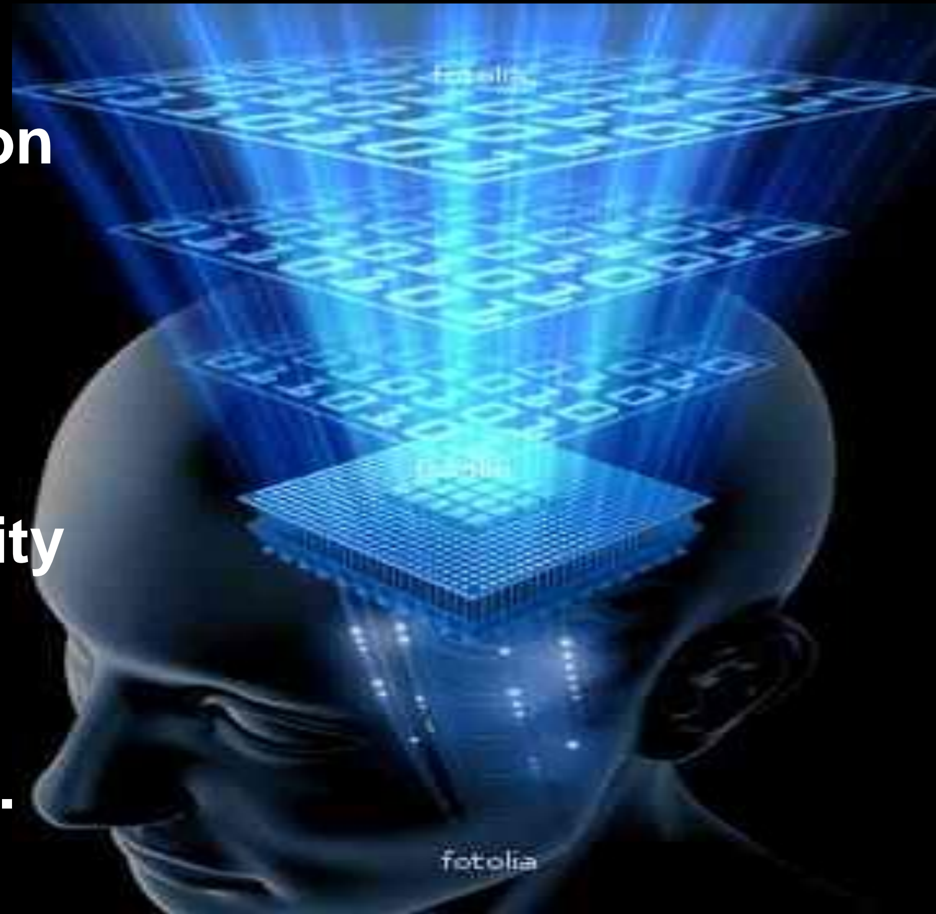




Note: DICE with 250nm and ProASIC3@130nm.

Take Away Points (1): Mitigation and Design Considerations

- Understand the target radiation environment:
  - Protons.
  - Heavy ions(LET range?).
- CMOS: As geometry and voltage decrease, susceptibility increases.
- TMR proves to be a better mitigation strategy over DICE.



- **Research** (modeling + LET curves) is currently being used to explain trends, mitigation strength, and supplies an upper-bound to upset rates… **It is not expected to supply exact upset rates per design.**

Take Away Points (2): Mitigation and Design Considerations

- Your safest and best design is your simplest design…
- Don't get twisted up in the details of research:
  - One example illustrates that adding more CL will reduce σ_{SEU}. Designer should not try this to reduce σ_{SEU}.
  - Design will be:
    - Too slow,
    - Too large,
    - Too much power, and
    - Too difficult to verify.

- Don't over or under design.
- If you need mitigation, then use a standard, proven approach (i.e. TMR).

