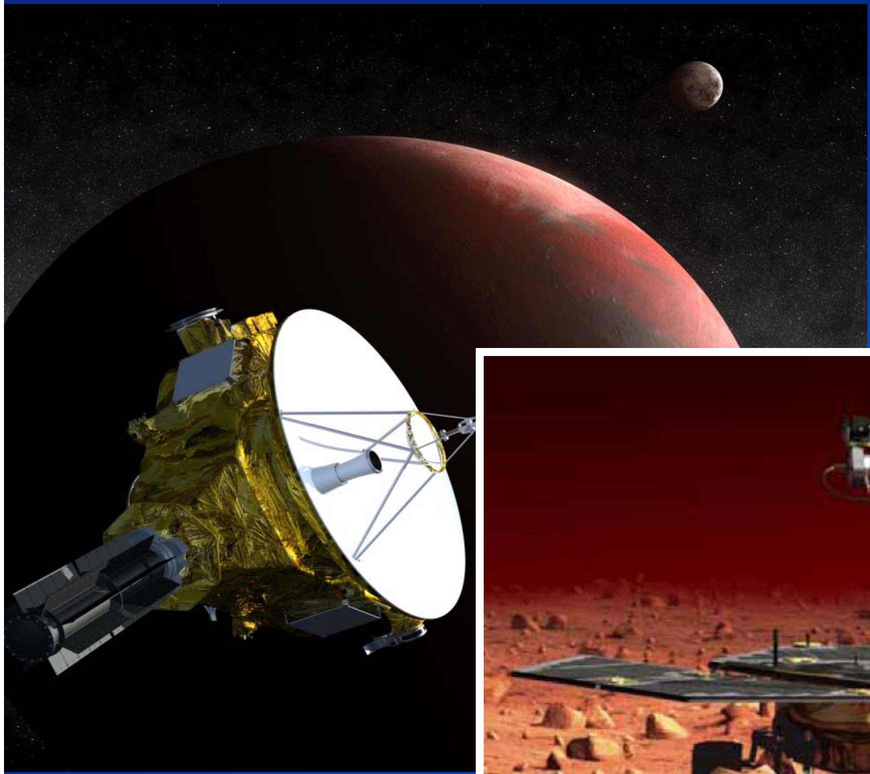
# FPGA Design Strategies for Critical NASA Goddard Flight Projects

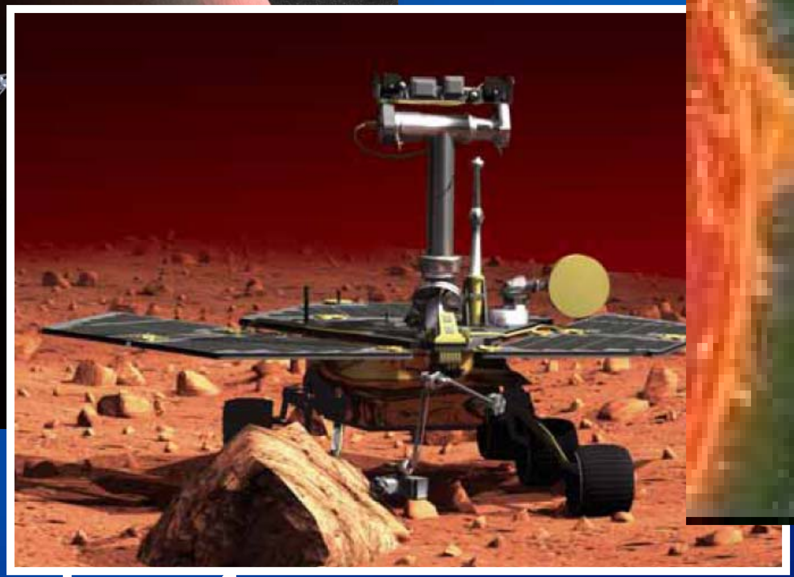
**Melanie Berg**
MEI Technologies- NASA/GSFC Radiation Effects and Analysis Group

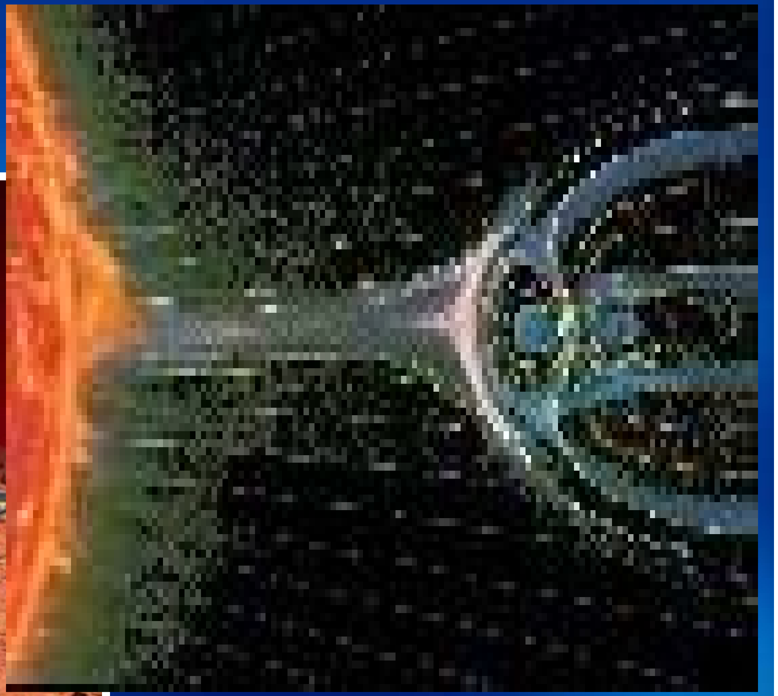
Presented by Melanie Berg at Mentor Graphics User to User Conference, San Clara CA, November 6th 2008

# What's the Issue?

**If something goes wrong…**



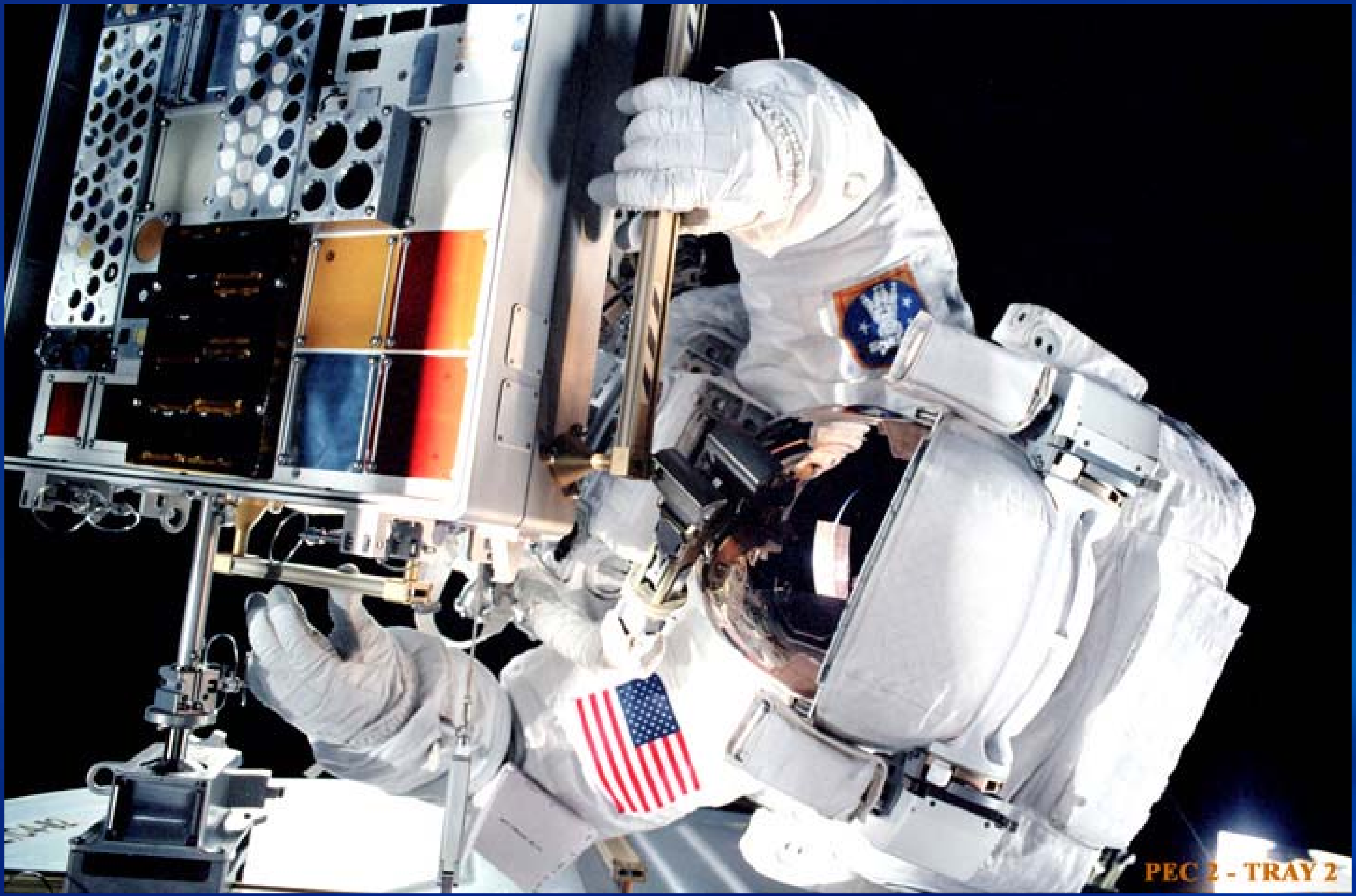




**Increasing number of FPGA devices inserted into space missions**

**Harsh Space Radiation Environment**

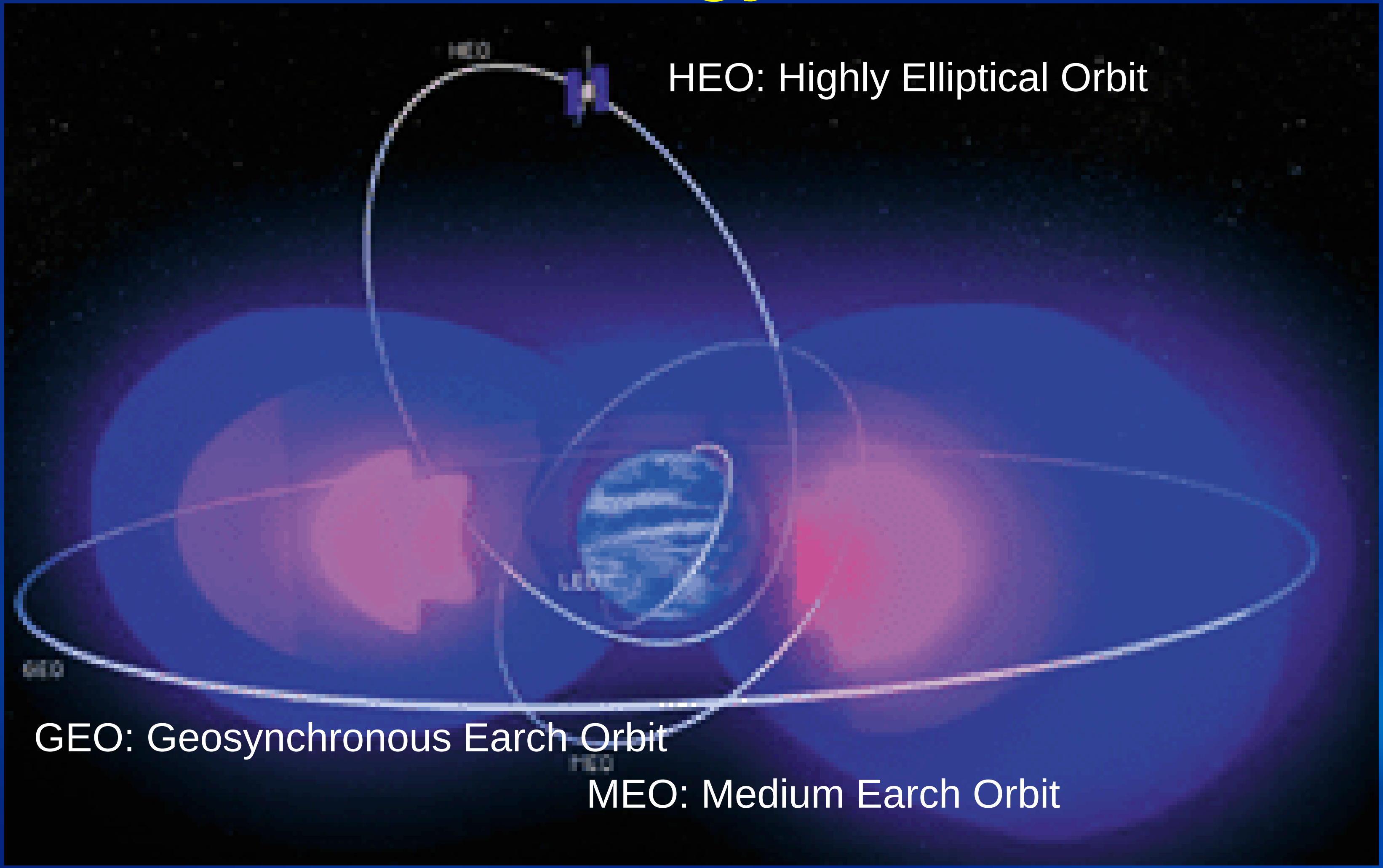
# We Can't Always do This…



# Overview: From Potential Faults to Fault Tolerance

- Space Radiation Environment and CMOS Technology
- Basic Synchronous Design Concepts and Potential Faults
- General Fault Tolerant Design Strategies for Mitigating Radiation Effects
- Finite State Machines and Fault Detection Strategies
- FPGA Characterization: Understanding the Differences to Effectively meet specifications
- Implementation Strategies- Radiation Effects and Commercial Tools

# Space Radiation Environment and CMOS Technology

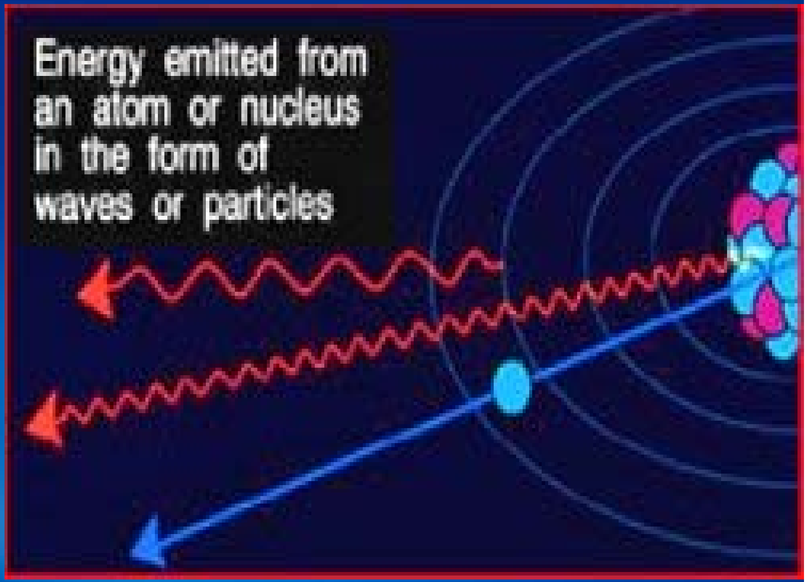



Van Allen Radiation Belts:

Illustrated by Aerospace Corp.

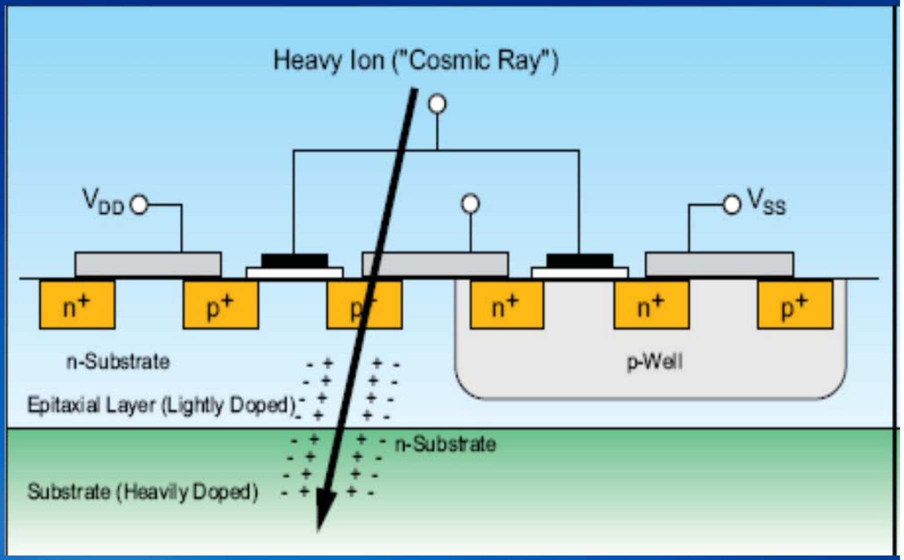
# Source of Faults: Ionizing Particles

- Single Event Effects (SEEs)
  - Terrestrial devices are susceptible to faults mostly due to:
    - **alpha particles**: from packaging and doping and
    - **Neutrons**: caused by Galactic Cosmic Ray (GCR) Interactions that enter into the earth's atmosphere.
  - Devices expected to operate at higher altitude (Aerospace and Military) are more prone to upsets caused by:
    - **Heavy ions**: direct ionization
    - **Protons**: secondary effects
- Composite Effects – Total Ionizing Dose (TID): The amount of energy imparted by ionizing particles to unit mass of irradiated material. Units are krads (Si). **Can be considered a limiting factor if TID is not within specifications**



# 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**

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

**Units**

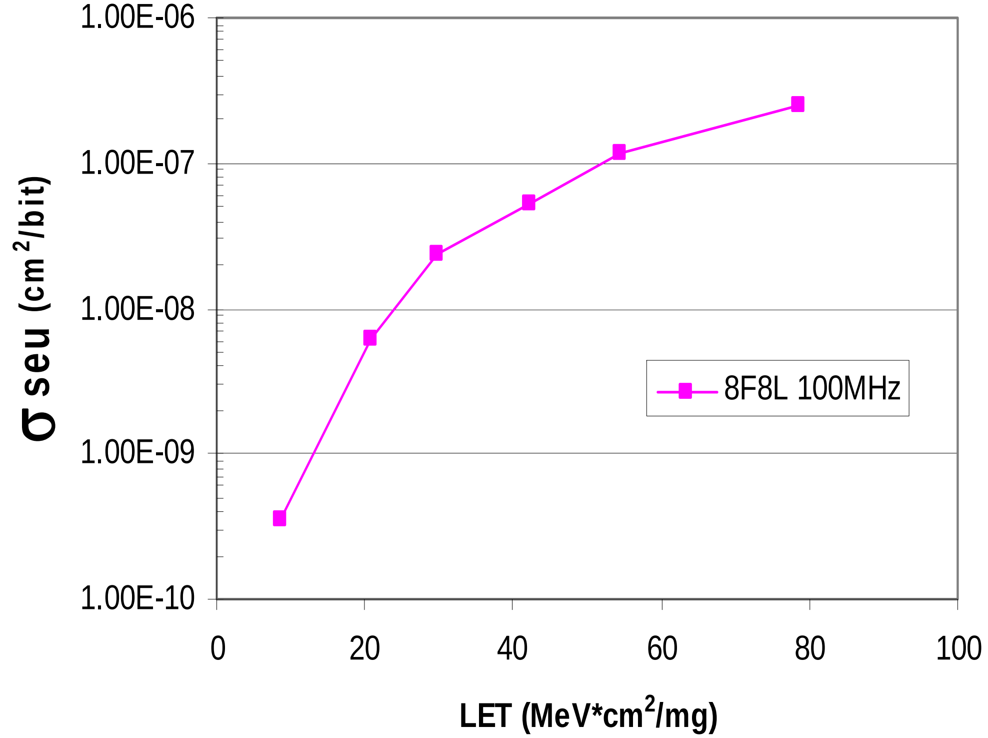
**Density of target material**

# LET vs. Error Cross Section Graph

- Error Cross Sections are calculated per LET value in order to characterize the number of potential faults and error rates in the space environment
- Terminology:
  - Flux: Particles/(sec-cm2)
  - Fluence: Particles/cm2
  - Error cross section($\sigma$): #errors normalized by fluence
  - Error cross section is calculated at several LET values (particle spectrum)

