New Developments in Field Programmable Gate Array (FPGA) Single Event Upsets (SEUs) and Fail-Safe Strategies

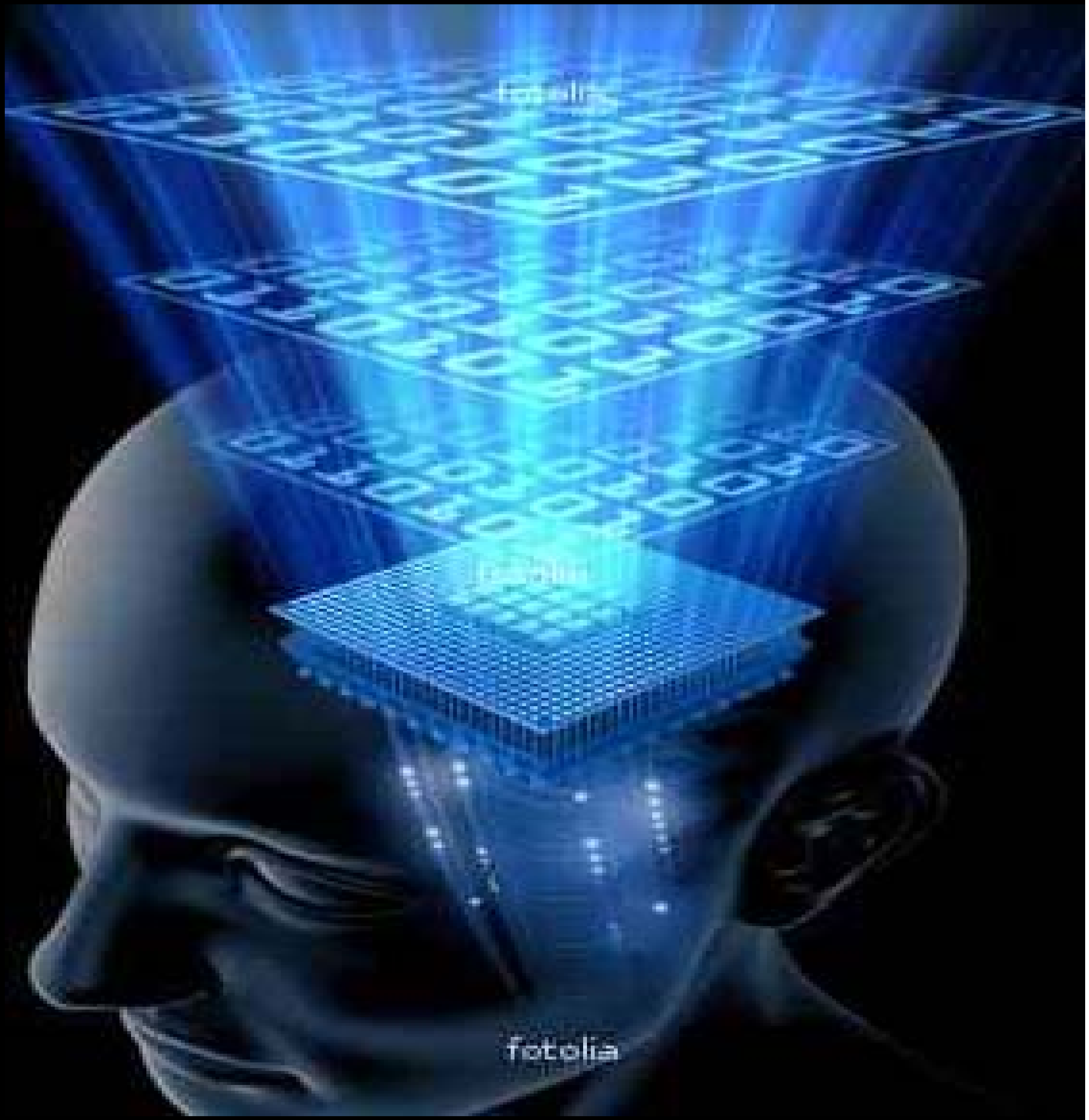
Melanie Berg, MEI Technologies in support of NASA/GSFC

To be presented by Melanie Berg at the Revolutionary Electronics in Space (ReSpace) / Military and Aerospace Programmable Logic Devices (MAPLD) 2011 Conference, Albuquerque, NM, August 22-25, 2011, and to be published on nepp.nasa.gov web site

Acknowledgements

- *Some of this work has been sponsored by the NASA Electronic Parts and Packaging (NEPP) Program and 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*
- *Thanks to the MAPLD committee for welcoming me back as a tutorial instructor*

Before we get started… here's some things to keep in mind during this very long presentation



Single Event Upsets (SEUs) and Field Programmable Gate Arrays (FPGAs)

- Ionizing particles cause upsets (SEUs)
- Each FPGA type has different error signatures
- Regarding SEUs, the question is how to avoid system failure
- The answer depends on
  - The system's requirements and the definition of failure
  - The target FPGA and surrounding circuit susceptibility
  - Implemented fail-safe strategies
  - Radiation environment
  - Trade space and decided risk

Fail-Safe Strategies…Don't Get The Actions Confused…There's A Difference

- **Detection:**
  - Watchdog (state or logic monitoring)
  - Simplistic Checking … Complex Decoding
  - Action (correction or recovery)
- **Masking**
  - Not letting an error propagate to other logic
  - Redundancy+mitigation or detection
  - Turn off faulty path
- **Correction**
  - Error state (memory) is changed
  - Need feedback
- **Recovery**
  - Bring system to a deterministic state
  - Might include correction

SEU Induced Fail-Safe Concerns for FPGA Based Systems

- Are you reducing error rate?
  - Be careful not all FPGAs have the same Single Event Upset (SEU) error signatures… don't be fooled
  - Poorly selected/implemented Mitigation scheme may increase upset rate instead of decrease
- Accumulation versus Multiple Bit Upsets (MBUs) may need to be handled differently (rate of correction versus correction technique)
- Tradeoffs: Is your scheme buying you anything?
  - May reduce system error rate at a high cost (area, power, complexity, cost)
  - STOP…. Requirements may not need Mitigation
  - If you can't validate that it meets requirements – then risk is high

Things To Look Out for In a Design Review: How Safe is Your Design?

- Are SEU error modes addressed properly?
- Did you mitigate where you expected to mitigate?
- Are there lock-up conditions in the design?
- Does your strategy protect the entire critical path?
- Is the synthesized design fail-safe?
- Can your watch-dog catch failure?
- Will your recovery scheme work?
- What are the limitations of your verification strategy?

The list goes on… Based on error signatures of the target FPGA, the designer must keep all points in mind at all stages of the design

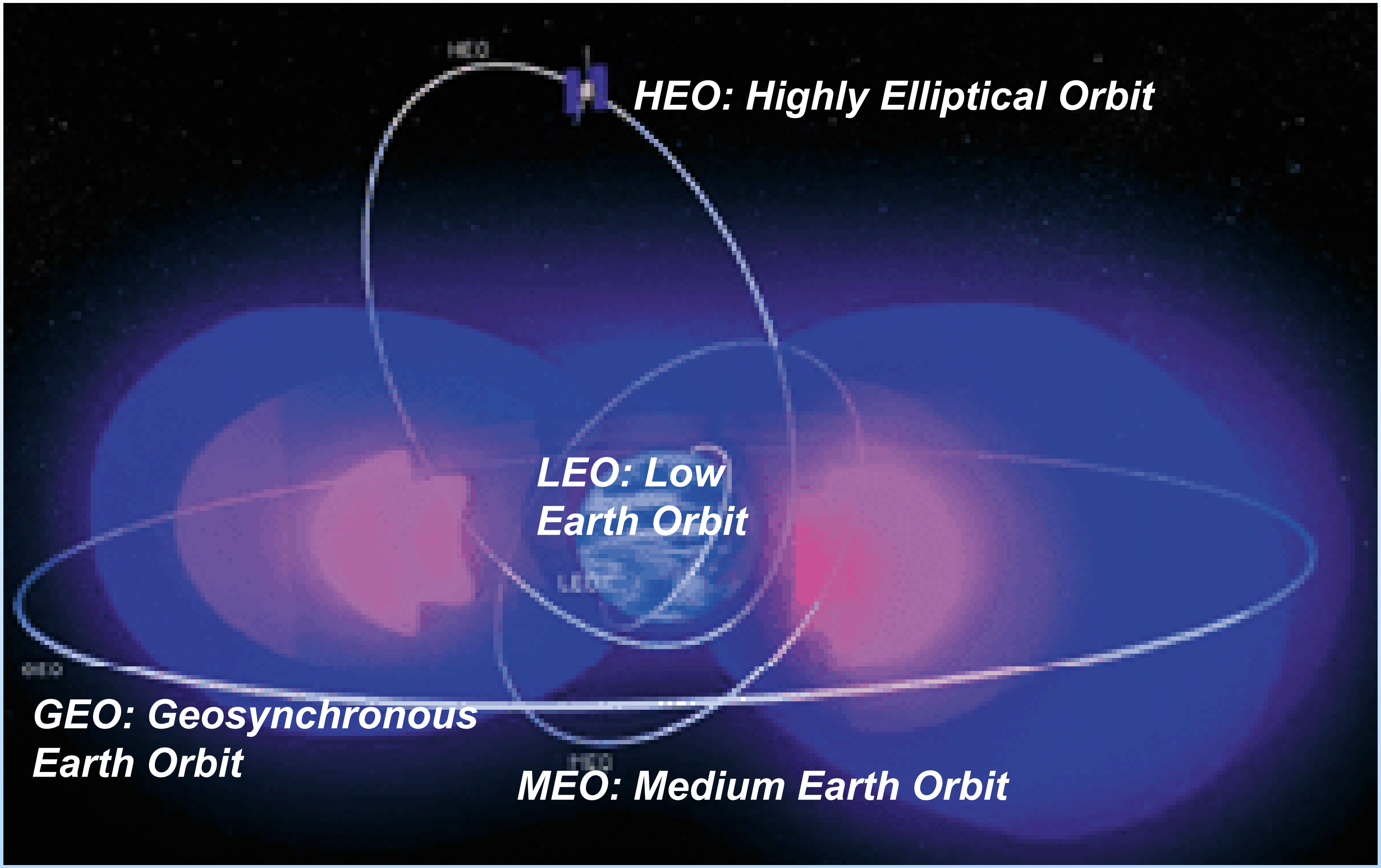
Agenda

- Section I: Single Event Effects (SEEs) in Digital Logic
- Section II: Application of the NASA Goddard Radiation Effects and Analysis Group (REAG) FPGA SEU Model
- Section III: Reducing System Error: Common Mitigation Techniques

Break

- Section IV: When Your Mitigation Fails
- Section V: Xilinx V4 and Mitigation
- Section VI: Fail-Safe Strategies

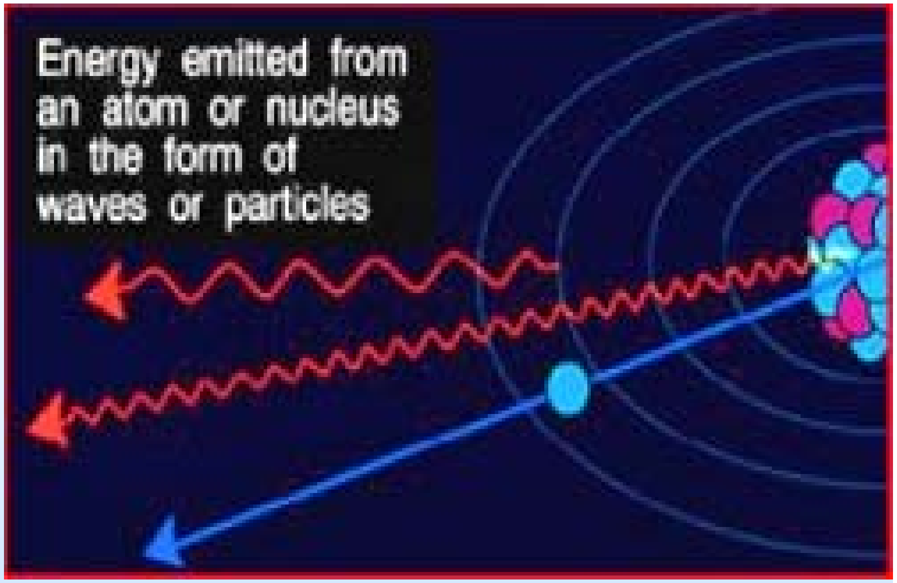
Van Allen Radiation Belts Have Varying Particle Spectra



Van Allen Radiation Belts: Illustrated by Aerospace Corp.

Source of Faults: SEEs and Ionizing Particles

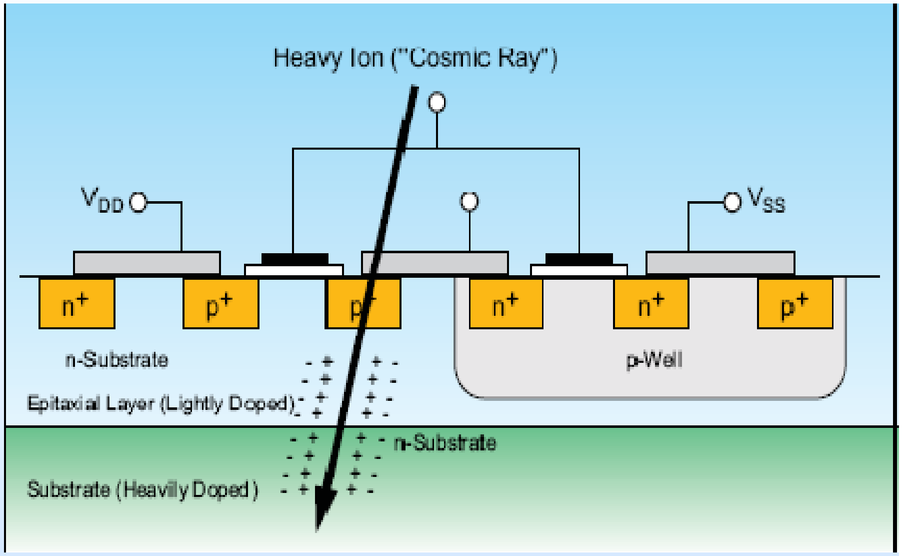
- Terrestrial devices are susceptible to faults mostly due to:
  - Alpha particles: from packaging and doping and
  - Neutrons: caused by Galactic Cosmic Ray (GCR) Interactions that enter into the earth's atmosphere.



- Devices expected to operate at higher altitude (Aerospace and Military) are more prone to upsets caused by:
  - Heavy ions: direct ionization
  - Protons: secondary effects

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

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

Units

Density of target material

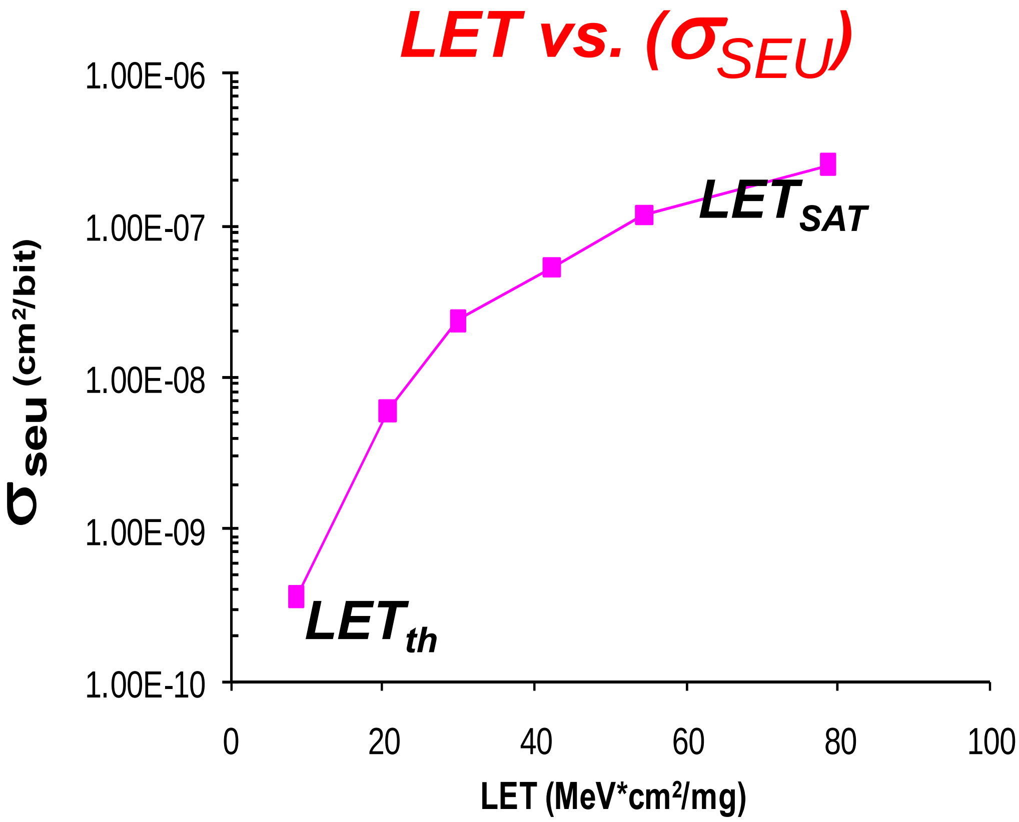
Terminology Used in Device Datasheets: LET vs. SEU

Error Cross Section (σ_{SEU})

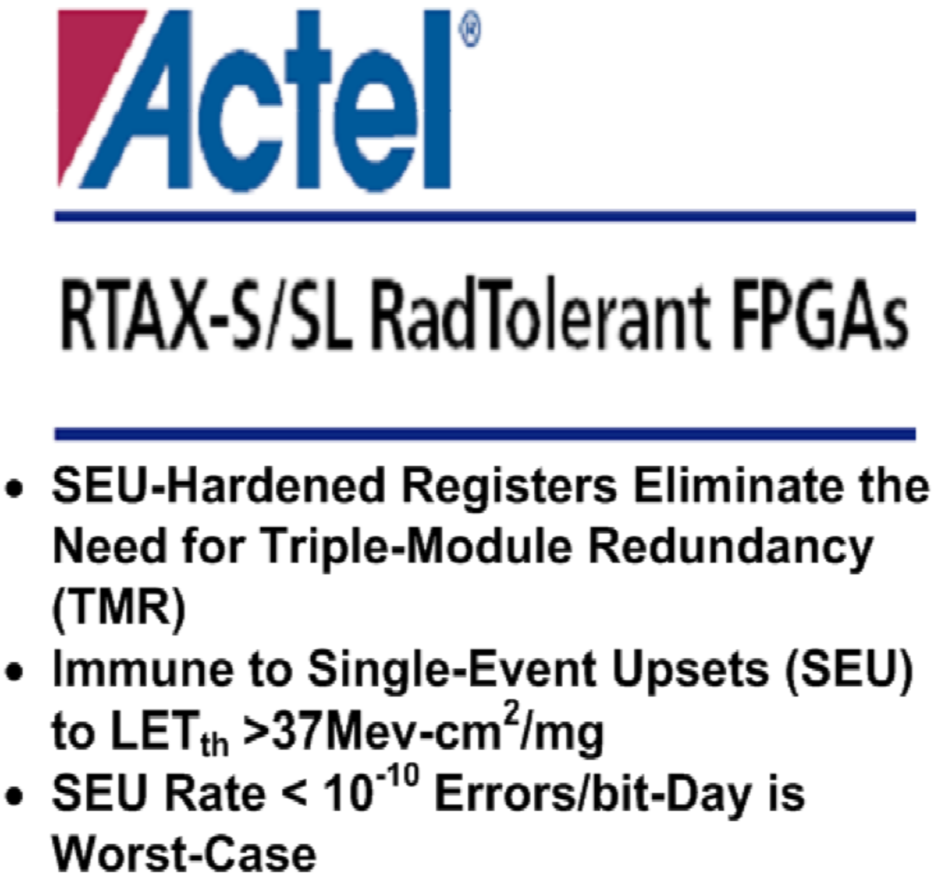
Terminology:

- **Flux: Particles/(sec-cm^2)**
- **Fluence: Particles/cm^2**
- **The σ_{SEU} is calculated at several LET values (particle spectrum)**
  - – **LET Threshold (LET$_{th}$) is the point where errors are first observed (on-set)**
  - – **LET Saturation (LET$_{SAT}$) is the point where errors stop statistically increasing with LET**

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



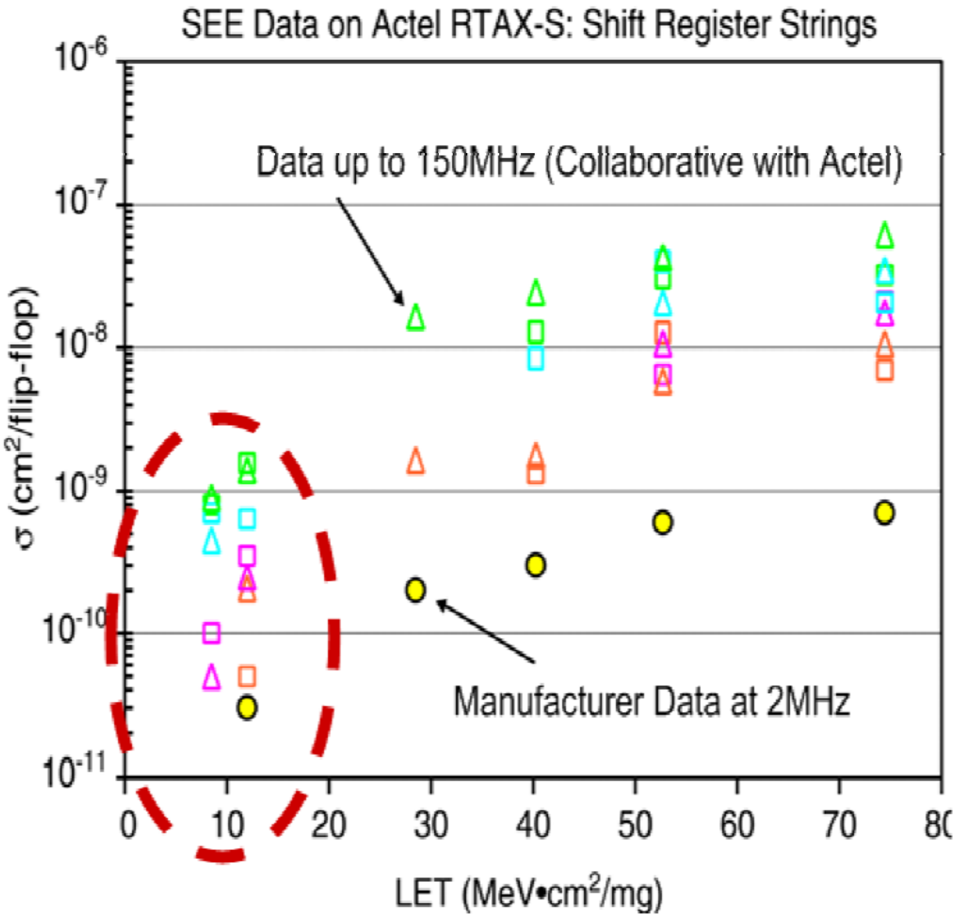
SEU Information: Manufacturer Datasheet Example