Agenda

- **Section I: General FPGA Description and Design Process.**
- **Section II: Single Event Effects (SEEs) in Digital Logic**
- **Section III: Application of the NASA Goddard Radiation Effects and Analysis Group (REAG) FPGA SEU Model**
- **Section IV: Reducing System Error: Common Mitigation Techniques**
- **Section V: When Your Mitigation Fails**
- **Section VI: Xilinx Virtex Series and Mitigation**

Xilinx Virtex Series Mitigation: What Are The First Schemes That Come To Mind?

- TMR and Scrubbing.
- Questions that should be asked prior to implementation:
  - Is mitigation necessary?
  - What type of TMR? (LTMR: NO! ;DTMR? GTMR?).
  - Should scrubbing be used? Answer: only if TMR is used.
  - How well can the inserted mitigation be verified?

SRAM Configuration Susceptibility vs. Functional logic Susceptibility:

Xilinx Consortium: ***VIRTEX-4VQ STATIC SEU CHARACTERIZATION SUMMARY: April/2008***

	Probability	Error Rate	$\frac{Upsets}{device-day}$	$\frac{Upsets}{device-day}$
Configuration Memory: XQR4VSX55	$P_{configuration}$	$\frac{dE_{configuraiton}}{dt}$	7.43	4.2
Combined SEFIs per device	P_{SEFI}	$\frac{dE_{SEFI}}{dt}$	7.5×10^{-5}	2.7×10^{-5}

- ***For non-mitigated designs, Consortium data and NASA data show that configuration upsets are most significant.***

$$P_{error} \propto P_{Configuraiton}$$

SRAM Based FPGAs are Considered Highly Susceptible to SEUs

Must test with flux≈100's/particles/(s*cm^2) during accelerated radiation testing. Otherwise, non-mitigated designs stop operating as soon as the radiaiton beam in turned on(NASA Goddard REAG).

$$P(fs)_{error} \propto P_{Configuraiton}$$

No Mitigation

- **For efficient mitigation, configuration upsets must be masked**
  - **Functional redundancy.**
  - **GTMR is theoretically the best solution – but the most expensive and difficult to implement due to clock skew.**

GTMR: $P(fs)_{error} \propto P_{Configuraiton} + P_{DFFSEU} + P(fs)_{SET \rightarrow SEU} + P_{SEFI}$

Low Low Low Low

DTMR: $P(fs)_{error} \propto P_{Configuraiton} + P_{DFFSEU} + P(fs)_{SET \rightarrow SEU} + P_{SEFI}$

Low Low Low

What Does This Mean Regarding Error Rate Prediction?

- **Designs with no mitigation – the upper bound *dE/dt* is:**
  - **Based on the configuration upset rate.**
  - **Based on the number of configuration bits that can directly affect your circuit:** $P(fs)_{error} \mu P_{Configuration} * \#UsedConfigurationBits$
- **Designs with DTMR – the upper bound *dE/dt* is:**
  - **Based on the global route upset,**
  - **Based on SEFI (hidden logic), and**
  - **Based on scrubber efficiency:**

$$P(fs)_{error} \mu P_{GlobalRoutes} + P_{HiddenLogic}$$

- **Designs with GTMR – the upper bound *dE/dt* is:**
  - **Based on SEFI (hidden logic), and**
  - **Based on scrubber efficiency:** $P(fs)_{error} \mu P_{GlobalRoutes}$

Scrubbing Definition

- The configuration memory of un-hardened static random access memory (SRAM)-Based Field Programmable Gate Arrays (FPGAs) is highly susceptible to single event upsets (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.
- Two questions are addressed:
  - How often should we scrub?
  - What is the difference between scrubbing in a space environment and scrubbing in an accelerated single event test environment?

Misperception #1: Space Application Scrub Rates

- It is a common misperception that space applications scrub configuration constantly. They don't:
  - Power,
  - Area, and
  - Risk with additional logic complexity.
- It is important to understand:
  - When configuration memory scrubbing is necessary and if so…
  - How often configuration memory scrubbing should occur (scrub rate).
- We use SEU cross sections (σ_{SEU}) to determine required scrub rates.