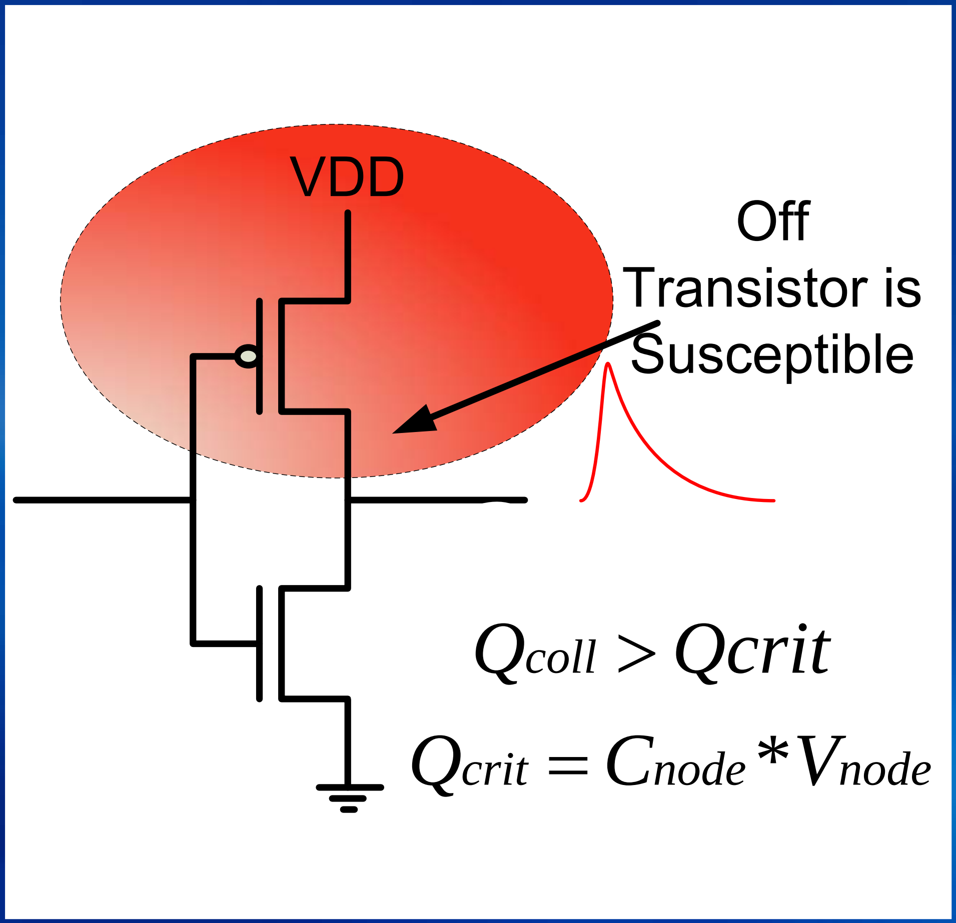
## LET vs. $\sigma$: Aeroflex UT6325

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



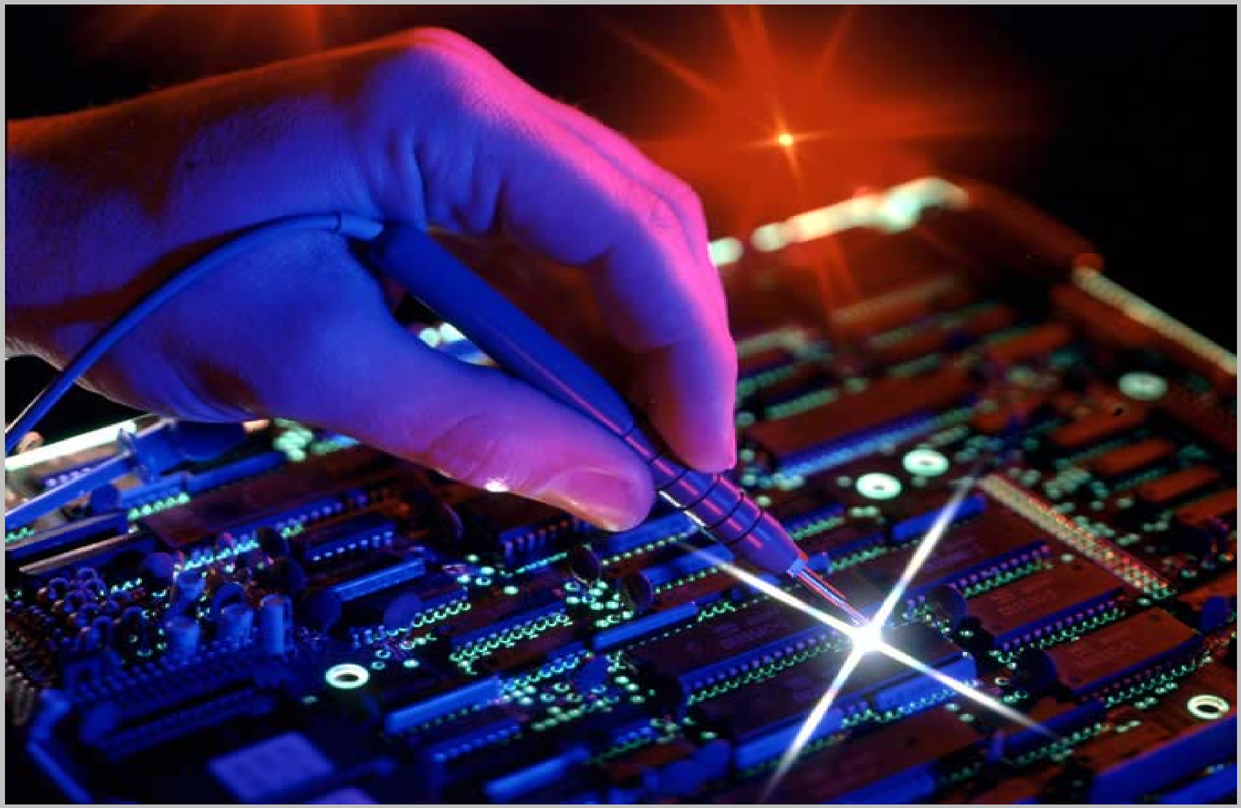
# Soft Faults: Single Event Transients (SETs) and Single Event Upsets (SEU)s

- CMOS transistors have become more susceptible to incurring faults due to:
  - the reduction in core voltage
  - decrease in transistor geometry, and
  - increase in switching speeds,
- Single Event Transient (SET): current spike due to ionization. Dissipates through bulk
- Single Event Upset (SEU): transient is caught by a memory element
- Single Event Functional Interrupt (SEFI) - upset disrupts function



# Radiation Tolerant (RT) vs Radiation Hard (RH): It's about The Dose Not SEU(s)

- RH is a device guaranteed to be fully operational up to a specified dosage (given in krad (Si)) .
- RH devices are generally 300 krad (Si) and above
- RT devices are usually between 100 krad (Si) and 300 krad (Si)
- Radiation Hardened By Design (RHBD) is a methodology:
  - inserting redundant circuitry
  - Changing the RC characteristics of gates or routes –filtration
  - Can sometimes refer to TID protected devices
- Beware: User's may still need to insert mitigation in RT or RH devices



# Basic Synchronous Design Concepts and Potential Faults

# Philosophy

- How we choose to implement and analyze our design has a direct impact on optimal fault tolerance insertion
- Bottom line is to efficiently insert fault protection when and where necessary
- Topics covered in this section:
  - Basic FPGA Library Components and sources of error:
    - Clock
    - DFF
    - Combinatorial logic
  - Methodology and Implementation
    - Static Timing Analysis
    - Asynchronous Resets
    - Repetition of logic

# Basic Synchronous Design Concepts and Potential Faults :

- FPGA Library Components:
  - Combinatorial logic
  - Sequential Logic: DFF
  - Clock
- Methodology and Implementation
  - Static Timing Analysis
  - Asynchronous Resets
  - Repetition of logic

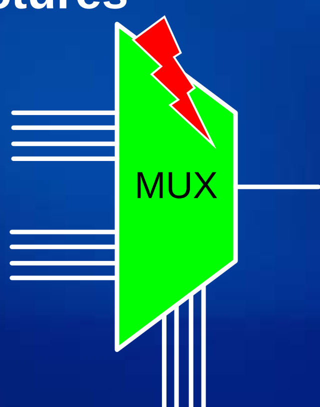
# Data Path: Some Common Basic Combinatorial Logic Blocks Found in FPGA Libraries

**HDL or Schematic**

```
If (input_a > inputb) then
        output <= '1';
Else
        output <= '0';
End if
```

**Synthesis**

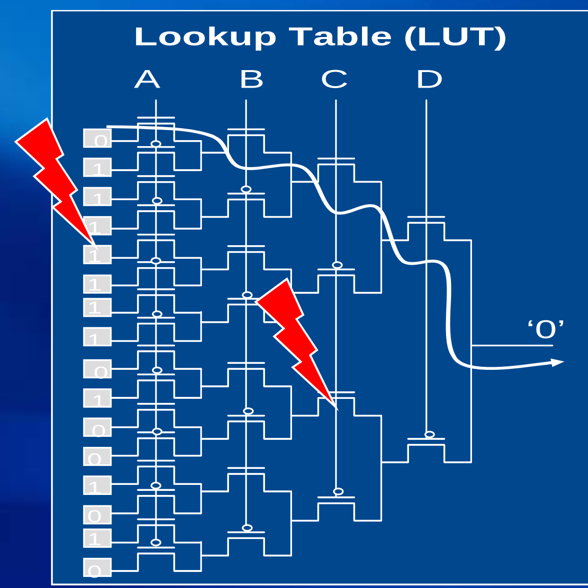
**MUX Structures**




**Gates:**

**Blocks of combinatorial logic dependent on FPGA library**

**LUT can incorporate SRAM + transistor logic**



# Generic DFF: Sequential Element

A DFF is clocked (sequential) logic where data is stored and reflected on the output at either the rising or the falling edge of a clock (following a clock to q delay).

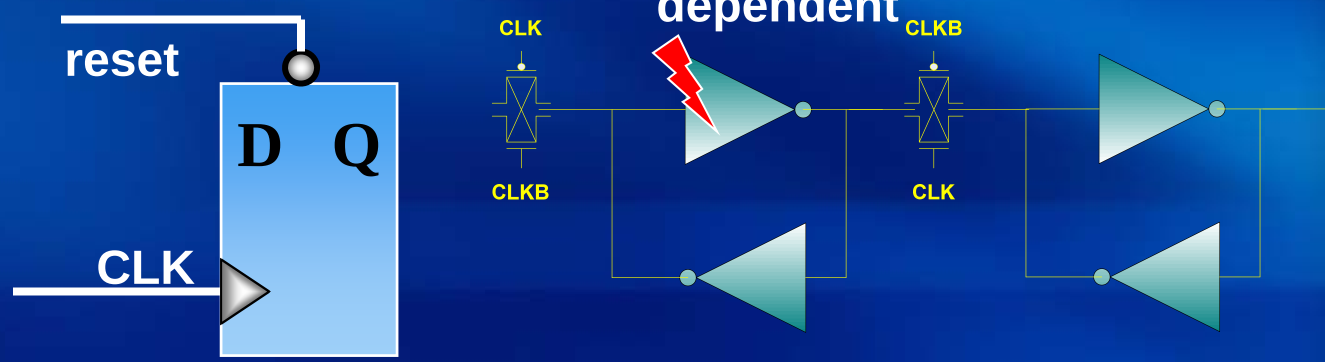
**HDL or Schematic**

```
If (reset = '0') then
        Q<= '0'
Elsif rising_edge(clk) then
        Q <= D
End if
```

Synthesis

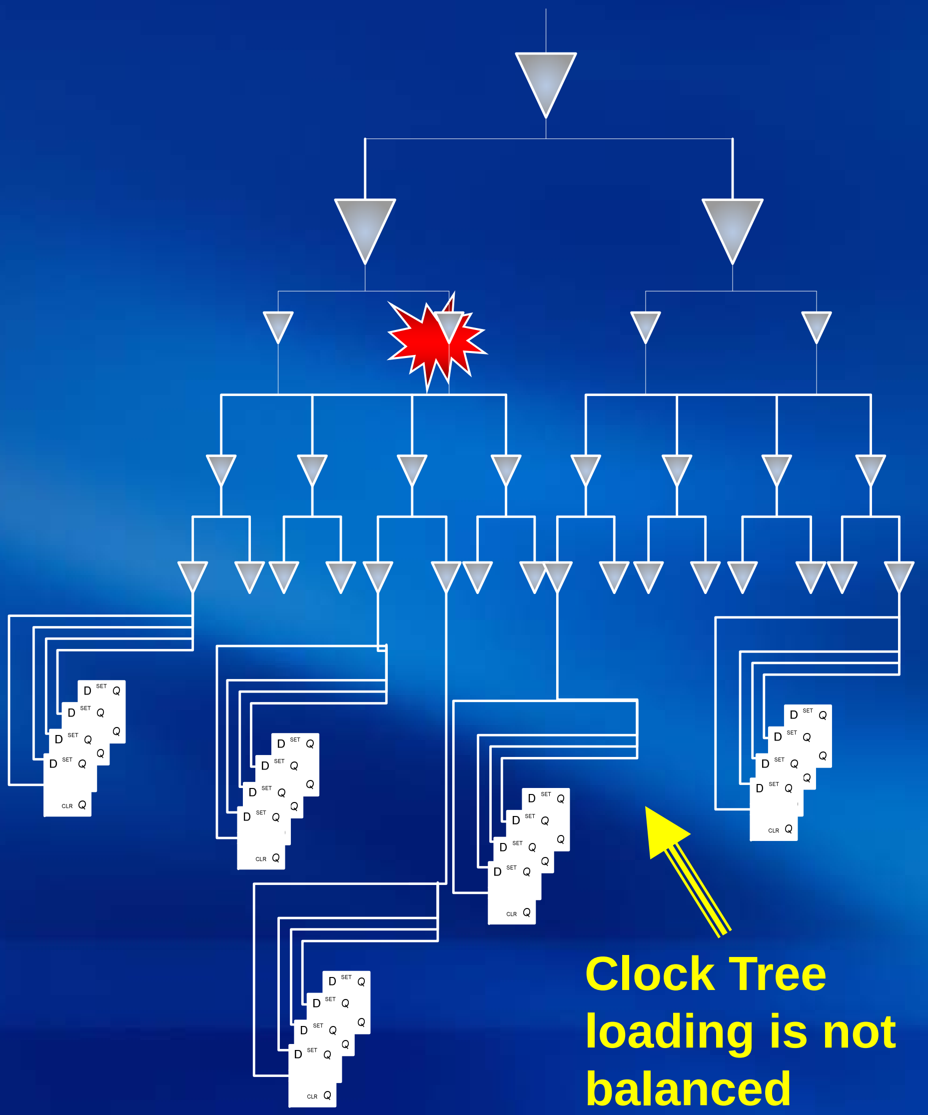
Gates

SEU strike is not frequency dependent



# Synchronous Design: Clock

- The clock creates discrete and deterministic intervals
- Every DFF is connected to a clock
- Necessary to minimize clock skew from DFF to DFF
  - It's capacitive loading must be balanced (no skew)
  - Subsequently, Must not enter the data path (only connect to the "clock" pin of a DFF)
- Clock Tree can be susceptible to faults
  - Clock Tree is made out of buffers, routes, and connects
  - Each FPGA has Design guidelines on clock tree usage and circuit criticality (i.e. ACTEL=> HCLK)



# Basic Synchronous Design Concepts and Potential Faults :

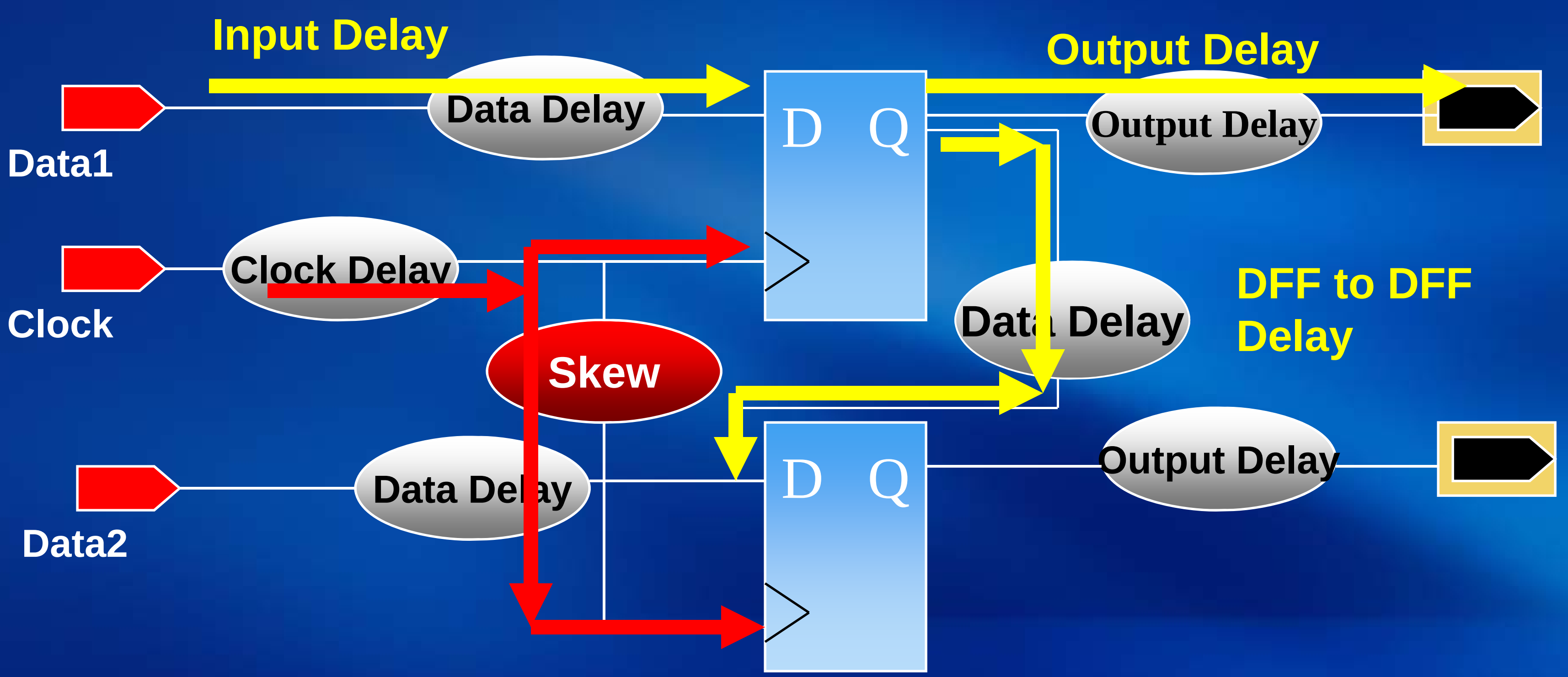
- FPGA Library Components:
  - Clock
  - DFF
  - Combinatorial logic
- **Methodology and Implementation**
  - Static Timing Analysis
  - Data Capture
  - Asynchronous Resets
  - Repetition of logic

# Static Timing Analysis (STA)

- Concept: When will Data arrive at its associated DFF relative to the clock
- Every data path delay contained solely within each clock domain must be strictly deterministic
- Analysis is not performed across clock domains
- Asynchronous behavior is not analyzed
- Analysis…Each path is defined as:
  - Input to DFF
  - DFF to output
  - DFF to DFF
  - Clock input latency (through clock Tree to DFF clock input)
  - Clock Skew: difference in clock arrival time with respect to each DFF clock pin on the same clock tree
  - Input to Output (highly not recommended design practice – inputs should pass through a DFF)

# Synchronous Clock Analysis
# Static Timing Analysis (STA)

Various delays within a Synchronous design

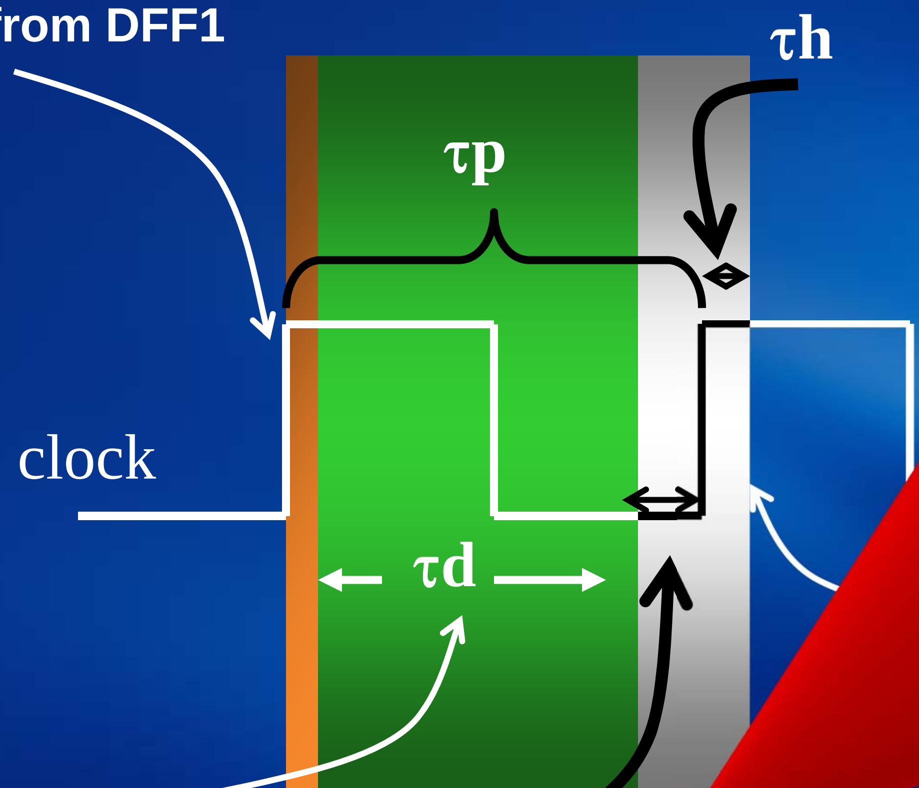


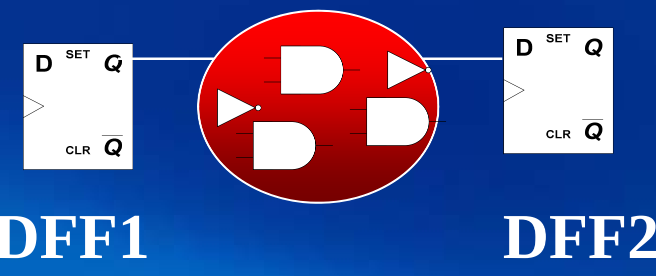

**Clock Latency: significant with synchronous I/O data capture and crossing internal clock domains (more during mitigation techniques section)**