Actel®

RTAX-S/SL RadTolerant FPGAs

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



SEE Data on Actel RTAX-S: Shift Register Strings

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

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
- Single Event Functional Interrupt (SEFI) - upset disrupts function

FPGA SEU Susceptibility

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

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

To be presented by Melanie Berg at the Revolutionary Electronics in Space (ReSpace) / Military and Aerospace Programmable Logic Devices (MAPLD) 2011 Conference, Albuquerque, NM, August 22-25, 2011, and to be published on nepp.nasa.gov web site

Agenda

- Section I: SEEs in Digital Logic
- **Section II: Application of the NASA REAG FPGA SEU Model**
  - Configuration σ_{SEU} ($P_{configuration}$)
  - Functional Data Path σ_{SEU} ($P_{functionalLogic}$)
  - Microsemi (Actel) ProASIC3 Example
- Section III: Reducing System Error: Common Mitigation Techniques

Break

- Section IV: When Your Mitigation Fails
- Section V: Xilinx V4 and Mitigation
- Section VI: Fail-Safe Strategies

The NASA Goddard REAG FPGA SEU Model : Top Down Approach

Top Level Model has 3 major categories of σ_{SEU} :

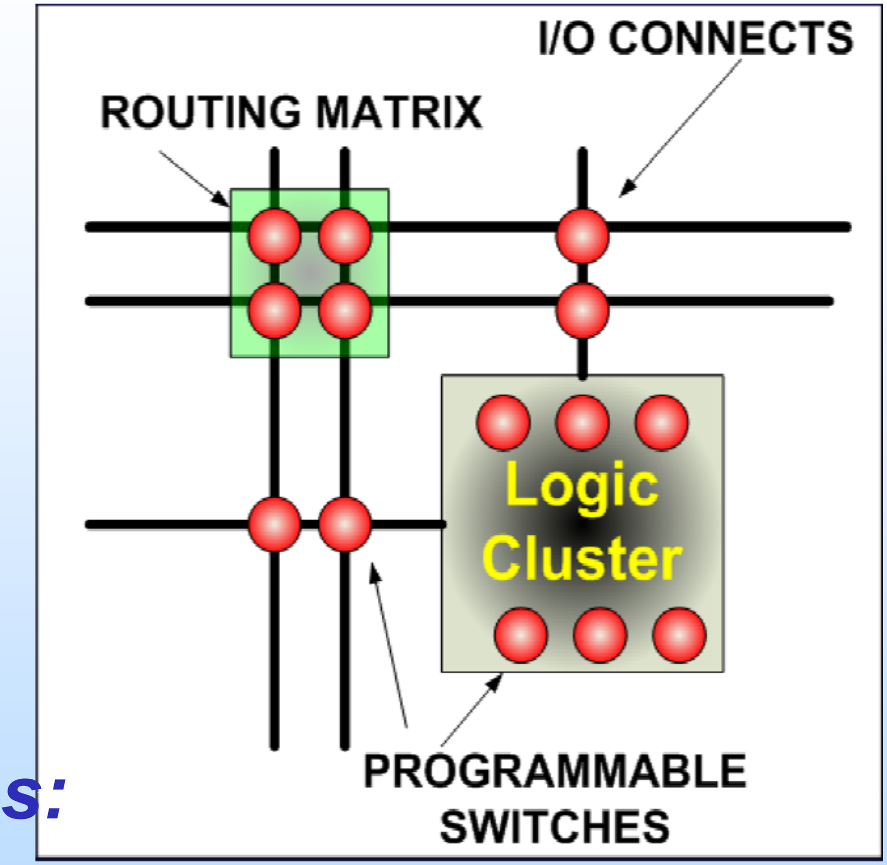
$$P\left(fs\right)_{error} \propto P_{Configurat\ ion} + P(\ fs)_{\ functional\ Logic} + P_{SEFI}$$

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

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

Configuration SEU Cross Sections

Place, Route, and Gate Utilization are Stored in the FPGA Configuration

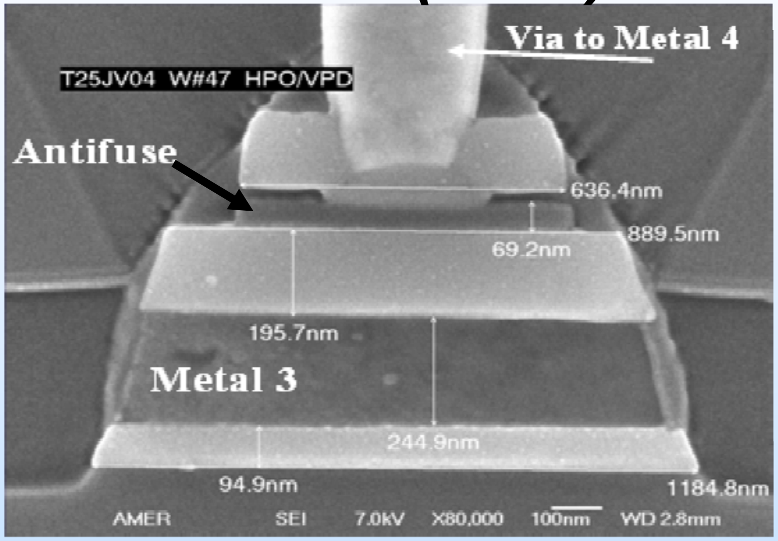


- ***Configuration Defines:***
  Arrangement of pre-existing logic via programmable switches
  - ***Functionality (logic cluster)***
  - ***Connectivity (routes)***
  - ***Placement***
- ***Programming Switch Types:***
  - ***Antifuse:* One time Programmable (OTP)**
  - ***SRAM:* Reprogrammable (RP)**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

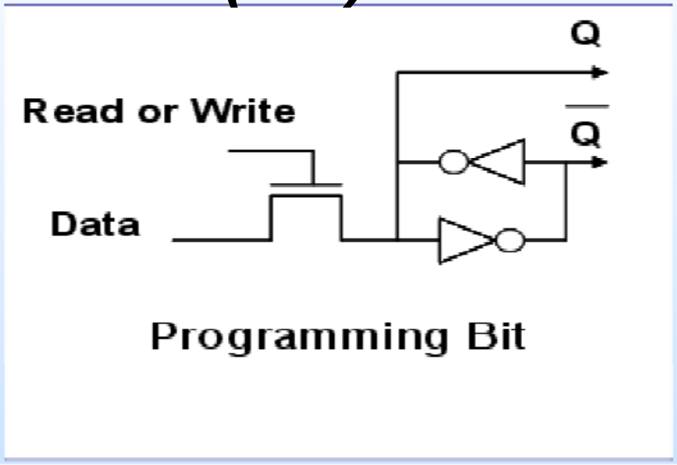
To be presented by Melanie Berg at the Revolutionary Electronics in Space (ReSpace) / Military and Aerospace Programmable Logic Devices (MAPLD) 2011 Conference, Albuquerque, NM, August 22-25, 2011, and to be published on nepp.nasa.gov web site

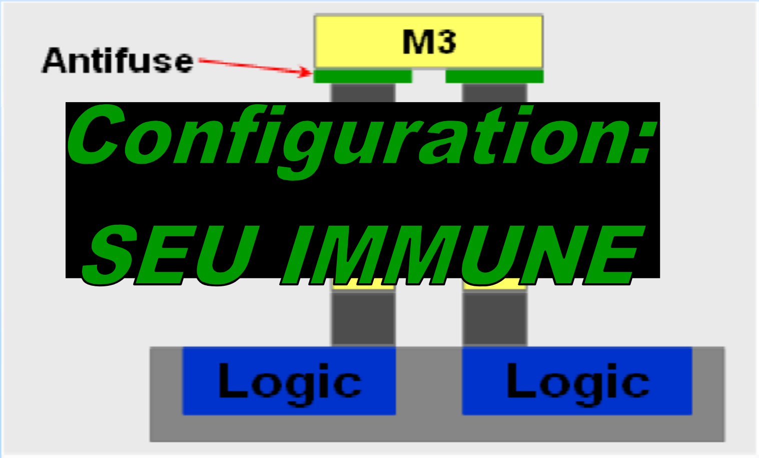
Programmable Switch Implementation and SEU Susceptibility

ANTIFUSE (OTP)

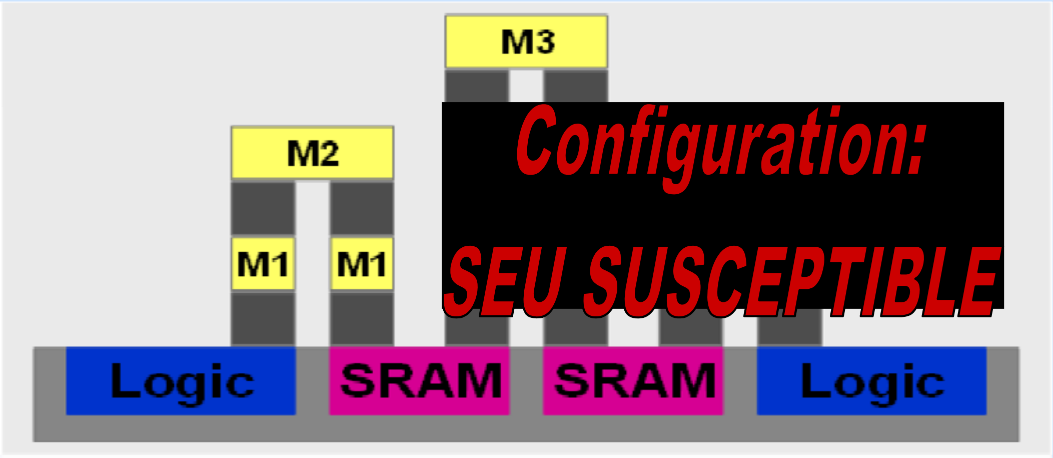


SRAM (RP)





Configuration: SEU IMMUNE



Configuration: SEU SUSCEPTIBLE

Configuration SEU Test Results and the REAG FPGA SEU Model

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

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

Functional Data Path SEU Cross Sections

Configuration versus Data Path (Functional Logic) SEUs

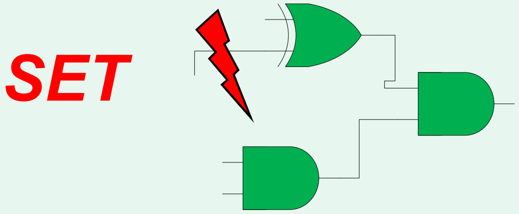
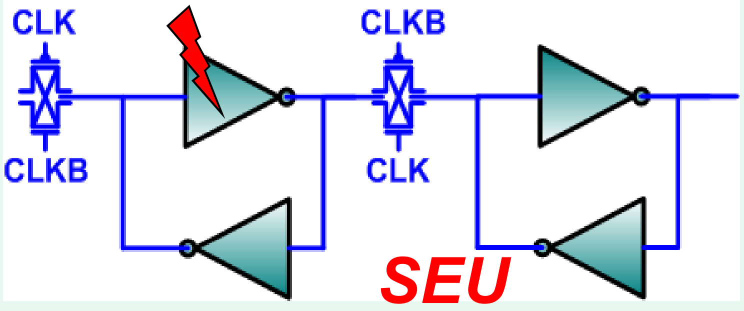
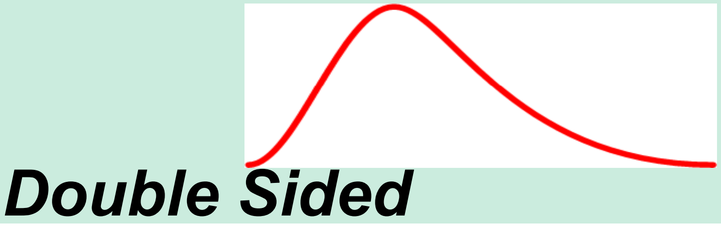
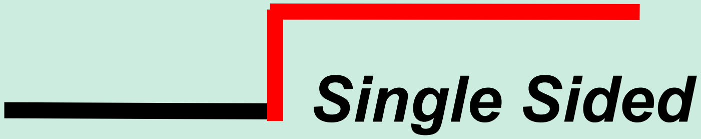
- Configuration and Functional data path circuitry are separate logic
- Can be implemented with different technologies within one device
- Configuration is static and data paths are not. Requires a different test and analysis approach

This explains why there are separate categories of error:

$$P_{configuration} \text{ vs. } P_{functionalLogic}$$

To be presented by Melanie Berg at the Revolutionary Electronics in Space (ReSpace) / Military and Aerospace Programmable Logic Devices (MAPLD) 2011 Conference, Albuquerque, NM, August 22-25, 2011, and to be published on nepp.nasa.gov web site

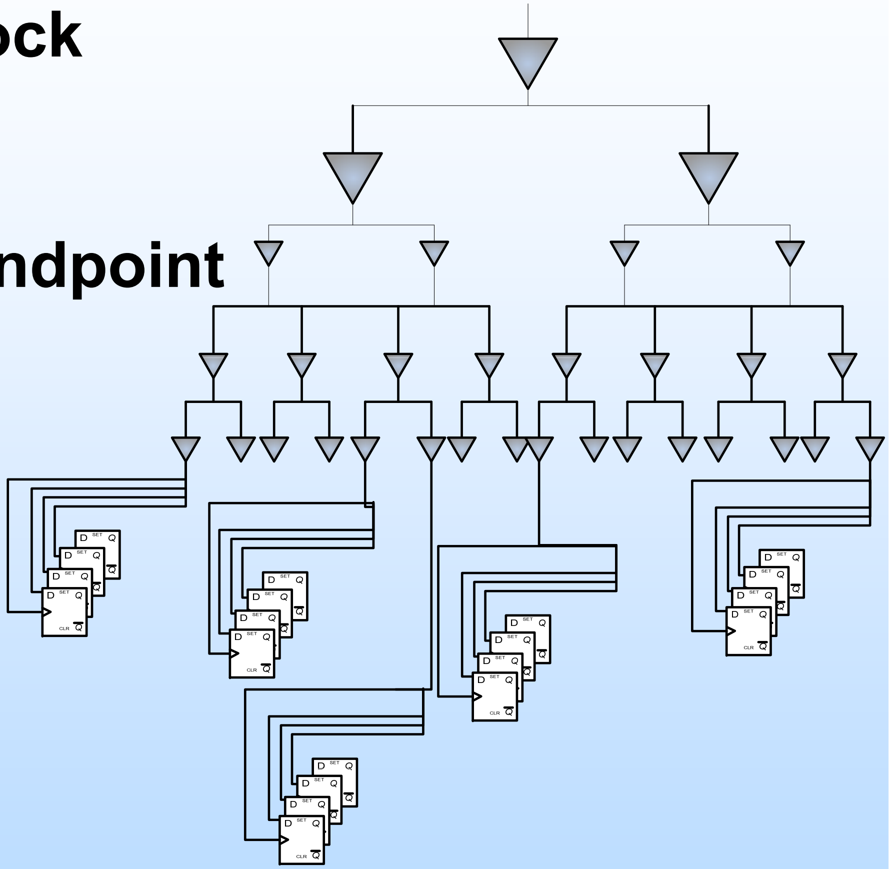
SEUs and SETs in a Data Path

Combinatorial	Sequential
Logic function generation (computation)	Captures and holds state of data input at rising edge of clock
	
SET: Glitch in the combinatorial logic: Capture is frequency dependent 	SEU: State changes until next cycle of enabled input: Next state capture can be frequency dependent 

DFF's in a Synchronous Design

- **All DFFs are connected to a clock**
- **Clock period: τ_{clk}**
- **Clock frequency: f_s**
- **Delay from Startpoint DFF to Endpoint DFF: τ_{dly}**

Clock Tree

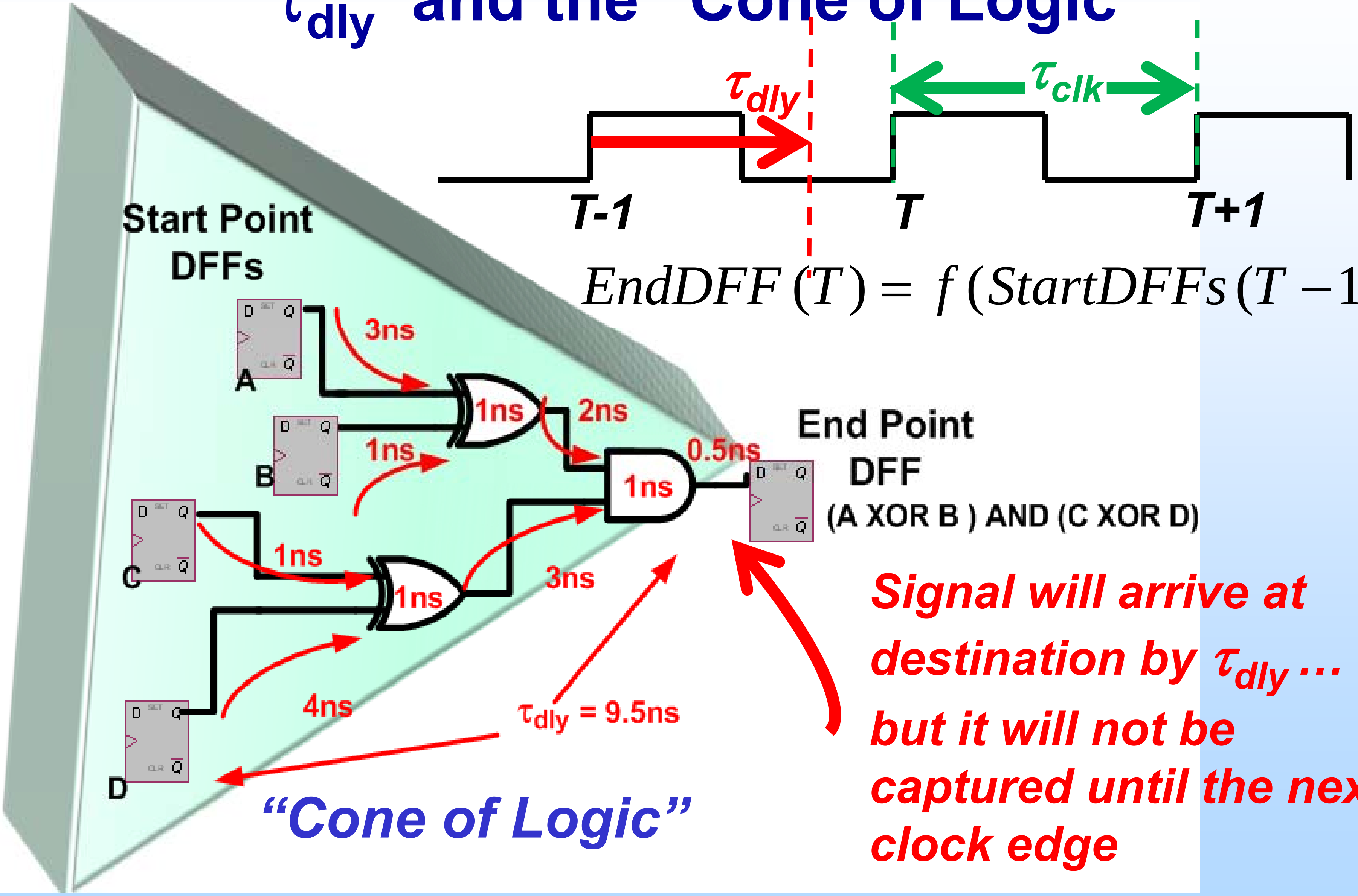


$$\tau_{clk} = \frac{1}{fs}$$

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

DFFs are BOUNDARY POINTs in a synchronous design

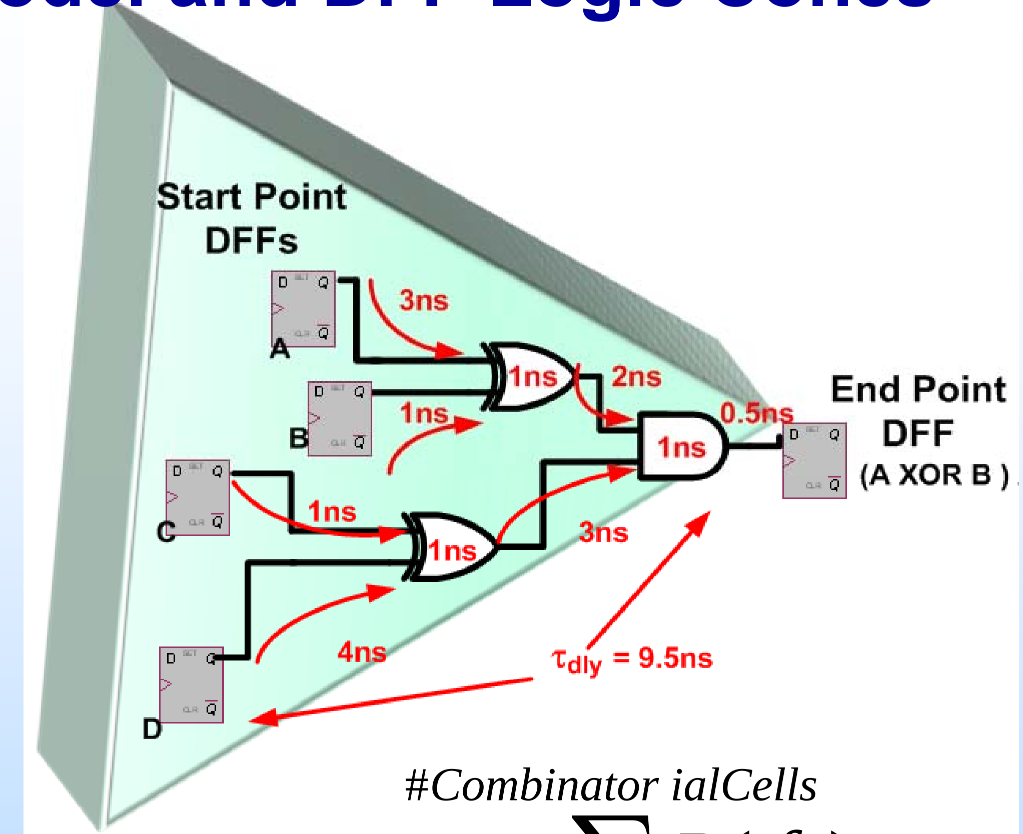
StartPoint DFFs → EndPoint DFFs
τ_{dly} and the "Cone of Logic"