Scrubbing…FPGA SEU Categorization as defined by NASA Goddard Radiation Effects and Analysis Group (REAG):

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

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

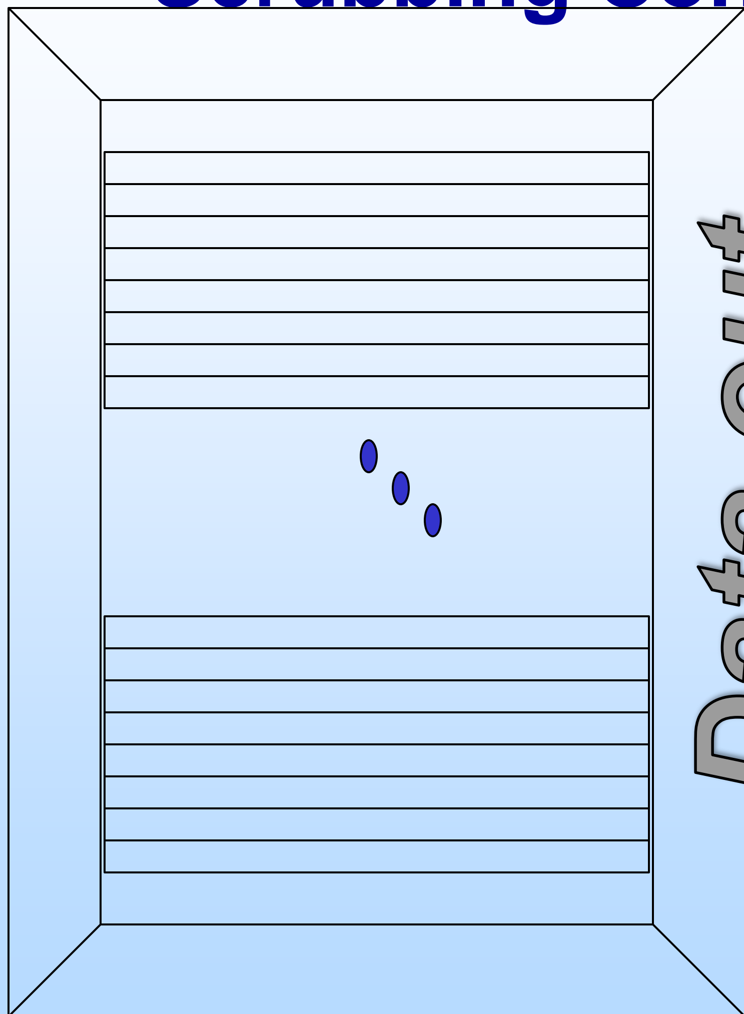
Sequential (DFF) and Combinatorial logic (CL) in data path

Single Event Functional Interrupt (SEFI): Global Routes and Hidden Logic

Depending on the FPGA type and/or mitigation, one of these σ_{SEU}s will be significantly more dominant than the others.

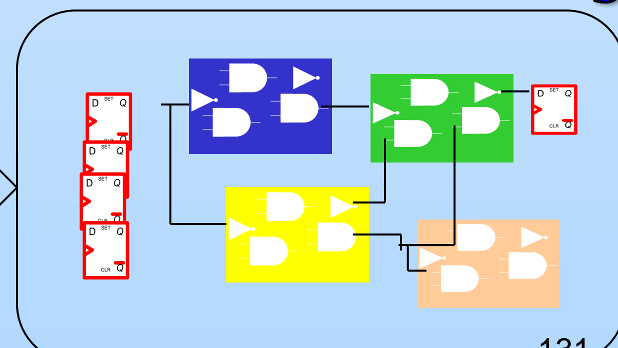
Scrubbing pertains to configuration σ_{SEU}s

Scrubbing Traditional SRAM … One Data Word at a Time (Not The Same as Scrubbing Configuration Memory)