# STA: Setup and Hold Time for a DFF

**Data arrival at any DFF must be stable between setup time (τsu) and hold time (τh)**

**Data Launch from DFF1**



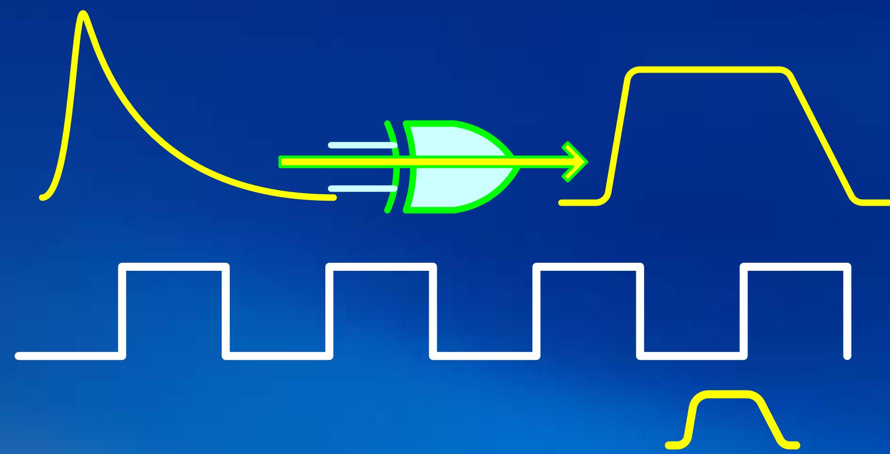

**Remember:**
**Analysis may not be performed across clock domains**

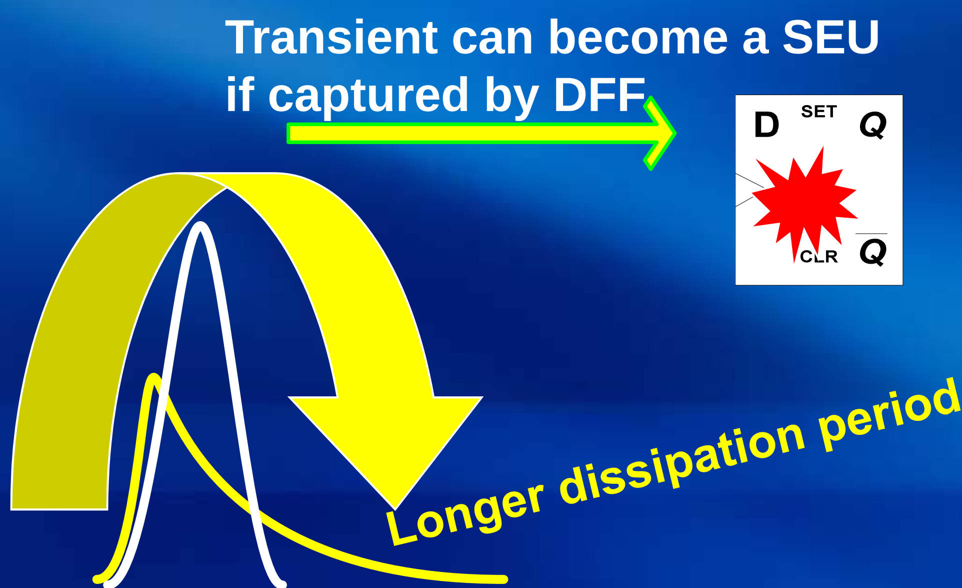
**τd : Data Delay through combinatorial logic and routes**

$$\tau_d < \tau_p - \tau_{su} - \tau_{skew} - \tau_{jitter}$$

# SET Capture with respect to Clocks, combinatorial logic and DFFs

- Transient Research:
  - Generation
  - Propagation
  - Capture
- Process dependent
- Synchronous Design
  - SET capture is frequency dependent
  - High clock speed Increase probability of SET capture
  - Capture is asynchronous – behavior is unpredictable in STA
- Newer Technology may be more susceptible to metastability



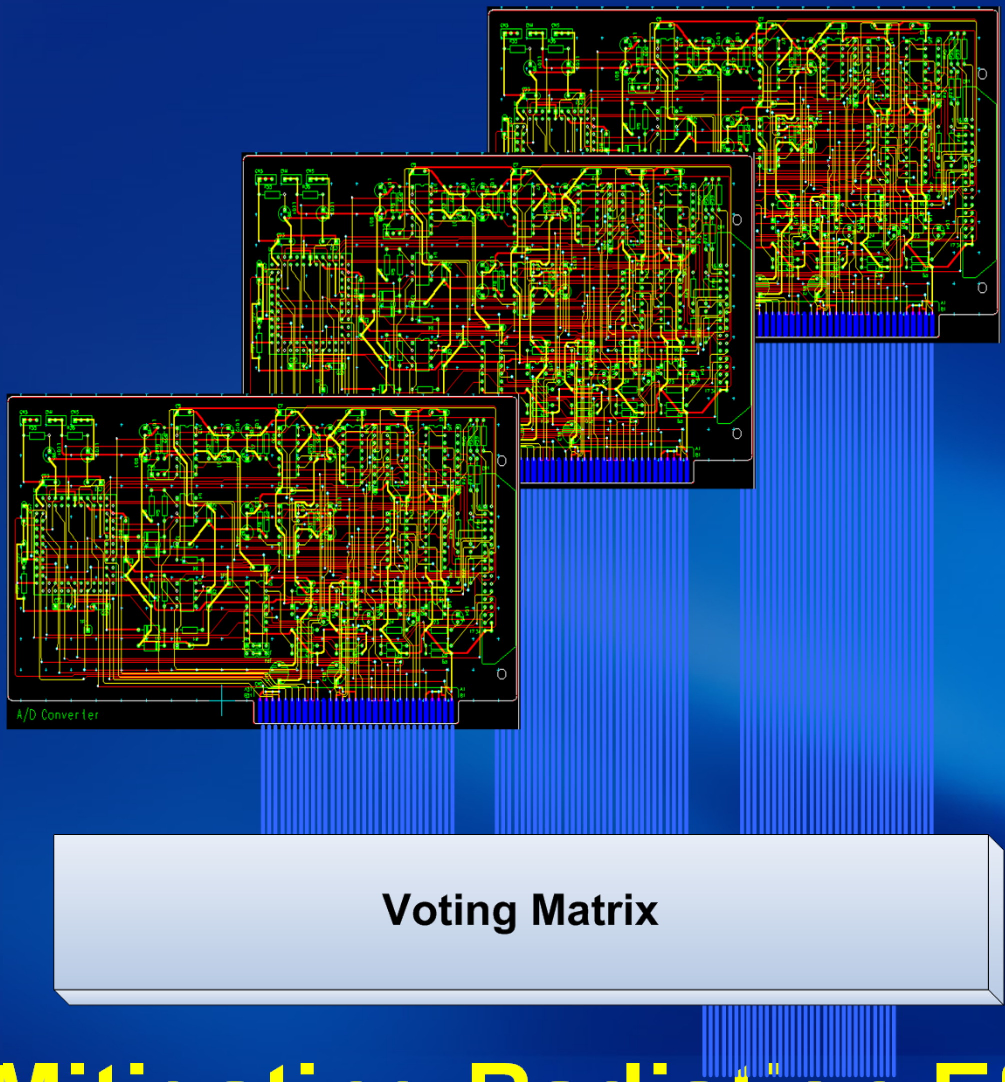


# Overview: Asynchronous Faults and the Impact to Synchronous Design Circuitry

- All SETs and SEUs are asynchronous and are nondeterministic events
- STA can accurately calculate every timing path in a synchronous circuit within ONE clock domain
  - Without SEUs: Must analyze all domain crossing manually (i.e. design reviews)
  - Asynchronous SEUs will take more than STA to analyze behavior
- Common Fault signatures
  - DFF's can Flip their state
  - Clocks can glitch
  - Resets can glitch
  - Potential metastability
  - Inputs can be missed by capture logic

**All must be taken into account while determining a mitigation scheme**

# No FPGA is 100% Fault Immune



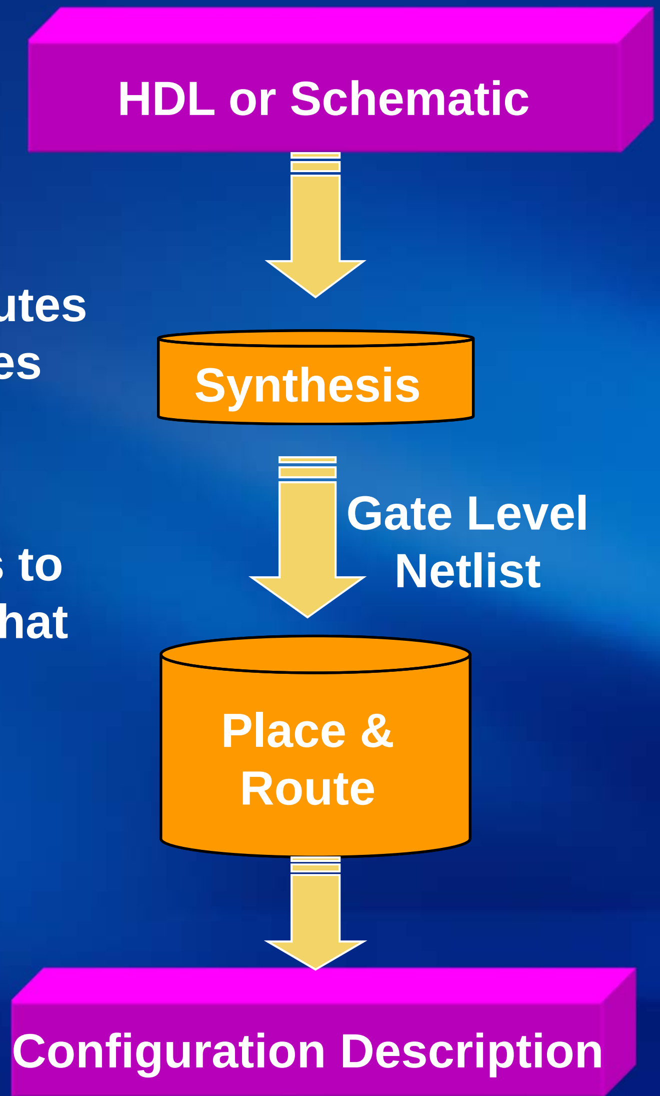
# SEUs and Common Triple Mode Mitigation (TMR)Techniques

- Most common type of mitigation in FPGA devices
- Involves triplication of circuitry and majority voter insertion
  - LTMR: Localized TMR
  - GTMR: Global TMR
  - DTMR: Distributed TMR

# Where Does the User Insert Mitigation in The Design Cycle

**RTL:**

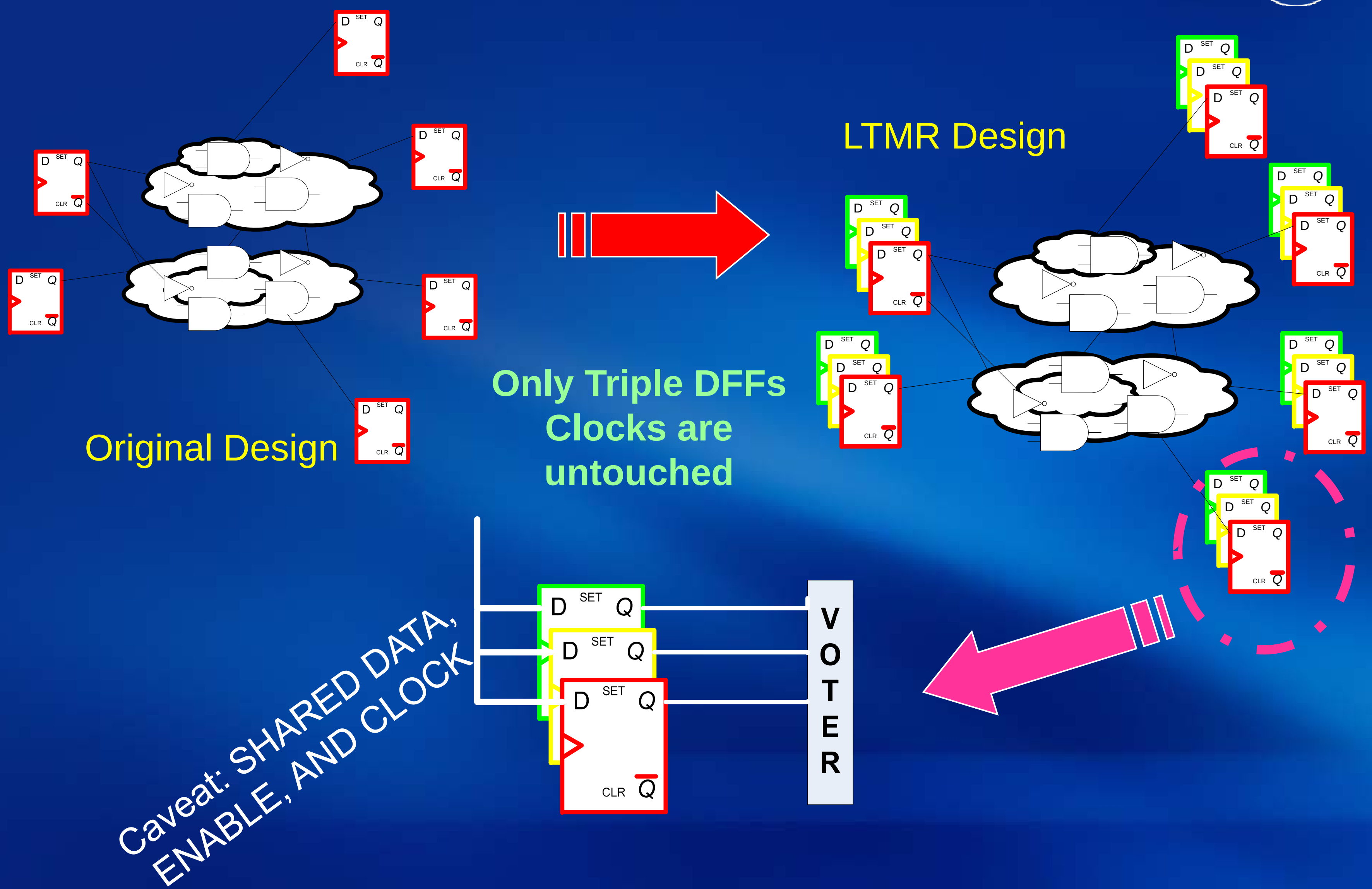
- User must include attributes so that synthesis tool does not optimize out the redundancy.
- User must add attributes to Place and Route tool so that redundancy does not get removed




**Gate Level Pre Place and Route:**

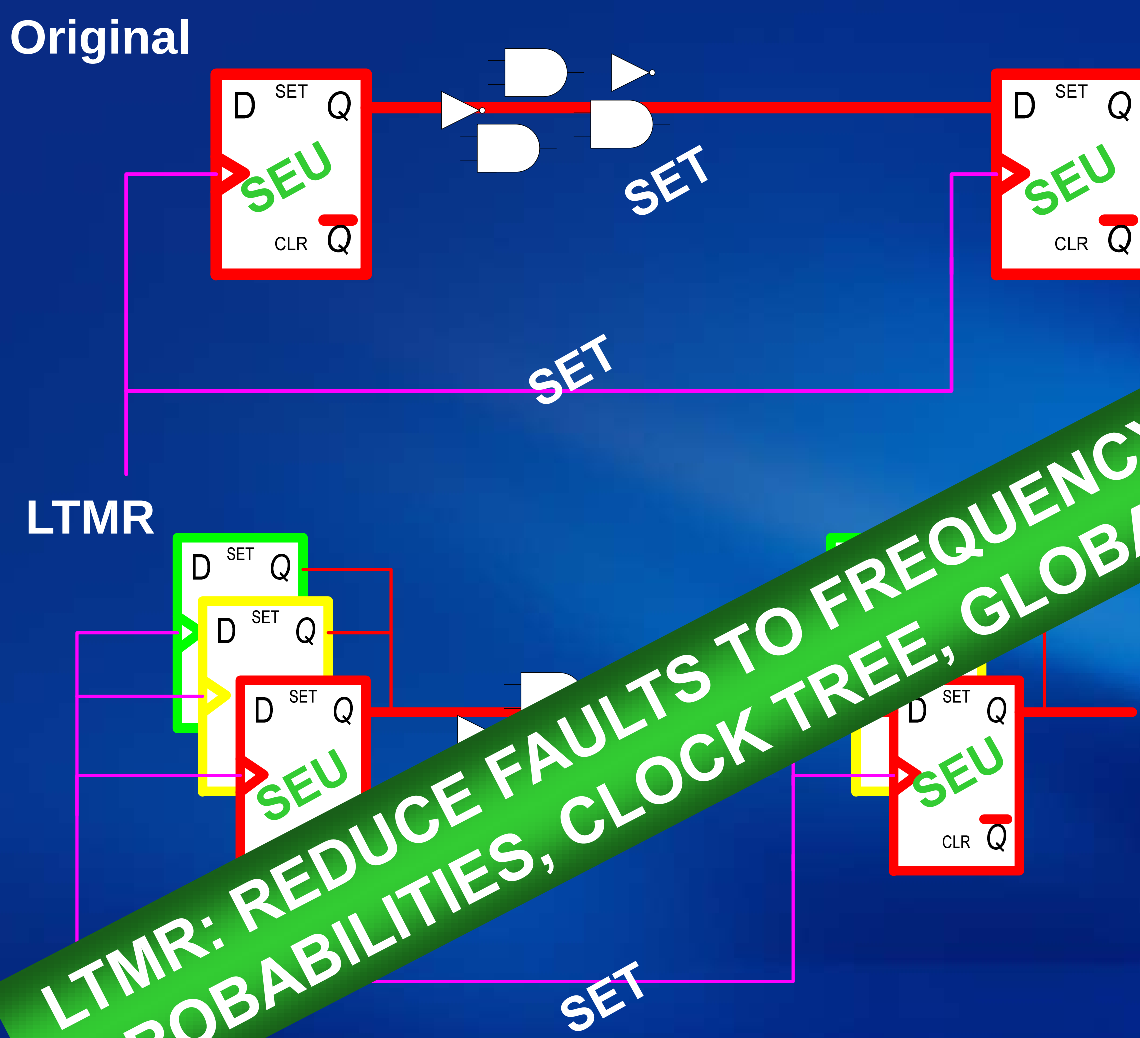
User must verify that the original functionality is not broken. User must add attributes to Place and Route tool so that redundancy does not get removed

# LTMR



Original Design
LTMR Design
Only Triple DFFs
Clocks are
untouched
VOTER
Caveat: SHARED DATA,
ENABLE, AND CLOCK

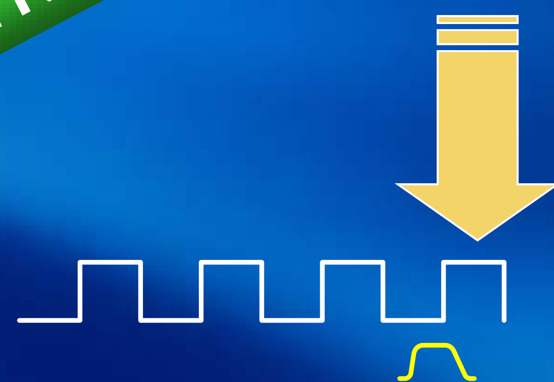
# Another Look at LTMR




Frequency independent SEU can occur fr… …hin the DFF

or SE… …lock … …h

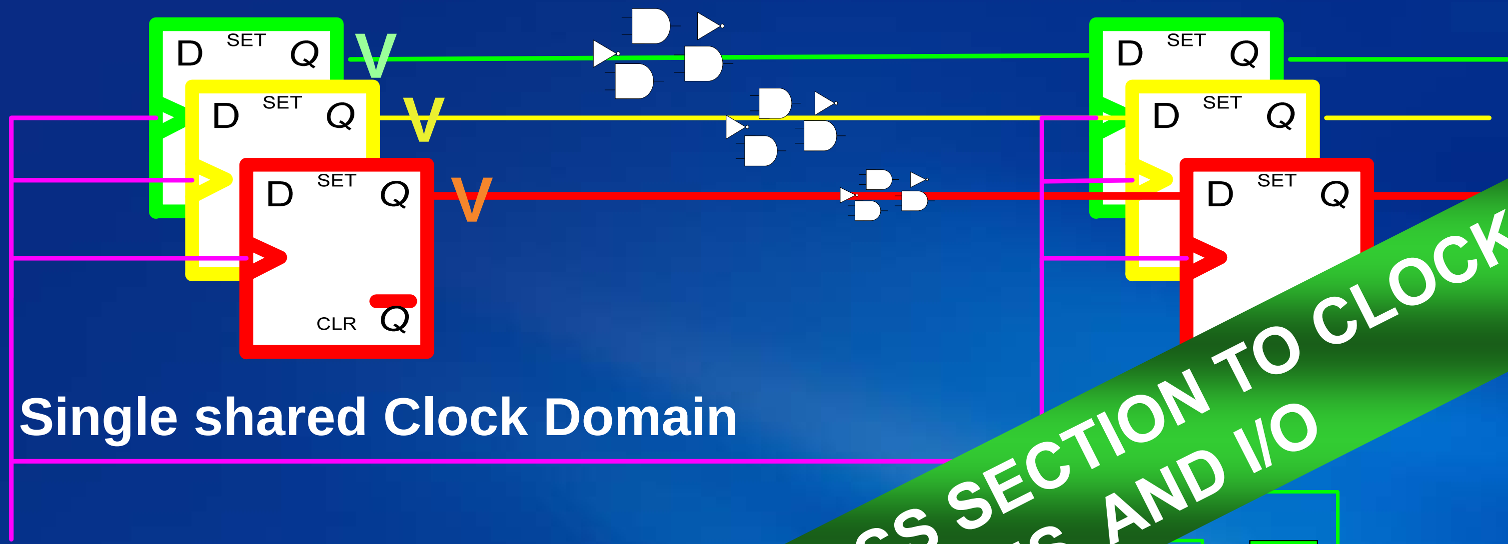
… CROSS SECTION



Frequency dependent SEU can occur from SET Capture by Clock edge or Clock Glitch

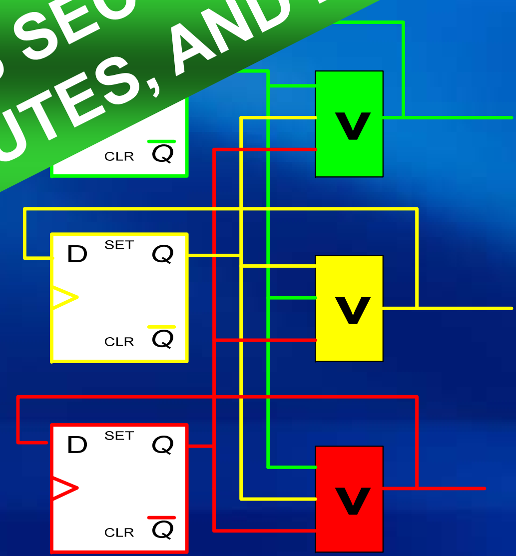
**LTMR: REDUCE FAULTS TO FREQUENCY DEPENDENT SEU PROBABILITIES, CLOCK TREE, GLOBAL ROUTES, AND I/O**

# DTMR: Separation of Data Paths (SET protection)




Single shared Clock Domain

Closer look at voters...
cross into voter...
possible f...