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

Signal will arrive at destination by τ_{dly} … but it will not be captured until the next clock edge

Data Path Model and DFF Logic Cones



$$P(fs)_{functional\ \ Logic}$$

Evaluate for Each DFF

$$\exists_{DFF}\left(\sum_{j=1}^{\#StartPo\,\mathrm{int}\,DFFs} P(fs)_{DFFSEU\rightarrow SEU(j)} + \sum_{i=1}^{\#Combinator\,ialCells} P(fs)_{SET\rightarrow SEU(i)}\right)$$

StartPoint DFFs

Combinatorial logic

DFF$_k$ Cone of Logic

To be presented by Melanie Berg at the Revolutionary Electronics in Space (ReSpace) / Military and Aerospace Programmable Logic Devices (MAPLD) 2011 Conference, Albuquerque, NM, August 22-25, 2011, and to be published on nepp.nasa.gov web site

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

Functional Data Path SEU Cross Sections and Combinatorial Logic Effects (Capturing SETs)

$$P(fs)_{functional\,Logic} \propto P(fs)_{DFFSEU \rightarrow SEU} + P(fs)_{SET \rightarrow SEU}$$

SETs and a Synchronous System

- Generation (P_{gen})
- Propagation (P_{prop})
- Logic Masking ($P_{logic.}$)
- Capture

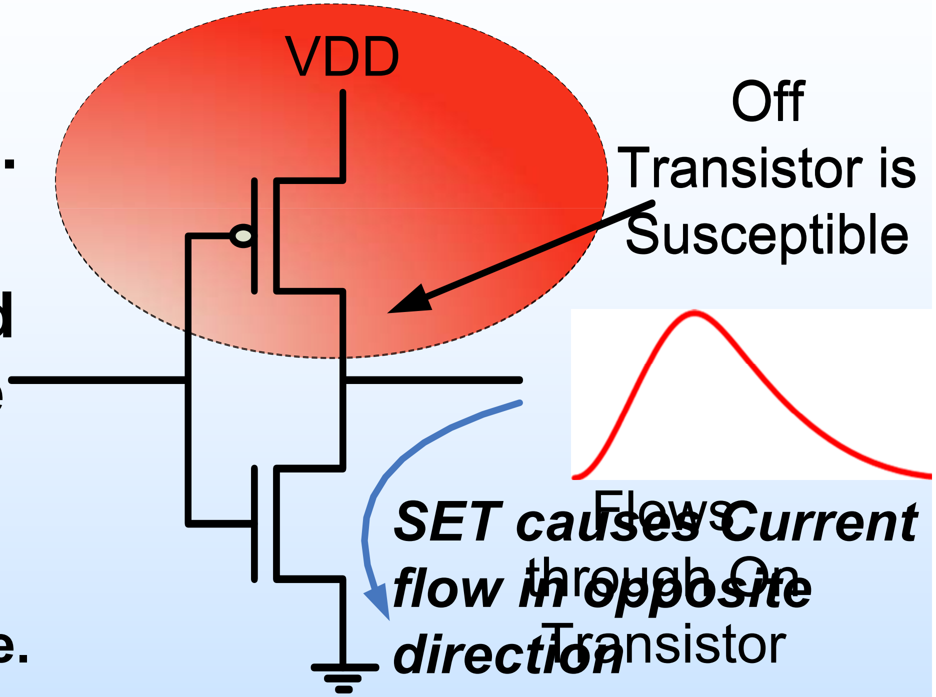
All Components comprise:

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

To be presented by Melanie Berg at the Revolutionary Electronics in Space (ReSpace) / Military and Aerospace Programmable Logic Devices (MAPLD) 2011 Conference, Albuquerque, NM, August 22-25, 2011, and to be published on nepp.nasa.gov web site

SET Generation: P_{gen}

- SET generation occurs due to an “off” gate turning “on”.
- CMOS SET: there is a push-pull between the on gate and the off gate collected charge
- SET has an amplitude and width (τ_{width}) based on:
  - Amount of collected charge (i.e. small LET →small SET)
  - The strength of the gate’s load
  - The strength of its complimentary “ON” gate
  - The dissipation strength of the process.



Off Transistor is Susceptible

SET causes Current flow in opposite direction

Flows through On Transistor

Collected Charge ***Critical Charge***

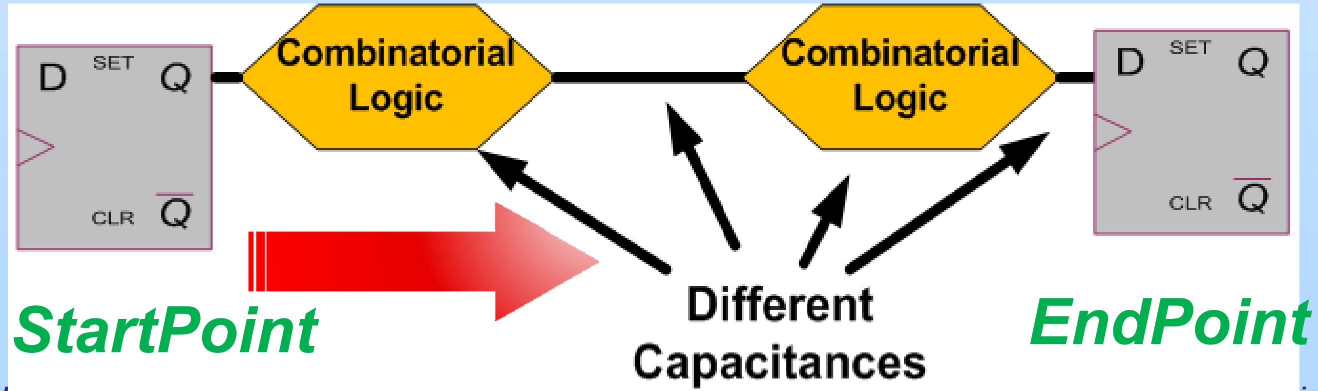
$$Q_{coll} > Qcrit$$

$$Q_{crit} = C_{node} * V_{node}$$

Node Capacitance ***Node Voltage***

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

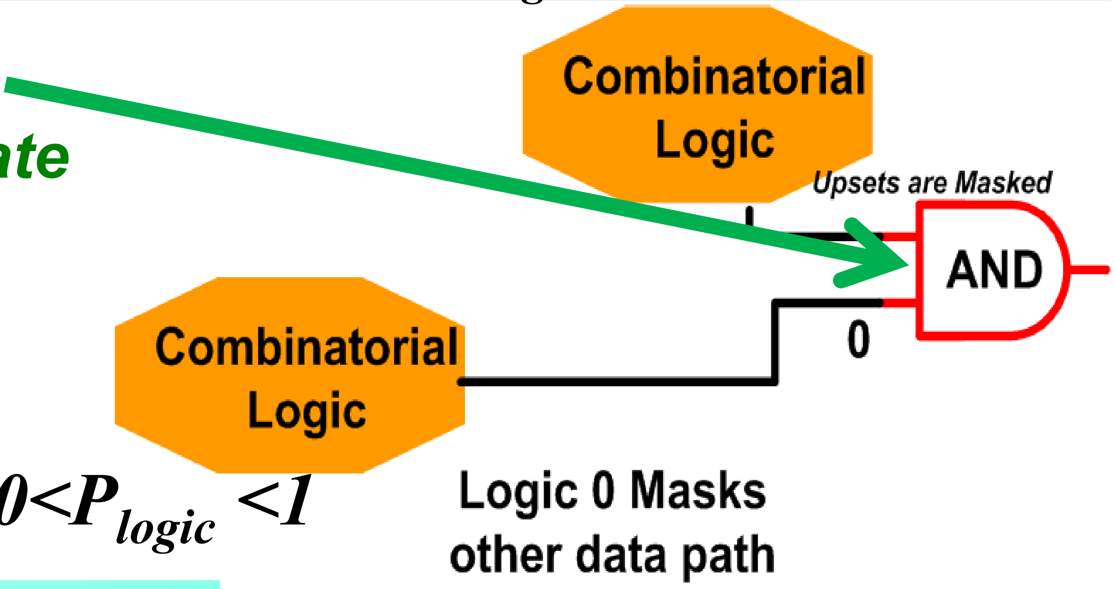


SET Logic Masking: P_{logic}

- 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

$0 < P_{logic} < 1$



Combinatorial Logic

Upsets are Masked

AND

0

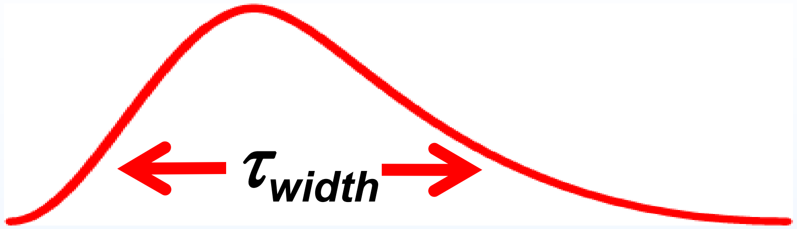
Combinatorial Logic

$0 < P_{logic} < 1$

Logic 0 Masks other data path

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.

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

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

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

To be presented by Melanie Berg at the Revolutionary Electronics in Space (ReSpace) / Military and Aerospace Programmable Logic Devices (MAPLD) 2011 Conference, Albuquerque, NM, August 22-25, 2011, and to be published on nepp.nasa.gov web site

Data Path Model and Combinatorial Logic SETs

$$\sum_{i=1}^{\#Combinator\,ialCells} P(fs)_{SET \rightarrow SEU\,(i)} \propto \sum_{i=1}^{\#CombinatorialCells} P_{gen(i)} P_{prop(i)} P_{\log ic(i)} \tau_{width(i)}\, fs$$

$$< \sum_{i=1}^{\#CombinatorialCells} P_{gen(i)} P_{prop(i)} \tau_{width(i)}\, fs$$

Upper Bound SET P_{logic}=1

Transient portion of σ_{SEU} suggests that operational frequency and number of combinatorial logic gates are directly proportional to σ_{SEU}

Have you always believed that if you decrease operational frequency, the σ_{SEU} will also decrease?

Or

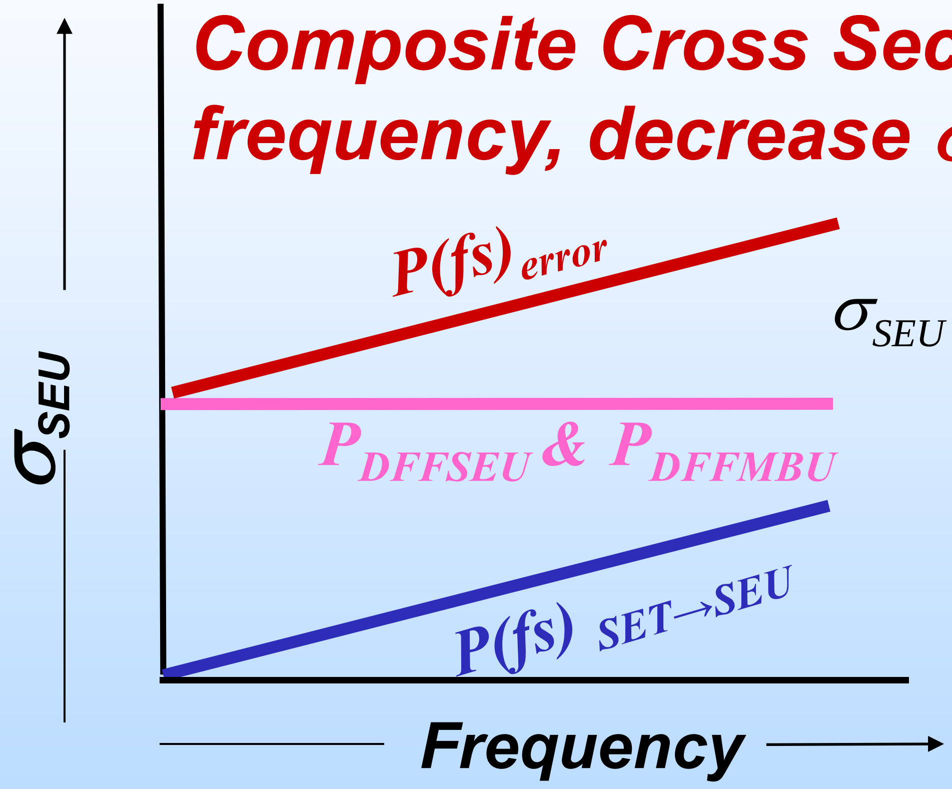
If you increase the amount of combinatorial logic, you will increase the σ_{SEU}

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

Functional Data Path SEU Cross Sections and DFF Effects (Capturing StartPoint SEUs)

$$P(fs)_{functional\,Logic} \propto \boxed{P(fs)_{DFFSEU \rightarrow SEU}} + P(fs)_{SET \rightarrow SEU}$$

Conventional Theory: System Upsets Have a Static Component+Dynamic Component



$$\sigma_{SEU} \propto P(fs)_{error} \propto P_{DFFSEU} + P(fs)_{SET\rightarrow SEU}$$

Takes into account upsets from combinatorial logic in DFF data path and the DFF potential for flipping its state

Does not fully characterize DFF upsets as they pertain to a synchronous system

StartPoint SEUs and a Synchronous System: New Stuff

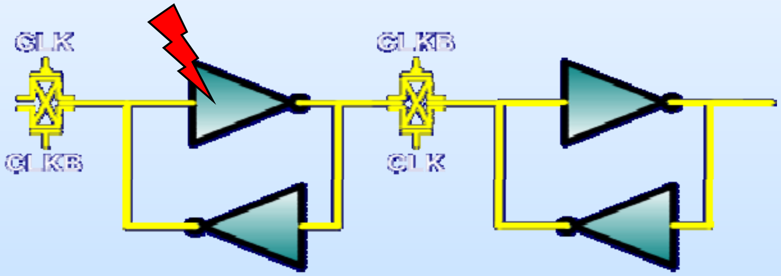
- Generation (P_{DFFSEU})
- P_{prop}=1 for hard state switch
- Logic Masking ($P_{logic.}$)
- Capture

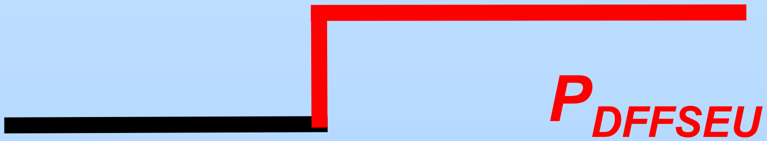
All Components comprise:

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

Generation of DFF Upsets: P_{DFFSEU}

- Probability that a DFF will flip its state
- Can be a hard flip:
  - Will not change until the next clock cycle
  - Amplitude and width are not affected as with a SET
- Can be a metastable flip
  - No real defined state
  - Otherwise known as a "weak" state
  - Can cause oscillations in the data path
  - Eventually settles to a state … not deterministic!





Generation P_{DFFSEU} versus Capture $P(fs)_{DFFSEU\rightarrow SEU}$

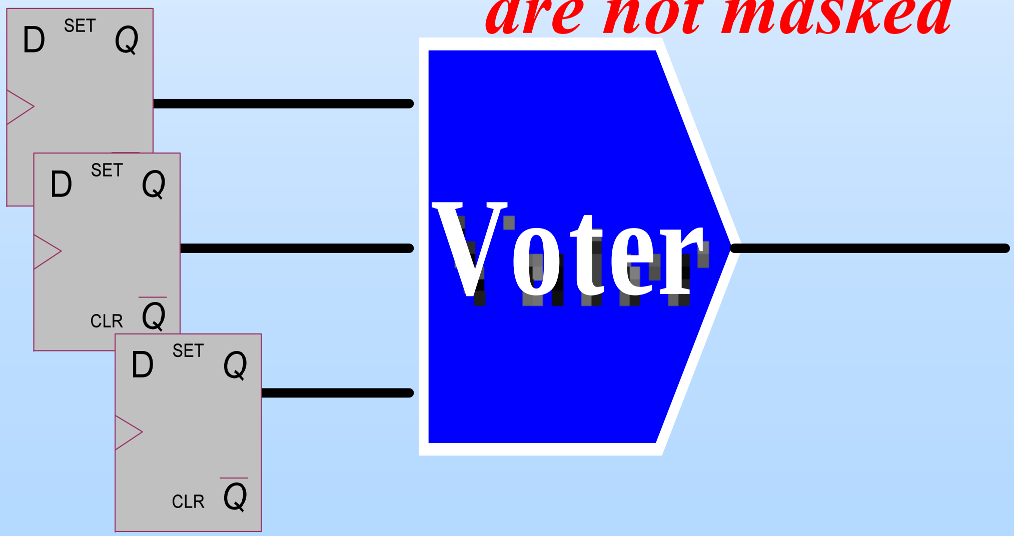
P_{DFFSEU}	$P(fs)_{DFFSEU\rightarrow SEU}$
Probability a StartPoint DFF becomes upset	Probability that the StartPoint upset is captured by the endpoint DFF
Occurs at some point in time within a clock period	Occurs at a clock edge (capture)
Not frequency dependent	Frequency dependent

Logic Masking DFFs… P_{logic}

- Logic masking for DFF StartPoints is similar to logic masking of combinatorial logic.
- DFF logic masking is generally the point where Triple Modular Redundancy (TMR) is inserted

P_{logic}>0

for Voter… its upsets are not masked



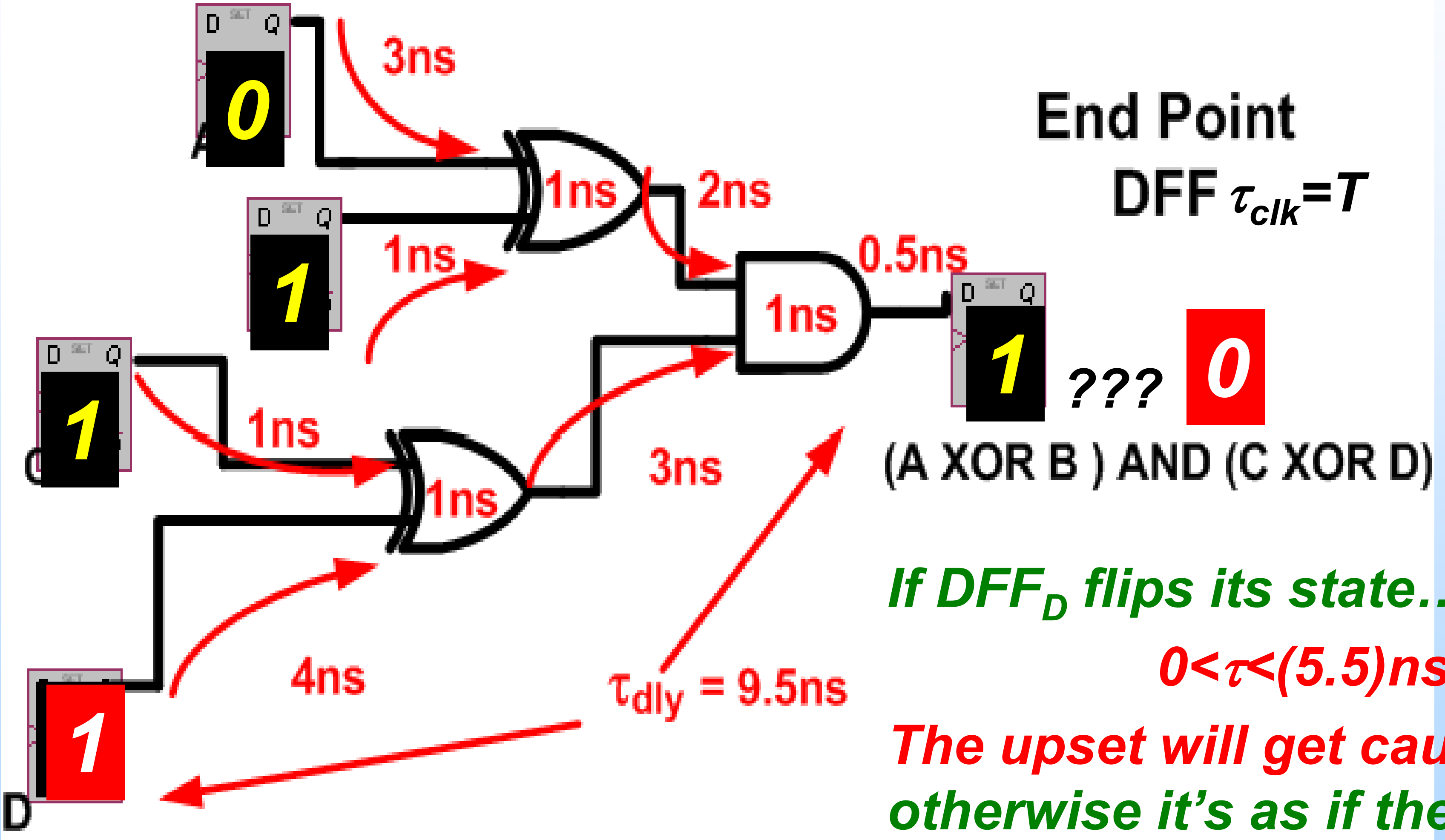
P_{logic}=0

for DFFs… their upsets are masked

StartPoint SEU Capture Example: Assume τ_{clk}=15ns

Start Point DFFs τ_{clk}=T-1

End Point DFF τ_{clk}=T



If DFF$_D$ flips its state…

0<τ<(5.5)ns

The upset will get caught… otherwise it's as if the event never occurred

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

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

upset capture with respect to to frequency

Data Path Upsets and StartPoint DFFs

$$\sum_{j=1}^{\#StartPo\operatorname{int} DFFs} P(fs)_{DFFSEU \rightarrow SEU(j)} \propto \sum_{j=1}^{\#StartPo\operatorname{int} DFFs} P_{DFFSEU(j)} P_{\mathrm{logic}(j)} (1-\tau_{dly(j)}\, fs)$$

$$< \sum_{j=1}^{\#StartPo\operatorname{int} DFFs} P_{DFFSEU(j)} (1-\tau_{dly(j)}\, fs)$$

$$< \sum_{j=1}^{\#StartPo\operatorname{int} DFFs} P_{DFFSEU(j)}$$

$P(fs)_{FunctionalLogic}$

Putting it all together:

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

Data Path Upsets

StartPoint DFF SEU capture

Combinatorial Logic SET capture

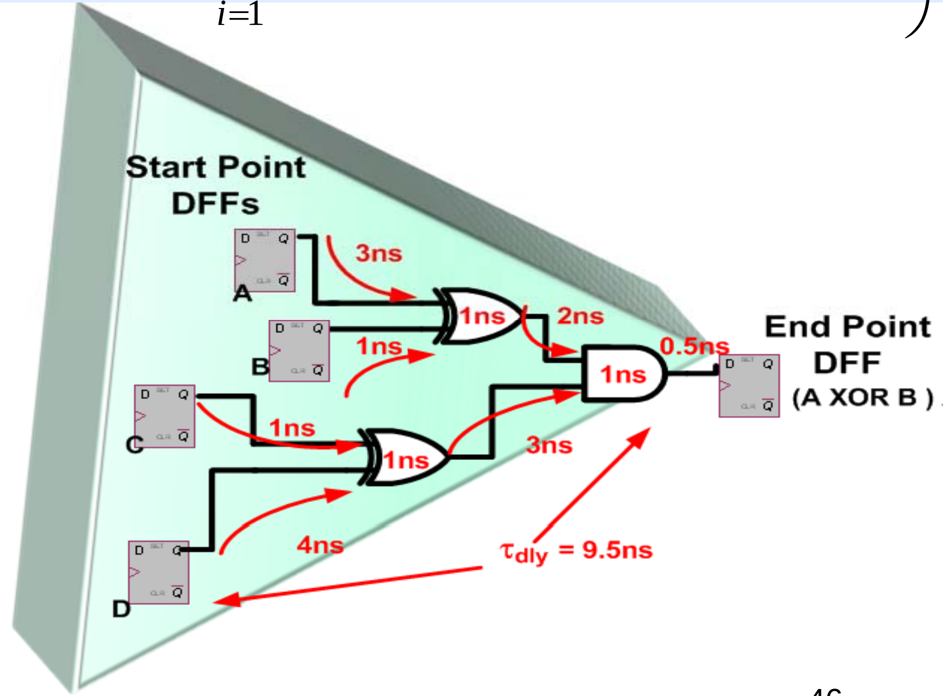
To be presented by Melanie Berg at the Revolutionary Electronics in Space (ReSpace) / Military and Aerospace Programmable Logic Devices (MAPLD) 2011 Conference, Albuquerque, NM, August 22-25, 2011, and to be published on nepp.nasa.gov web site

NASA REAG FPGA Data Path Functional Logic Susceptibility Model

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

$$\exists_{DFF}\left(P(fs)_{DFFSEU\rightarrow SEU} + P(fs)_{SET\rightarrow SEU}\right)$$

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



NASA REAG FPGA Upper Bound Susceptibility Model

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

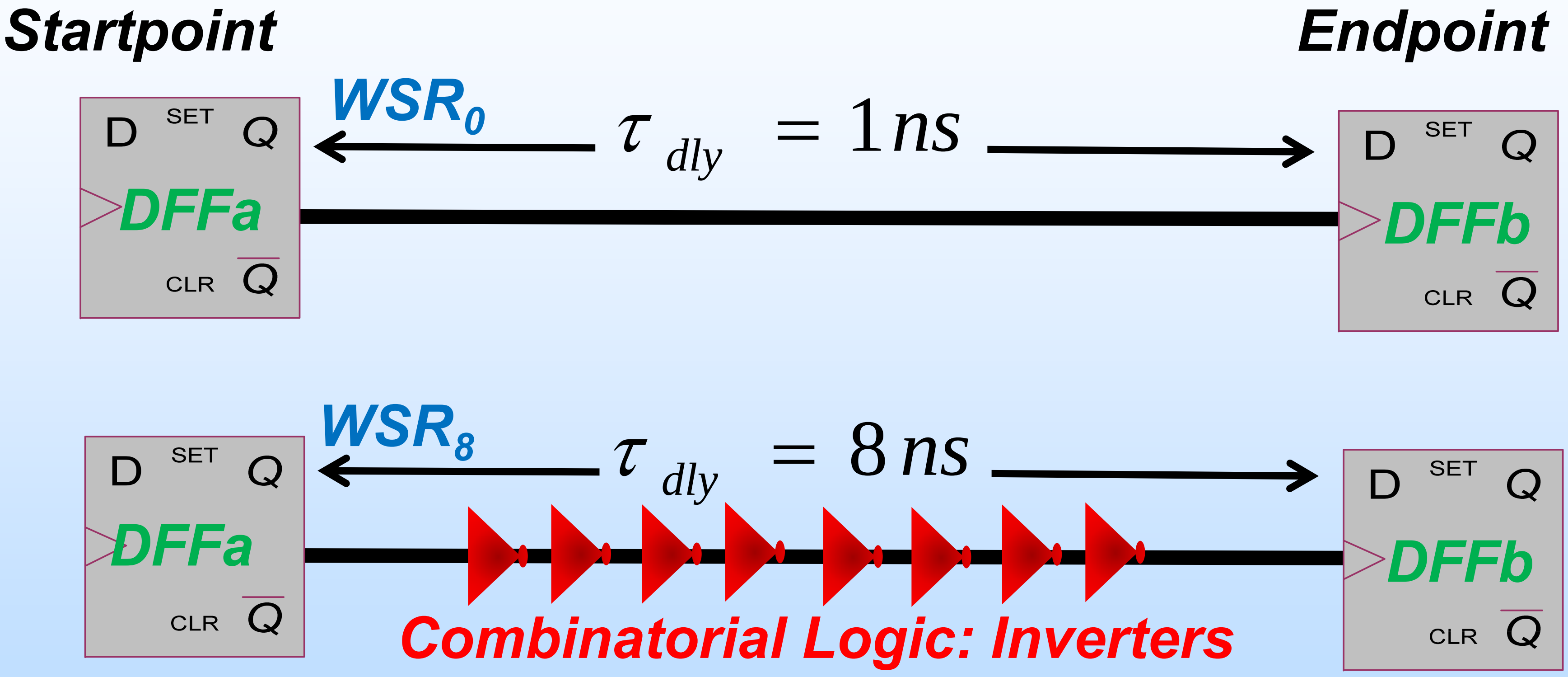
Upper-bound assumes P_{logic}=1 (no mitigation) and NO DFF frequency (fs) dependency

$$\exists_{DFF}\left(\sum_{j=1}^{\#StartPoint\,DFFs} P_{DFFSEU} + \sum_{i=1}^{\#Combinatorial\,Cells} (P_{gen(i)}P_{prop(i)}\tau_{width(i)}\, fs) \right)$$

How DFF or Combinatorial Logic Susceptibility Dominance Affects σ_{SEU}

	$P(fs)_{DFFSEU \rightarrow SEU}$	$P(fs)_{SET \rightarrow SEU}$
Logic	**DFF Capture**	**Combinatorial SET Capture**
Capture percentage of clock period	$(1 - \frac{\tau_{dly}}{\tau_{clk}}) = (1 - \tau_{dly} fs)$	$\frac{\tau_{width}}{\tau_{clk}} = \tau_{width} \quad fs$
Frequency Dependency	**Increase Frequency decrease σ_{SEU}**	**Increase Frequency Increase σ_{SEU}**
Combinatorial Logic Effects	**Increase Combinatorial logic increases τ_{dly} and decreases σ_{SEU}**	**Increase in combinatorial logic increases P_{gen} and increases σ_{SEU}**

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



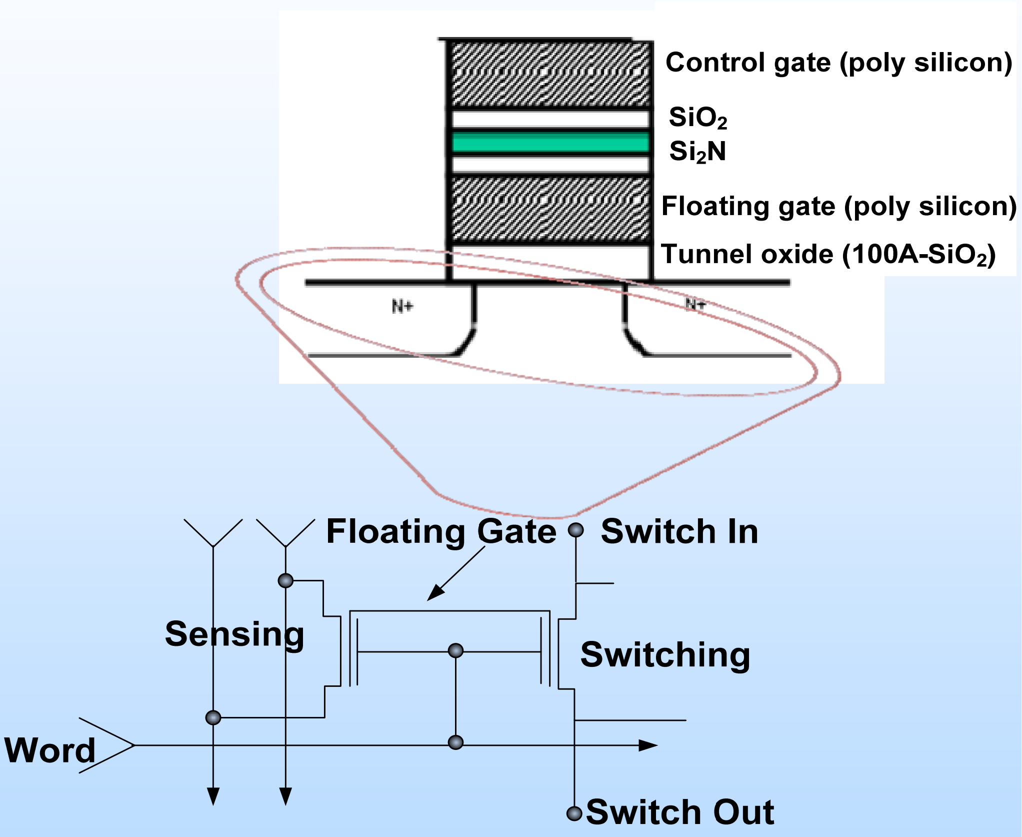
You can't answer the question until you understand the relative σ_{SEU} contribution of DFFs to Combinatorial Logic…

Is there Logic Mitigation?

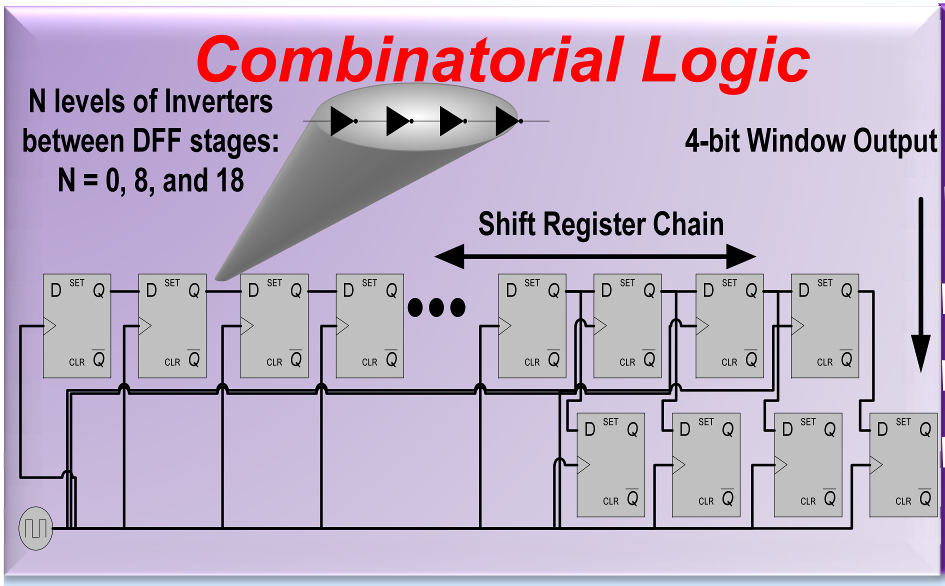
NASA REAG Models + Heavy Ion Data: Microsemi (Actel) ProASIC3

Background: 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.


Control gate (poly silicon)
SiO_2
Si_2N
Floating gate (poly silicon)
Tunnel oxide (100A-SiO_2)
N+
N+
Floating Gate
Switch In
Sensing
Switching
Word
Switch Out

ProASIC3 Analysis: Combinatorial Logic Contributions to σ_{SEU} using Shift Registers

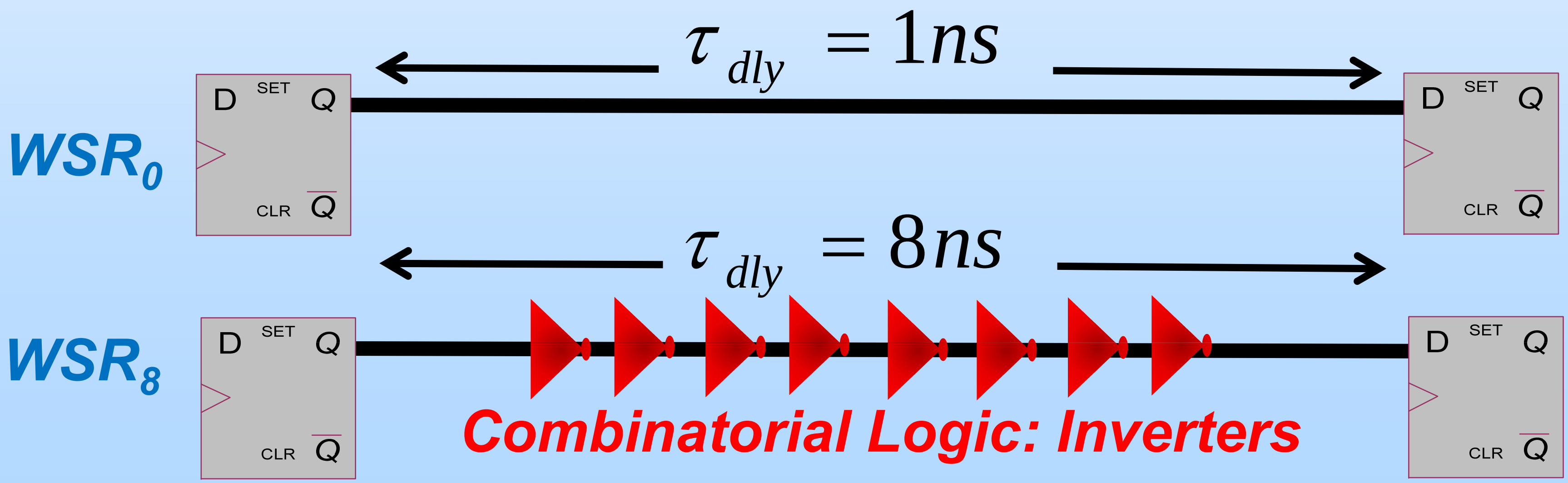


Windowed Shift Register (WSR)

WSR_0: N=0 Chain … Only DFFs

WSR_8: N=8 Chain… 8 Inverters per 1 DFF

WSR_{16}: N=16 Chain… 16 Inverters per 1 DFF



Microsemi (Actel) ProASIC3 Shift Register Study τ_{dly} and Adding Combinatorial Logic

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

Insignificant σ_{SEU}

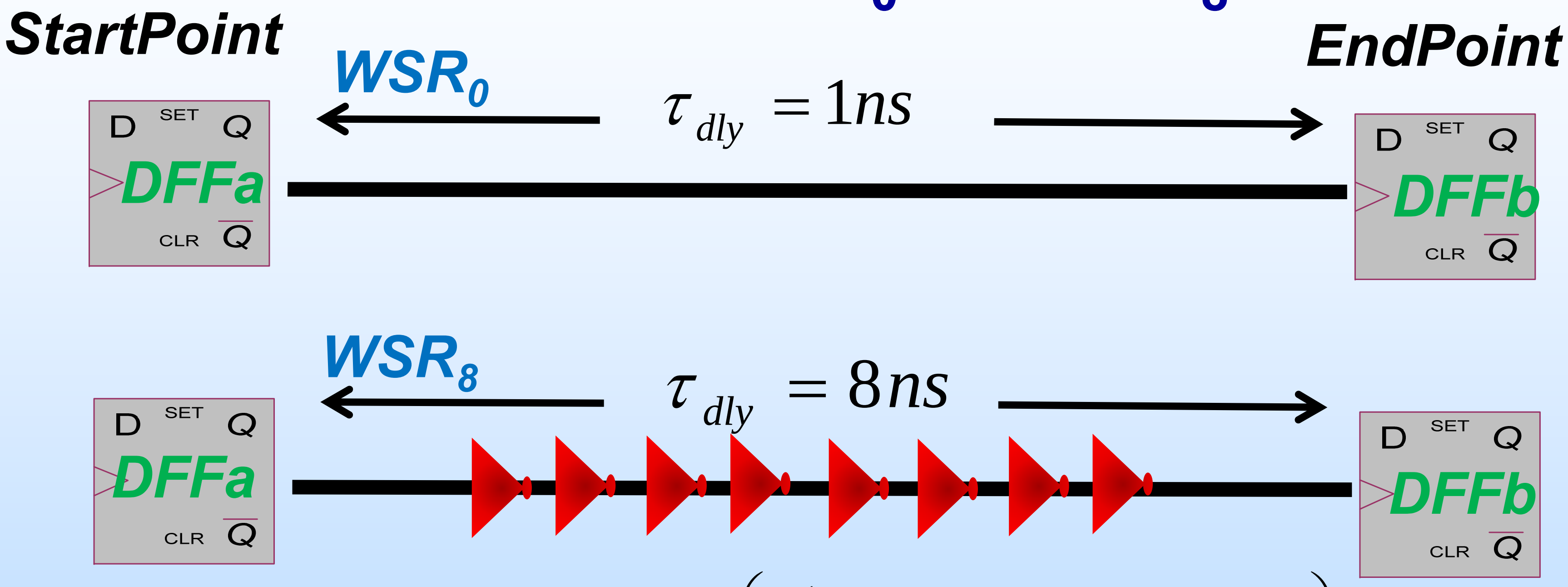
$$P(fs)_{functional\ Logic} \propto \exists_{DFF}\left(\sum_{j=1}^{1} P(fs)_{DFFSEU \rightarrow SEU(j)} + \sum_{i=0}^{\#Inverters} \cancel{P(fs)_{SET \rightarrow SEU(i)}}\right)$$

If the DFFs are not mitigated they will have the dominant σ_{SEU}

$$P(fs)_{functional\,Logic} \propto \exists_{DFF}\left(\left(\sum_{j=1}^{1} P_{DFFSEU(j)}(1-\tau_{dly(j)}\, fs)\right) \right)$$

Only One StartPoint per EndPoint DFF and P_{logic}=1

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



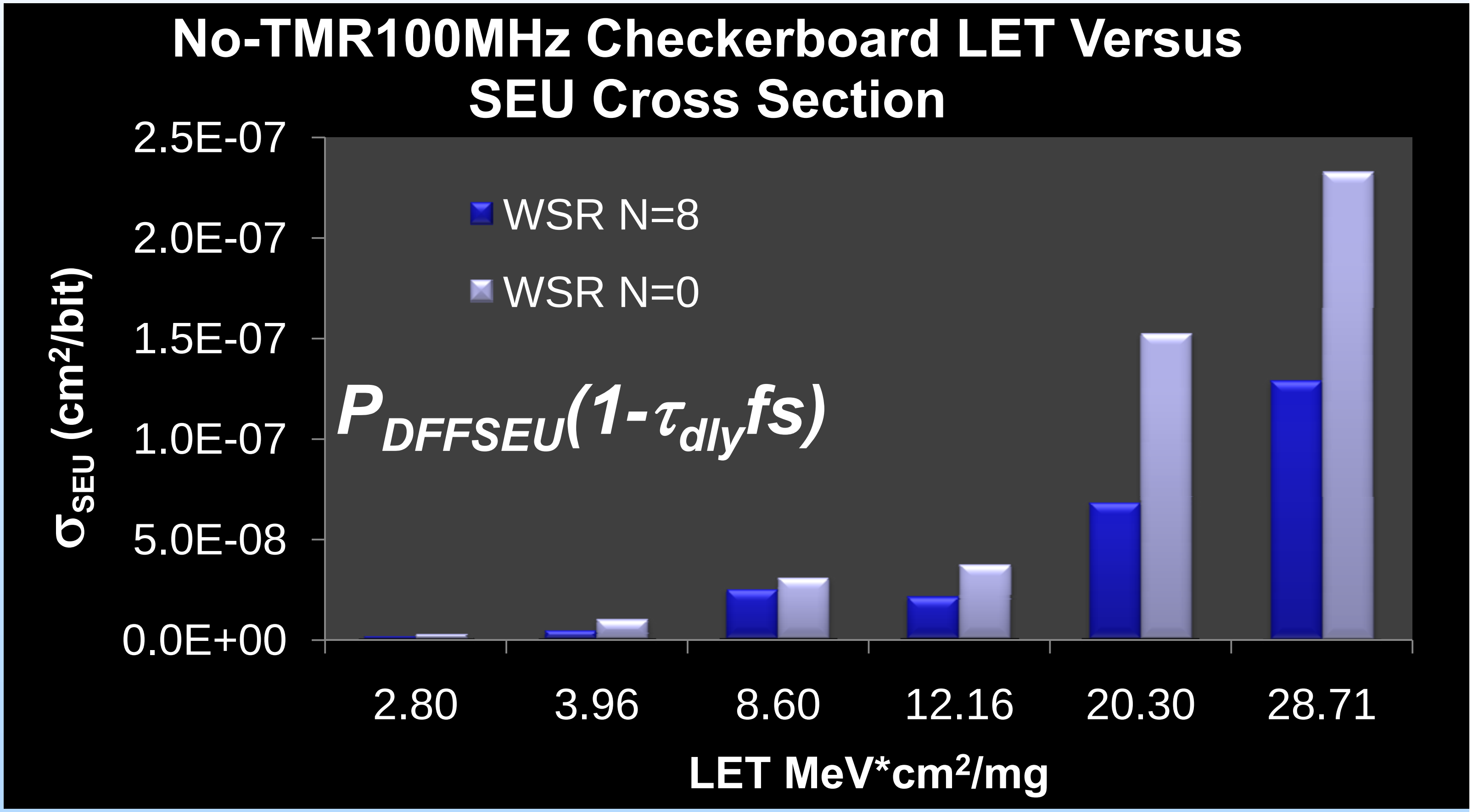
$$P(fs)_{functional\,Logic} \propto \underset{DFF}{\exists}\left(\left(\sum_{j=1}^{1} P_{DFFSEU(j)}(1-\tau_{dly(j)}\, fs)\right) \right)$$