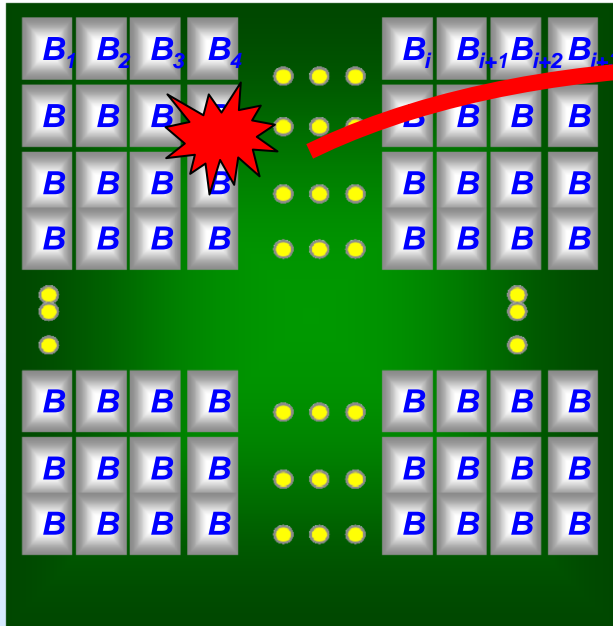


- Upsets have no effect until Address containing upset is read out of SRAM.
- Error detection and correction (EDAC) are placed after data out.
- EDAC circuits only work one data word at a time.

User Circuitry



Configuration SRAM is NOT Utilized the Same Way as Traditional SRAM



Every used bit is visible

- Direct connections from configuration to user logic
- Upset occurs in a used configuration bit then, upset occurs in logic

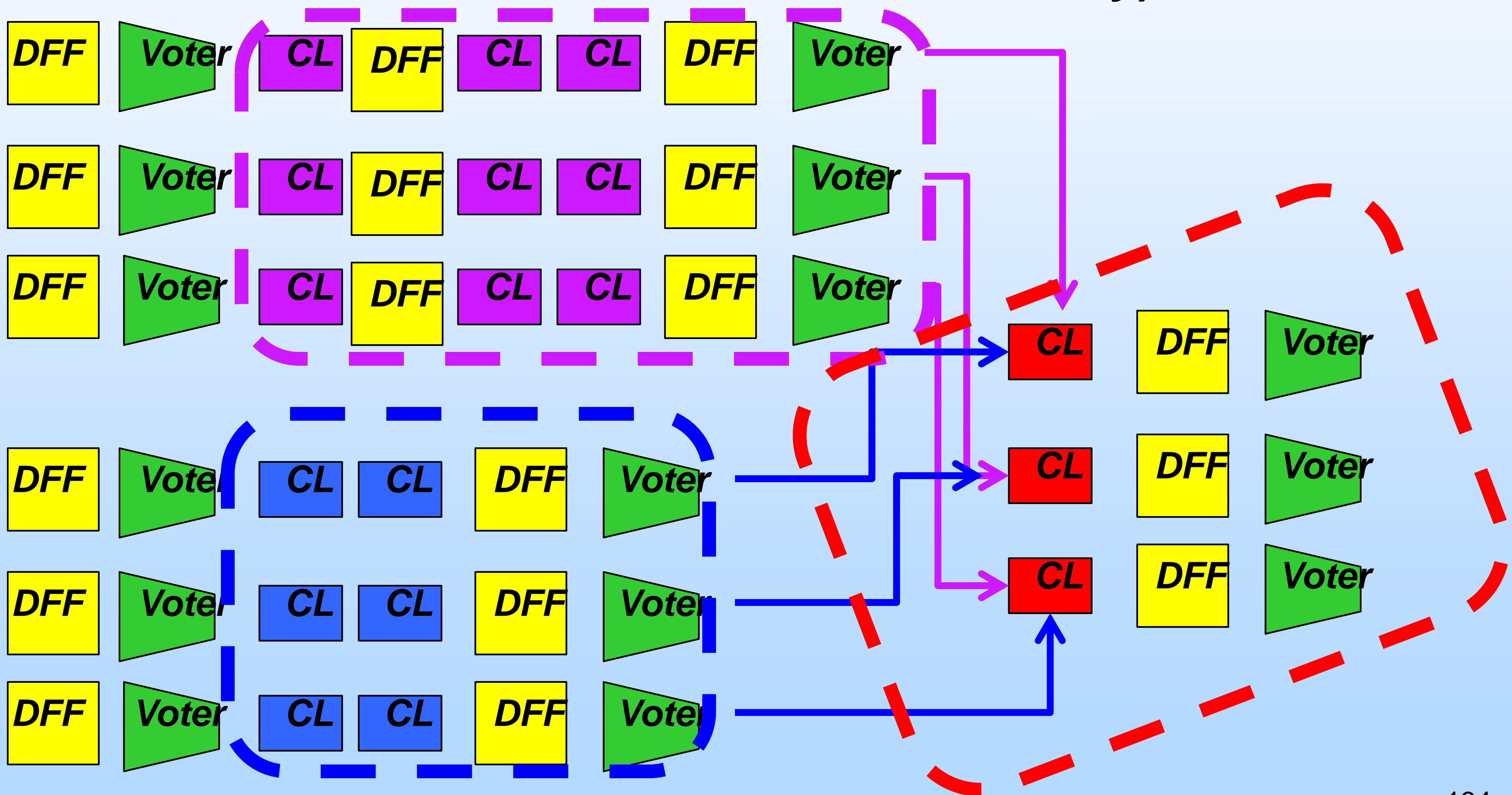
- We're not dealing with data words anymore. Traditional SRAM EDAC schemes don't quite apply for configuration SRAM