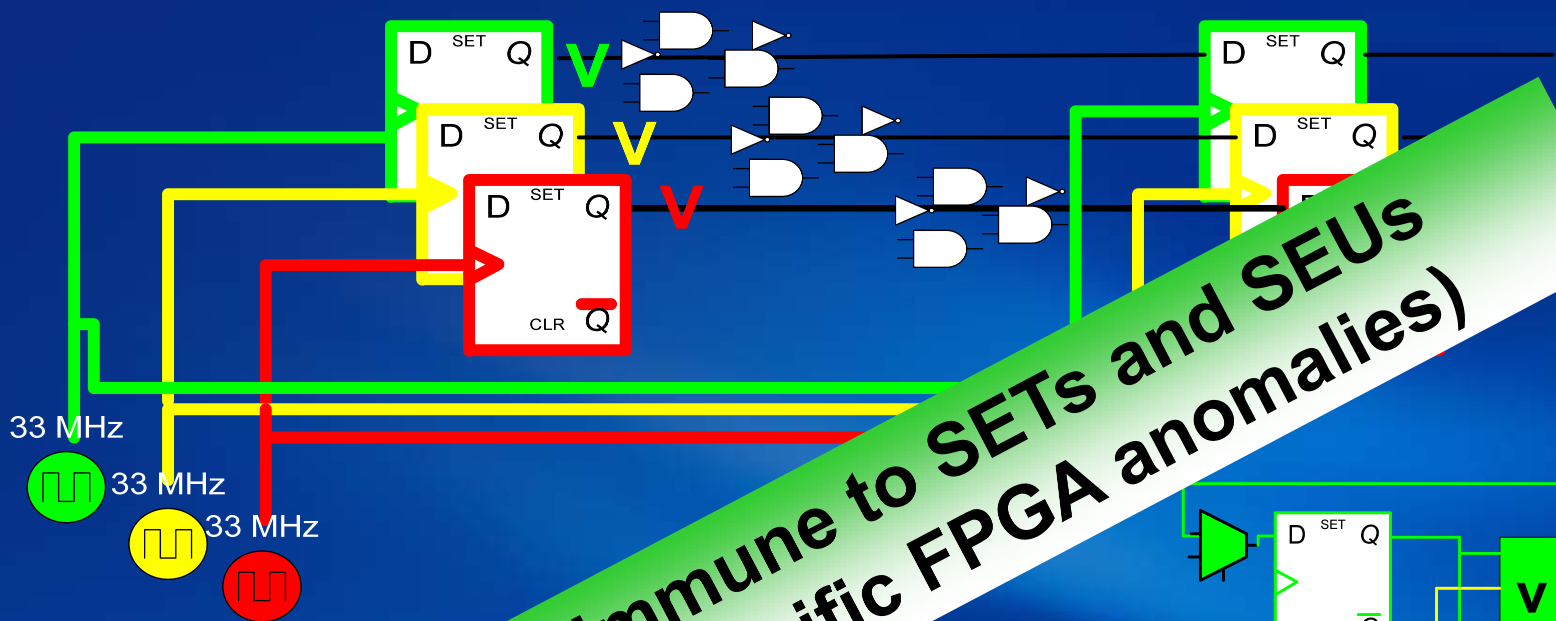



**DTMR REDUCES SEU CROSS SECTION TO CLOCK TREE, GLOBAL ROUTES, AND I/O**

# Potential DTMR Caveats

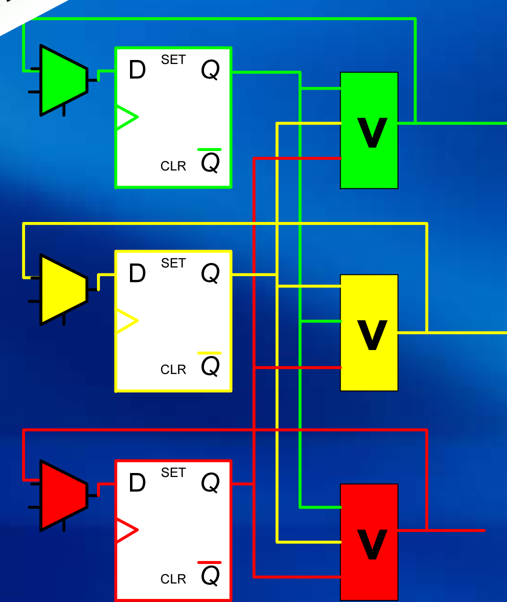
- Clock Glitch
- Glitch on Global Routes (Resets)
- I/O
- Placement of redundant stings in blocks that share logic:
  - A fault can cause a short between redundant paths
    - Shared routing matrix
    - Shared route link from Cluster
    - Shared MUX
  - Can’t vote out contention fault.
- Asynchronous Data Capture

# Global TMR (GTMR): Separation of Clock Domains and I/O




**Theoretically immune to SETs and SEUs (not including specific FPGA anomalies)**

- All logic is tri... including clocks a...
- Vot... vote across clock d...

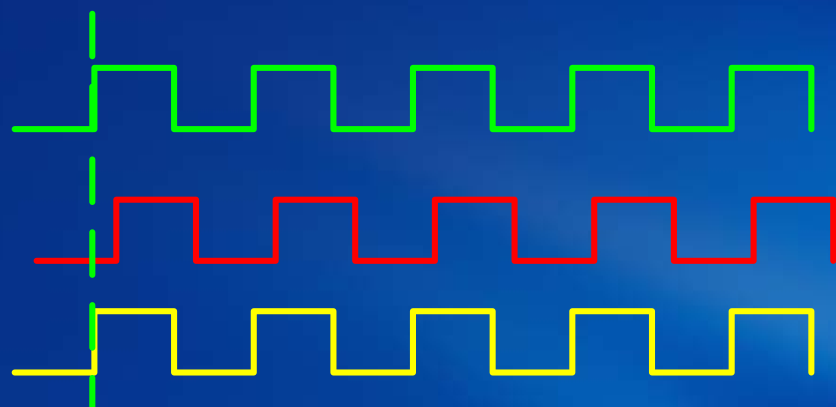


# Potential GTMR Caveats

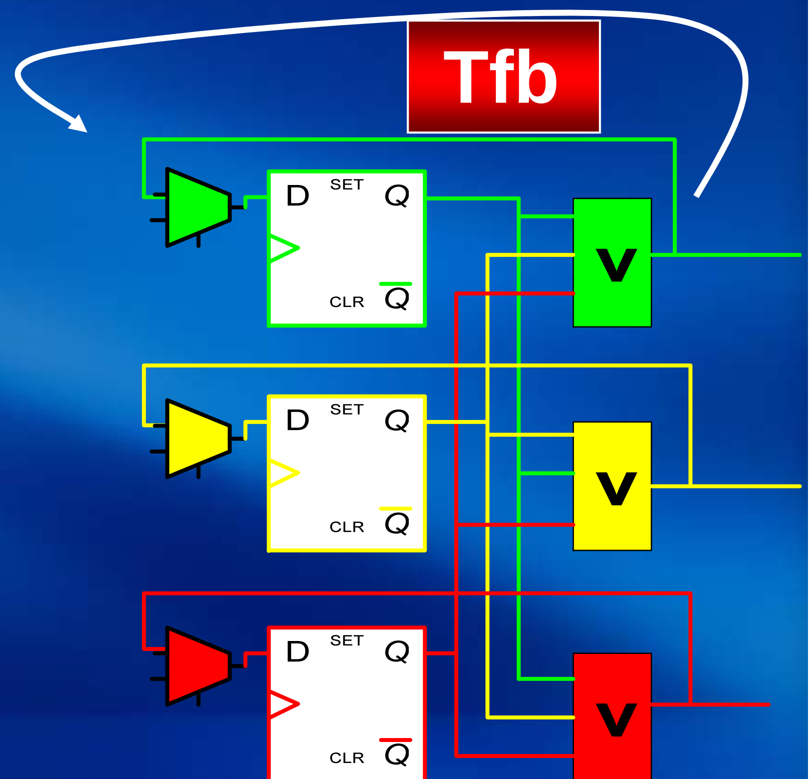
- Placement (as in DTMR)
- Asynchronous Data Capture (as in DTMR)
- **Clock Skew: GTMR necessitates communication across multiple clock domains**
  - **User must take care of clock skew at the board level**
  - **User must take care of clock skew internal to FPGA**

# GTMR Clock Skew and Race Conditions

- Example: Red Clock domain (as seen by DFFs) has skew relative to other domains



**Race condition on feedback path if its delay is faster than clock skew**




**Total Skew:**

$$\mathbf{Tsk} = S_{io} + S_{route} + S_{int\_max}$$

**$S_{io}$ : Skew Measured at Input Boundary**

**$S_{route}$: Skew of route from Input to Clock tree buffer**

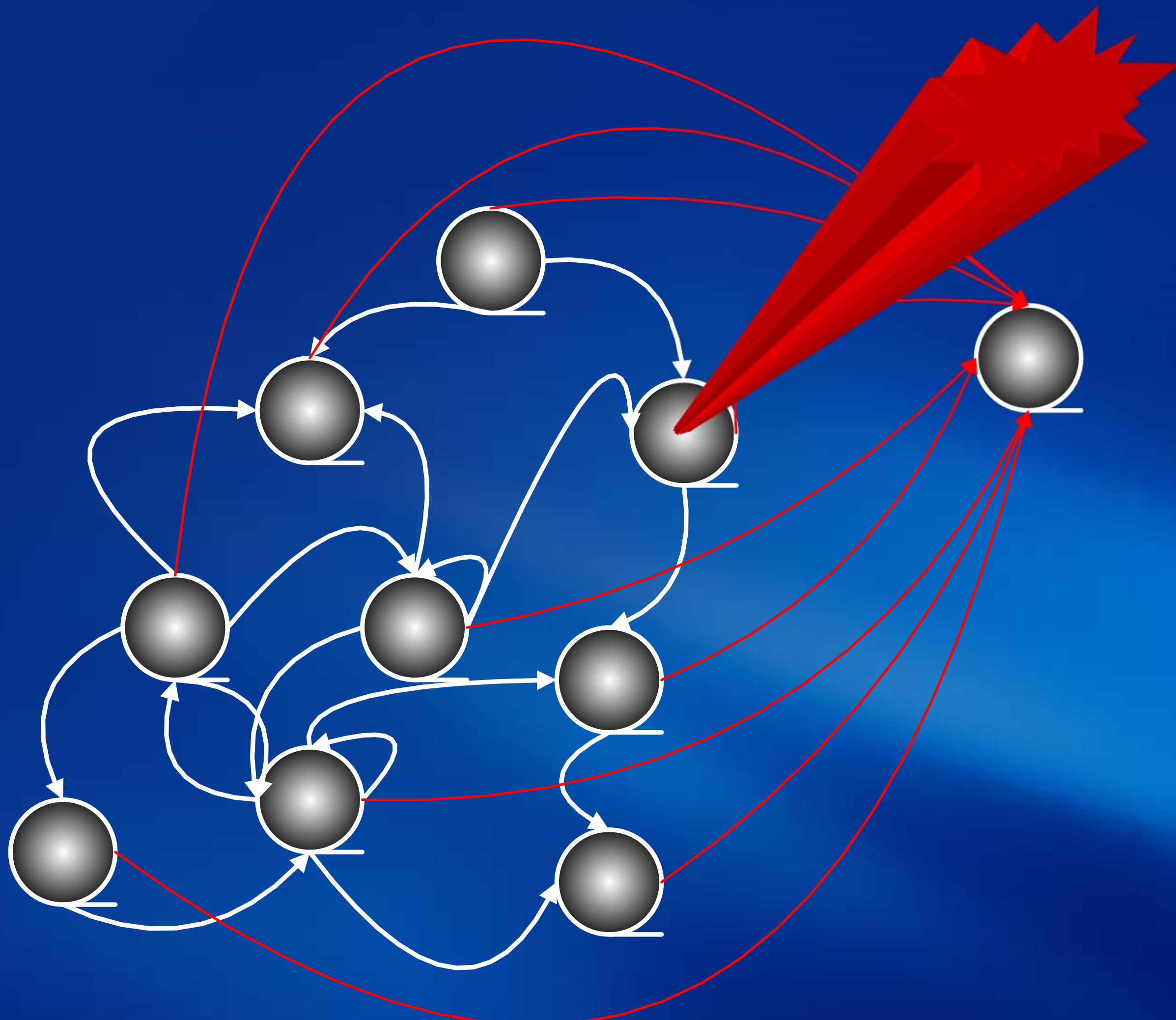
**$S_{int\_max}$: Static Timing Analysis max Skew**

# Clock Skew and Race Conditions

- Board design:
  - Input Clocks: select I/O that will guarantee minimal skew from input to clock tree connect
  - Balance traces to inputs so that the three signals arrive with minimal skew
- FPGA must contain clock buffers that have minimal skew from each other
- FPGA must contain routes from Input to clock buffers that are almost the same distance
- Static Timing Analysis must be performed in order to validate
  - Maximum feedback path timing (Tfb)
  - Maximum skew from clock inputs to DFFs

$$T_{sk} < T_{fb}$$

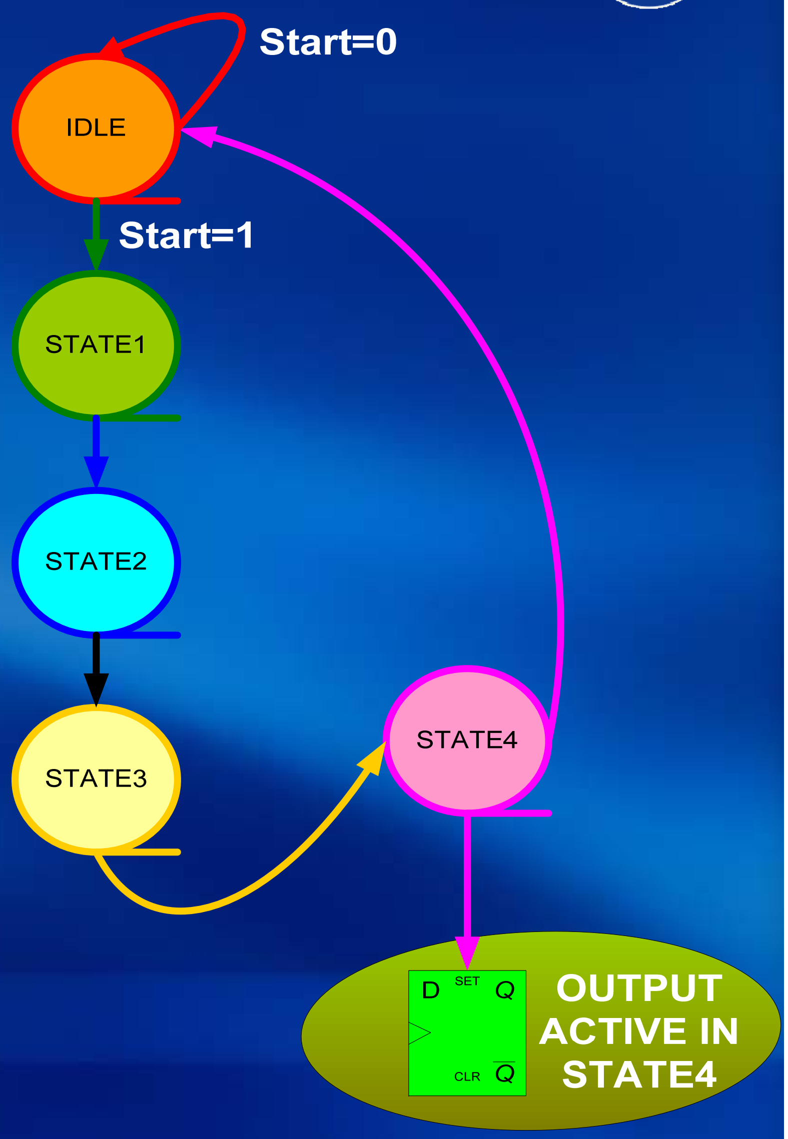
- Try to validate via fault injection if possible



# Finite State Machines (FSMs) and Fault Detection Strategies

# Encoding Schemes: 5 State Example

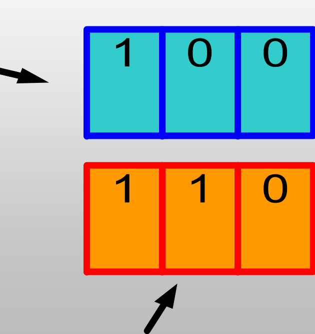
- Each state of a FSM must be mapped into some type of encoding (pattern of bits)
- Once the state is mapped, it is then considered a defined (legal) state
- Unmapped bit patterns are illegal states



# Encoding Schemes

## Registers: Binary Encoding

Good state : SEND_DATA




Bad state: unmapped

Int(Log2(states))

**STATES (5):**

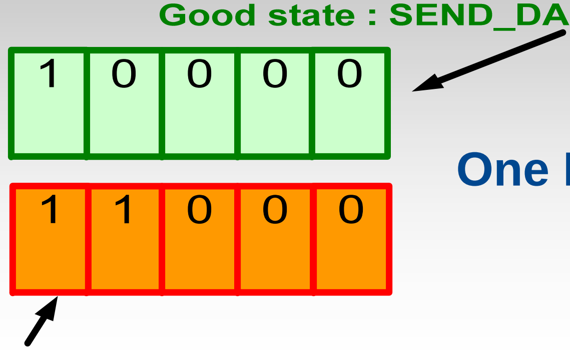
| | |
|---|---|
| IDLE | :000 |
| STATE1 | :001 |
| STATE2 | :010 |
| STATE3 | :011 |
| STATE4 | :100 |

## Registers: One Hot encoding

**STATES (5):**

| | |
|---|---|
| IDLE | :00001 |
| STATE1 | :00010 |
| STATE2 | :00100 |
| STATE3 | :01000 |
| STATE4 | :10000 |

Good state : SEND_DATA




Bad state: unmapped

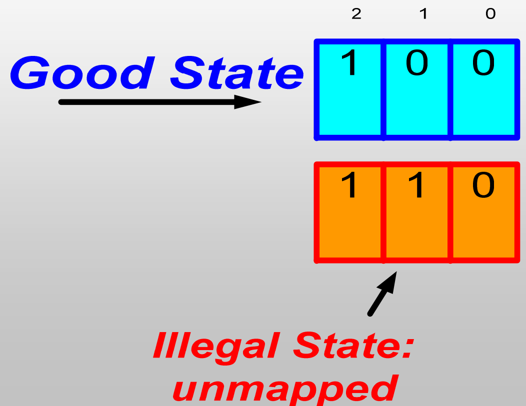
One Bit per State

# Safe State Machines???

- A “Safe” State Machine has been defined as one that:
  - Has a set of defined states
  - Can deterministically jump to a defined state if an illegal state has been reached (due to a SEU).
- Subsequently (by definition):
  - Does not reduce error cross section (no redundancy).
  - Does not necessarily provide error detection for the rest of the circuitry
  - Will insert a substantial amount of additional logic for implementation
  - Is itself susceptible because there is no redundancy
- **Question… How safe is this?**

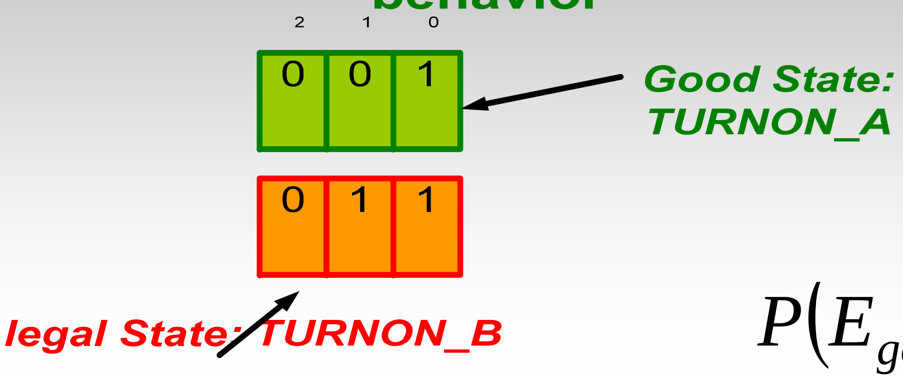
# Binary Safe State Machines???