No-TMR: σ_{SEU} is inversely proportional to τ_{dly}

σ_{SEU} WSR_0 > σ_{SEU} WSR_8

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 combinatorial logic does not increase σ_{SEU} because increase in τ_{dly}



To be presented by Melanie Berg at the Revolutionary Electronics in Space (ReSpace) / Military and Aerospace Programmable Logic Devices (MAPLD) 2011 Conference, Albuquerque, NM, August 22-25, 2011, and to be published on nepp.nasa.gov web site

Agenda

- Section I: Single Event Effects (SEEs) in Digital Logic
- Section II: Application of the NASA Goddard Radiation Effects and Analysis Group (REAG) FPGA SEU Model
- **Section III: Reducing System Error: Common Mitigation Techniques**
  - **Triple Modular Redundancy (TMR)**
  - Embedded Radiation Hardened by Design (RHBD)

Break

- Section IV: When Your Mitigation Fails
- Section V: Xilinx V4 and Mitigation
- Section VI: Fail-Safe Strategies

Example: TMR Mitigation Schemes will use Majority Voting

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

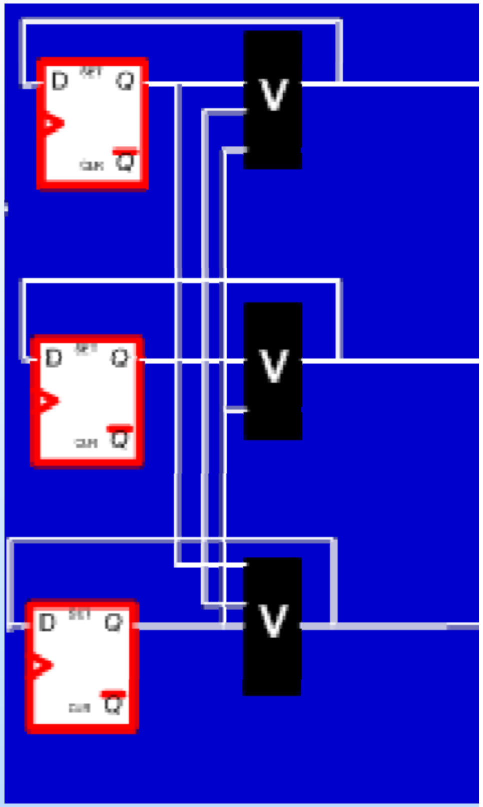
I0	I1	I2	Majority Voter
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

Best 2 out of 3

Triplicate and Vote

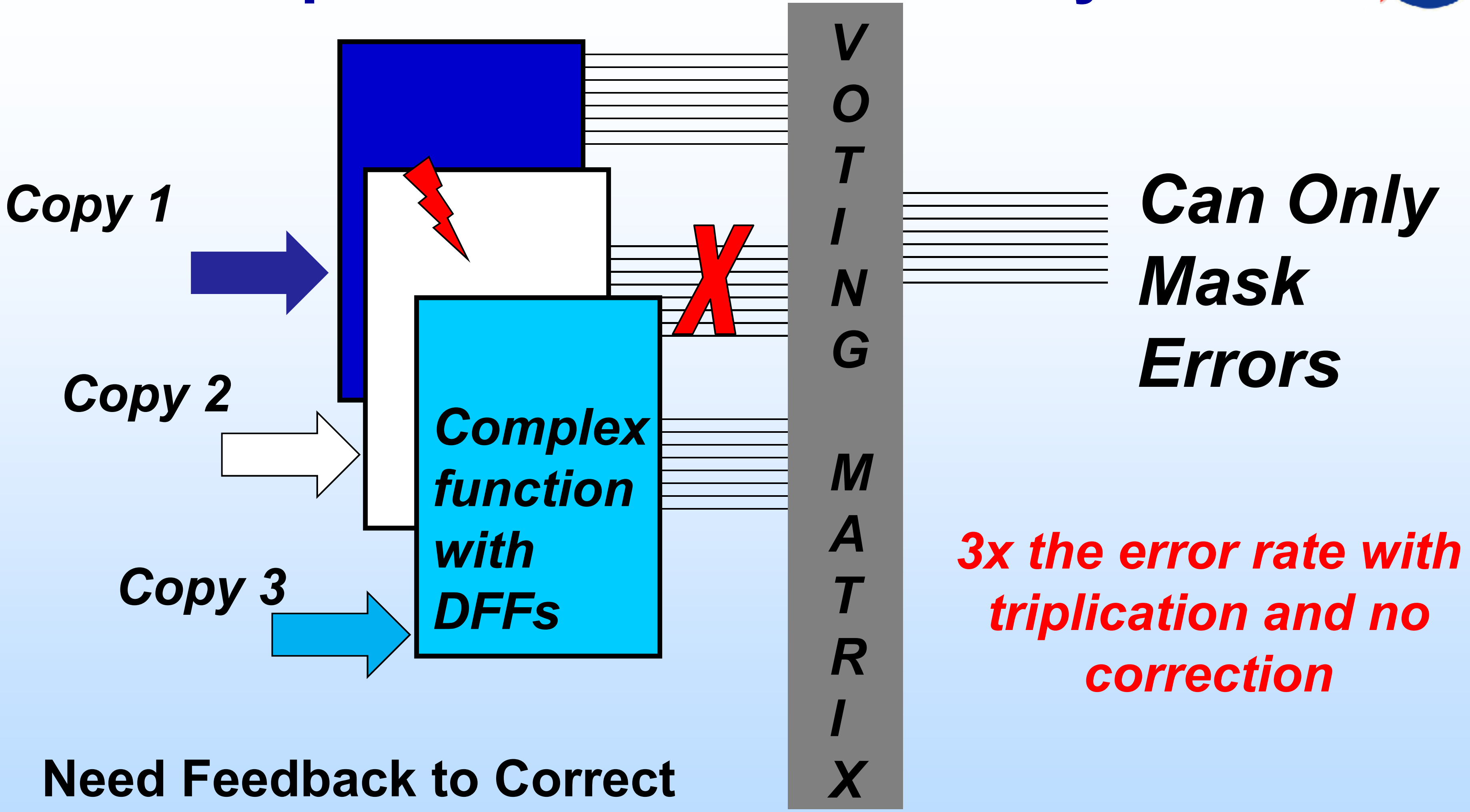
TMR: Correction vs Masking

- TMR with feedback will mask and correct an error
- TMR with no feedback will only mask an error
  - May not buy you anything if a large amount of circuitry has no correction capability
  - Triple the circuitry without correction:
    - triples the upset rate
    - may end up with the same upset rate using this scheme



To be presented by Melanie Berg at the Revolutionary Electronics in Space (ReSpace) / Military and Aerospace Programmable Logic Devices (MAPLD) 2011 Conference, Albuquerque, NM, August 22-25, 2011, and to be published on nepp.nasa.gov web site

Block Triple Modular Redundancy: BTMR

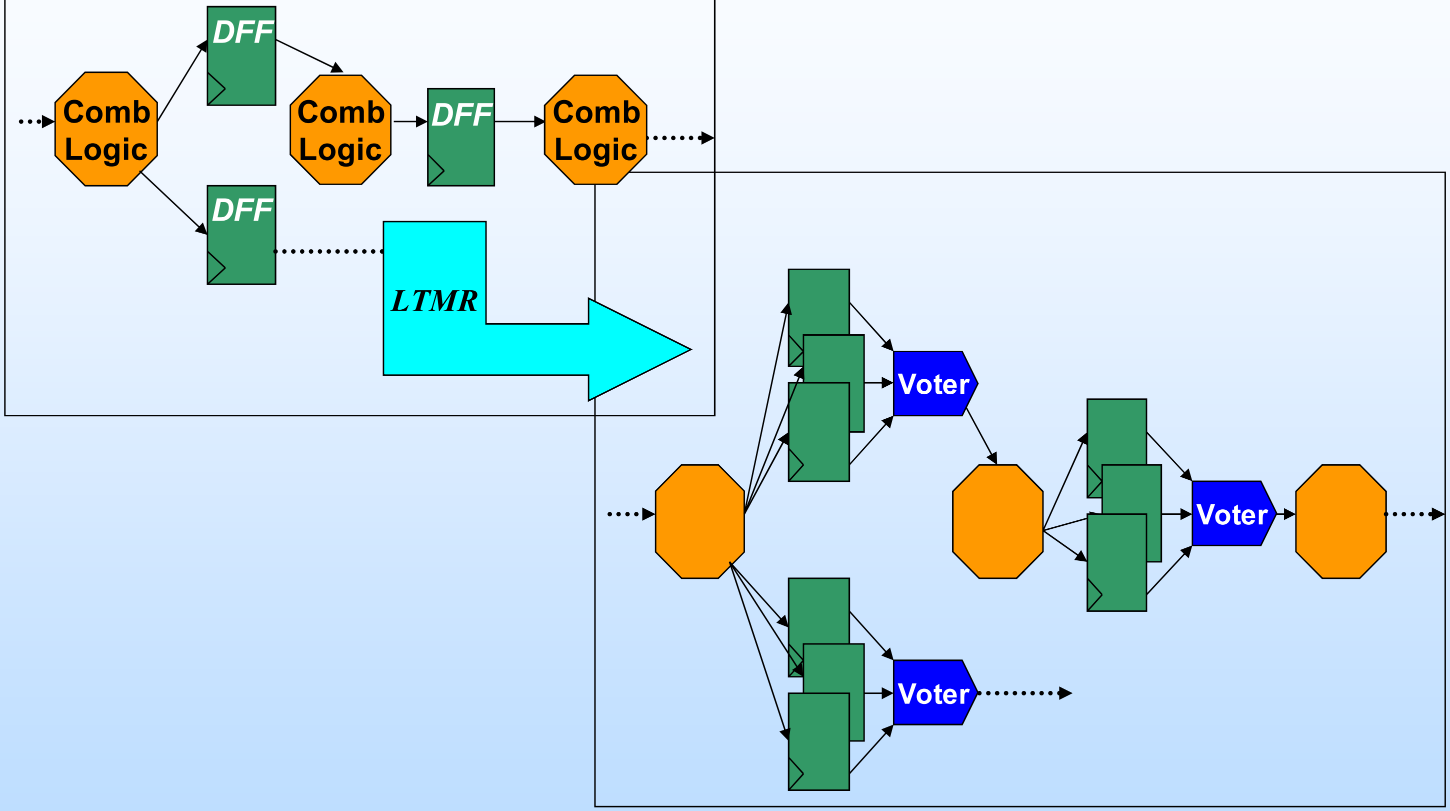


3x the error rate with triplication and no correction

- Need Feedback to Correct
- Generally can not apply internal correction from voted outputs
- Errors can accumulate – not an effective technique

To be presented by Melanie Berg at the Revolutionary Electronics in Space (ReSpace) / Military and Aerospace Programmable Logic Devices (MAPLD) 2011 Conference, Albuquerque, NM, August 22-25, 2011, and to be published on nepp.nasa.gov web site

Local Triple Modular Redundancy (LTMR): Only DFFs Voter+Feedback=Correction



$$P(fs)_{error} \propto P_{configuration} + \underbrace{P(fs)_{functionalLogic}}_{\overset{0}{P(fs)_{DFFSEU \rightarrow SEU}} + P(fs)_{SET \rightarrow SEU}} + P_{SEFI}$$

ProASIC3 LTMR Shift Register Data Path Model

$$\underset{DFF}{\exists}\left(\sum_{j=1}^{\text{\#StartPo int DFFs}} P(fs)_{DFFSEU\rightarrow SEU\,(j)} + \sum_{i=1}^{\text{\#Combinator ialLogicGa tes}} P(fs)_{SET\rightarrow SEU\,(i)}\right)$$

LTMR: P_{logic}=0

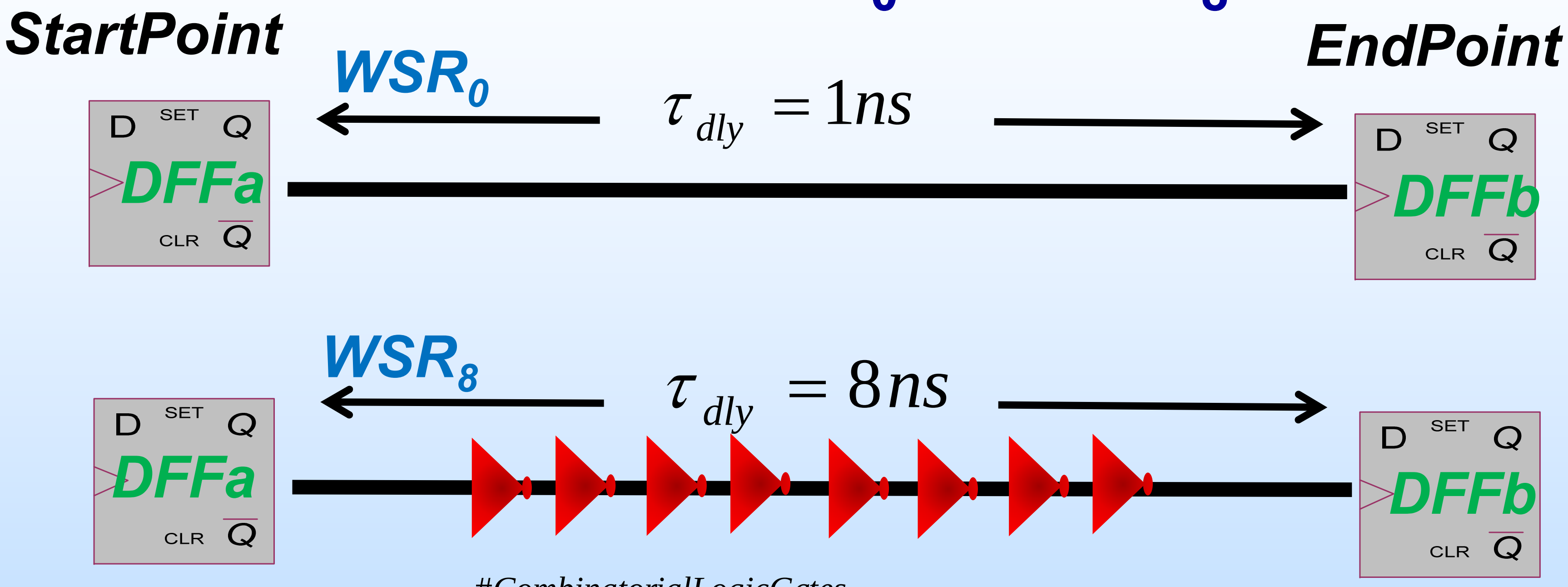
$$\underset{DFF}{\exists}\left(\cancel{P(fs)_{DFFSEU\rightarrow SEU}} + \sum_{i=1}^{\text{\#Combinator ialLogicGa tes}} P(fs)_{SET\rightarrow SEU\,(i)}\right)$$

Becomes significant

$$\sum_{i=1}^{\text{\#Combinator ialLogicGa tes}} P(fs)_{SET\rightarrow SEU\,(i)} \propto P_{gen\,(i)}P_{prop\,(i)}P_{\log ic}\,\tau_{width\,(i)}\,fs$$

LTMR removes DFF σ_{SEU} contribution. Dominance is now with SET capture ($P(fs)_{SET\rightarrow SEU}$)

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



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

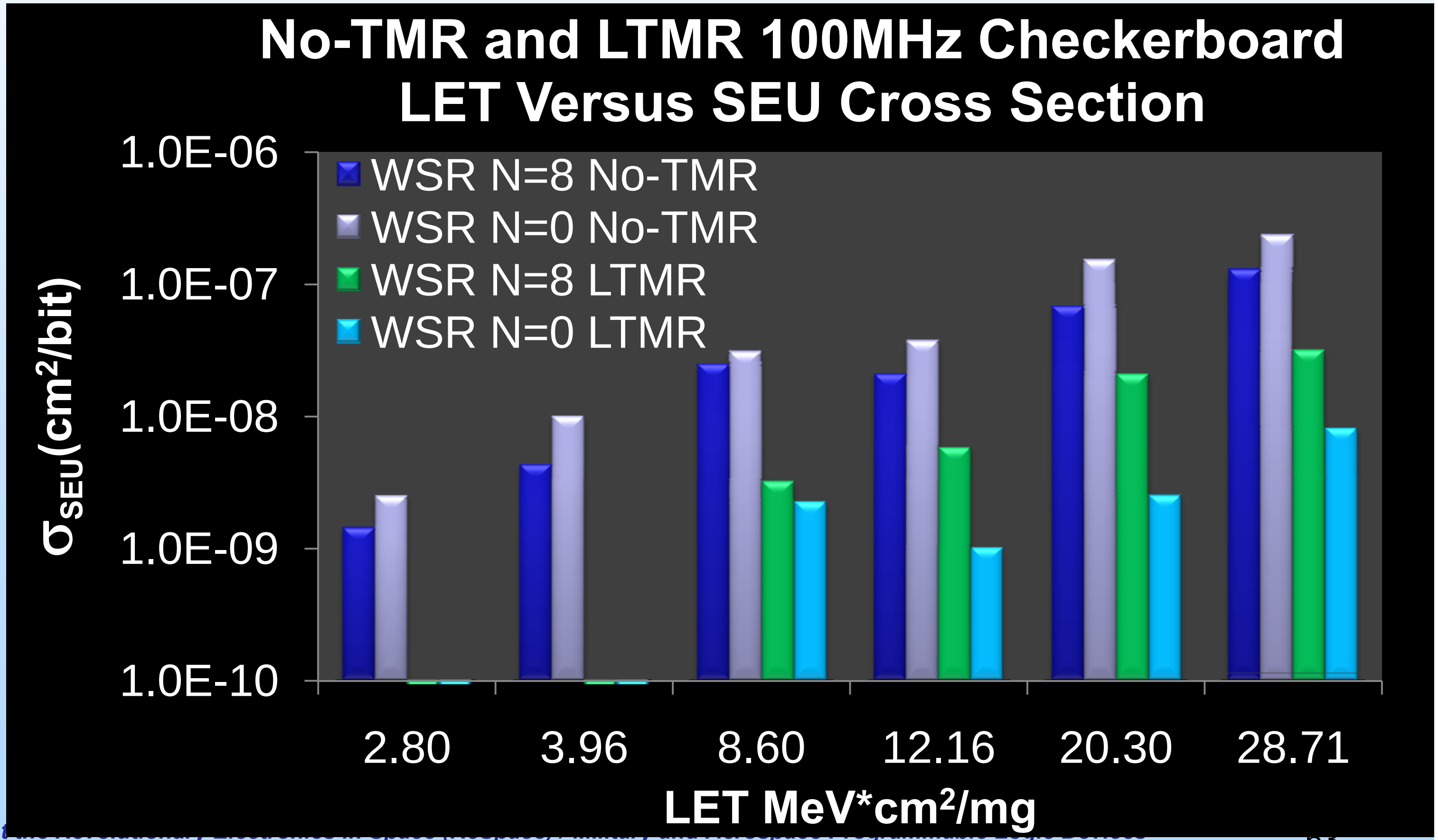
As we increase #combinatorial logic gates we increase σ_{SEU}

Hence for LTMR (disregarding P_{prop}),

σ_{SEU} WSR_8> σ_{SEU} WSR_0

σ_{SEU} Test Results: Windowed Shift Registers (WSRs) Trend Reverses with TMR

- LTMR is effective and has mitigated $P(fs)_{DFFSEU \rightarrow SEU}$
- LTMR: σ_{SEU} $WSR_0 < \sigma_{SEU}$ WSR_8 For every LET
- Increasing combinatorial logic increases σ_{SEU}



Summary of No-TMR vs. LTMR: Combinatorial Logic Effects

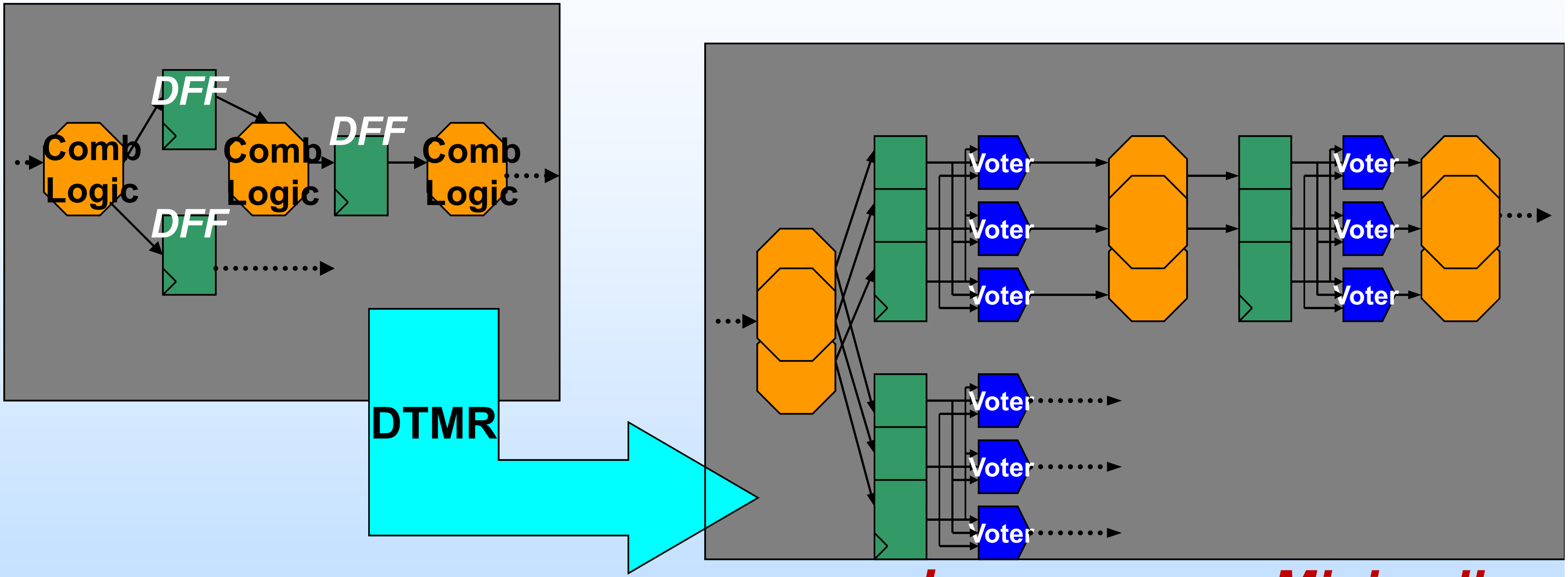
	No-TMR ProASIC3	LTMR ProASIC3
Significant circuit type	StartPoint DFF (sequential): SEU capture	Combinatorial: SET capture
Significant Model component	$P_{DFFSEU}(1-\tau_{dly}fs)$	$P_{gen}P_{prop}\tau_{width}fs$
Error Type	One sided function	Two-sided function
σ_{SEU} WSR$_8$ vs. σ_{SEU} WSR$_0$	σ_{SEU} WSR$_8$ <σ_{SEU} WSR$_0$	σ_{SEU} WSR$_8$ >σ_{SEU} WSR$_0$
Relative σ_{SEU} reasoning	WSR$_8$ has more combinatorial Logic and more τ_{dly} between DFFs hence σ_{SEU} is reduced	WSR$_8$ has more combinatorial Logic and has more opportunity for SET generation hence σ_{SEU} is increased