Rehash: Mitigation and SRAM Based FPGAs

- Mitigation is the act of dealing with an upset.
- Upsets need to be clearly defined:
  - Does an upset mean that the circuit is malfunctioning?
  - Does an upset mean non-recoverable failure or is recoverable but needs a reset?
  - Configuration upsets versus Functional upsets?
- There is a difference with how we handle upsets:
  - Detecting: Determining an upset exists in the circuitry.
  - Masking: blocking an error from affecting functional behavior.
  - Correcting:
    - Can use detection circuitry to correct; e.g. error correction and detection (EDAC).
    - Can blindly (automatically) correct (e.g. triple modular redundancy (TMR)).

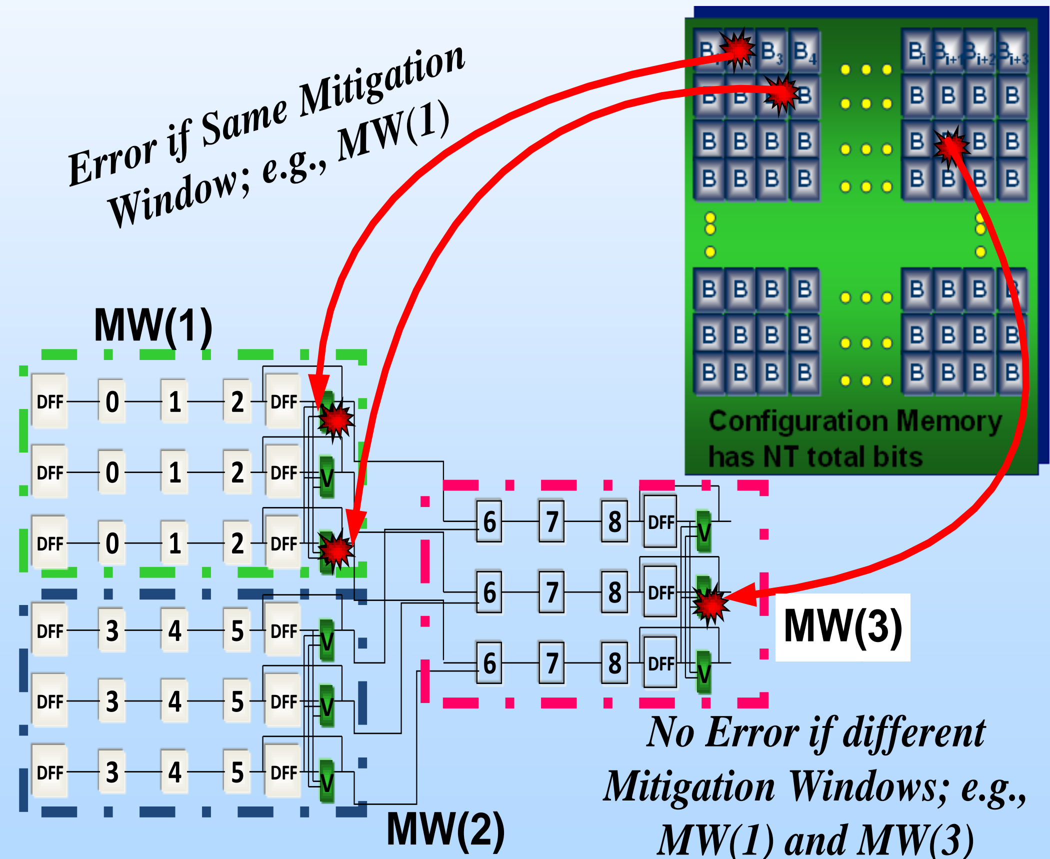
TMR Mitigation Window Definition

- MW boundaries: Start at a voter-output or a device input; End at a DFF-voter pair or a device output.
- Internal MW elements can be CL or DFFs (i.e., if a DFF does not have a voter, then it is not a MW boundary).



TMR in SRAM Based FPGAs and Mitigation Windows (MW)

- **Two upsets in the same MW and in different redundant paths will break the TMR protection.**
- **Uncorrected upsets accumulate in MWs and can eventually break TMR.**
- **Large MWs can be misguiding and will not provide the expected protection (too many bits in a MW).**
- **Strong mitigation has correction+masking.**



Global TMR (GTMR) and Scrubbing

- GTMR only masks configuration upsets it does not correct configuration upsets.
- We scrub to reduce accumulation in order to help protect the GTMR mitigation.
- Scrubbing corrects the configuration memory:
  - Does not reduce $dE_{Configuration}/dt$.
  - Reduces the accumulation bit error rate.
  - Does not correct functional upsets.
- Scrub Rate (dC/dt) must be fast enough to beat accumulation.
- Misperception #2:

Source: Xilinx Consortium:V4QV

Reported: $dC/dt > 10x(dE_{configuration}/dt)$

Problem: Only considers configuration, does not take into account mitigation strength; e.g., MW size and GTMR configuration masking