**Using the "Safe" attribute will transition the user to a specified legal state upon a SEU**




**STATES (5):**
IDLE :000
TURNON_A :001
TURNOFF_A :010
TURNON_B :011
TURNOFF_B :100

**Using the "Safe" attribute will not detect this SEU: Could cause detrimental behavior**




**P(E): probability of flipping into alternate state**

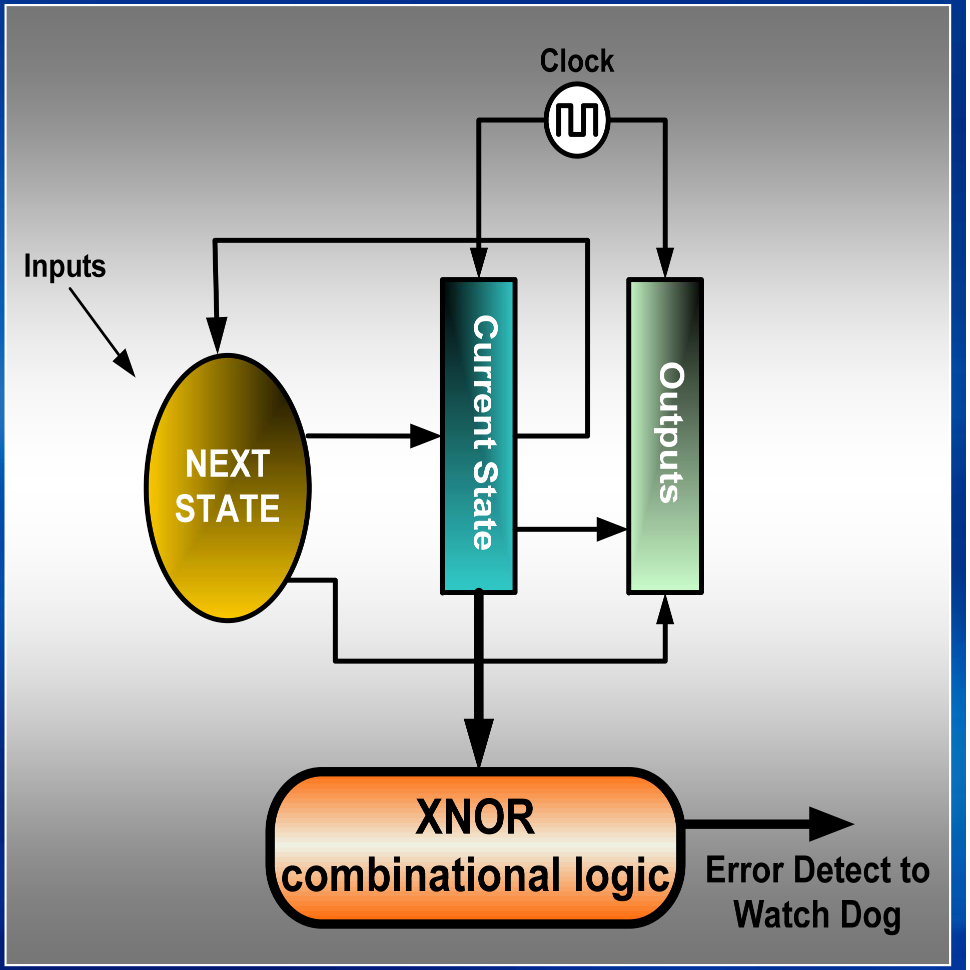
$$P(E_{goodState}) > P(E_{badState})$$

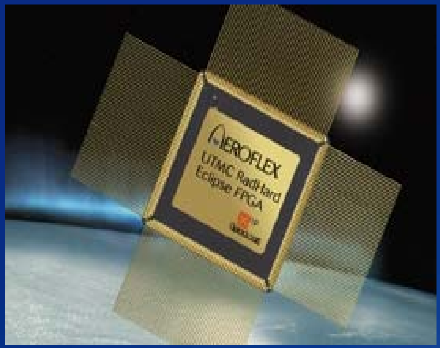
# One-Hot vs. Binary

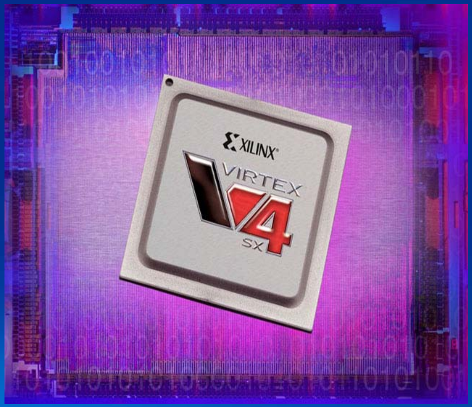
- Implementation:
  - Binary: Number of DFFs = log2(states). Uses decoding logic for next state circuit
  - One Hot: Implemented as a shift register. Minimal decoding logic for next state circuit
- Outputs
  - Binary: outputs depend on every dff + decoding logic
  - One hot: outputs depend on dff of active state – will reduce error cross section in an antifuse device
- Error Detection
  - Binary:
    - can not detect an unanticipated move into a mapped state
    - Can only detect moves into unmapped states
  - One hot:
    - very difficult to erroneously move from one mapped state to another (takes two flips – one must include the bit that is turned on)
    - Subsequently, Can easily detect moves into unmapped states

# Proposed SEU Error Detection: One-Hot

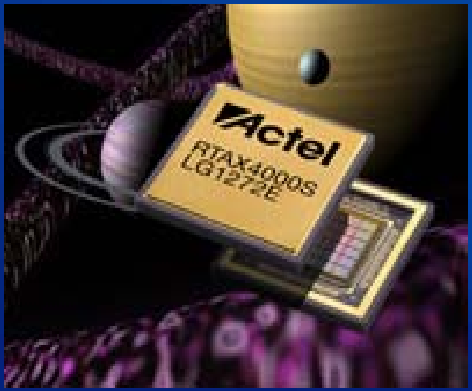
- One-Hot requires only one bit be active high per clock period
- If an SEU occurs,, then an error will be detected by the XNOR.
- Combinational XNOR over the FSM bits is sufficient for SEU detection
- Error Detection can be used to deal with the upset (i.e. reset FPGA)
- XNOR is redundant circuitry. The designer must add appropriate attributes or it will be removed by synthesis



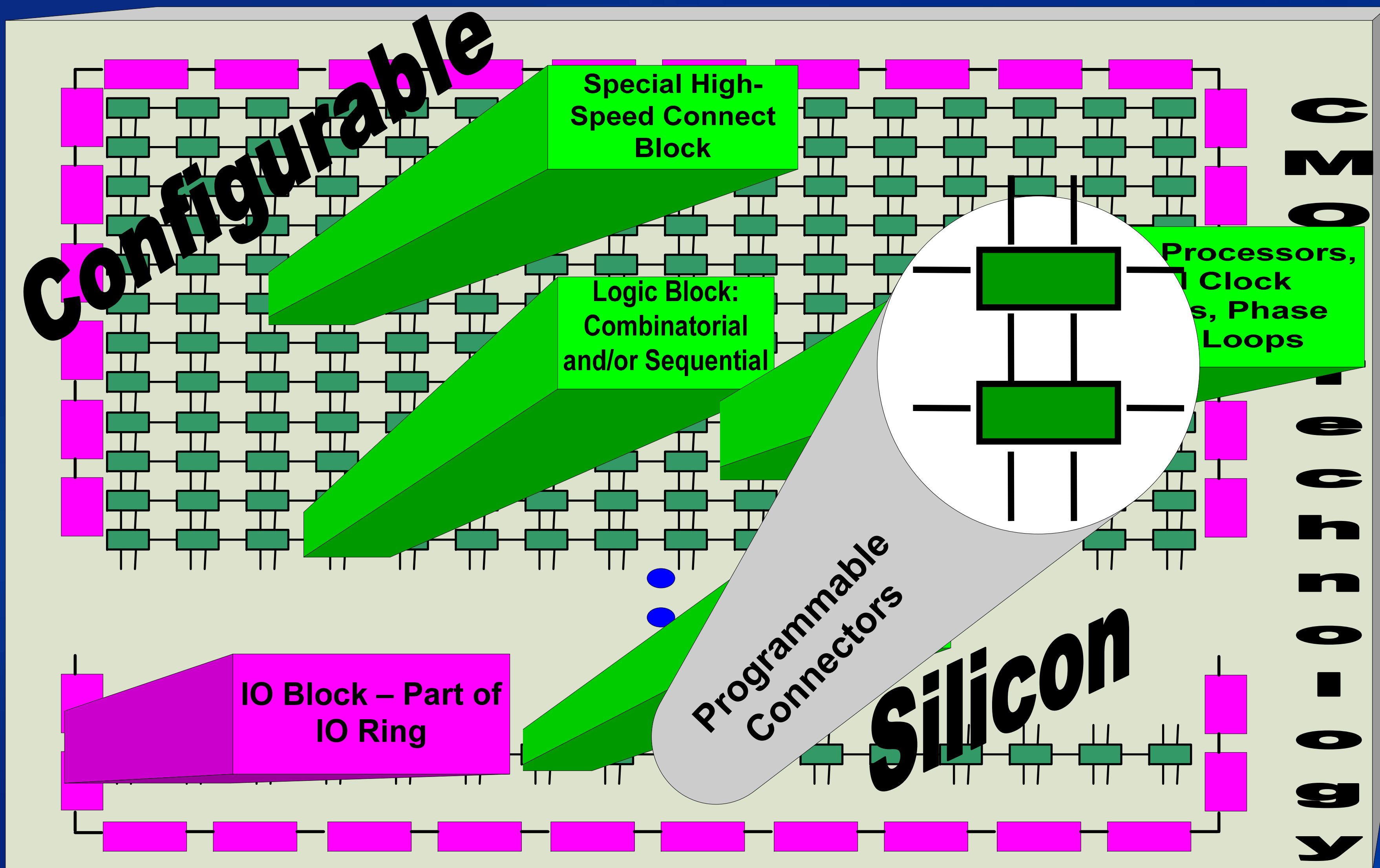



# FPGA Characterization: Understanding the Differences to Effectively Meet Specifications




# General FPGA Architecture



# Configuration: A Major Difference between FPGA Classes

- FPGAs contain groups of preexisting logic: HARDWARE
- Configuration:
  - Arrangement of pre-existing logic
  - Defines Functionality
  - Defines Connectivity
- Common types
  - One time configurable
  - Re-configurable

**CONFIGURATION TYPES**

One Time Configurable → Antifuse

Re-Configurable → SRAM - Based; FLASH - Based

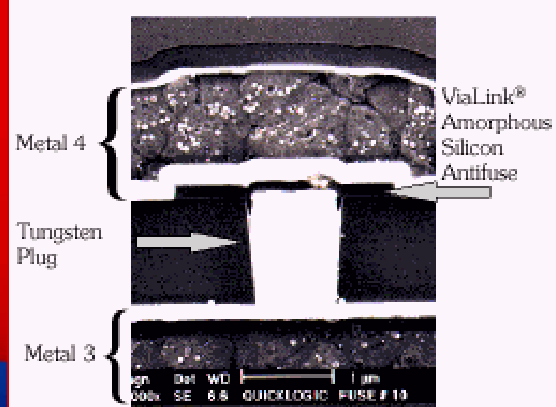
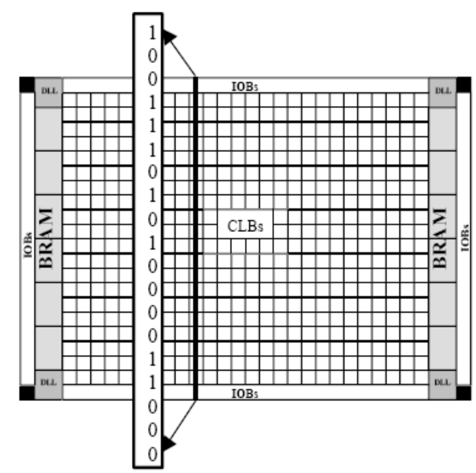



FIGURE 2. ViaLink® Element



# Configuration SEU Susceptibility

- SEU Hardened - Antifuse
  - Configuration is fused into the device –
    - one time configurable
    - Available at power up cycle
  - Not susceptible to SEU
  - Reliability and Dose characterized to acceptable bound
- SEU Tolerant– non-volatile (FLASH)
  - Configuration is store in non volatile memory –
    - Re-configurable
    - Available at power up cycle
  - Slightly susceptible to SEU
  - Dose is a big issue (currently 10-20KRad)'
- SEU Susceptible - SRAM
  - Configuration is store in SRAM memory –
    - Re-configurable
    - Must re-configure during every power cycle
  - Reliability and Dose characterized to acceptable bound

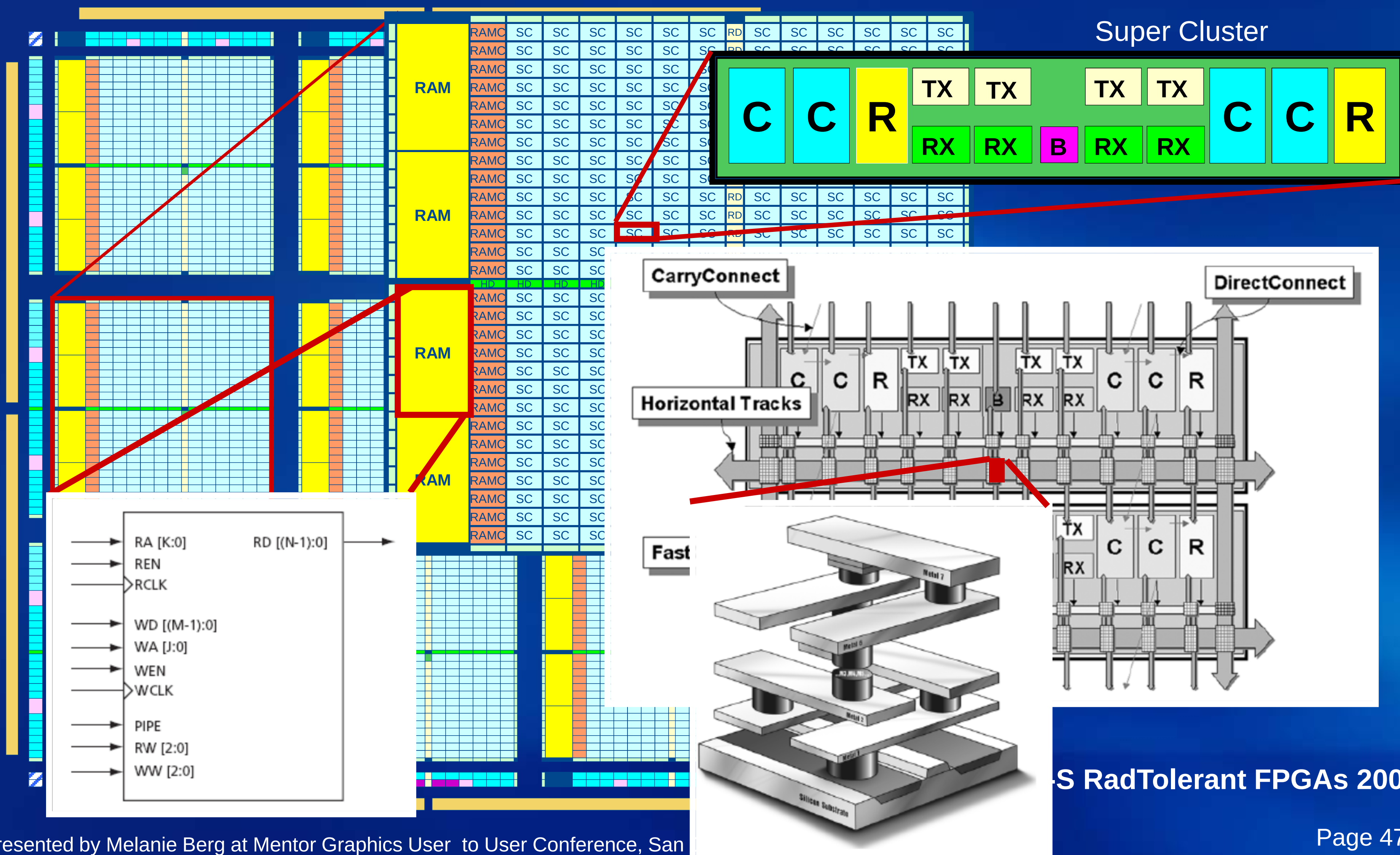
# Antifuse Example: ACTEL

- **Architecture**
- User Inserted Mitigation

# SEU Hardened Antifuse FPGA Devices (Actel)

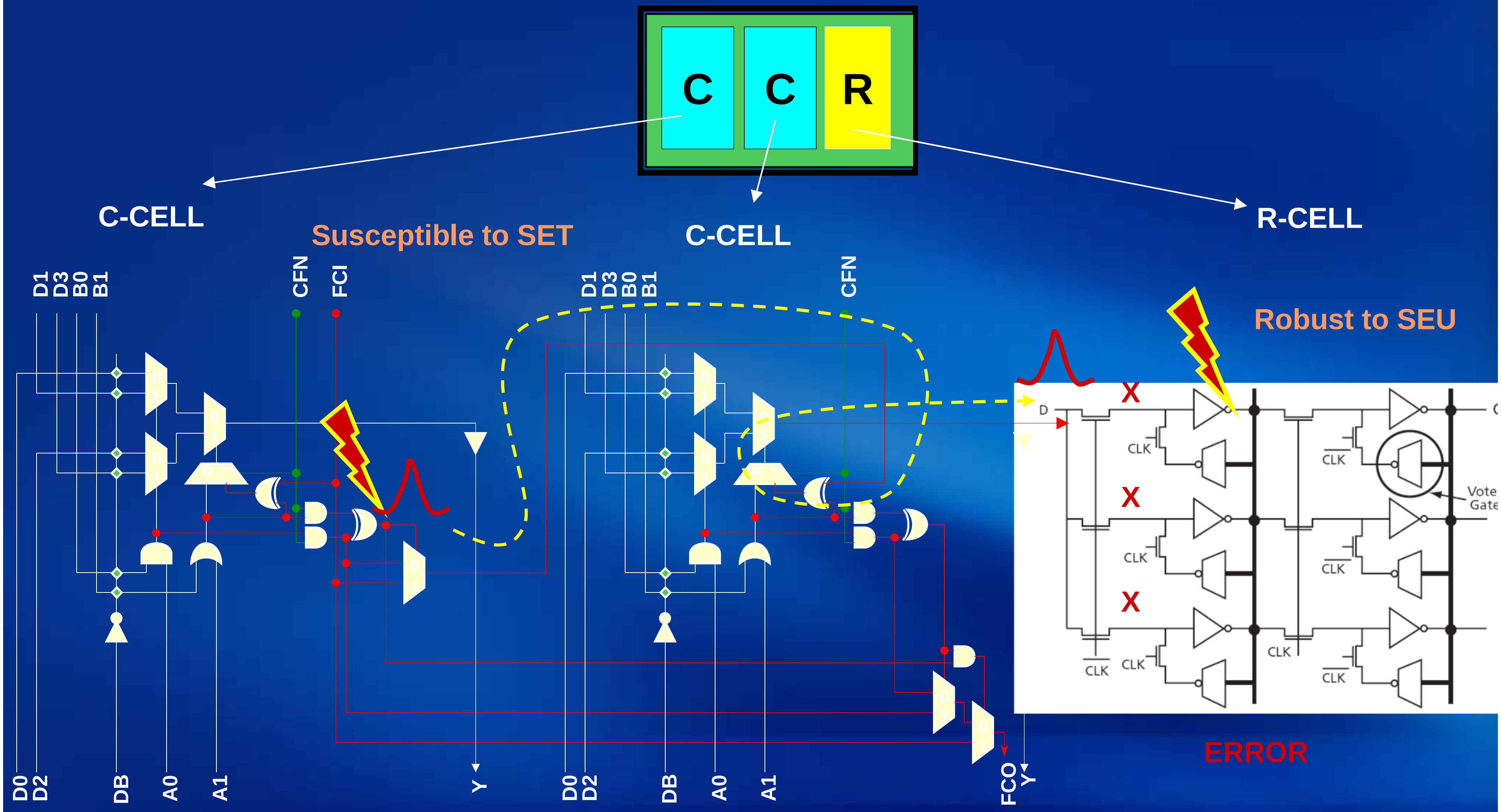
- Hardened Global Clocks with minimal skew (HCLK)
- Hardened Global routes (used for resets)
- Configuration is fused (no transistors) and is thus "HARDENED" – not affected by SEUs
- LTMR at each DFF: Voters are glitch free and are not susceptible (tied together)
  - Uses a wired "OR" to create voter and is embedded inside DFF CELL (RCELL)
  - Wired "OR" is not available to the designer as a usable component
  - Subsequently, a voter created by using library components is susceptible and not as efficient as a RCELL
  - Users should not try to create their own DFF cells (tying together combinatorial logic and inserting their own voter).
    - SEU susceptible
    - Will not adhere to skew requirements as the RCELLs connected to HCLK
- Each cell instantiation contains extra combinatorial logic and can be SET susceptible: Enables, MUXes and route connects

# ACTEL: RTAX-S device




-S RadTolerant FPGAs 2007]

# ACTEL: RTAX-S Device; LTMR and Single Event Upsets




[Actel, RTAX-S RadTolerant FPGAs 2007]