No-TMR vs. LTMR: Frequency Effects

- The same reasoning for τ_{dly} can be used for Frequency
- No-TMR:
  - Inversely proportional to frequency: $P_{DFFSEU}(1-\tau_{dly}fs)$
  - Increase Frequency Decrease σ_{SEU}
- LTMR
  - Directly proportional to frequency: $P_{gen}P_{prop}\tau_{width}fs$
  - Increase Frequency increase σ_{SEU}

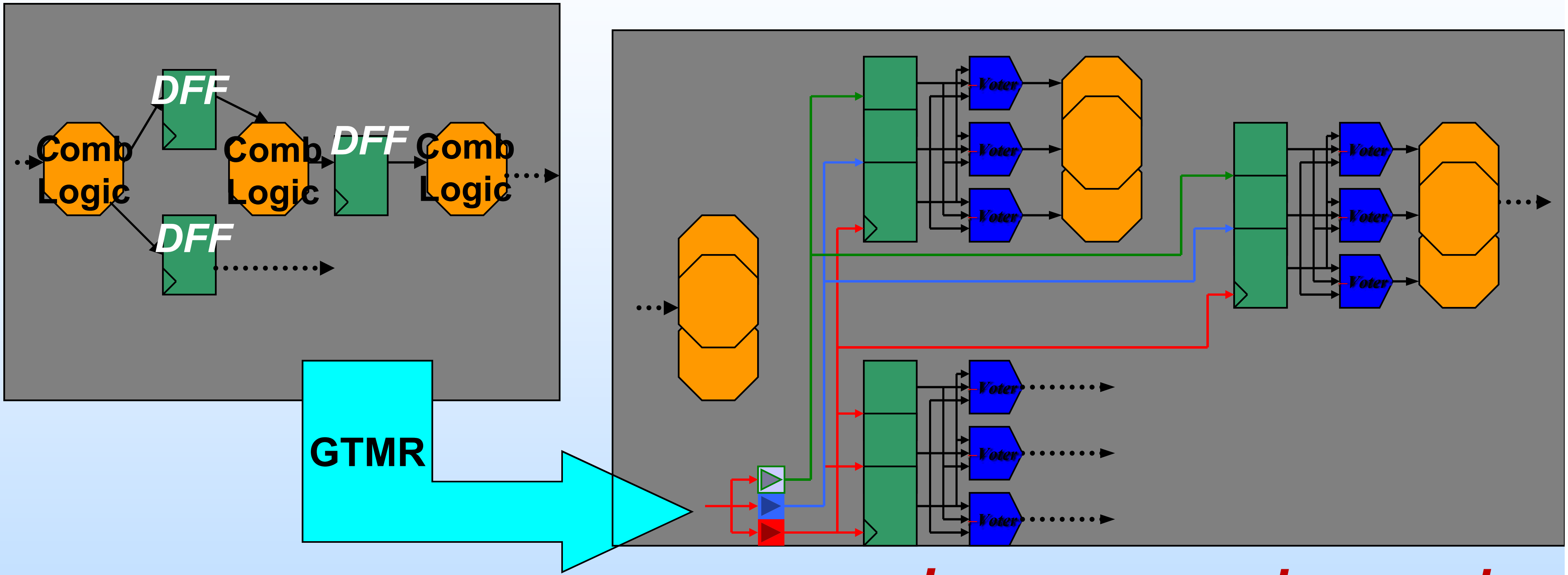
The assumption: If you operate your circuit slower then you will decrease your σ_{SEU} is NOT always valid!!!!!!!!!

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



$$P(f_s)_{error} \propto P_{configuration} + \underbrace{\overset{Low}{P(f_s)_{functionalLogic}}}_{\overset{0}{P(fs)_{DFFSEU\to SEU}} + \overset{Low}{P(f_s)_{SET\to SEU}}} + \overset{Minimally\ Lowered}{P_{SEFI}}$$

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



$$P(f_s)_{error} \propto P_{configuration} + \underbrace{\overset{\text{Low}}{P(f_s)_{functionalLogic}}}_{\overset{\text{Low}}{P(fs)_{DFFSEU \rightarrow SEU}} + \overset{\text{Low}}{P(f_s)_{SET \rightarrow SEU}}} + \overset{\text{Lowered}}{P_{SEFI}}$$

GTMR Proves 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
- Difficult to verify
- Does the FPGA contain enough low skew clock trees? (each clock + its synchronized reset)x3

Agenda

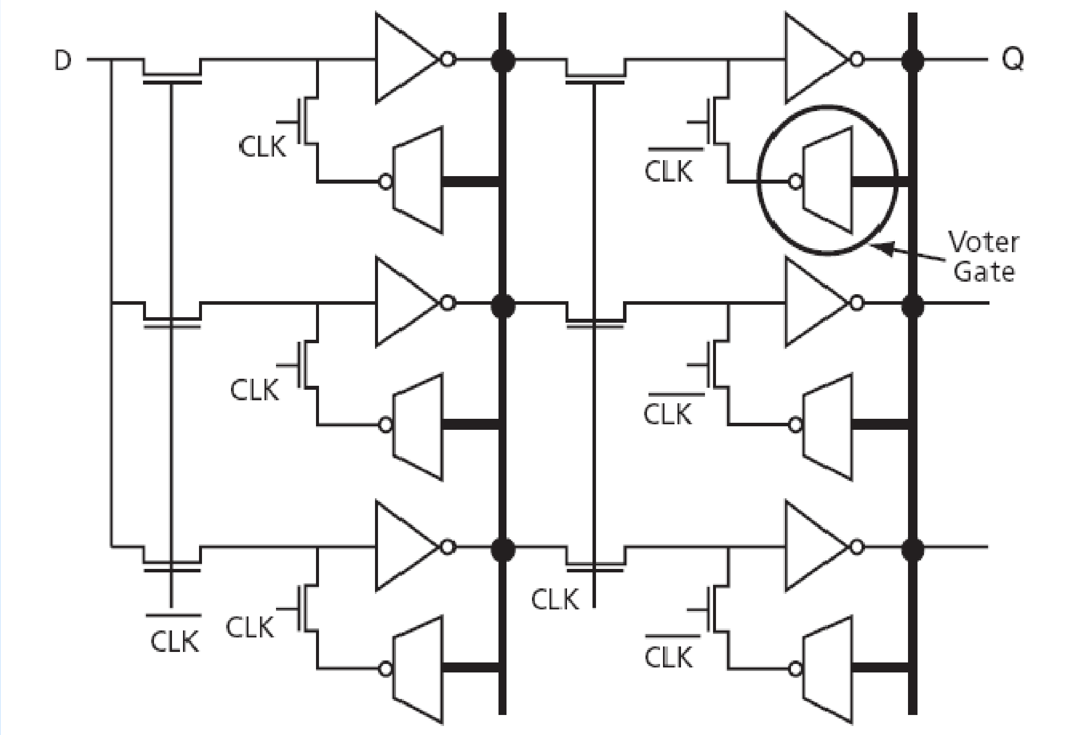
- Section I: Single Event Effects (SEEs) in Digital Logic
- Section II: Application of the NASA Goddard Radiation Effects and Analysis Group (REAG) FPGA SEU Model
- Section III: Reducing System Error: Common Mitigation Techniques
  - Triple Modular Redundancy (TMR)
  - Embedded Radiation Hardened by Design (RHBD)

Break

- Section IV: When Your Mitigation Fails
- Section V: Xilinx V4 and Mitigation
- Section VI: Fail-Safe Strategies

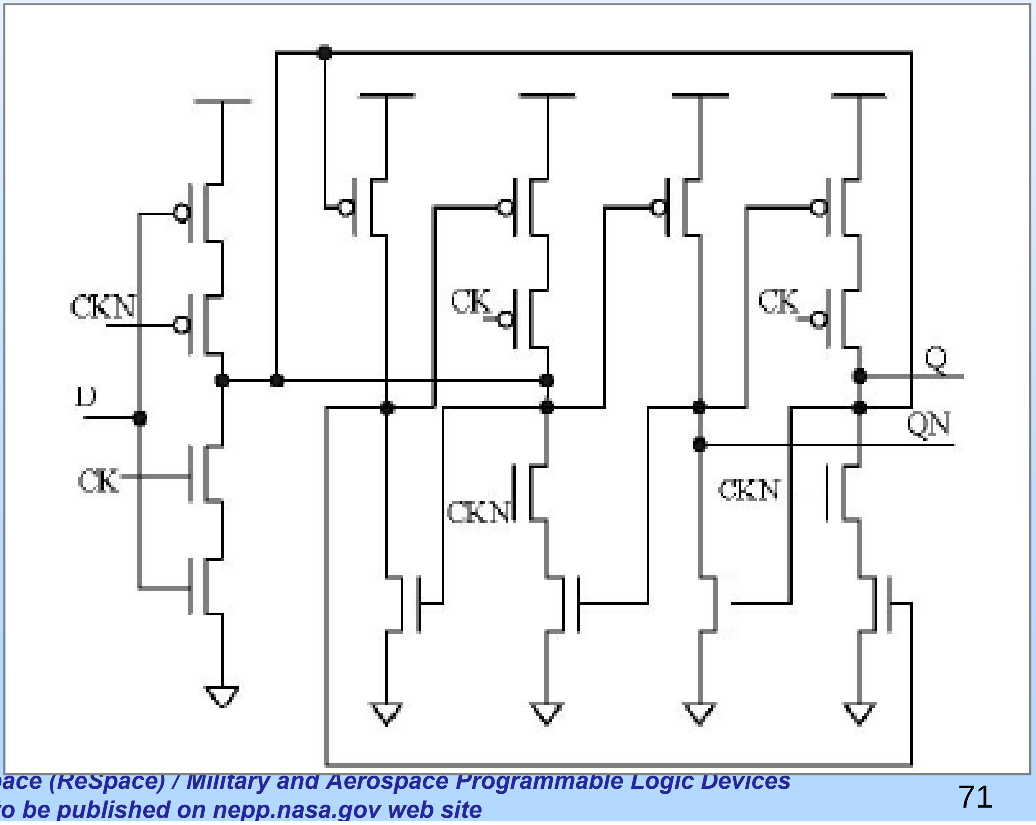
DFF with Embedded LTMR: Microsemi (Actel) RTAXs Family of FPGA

- Localized (only at DFF)
- Microsemi uses Wired "OR" approach to voting – no SETs on voters



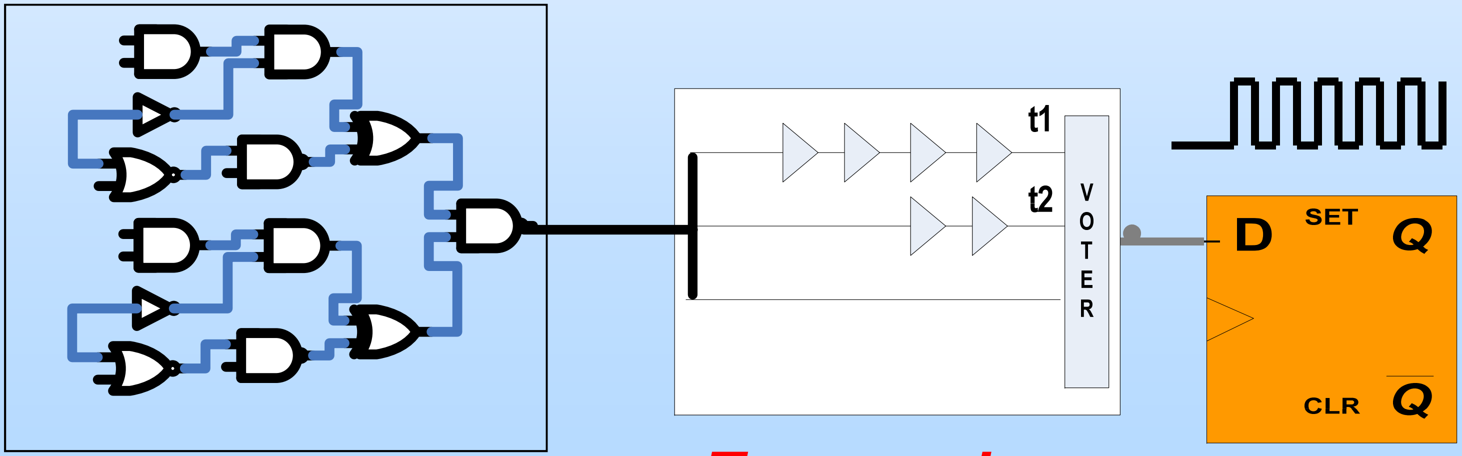
DFF with Embedded Dual Interlock Cell (DICE): Aeroflex Eclipse FPGA

- 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. FPGA designers should not implement– best if embedded
- Maximum Clock frequency is reduced by the amount of new delay



Combining Embedded Schemes

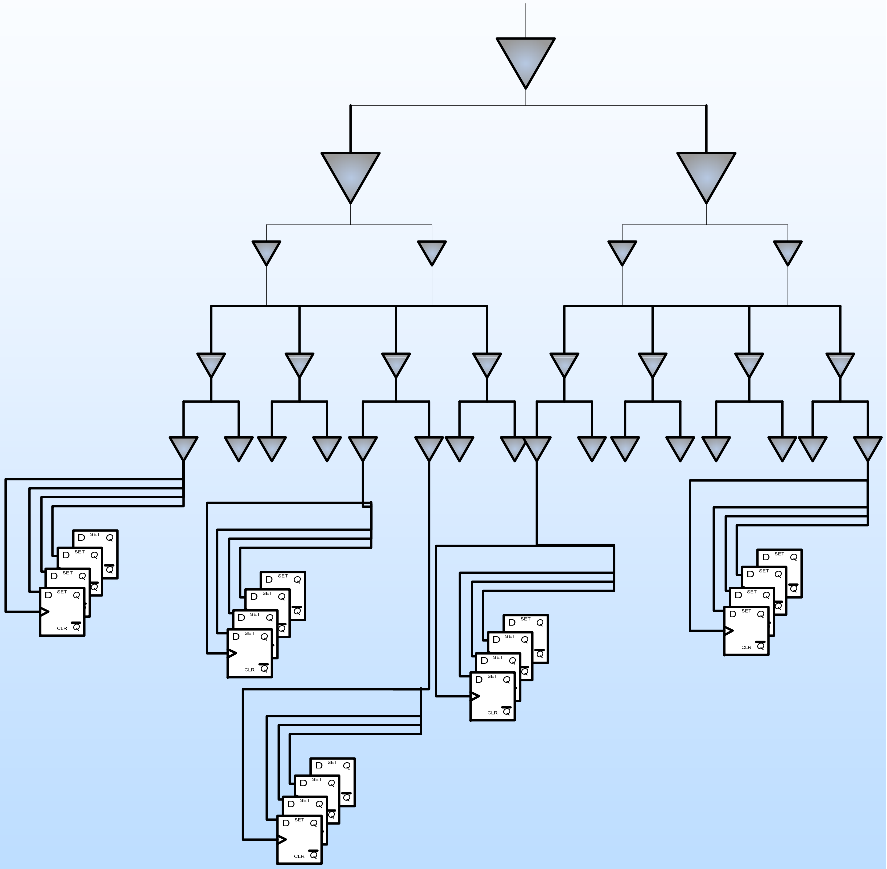
- Some Radiation Hardened by Design (RHBD) schemes combine embedded temporal redundancy with localized redundant latches:
  - TR+LTMR
  - TR+DICE
- New Xilinx RHBD FPGA (Virtex 5QV) has embedded TR+DICE

$$P(f_s)_{error} \propto P_{configuration} + \underbrace{\overset{\text{Low}}{P(f_s)_{functionalLogic}}}_{\overset{\text{Low}}{P(fs)_{DFFSEU \rightarrow SEU}} + \overset{\text{Lowered}}{P(f_s)_{SET \rightarrow SEU}}} + P_{SEFI}$$

RHBD for Global Routes

- Some RHBD FPGAs contain hardened clock trees and other global routes
- Global structures are generally hardened by using larger buffers
- TR will not work on a global network (signal integrity, skew balancing, speed and area would be significantly affected)

Clock Tree



Break!
10 minutes

Agenda

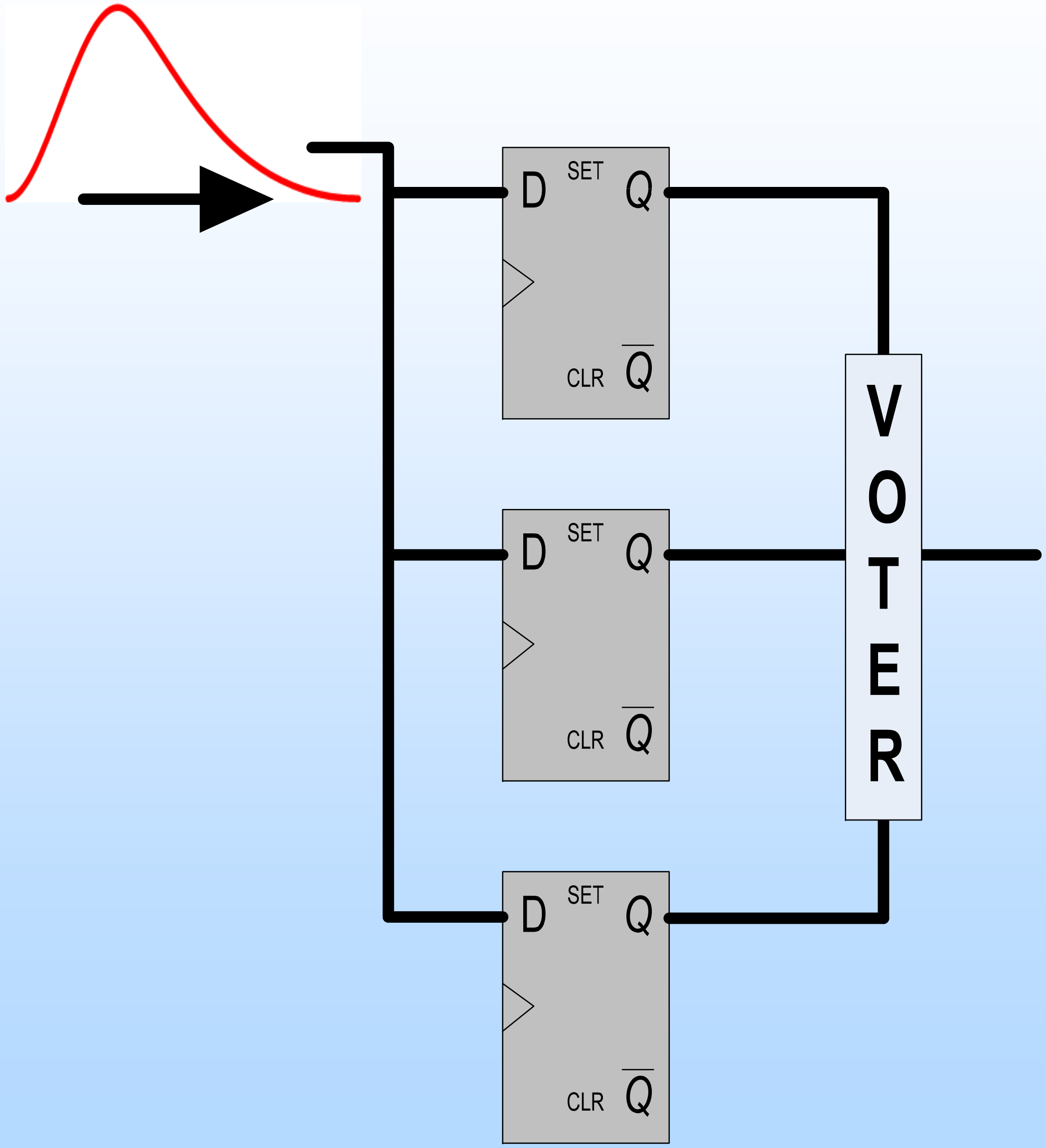
- Section I: Single Event Effects in Digital Logic
- Section II: Application of the NASA Goddard Radiation Effects and Analysis Group (REAG) FPGA SEU Model
- Section III: Reducing System Error: Common Mitigation Techniques

Break

- **Section IV: When Your Mitigation Fails**
- Section V: Xilinx V4 and Mitigation
- Section VI: Fail-Safe Strategies

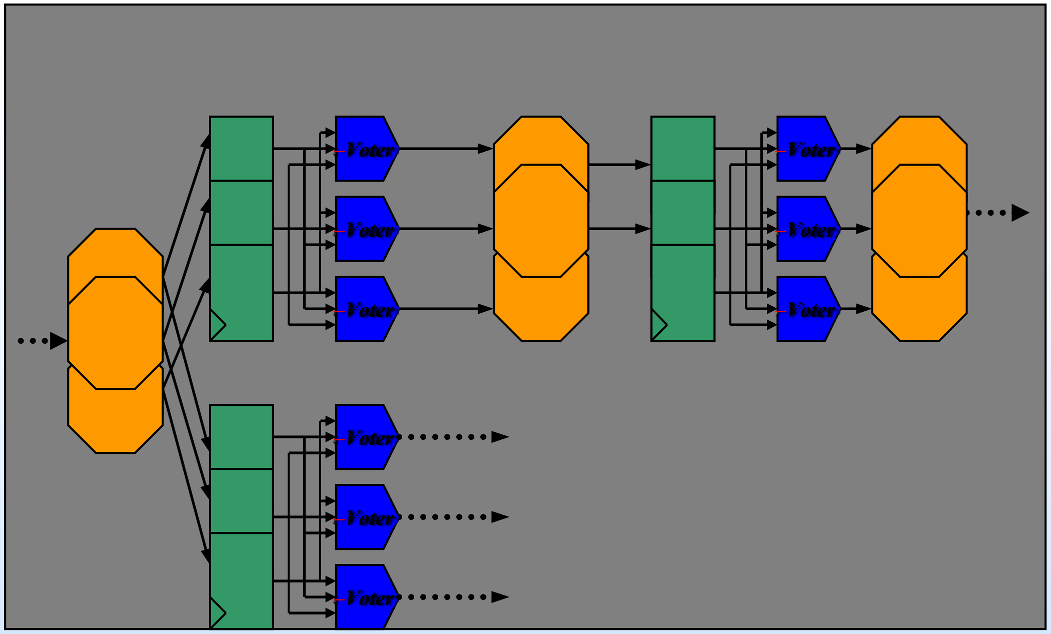
LTMR Failure

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



To be presented by Melanie Berg at the Revolutionary Electronics in Space (ReSpace) / Military and Aerospace Programmable Logic Devices (MAPLD) 2011 Conference, Albuquerque, NM, August 22-25, 2011, and to be published on nepp.nasa.gov web site