Do Not Scrub Configuration Memory of Non-Mitigated Designs

- **Scrubbing is a weak/secondary mitigation strategy:**
  - **It only protects against accumulation.**
  - **There is no masking when only scrubbing configuration memory (masking needs mitigation) :**
    - **If a utilized bit is hit – and its circuitry is active, then…**
    - **A functional error can occur with no time allowable to wait for a scrubber .**
- **If the decision is made to use a non-mitigated design in a radiation environment, then the application is expected to have a high upset rate.**
- **Do not add additional circuitry if it doesn't improve upset rates:**
  - **Power, area, Risk to project completion.**

Scrubbers: Blind versus Read-back

Blind Scrubber

- Write golden configuration into configuration.
- Scrub cycle in the order of *ms*.
- Pros: simple, less area and power, no need for additional non-volatile memory, very fast (great for accelerated testing).
- Cons: Write pointer can get hit during writing and write bad data into configuration- however, insignificant probability of occurrence (proven in heavy ion SEU testing).

Read-back

- Read configuration, calculate correct data; if there is an upset, write corrected data.
- Scrub cycle in the order of *s*.
- Pros, only writes if there is an upset.
- Cons, additional non-volatile memory required; slow (only a problem for accelerated testing); takes more area and power; Correction scheme can break (e.g. be limited to detecting and correcting one upset) and consequently write bad data to configuration.

Differentiate Scrubbing for Space Applications and Scrubbing for Radiation Testing

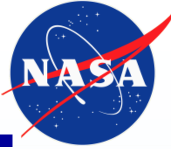
Space Application

- Only scrub if there is mitigation.
- Make scrubber simple to reduce project risk.
- Do not scrub constantly – not necessary and not good for the system.
- Single error correction double error detection (SECDED) scrubbers may not work well due to multiple bit upsets (MBUs).
- Blind scrubbing is the simplest scheme yet read-back will also work.