# Antifuse: User Insertion of Additional Mitigation for Highly Critical Applications

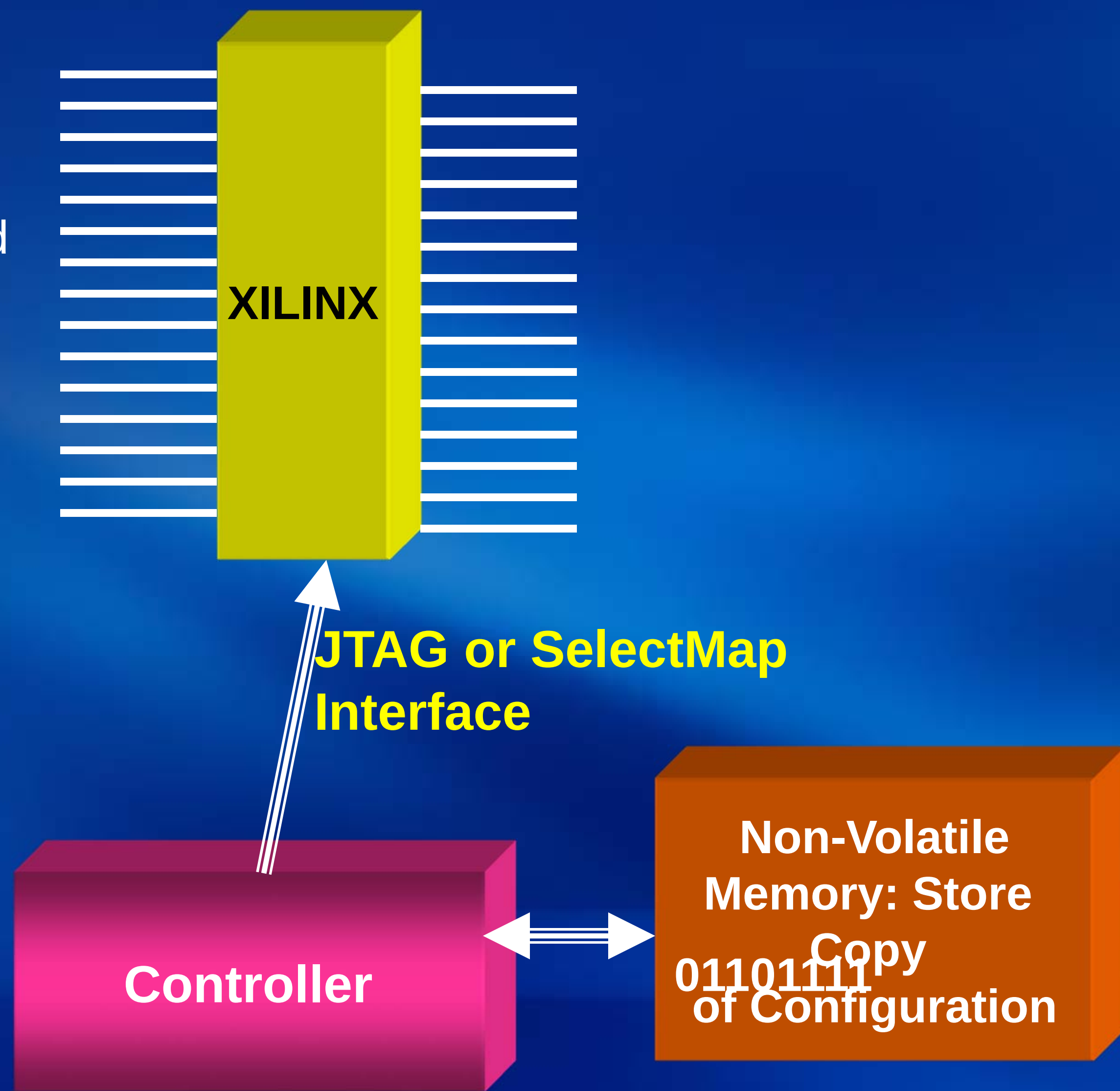
- LTMR: Device already contains LTMR. The susceptibility is in the shared data path. Additional LTMR will not remove SETs (may even increase SETs because of additional voter insertion)
- DTMR: best solution. Clocks, Resets, and configuration are hardened. This will protect against SETs
- GTMR: Can be overkill. But for highly critical missions will give extra protection on clock and reset trees plus I/O
- Remember: most missions do not require the additional mitigation to antifuse devices

# SRAM Based FPGA Example: Xilinx Virtex 4 Series

- **Architecture**
- User Inserted Mitigation

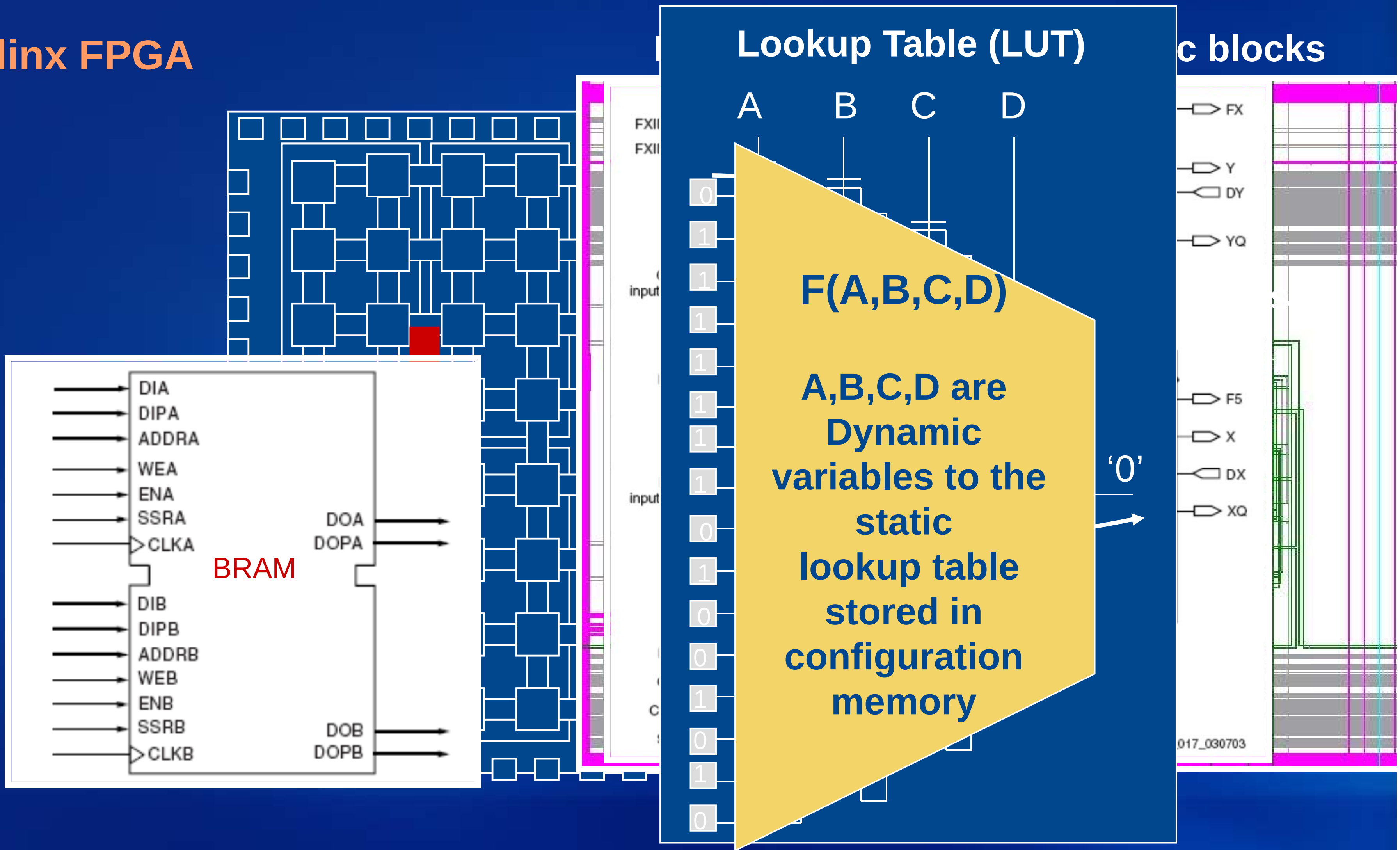
# Configuring Xilinx Devices

- Configuration is stored in SRAM
  - Advantage: Can change functionality while in flight
  - Disadvantage: Configuration is stored in SRAM and is SEU susceptible
  - Devices need to be configured at power up
  - Configuration is loaded into the Xilinx Device through the JTAG or SelectMap Interface
- Additional hardware necessary for (re)configuration
- There are no SEU hardened structures
- Additional design complexity necessary for mitigation



# General Virtex FPGA Architecture

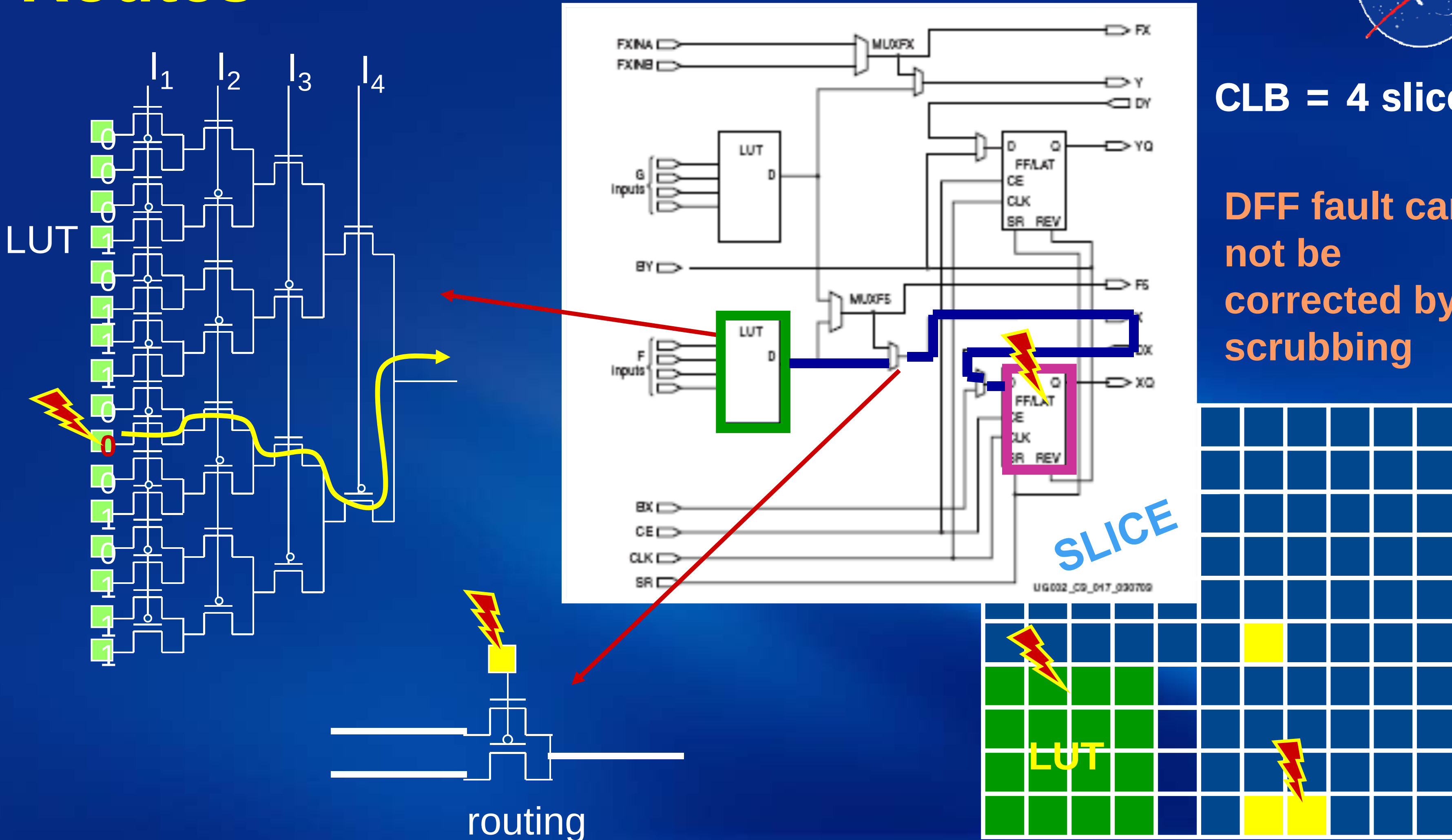
## Xilinx FPGA



# SEUs in SRAM Configuration: CLB and Routes

**CLB = 4 slices**

**DFF fault can not be corrected by scrubbing**

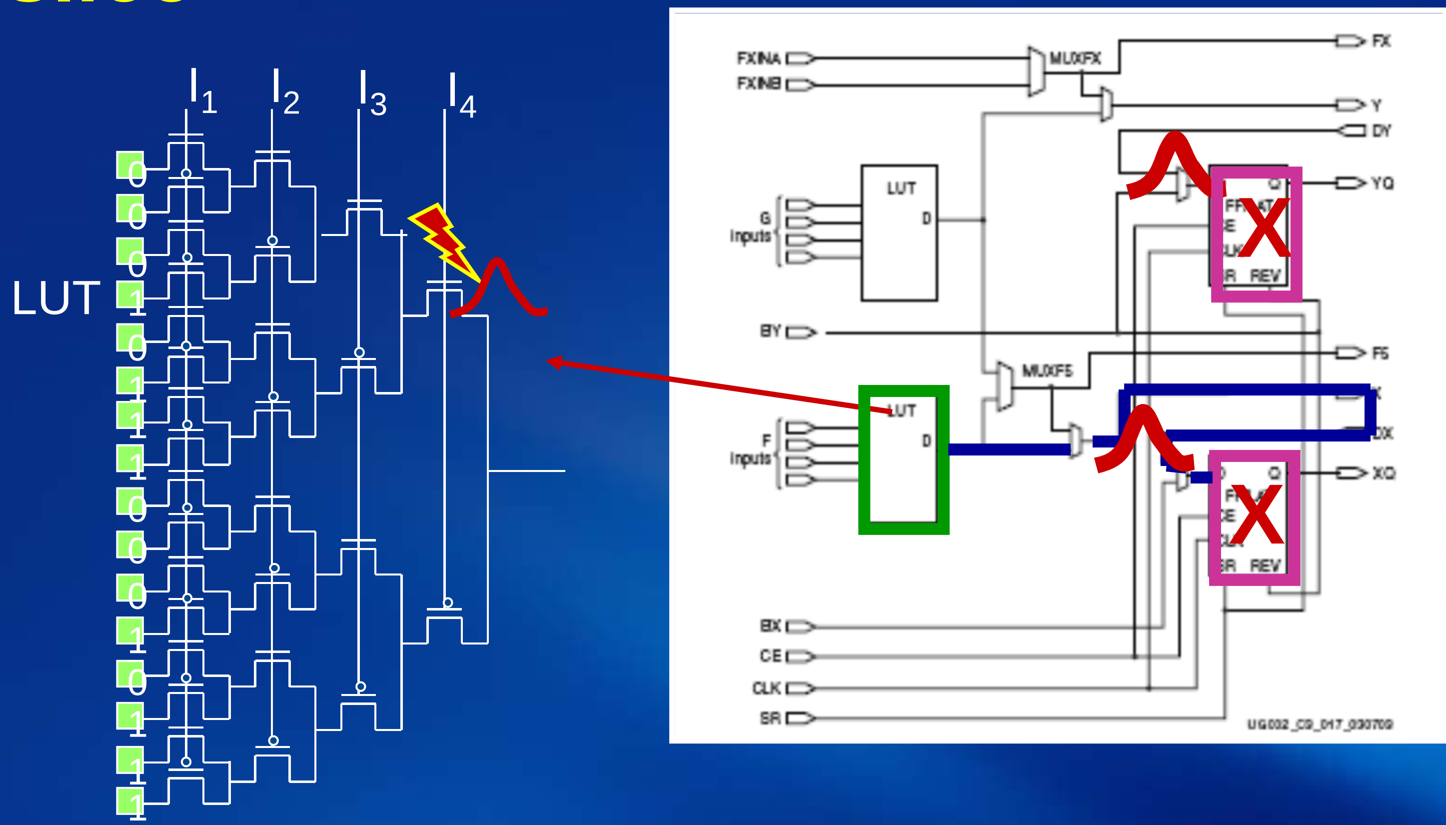



**Configuration memory bits**

**LUT function is incorrect with a Configuration bit flip (corrected by scrubbing)**

**– Its affects of incorrect functionality is not corrected by scrubbing**

# SETs in SRAM-based FPGAs : CLB slice



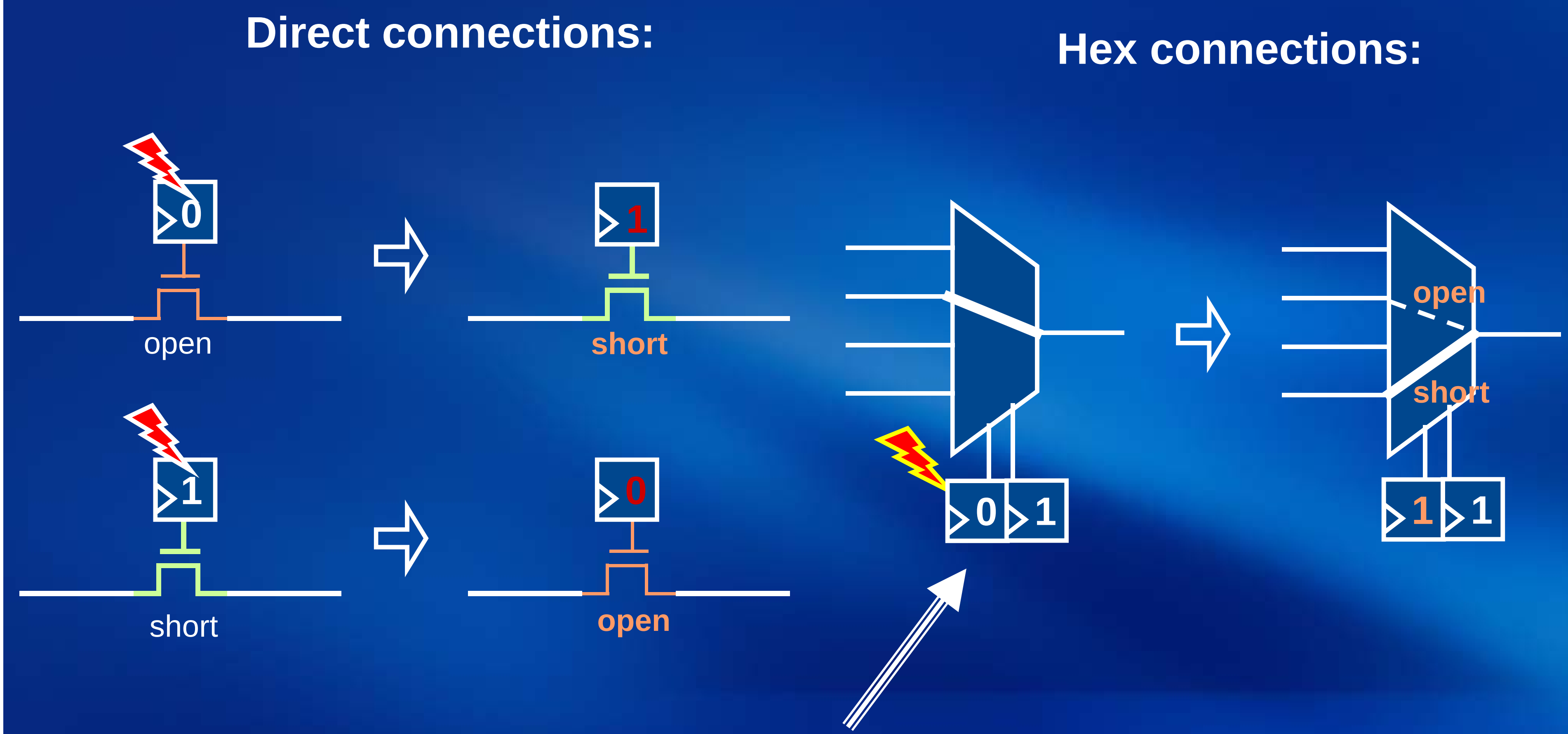

**CLB = 4 slices**

**SET may be captured by the DFF.**

**P(SET) <<P(SEUConfig)**

# Routing SEUs






**Each have static connections to the configuration memory and are all susceptible to SEU strikes**

# Other sensitive structures

## *SEFIs*

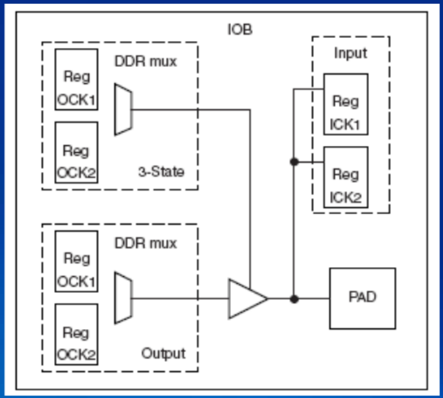
## Can't Mitigate

- **Power on Reset (POR)**
  - **Low probability of occurrence**
  - **Signature: done pin transitions low, I/O becomes tri-stated, no user functionality available**
  - **Solution: reconfigure device**
- **SelectMap and JTAG Controllers**
  - **Low probability of occurrence**
  - **Signature: loss of communication, read access to configuration memory returns constant value.**
  - **Solution: reconfigure device**

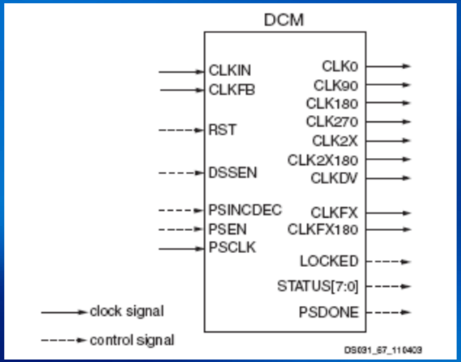
## Can Mitigate

- **Global Routes (Clocks and Resets)**
  - **Clock tree or reset tree**
  - **Probability of occurrence is significant**
  - **Signature: State space is totally disrupted Solution: reset device**

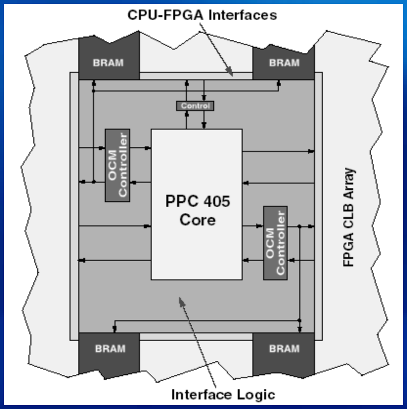
**Input and Output Blocks (IOB)**



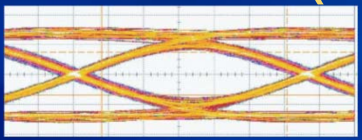

**Digital Clock Manager (DCM)**




**Power-PC Hard IP**




**Multi-Gigabit Transceivers (MGT)**



# SRAM Based FPGA Example: Xilinx Virtex 4 Series

- Architecture
- User Inserted Mitigation
  - Logic
  - Scrubbing
- Verification (beyond Simulation)

# Xilinx: Radiation Tolerant Parts

Guaranteed Radia... ...rance

Virtex-4QV FPGAs are g... single-event latch-up (SE... of SRAM-based FPGAs in h... with JPL founded the SEE Co... characterization and report th... *http://parts.jpl.nasa.gov/resou...*

- **Total Ionizing Dose**–Xilinx ... ensure that device perform... electrical specification re... parameters at 300 krad...
- **Single-Event Latchu...** incorporates a thin ... for latch-up immu... consortium verifi... temperature, s... particles/cm2... MeV-cm²/m...
- **Single-E...** ...conducts additional experiments in heavy ion, proton, and neutron environments in order to measure and document the susceptibility and consequence of SEU(s). The SEE Consortium oversees and validates the test methods, empirical data collected, and resulting analysis.