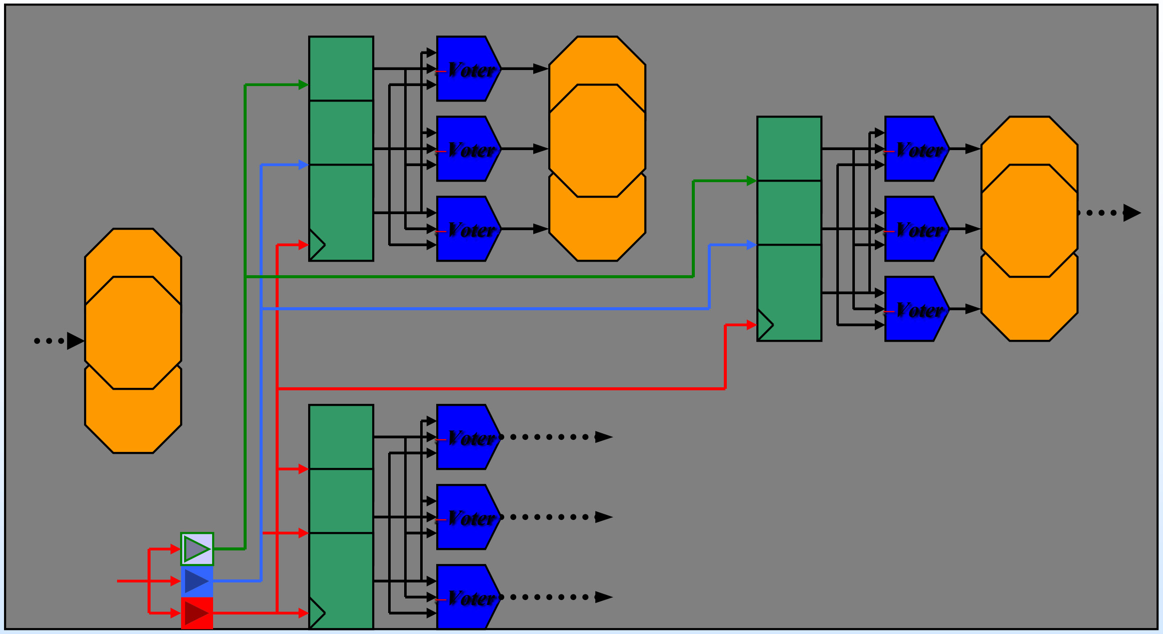
DTMR Failures



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

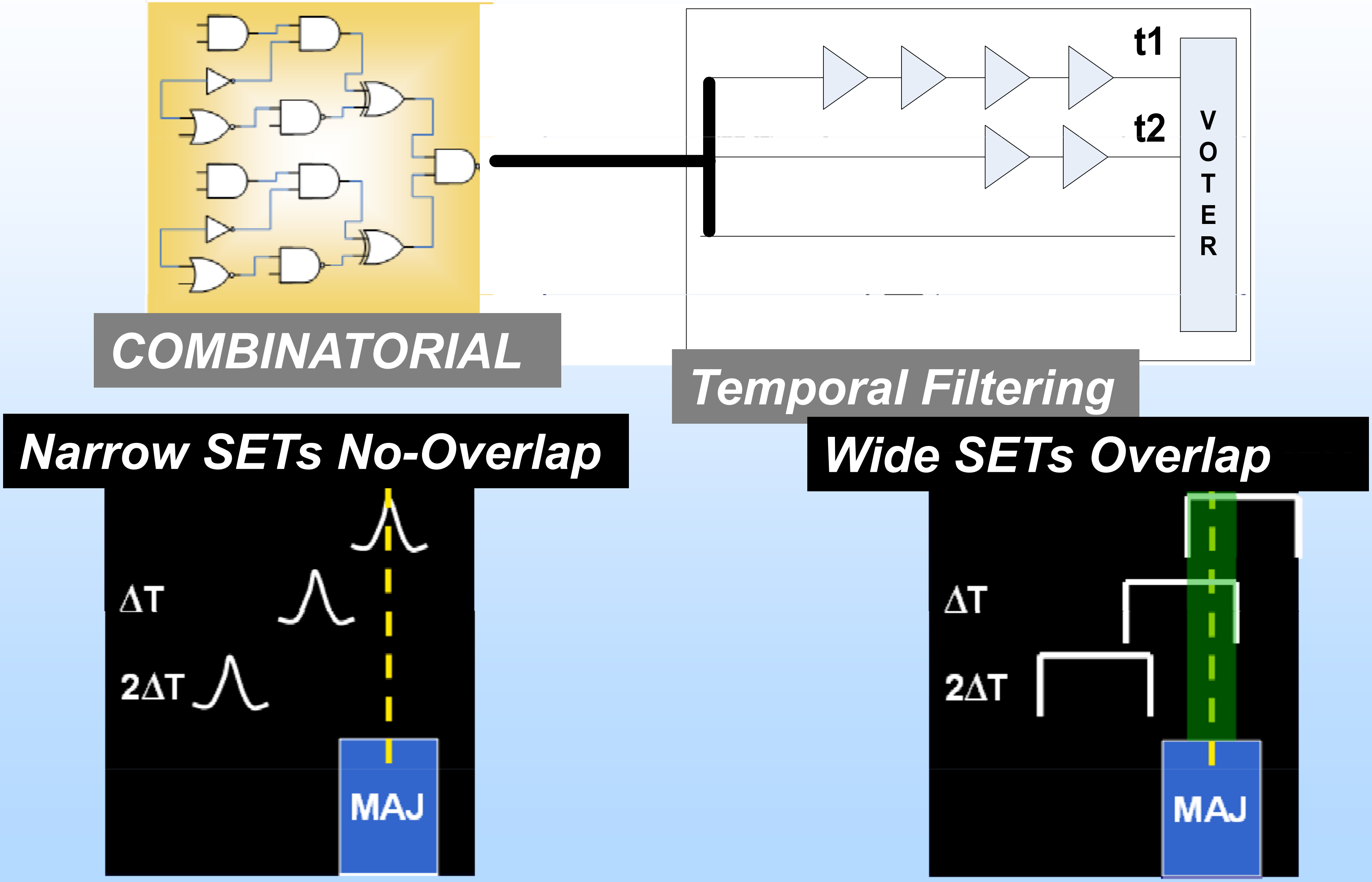
To be presented by Melanie Berg at the Revolutionary Electronics in Space (ReSpace) / Military and Aerospace Programmable Logic Devices (MAPLD) 2011 Conference, Albuquerque, NM, August 22-25, 2011, and to be published on nepp.nasa.gov web site

GTMR Failures



- **Domain placement**
  - **possible for domains to share common routing matrix**
  - **Hit to shared routing matrix can take out two domains**
- **Clock Skew**
- **Asynchronous clock domain crossings need additional voter insertion – tools don't auto handle**

TR Failures



COMBINATORIAL

Temporal Filtering

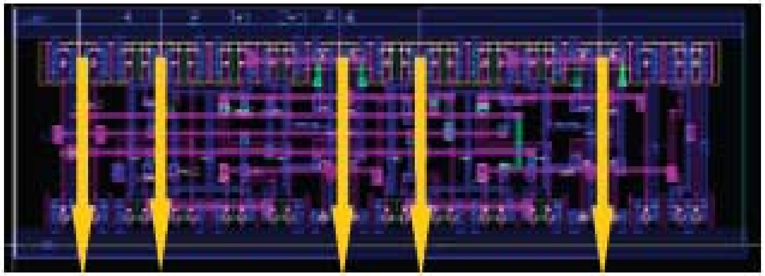
Narrow SETs No-Overlap

Wide SETs Overlap

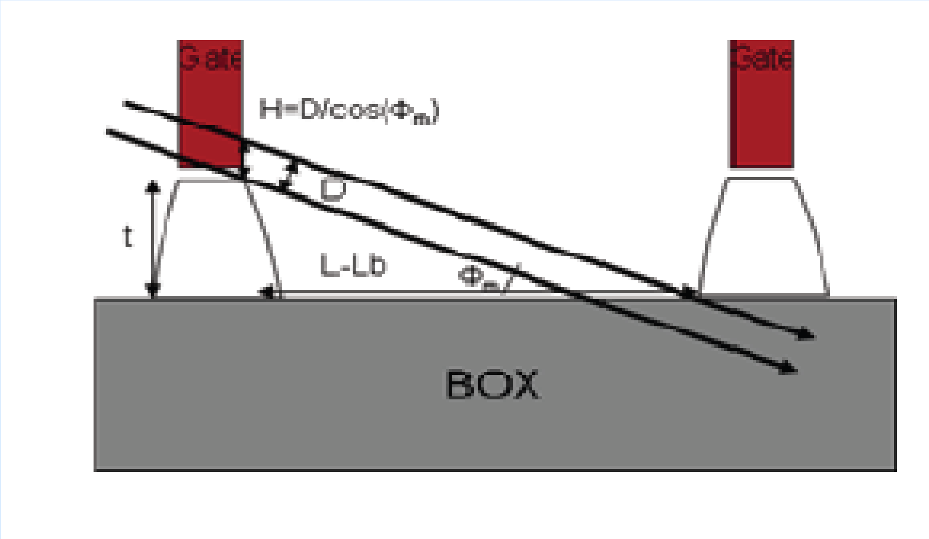
DICE Susceptibility

- **One particle strike can take out 2 nodes and break Dice**

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



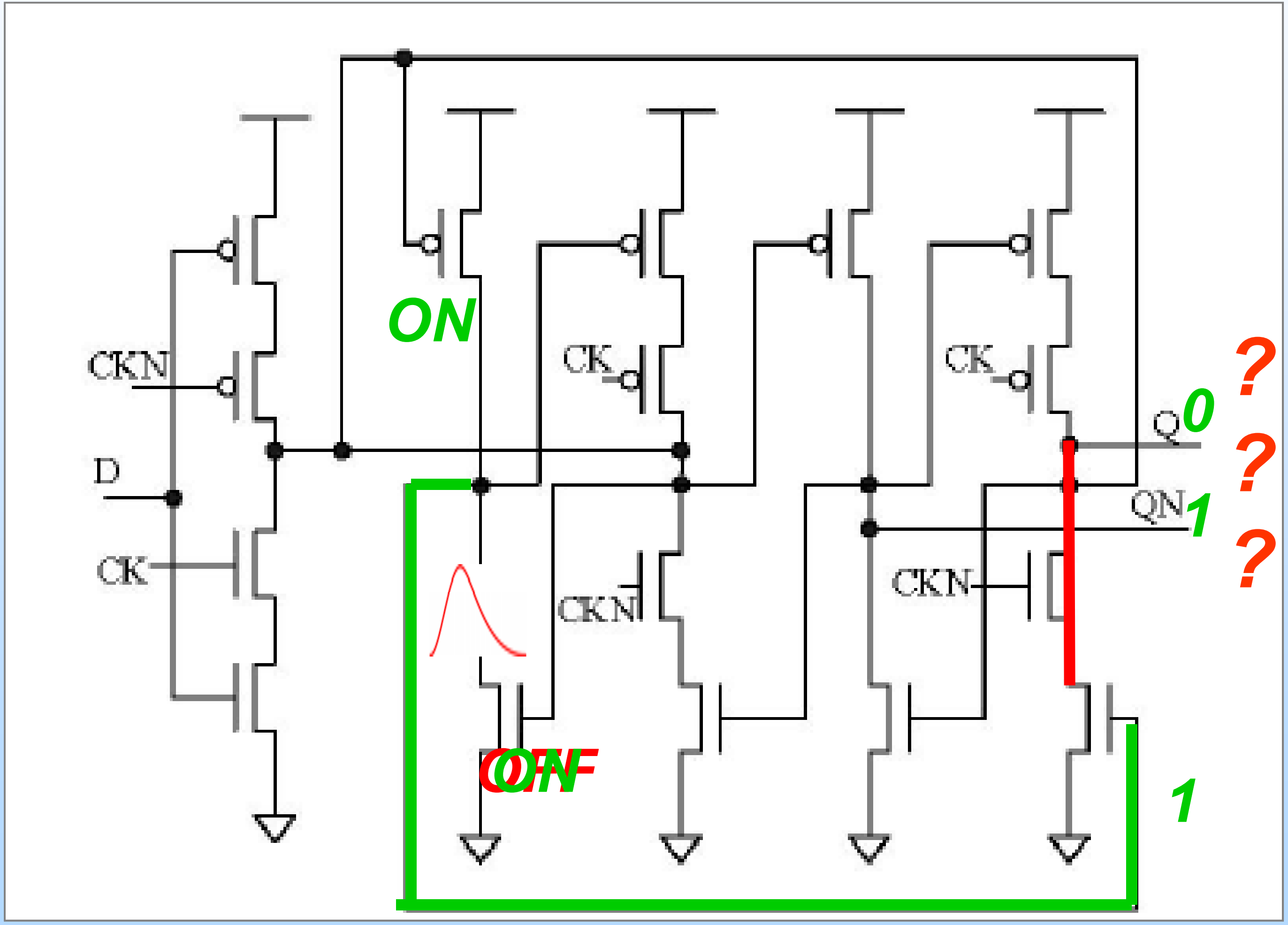
Minimum spaced DICE flip flop
(lines show critical node)



Multiple bit upset ion strike simplified geometric model

DICE Susceptibility: Not So Bad for a SRAM Cell – However, Can Cause Metastability Problems in a High Speed Master-Slave DFF

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



To be presented by Melanie Berg at the Revolutionary Electronics in Space (ReSpace) / Military and Aerospace Programmable Logic Devices (MAPLD) 2011 Conference, Albuquerque, NM, August 22-25, 2011, and to be published on nepp.nasa.gov web site

Agenda

- Section I: Single Event Effects in Digital Logic
- Section II: Application of the NASA Goddard Radiation Effects and Analysis Group (REAG) FPGA SEU Model
- Section III: Reducing System Error: Common Mitigation Techniques

Break

- Section IV: When Your Mitigation Fails
- **Section V: Xilinx V4 and Mitigation**
- Section VI: Fail-Safe Strategies

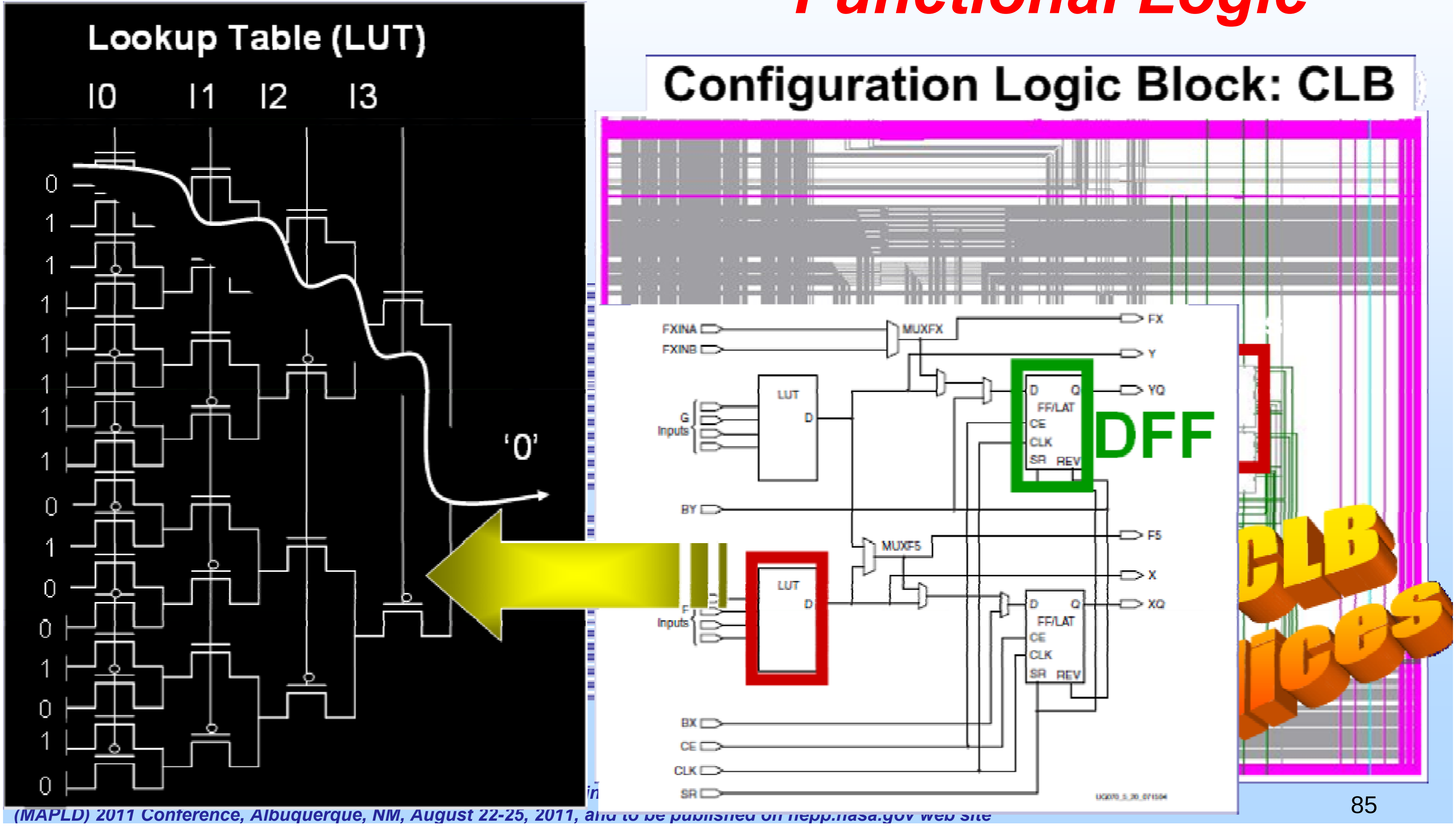
Commercial Devices in Critical Applications

- Why are commercial devices being considered?
  - Fast
  - Cheap
  - Easier to design with (especially with reprogramability option)
- Commercial devices were not designed for critical applications…. Considerations:
  - Requires extensive knowledge of SEU error signatures
  - Requires knowledge of proper mitigation techniques
  - Requires additional tool costs
  - Watchdogs become more complex
  - Recovery becomes more complex
  - Verification becomes more complex

The following slides illustrate some of the considerations regarding using commercial devices for critical applications

General Xilinx Virtex 4 FPGA Architecture

Functional Logic



Xilinx SX55: Radiation Test Data

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

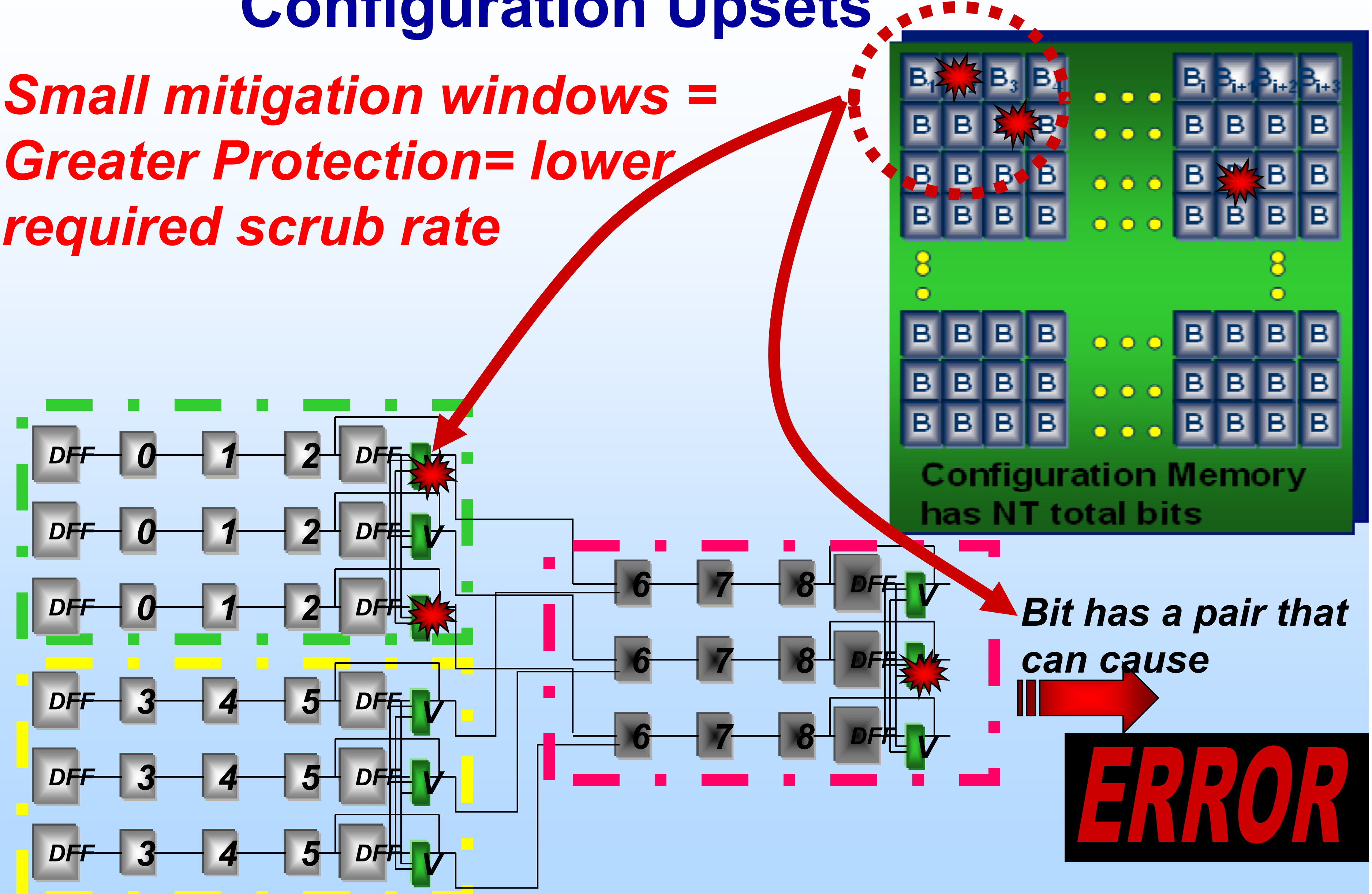
	Probability	Error Rate	LEO $\frac{Upsets}{device-day}$	GEO $\frac{Upsets}{device-day}$
Configuration Memory: XQR4VSX55	$P_{configuration}$	$\frac{dE_{configuration}}{dt}$	**7.43**	**4.2**
Combined SEFIs per device	P_{SEFI}	$\frac{dE_{SEFI}}{dt}$	**$7.5x10^{-5}$**	**$2.7x10^{-5}$**