Accelerated SEU Testing

- We must scrub!
- Particles cannot overtake the scrubber – i.e., scrubber must be fast enough to stop fast accumulation of configuration SEUs – SCRUB CONSTANTLY.
- SECDED scrubbing schemes do not work well during accelerated testing because of MBUs and accumulation.
- Generally no time for read-back of configuration – hence blind scrubber is the best fit for accelerated testing.

Not All Configuration is used for a Design.



Probability that a used configuration bit will incur an SEU based on $dE_{configuration}/dt$:

Total Number of configuration bits (NT).

#used bits is a fraction of NT.

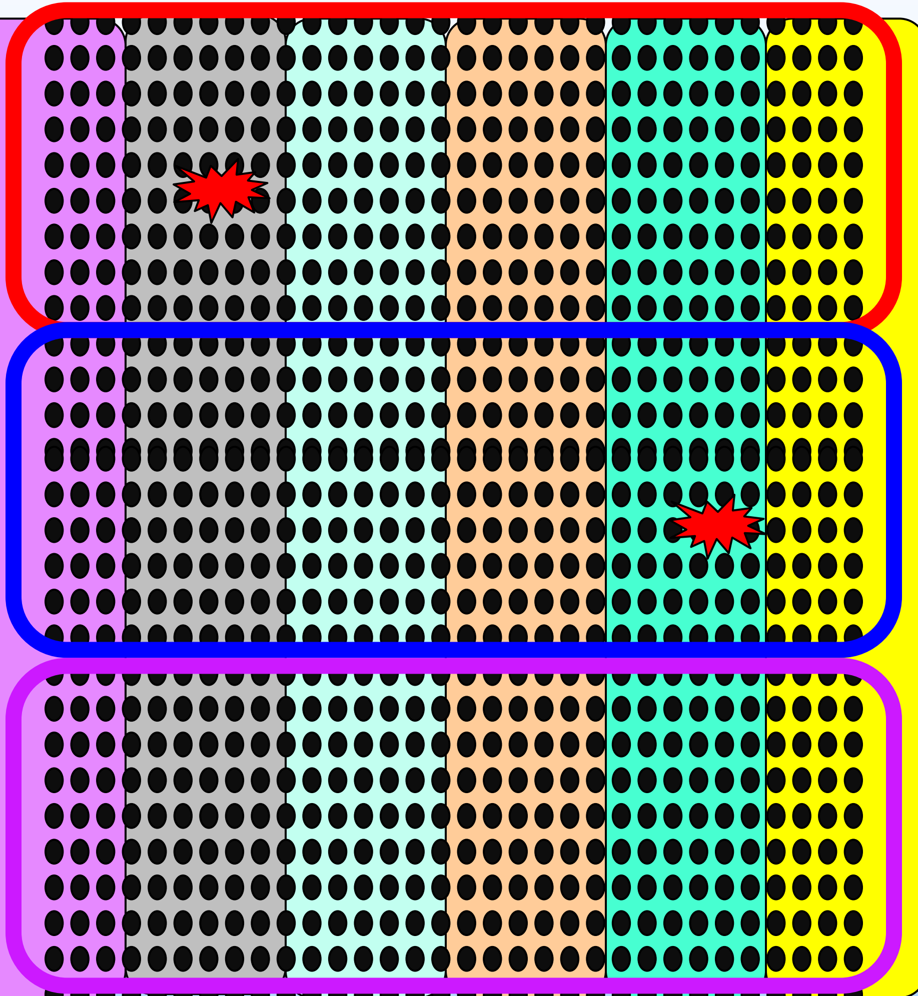
Event that one used bit is upset =

$$\left(\frac{dE_{configuration}}{dt}\right) * \left(\frac{\#usedbits}{NT}\right)$$

MWs Are Created out of Used Bits.