In conjunction with the SEE Consortium, Xilinx develops beam-tested, upset mitigation solutions. For mitigation, Xilinx provides triple modular redundant reference designs, configuration memory scrubbing application notes, and the TMRtool™ for automating error-free triplication of designs destined for space.

**Single Event Upset** –

Xilinx conducts additional experiments in heavy ion, porton, and neutron environments in order to measure and document the susceptibility and consequence of SEU(s)

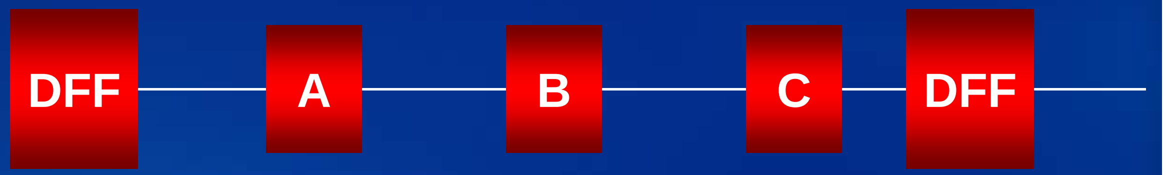
...izing dose (TID)

...SEL) immunity

- Not to be confused with RHBD – Logic is still susceptible
- Many systems may not require mitigation:
  - Data processing
  - Non critical controllers
- User must insert mitigation to reduce SEU cross section if required by system
- Xilinx has a TMR tool and support for mitigation insertion (XTMR)

# Xilinx and XTMR

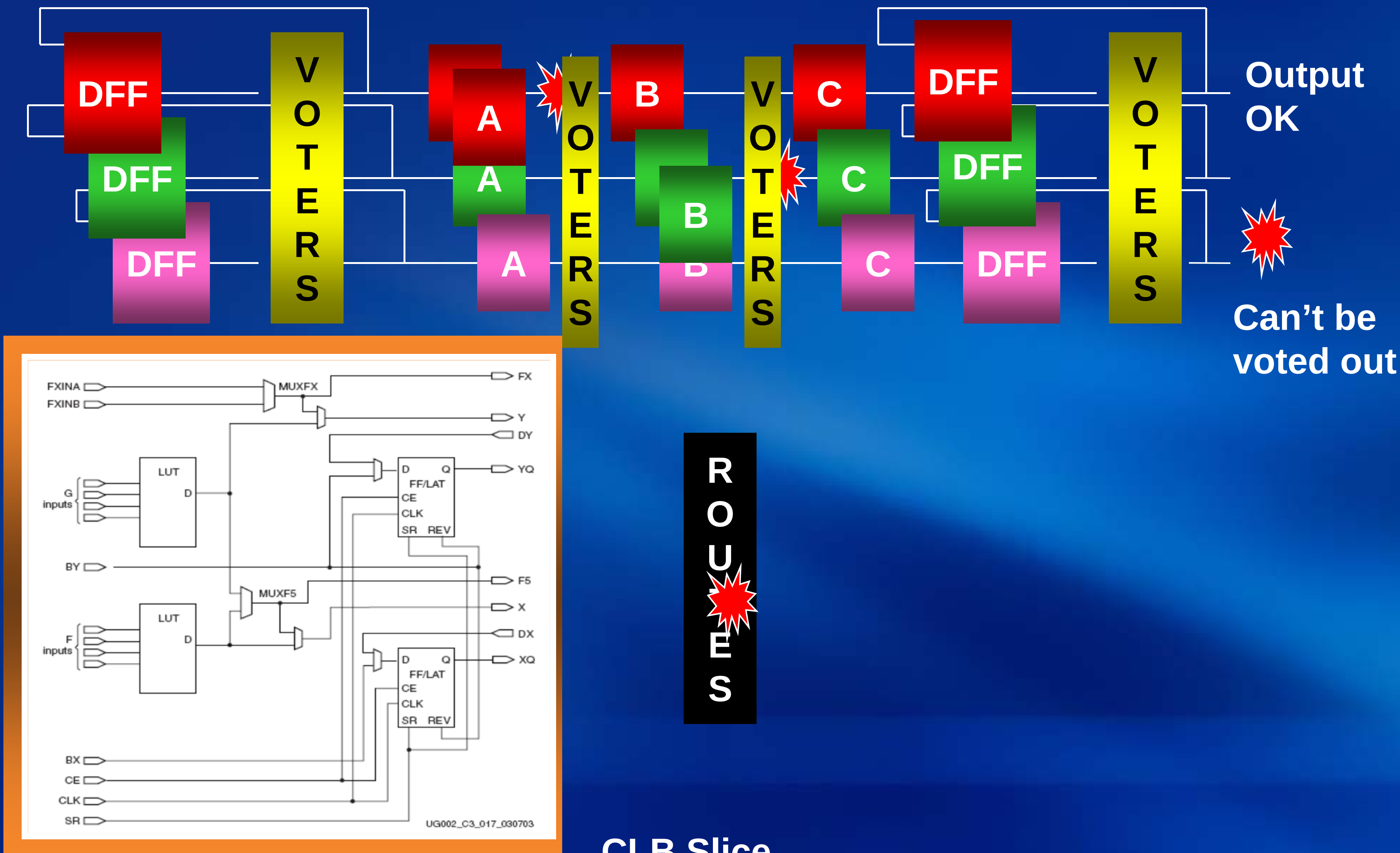
- Configuration is sensitive: Need to triplicate data paths to protect:
  - Logic
  - Routes
- Clocks are sensitive: Need to triplicate clock domains
- GTMR is the best solution
- Xilinx offers XTMR




**Start with non redundant paths of logic**

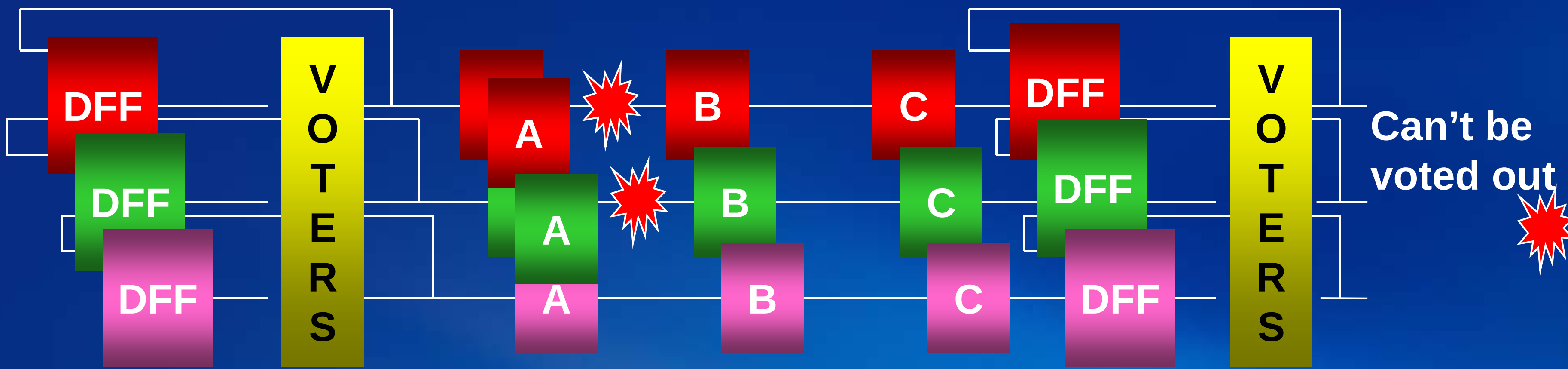
1. **Triple everything**
2. **Insert voters after DFFs that contain feedback**
3. **Place and route must be taken into consideration with mitigation approach**

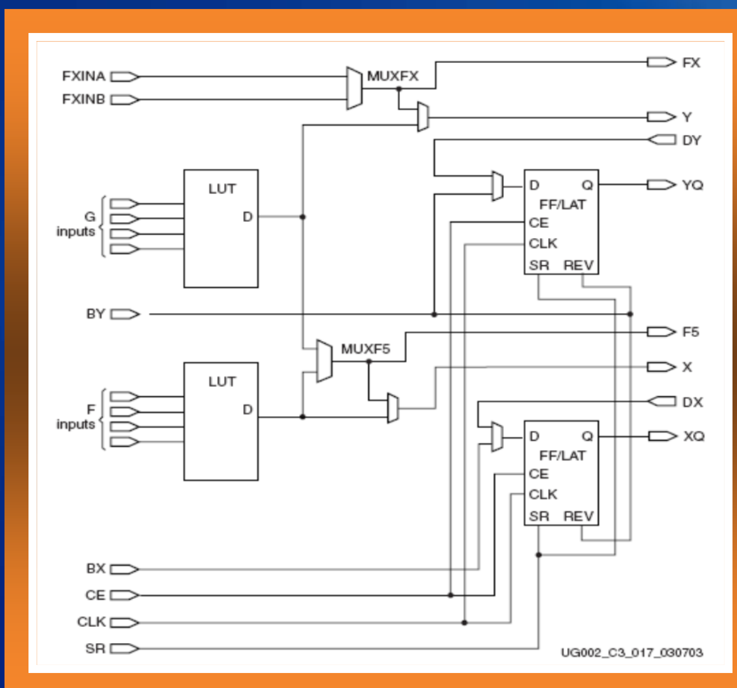
# Mitigation with Respect to Place and Route

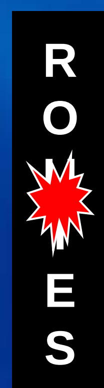



CLB Slice

# Mitigation with Respect to Place and Route

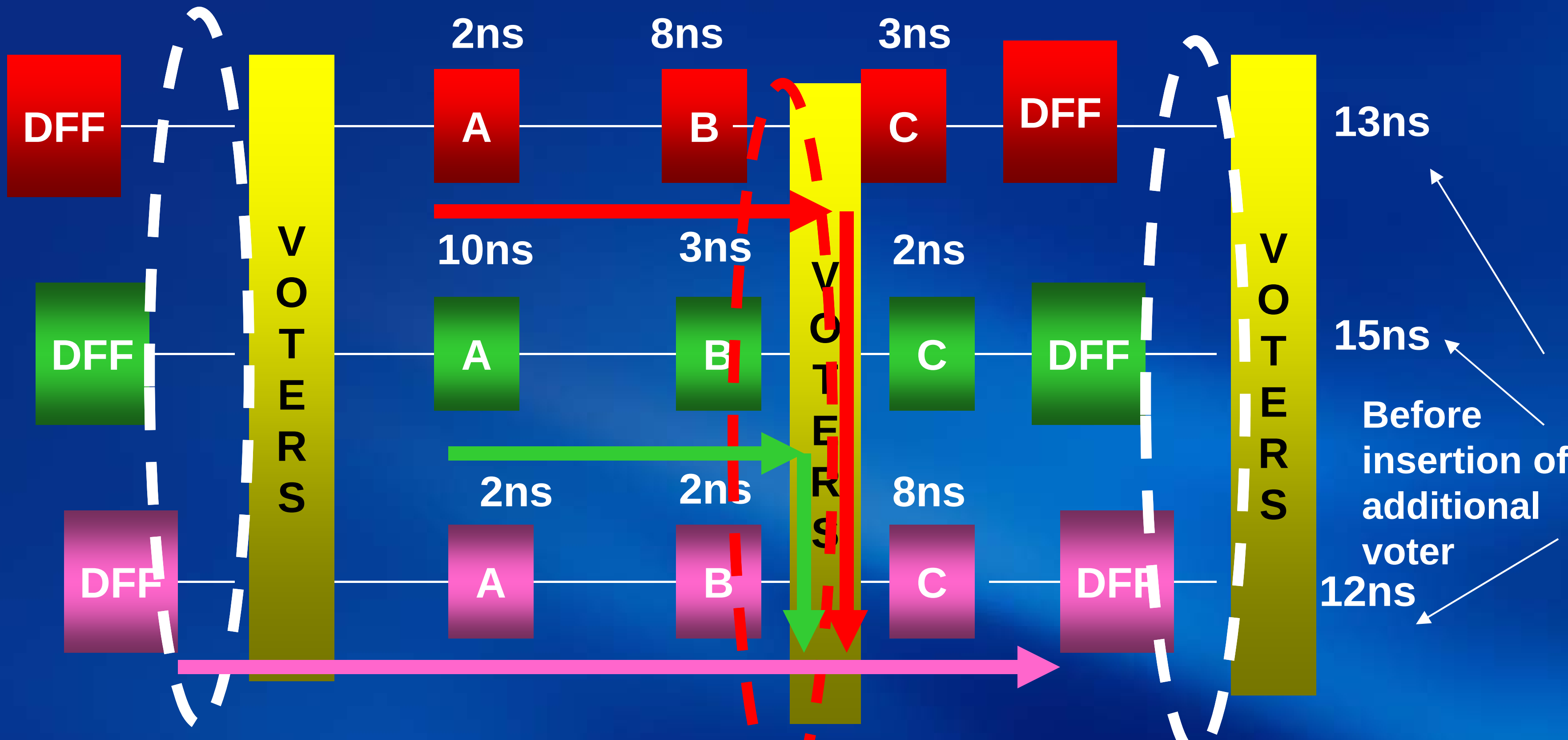



1. Additional voters will not help this situation
2. Additional voters adds to congestion
3. Additional voters lead to higher probability of two voters being placed in a common CLB (increase in error cross section)
4. Additional voters adds to STA problems

# Additional Voters and STA: Example with 16ns Time Constraint




Will not make timing: 10n + 3ns+8ns > 16ns constraint

Guaranteed minimal skew

Too much skew

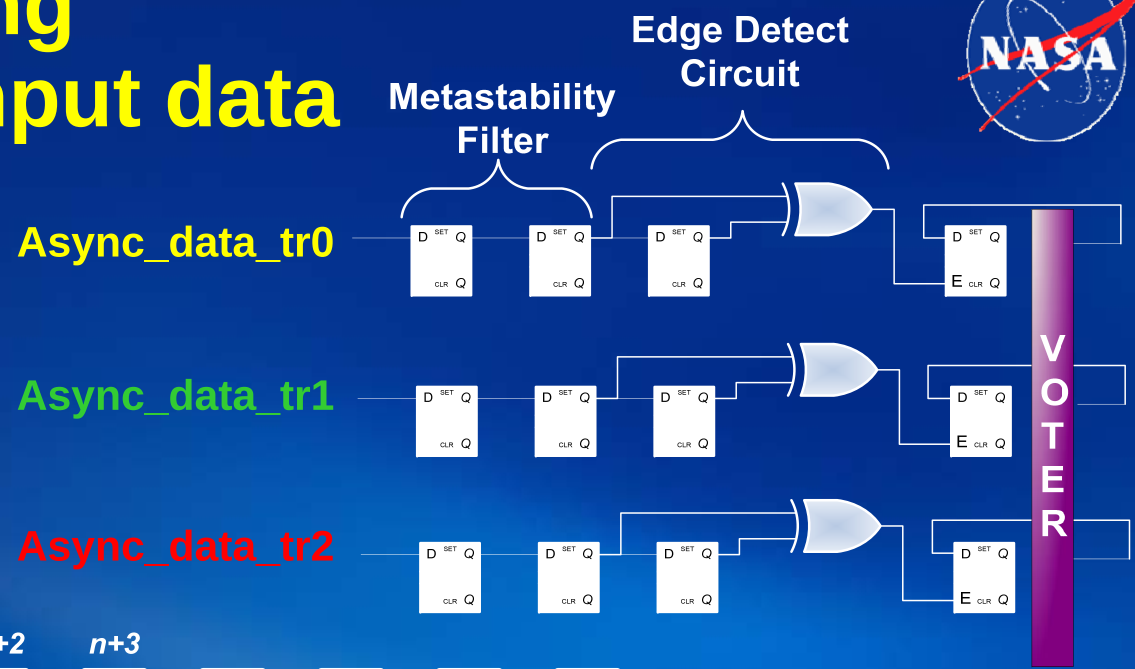
STA will not report the timing error because it is across clock domains

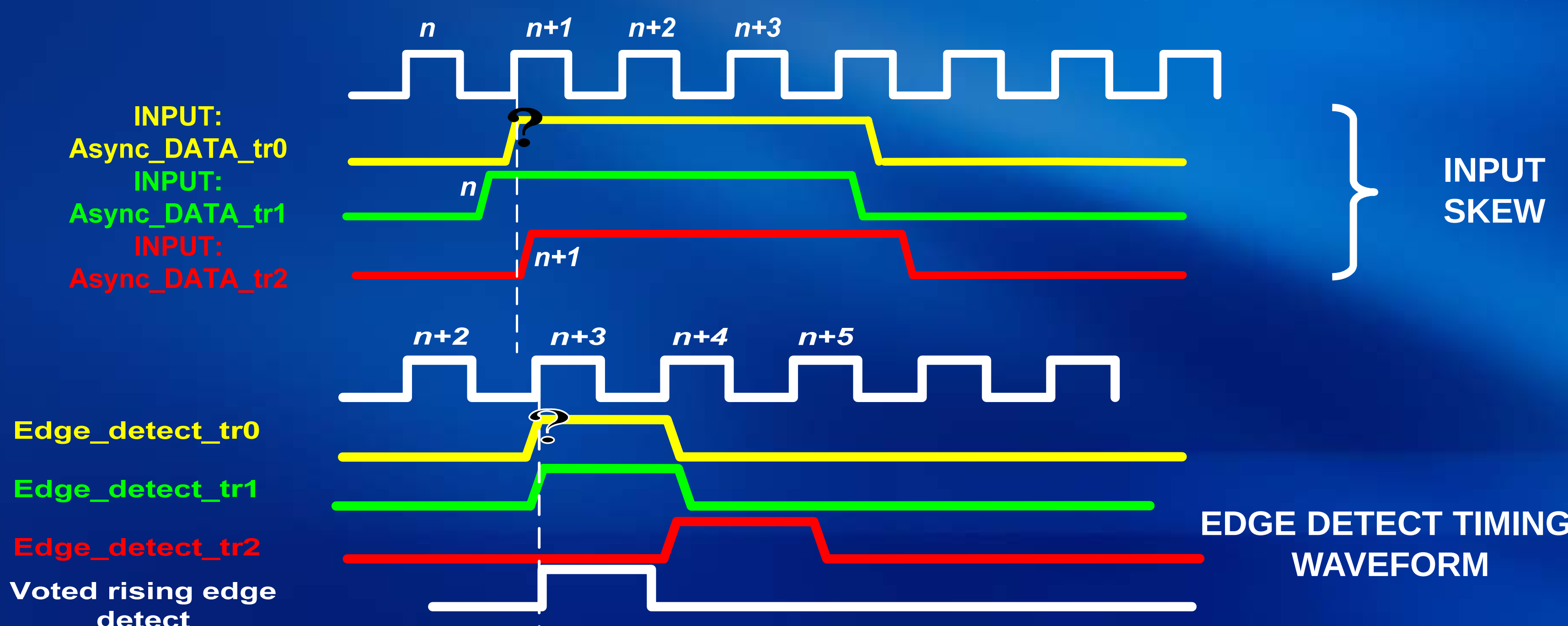
Best to have Voters anchored at DFF Boundaries

# XTMR – Capturing Asynchronous Input data

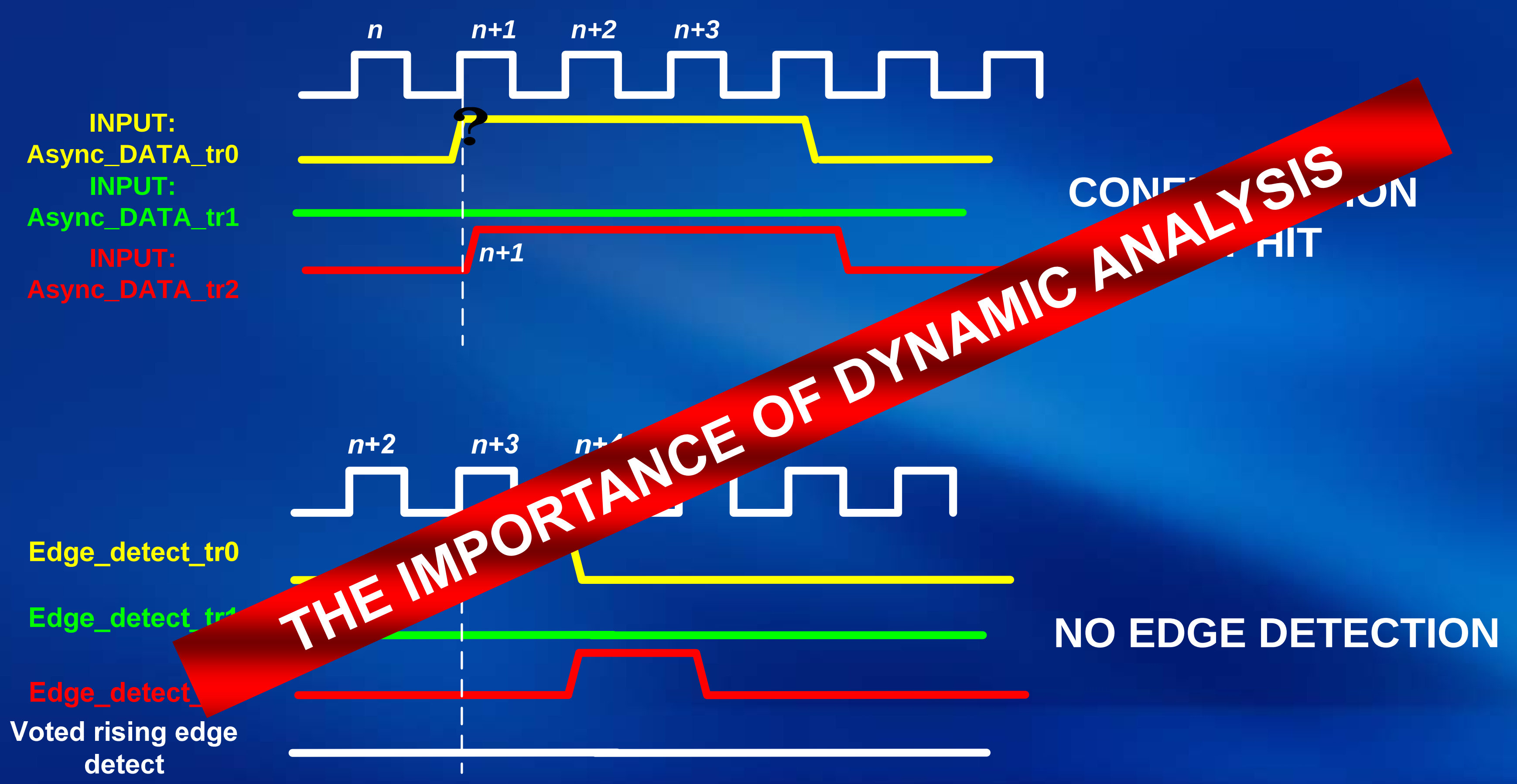
## Dynamic Analysis:

- Timing wrt to operational clocks and changing data
- Takes into account asynchronous signals



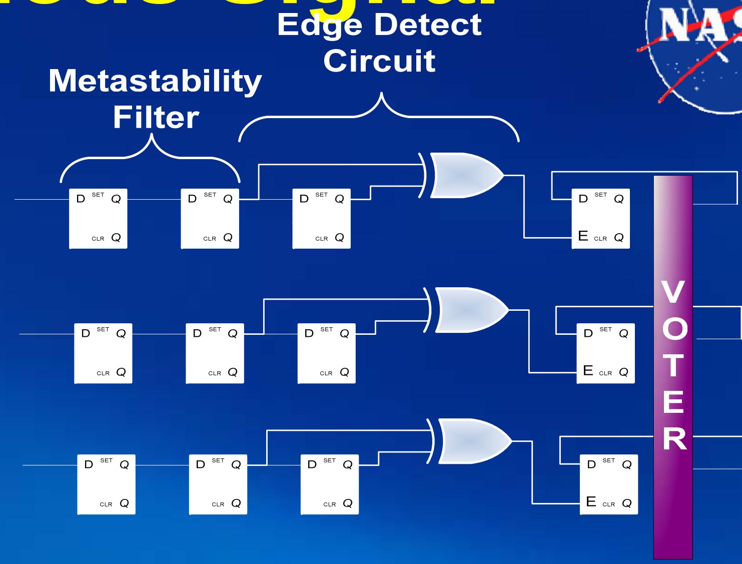



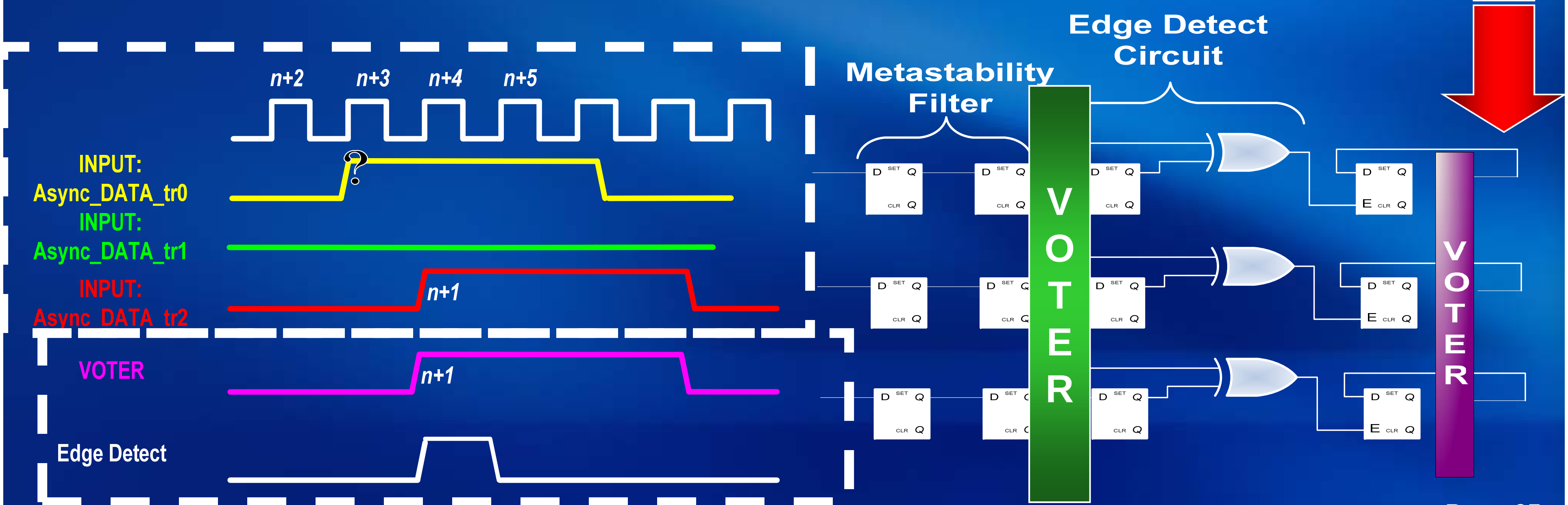
# Time Domain Considerations: GTMR Single Bit Failures …Not Detected by Static Node Analysis



# Voters and Asynchronous Signal Capture

- Place voter after metastability filters
- It satisfies skew constraints because voter is anchored at DFF control points



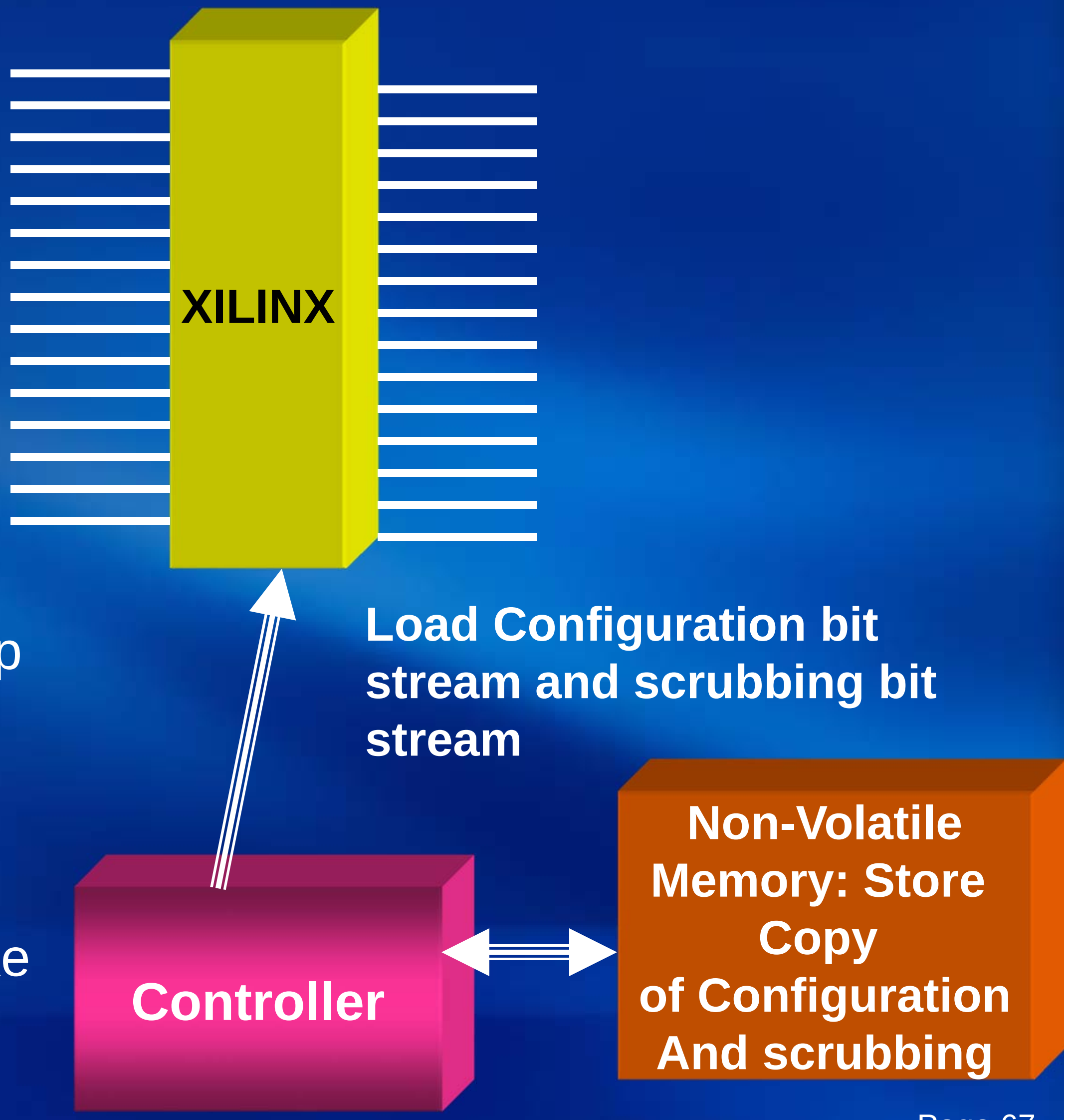



# SRAM Based FPGAs and other Mitigation Strategies

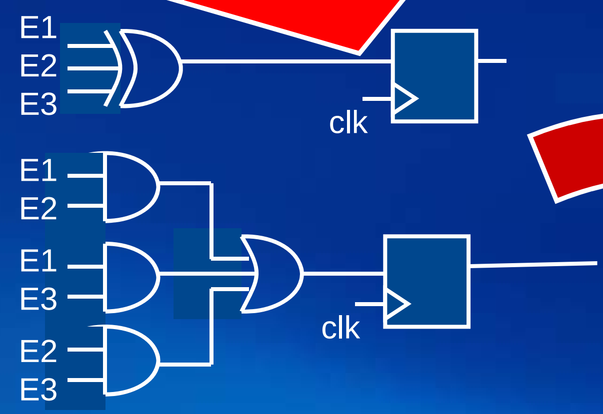
- Beware of mitigation schemes that do not triplicate the clock domains
  - Should triplicate because clocks are susceptible
  - DTMR may not be sufficient
  - Partial mitigation schemes may not be sufficient
- Other Mitigation strategies can be placed in conjunction with GTMR (XTMR)
  - Scrubbing (obvious)
  - State Machine error detection
  - General EDAC circuitry

# Scrubbing: An Enhancement to XTMR

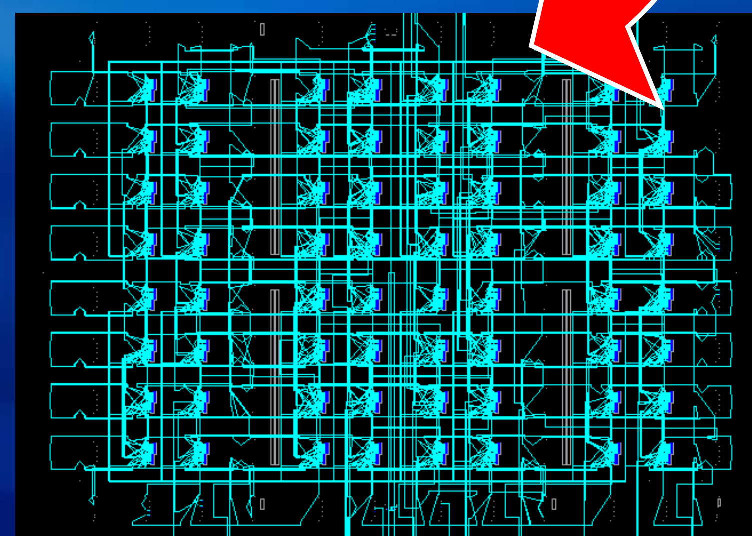
- Does not decrease SEU Rate
- Protects against accumulation of configuration upsets
  - Writes over incorrect bit with correct data while system is fully operational
  - Multiple errors can break XTMR.  Scrubbing will help because it will decrease accumulation
  - Scrubbing will not protect XTMR against a Multiple Bit Upset (MBU) (one strike hitting multiple nodes at one time).



```
Process(sysclk,reset)
If reset = ‘0’ then
        dff <= ‘0’;
Elsif (rising_edge(sysclk) then
        Dff <= E1 xor E2 xor E3
End if;
End process
```




# Radiation Effects and Commercial Tools



# Antifuse devices: Clock and Reset instantiation

- Space-grade antifuse devices contain hardened global routes:
  - Clocks
  - Resets
- It is best practice for the designer to instantiate the appropriate clock tree buffers
- Synthesis (Precision) understands the hardened routes and will automatically place clocks and resets on the global routes:
  - Clock can only feed clock pin of DFF
  - Resets must only go to reset pins (for guarantee)
  - Aeroflex: clock can only be rising edge
  - Aeroflex: reset can only be active high

# Safe State Machines

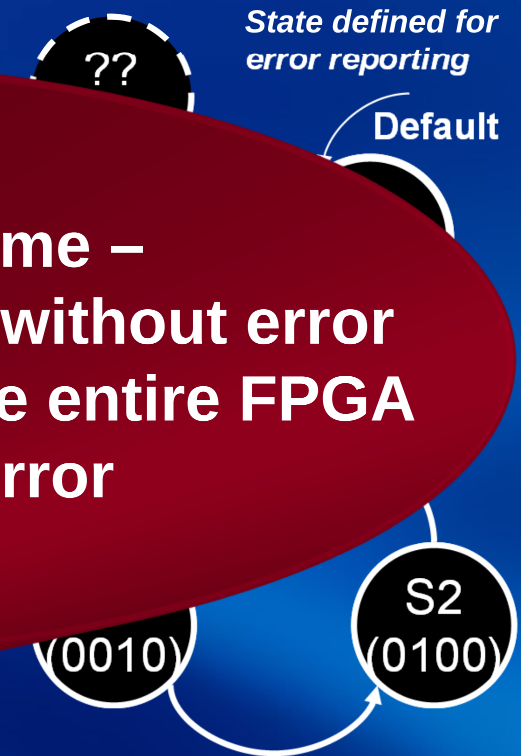
- Normal Mode (with no safe attribute applied):
  - Default/others clause is ignored by synthesis tool
  - Impossible to code in unreachable states (states that occur by just a bit flip and not actual next state logic)
- Precision has responded to the Aerospace industry and have provided a "safe" state machine option
  - Does not provide mitigation
  - Provides detection and a jump to a designated state
  - User must insert additional recovery circuitry (i.e. do not just only apply the safe attribute to a state machine)
- If not utilized correctly, recovery can either be unsafe or unfeasible

# Example: Safe FSM Operation in Precision

- Implements all possible states (including those unspecified in RTL)
- "Invalid States' transition to state specified in "default"/"when others"
- Subsequently, Go... behavior for ... bits
- Precisio... indicat... be pro...
- user...

**Have a recovery scheme –**
**don't just use the safe option without error Detection/indication so that the entire FPGA can respond to the error**

## Safe FSM





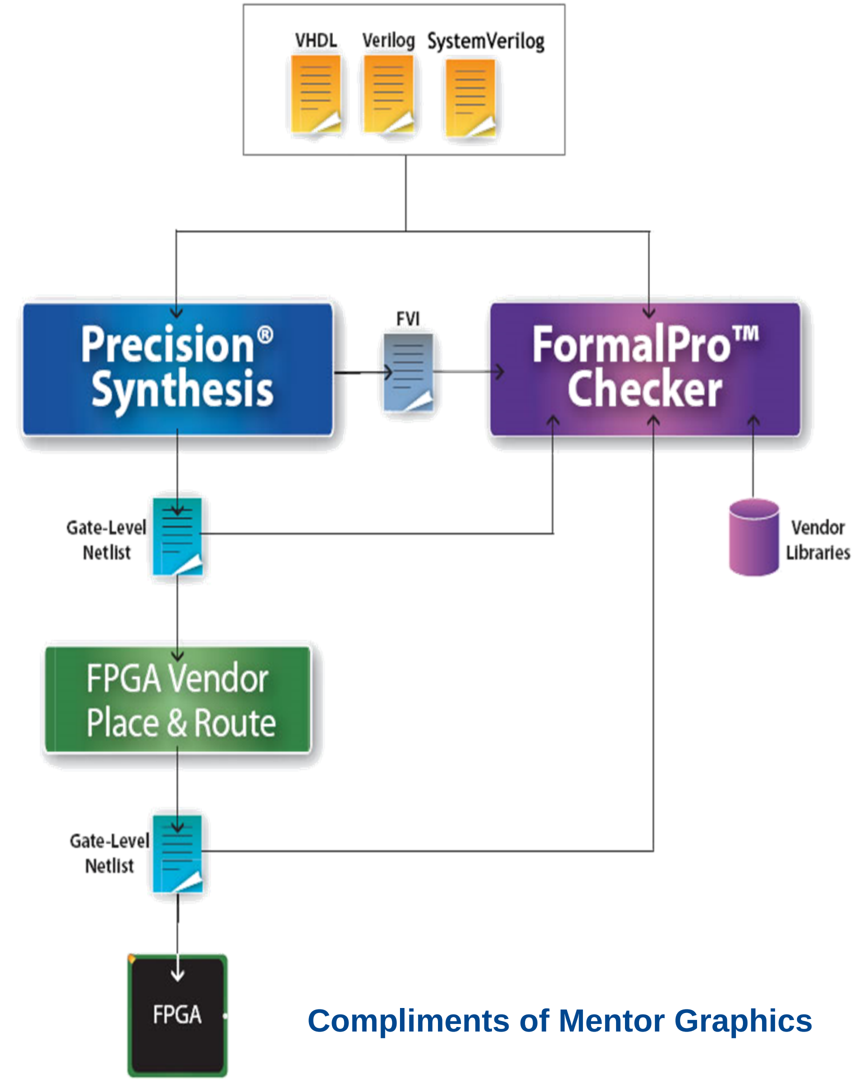
Compliments of Mentor Graphics

# XTMR

- XTMR is a tool offered by Xilinx specifically for Virtex II, Virtex IV, and Virtex V families
- Implements GTMR
- Module selectable
- I/O selectable
- Removes Half latches and brings the constants to output pins (found not to be 100% in Virtex IV devices)
- Currently, best choice for Xilinx devices – recognizes SEU and SET Virtex specifics
- User must take the responsibility of skew minimization as earlier presented
- Always check that voters have been inserted properly
- **Biggest Caveat – how do you verify that original circuit is not broken by XTMR insertion**

# Mentor Graphics FPGA Synthesis and Equivalence Checking

- RTL ⇔ Gate-Level equivalency checking with FormalPro™
- Catch all functional errors without simulation time or setup
  - Increases confidence in mapping result
- Supports all major vendors
- Automated setup of FormalPro from Precision
- Supported Precision optimizations
  - Merged registers
  - Duplicated registers
  - Inferred counters
  - Inferred static SRL
  - Eliminated registers
  - Re-encoded FSM
- *Supports LTMR*
- *Promising for XTMR*




**Compliments of Mentor Graphics**

# User Implemented Redundancy

- Redundancy is required for mitigation
- Synthesis tool removes redundant logic (area optimization)
- Verify that mitigation reduces the error cross section – otherwise it adds to the system complexity
  - verification
  - implementation
- User must place attributes on mitigation logic during synthesis and during place and route
  - Syn_keep
  - Syn_preserve
  - Don’t_touch
  - No_optimize

# Summary

- Space Radiation Environment and CMOS Technology:
  - CMOS transistors have become more susceptible to incurring faults due to:
    - the reduction in core voltage
    - decrease in transistor geometry, and
    - increase in switching speeds,
  - Defined Key terms: SEU, SET, SEL, TID, SEFI
  - Illustrated ionization effects
- Basic Synchronous Design Concepts and Potential Faults
  - Discussed synchronous deterministic behavior and analysis embedded in asynchronous fault environments requires additional inspection beyond STA
  - Defined the role of a clocks, DFFs, global routes, and combinatorial logic with respect to SEUs
- General Fault Tolerant Design Strategies for Mitigating Radiation Effects
  - The Benefits and caveats of various mitigation implementations were presented
    - LTMR, DTMR, GTMR
    - DICE

# Summary

- State Machines
  - Discussed how safe “safe” state machines actually are
  - Safe state machines do not contain redundancy (no correction just detection)
  - Users must be aware of encoding schemes and potential increase in error cross section
  - One hot is the safest encoding scheme
- FPGA Characterization: Understanding the Differences to Develop a Comprehensive Analysis
  - Illustrated the difference between Antifuse and SRAM devices
  - Discussed Data Sheet interpretation
  - Presented common mitigation techniques specifically per device:
    - LTMR – Antifuse
    - GTMR (XTMR) and scrubbing for SRAM based
  - Noted the existence of hardened global routes and configuration in antifuse devices
  - Listed an estimate of usable DFFs for SRAM vs. antifuse devices in a mitigated environment

# Summary (continued)

- Radiation Effects and Commercial Tools
  - Presented specific attributes of common user tools geared towards space grade FPGA designs
  - Discussed the use of Formal checkers
  - Noted the importance of including attributes to synthesis and place and route tools when implementing mitigation

## Most Importantly:

- **There is no one button solution for insertion of complex devices with complex applications into critical space systems.**
- **It is now a necessity for designers to take into account Radiation Effects Information at all levels of the Design Cycle**