Probability that an SEU occurs in a MW Based on $dE_{configuration}/dt$

Used Configuration bits



1 MW: Divide the used bit space into 3 (crude estimate).

Broken mitigation if upsets are in 2 separate MWs.

As the number of MW increases, the number of used bits per MW decreases.

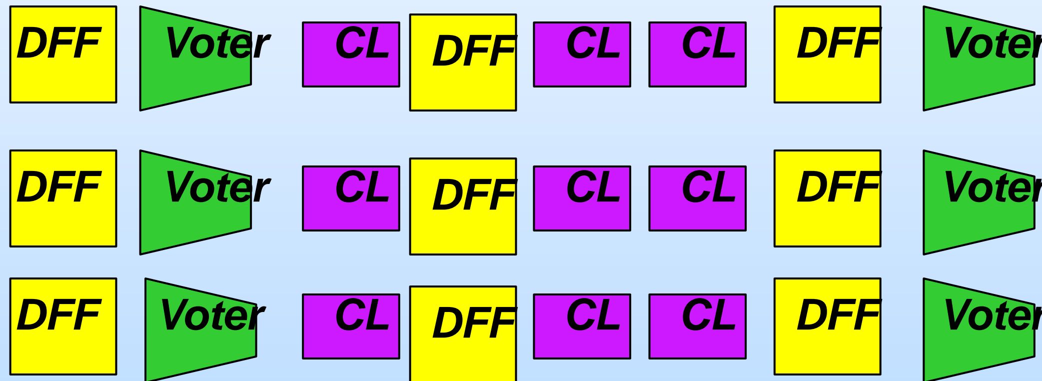
Event that an upset occurs in an MW =

$$\left(\frac{dE_{configuration}}{dt}\right) * \left(\frac{\#usedbits}{NT}\right) * \left(\frac{1}{NMW}\right)$$

NMW = number of MWs.

Scrub Rate Requirement Based on the Probability of Breaking the GTMR Mitigation:

$$\frac{dC}{dt} > \left(\frac{dE_{configuration}}{dt}\right)^2 * \left(\frac{\#usedbits}{NT}\right)^2 * 2/3 * \frac{1}{NMW} * j$$



MW fan-out: a bit can be within multiple MWs

Makes a crude assumption that all MWs are the same size – but they are not. This is a rough estimate – but good enough

Scrub Rate Example for Accelerated Testing

Variable Definition	Variable	Example Number
Fraction of used bits per total number of bits	#usedbits/NT	0.1
Configuration Error rate	$dE_{configuration}/dt$	1000 bit-errors/device-s
Average fanout from MW	φ ; 1<φ<NMW	10
Number of GTMR MWs	NMW	5000
Fraction of bits in the same MW	$\left(\frac{\#usedbits}{NT}\right) * \frac{1}{NMW}$	2.0E-5

Calculated at Texas A&M with a flux of $1e^4$ and LET=26MeVcm²/mg

$$\frac{dC}{dt} > \left(\frac{dE_{configuration}}{dt}\right)^2 * \left(\frac{\#usedbits}{NT}\right)^2 * {}^{2}\!/_{3} * \frac{1}{NMW} * j$$

$$\frac{dC}{dt} > 14 \, {}^{scrub}\!/_{s}$$

Scrub cycle must be in the order of ms; Blind scrubber works best.

Scrub Rate Example for Space: Variation of Number of Mitigation Windows

$dE_{configuration}/dt$ = 4 bit errors/day; #userbits/NT = 0.1; $\varphi = 10$

Number of MWs	dC/dt (scrub rate per day)	
1	1.07E+00	Once a day
10	1.07E-01	Once every 10 days
50	2.13E-02	Once every 50 days
500	2.13E-03	Once every 500 days
1000	1.07E-03	Once every 1000 days
5000	2.13E-04	Once every 5000 days

Accelerated $E_{configuration}/dt$ is 8.64×10^7 times faster than the space environment $E_{configuration}/dt$…

Hence the required scrub rates are significantly reduced.

Take Away Points : Scrubbing