- **For non-mitigated designs the most significant upset factor is:** $P_{Configuration}$
- **Localized redundancy is NOT effective. Designer must use DTMR or GTMR**

To be presented by Melanie Berg at the Revolutionary Electronics in Space (ReSpace) / Military and Aerospace Programmable Logic Devices (MAPLD) 2011 Conference, Albuquerque, NM, August 22-25, 2011, and to be published on nepp.nasa.gov web site

TMR Mitigation Windows and Configuration Upsets

Small mitigation windows = Greater Protection= lower required scrub rate



Configuration Memory has NT total bits

Bit has a pair that can cause

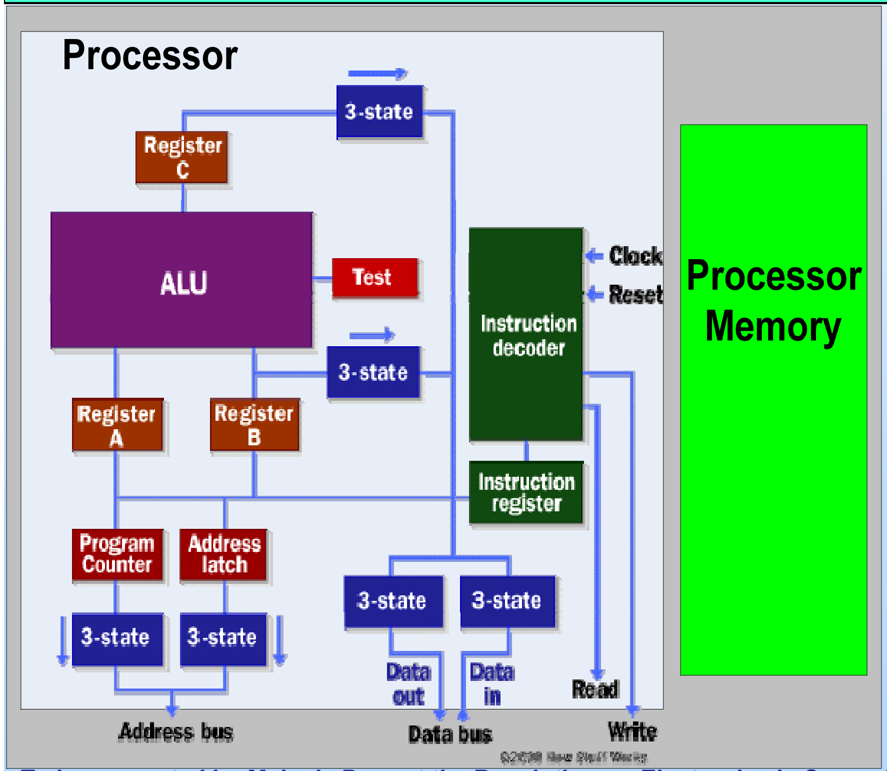
Is GTMR (a.k.a. XTMR) or DTMR All We Need?

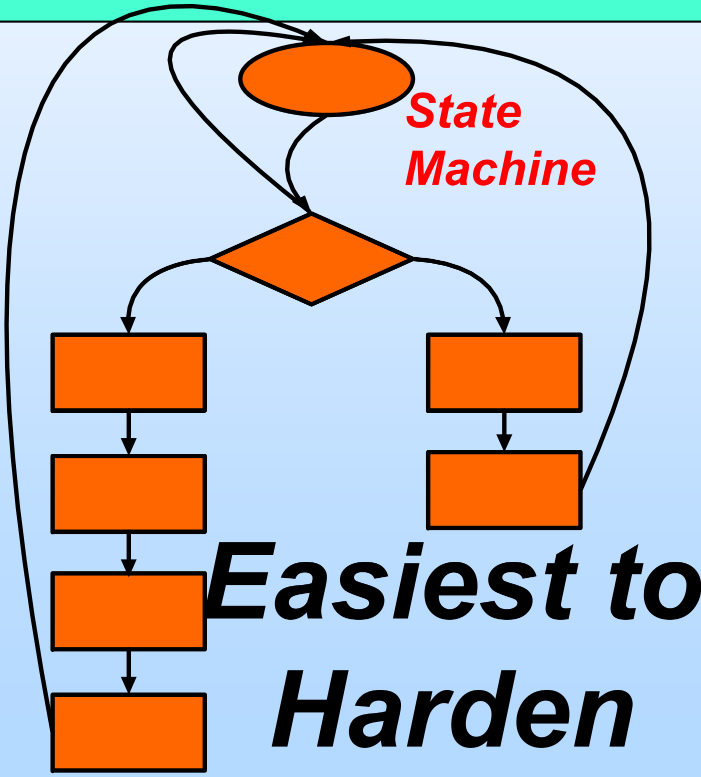
- GTMR only:
  - Masks and corrects the Functional logic
  - Masks most configuration upsets (no correction to configuration bits)
- Two upsets in a mitigation window can cause a system upset
- Accumulation in Configuration can occur and eventually break the GTMR
- Scrubbing corrects the configuration memory
  - Does not reduce Configuration Upset Rate
  - Reduces the accumulation bit error rate
  - Does not correct functional upsets
  - Will not disrupt device operation

Variations of Scrubbing Implementation

Scrubber Location → *Internal to FPGA*
Scrubber Location → *External to FPGA*

Control → *Processor Based*
Control → *State Machine Based*



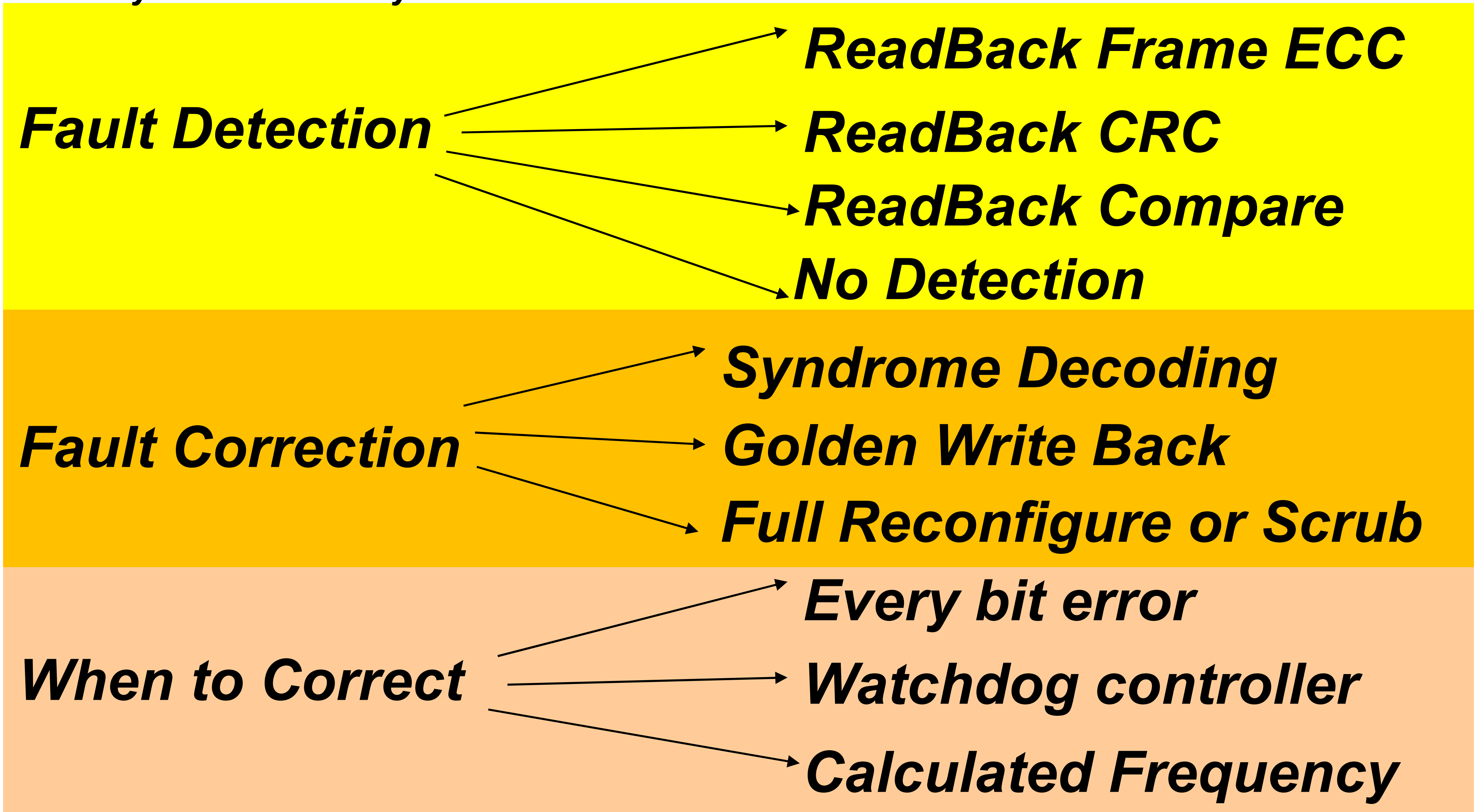


Easiest to Harden

Scrubber Fault Detection and Correction

ECC: Error Correction Code

CRC: Cyclic Redundancy Code



Fault Detection

- ***ReadBack Frame ECC***
- ***ReadBack CRC***
- ***ReadBack Compare***
- ***No Detection***

Fault Correction

- ***Syndrome Decoding***
- ***Golden Write Back***
- ***Full Reconfigure or Scrub***

When to Correct

- ***Every bit error***
- ***Watchdog controller***
- ***Calculated Frequency***

System Operation and Scrubbing

- System retains its state of operation during scrubbing cycles
- Scrubbing can NOT correct DFFs (state machines, counters, control registers, etc…)
- If operation has been affected, correcting the configuration bit may not recover operation
  - Reset
  - Re-synchronize
  - Correction circuitry
  - Re-power or Reconfigure
    - FPGA becomes inactive
    - Peripheral devices that require control will require an alternate source of control

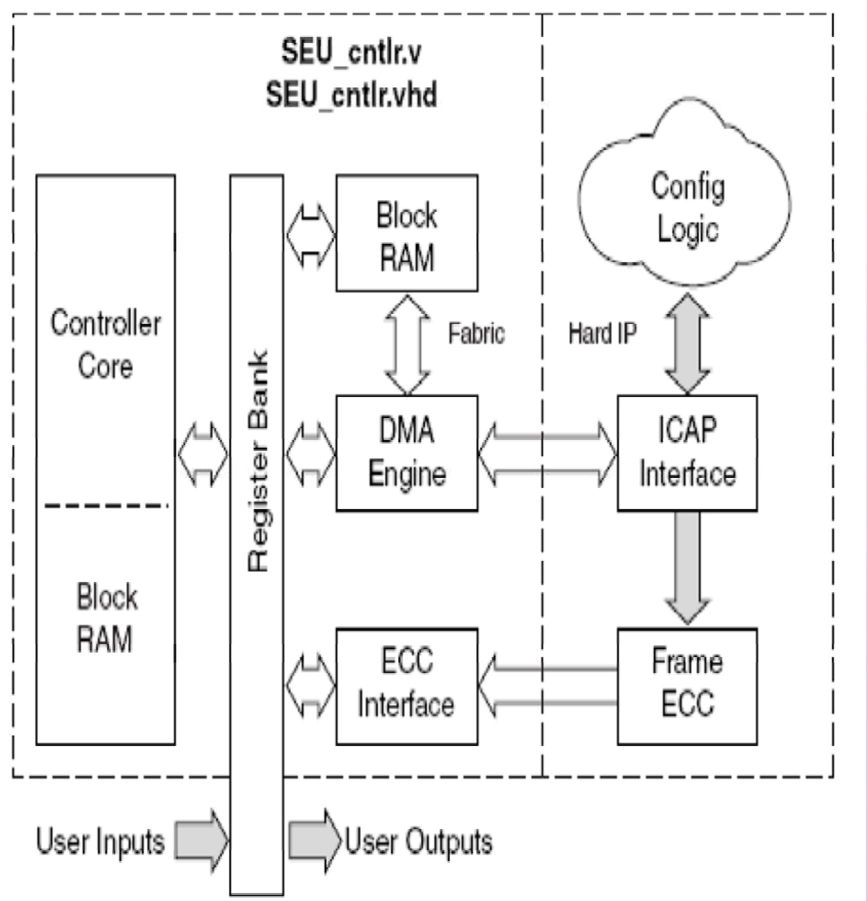
Mitigation and Scrubbing

- For a critical design Scrubbing is secondary but should be implemented
- As mitigation windows increase (partial mitigation), scrubbing becomes more of a significant factor
  - Larger windows = more bits with pairs that can break mitigation
  - Several bit upsets per day makes accumulation significant

Processor Based Internal Scrubber

- **Failed Miserably during radiation testing**
  - **Processor was internal to FPGA with no mitigation**
  - **Processor used memory with no mitigation**
  - **Detection and correction scheme was Single Error Correct Double Error Detect (SECDED)**
- **Presented: RADECs 2007, "*Effectiveness of Internal versus External SEU Scrubbing Mitigation Strategies in a Xilinx FPGA: Design, Test, and Analysis*"; M. Berg et. al**

Processor for Fault Correction



Frame ECC is only SECDED

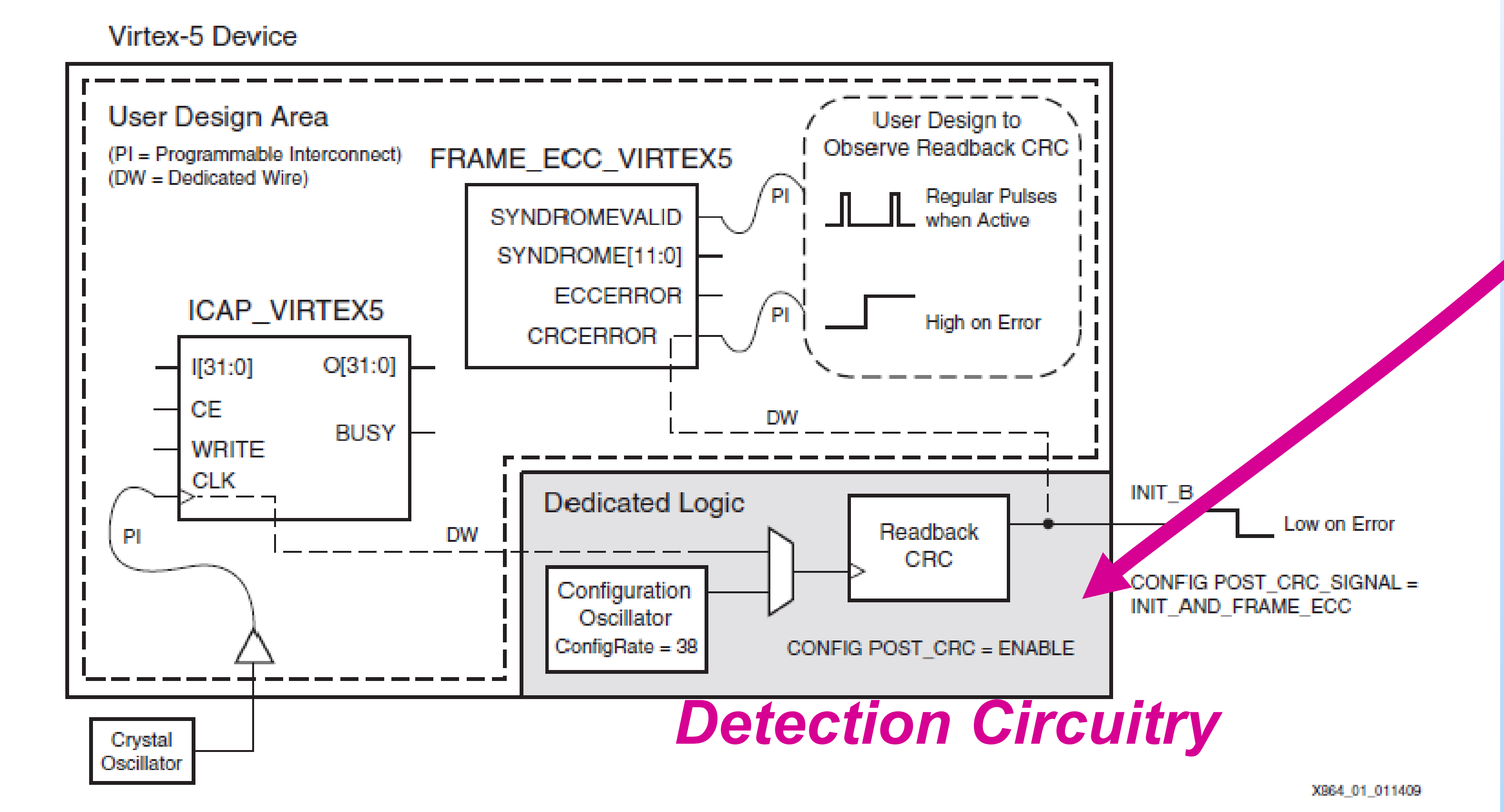
To be presented by Melanie Berg at the Revolutionary Electronics in Space (ReSpace) / Military and Aerospace Programmable Logic Devices (MAPLD) 2011 Conference, Albuquerque, NM, August 22-25, 2011, and to be published on nepp.nasa.gov web site

Be Aware and Take Caution When Using SECDED Based Scrubbers

- SECDED Scrubbers can only correct one upset per configuration frame
- MBU or an accumulation of upsets within a frame can cause SECDED correction circuitry to write incorrect values into the frame

Virtex5 Scrubbing

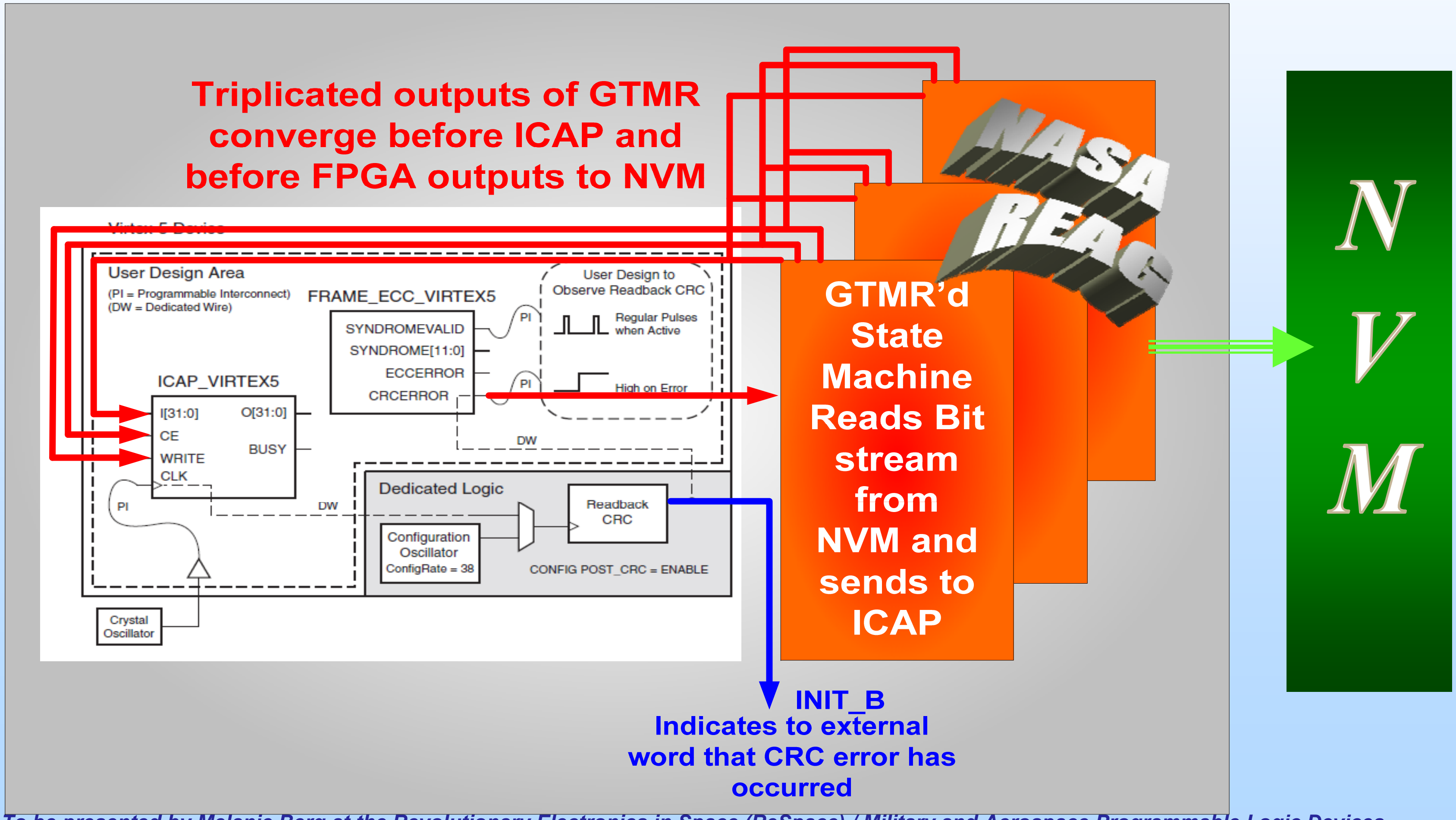
- New embedded logic that performs readback in the background
- Read back is now free
- Lower power - does not require I/O switching



Detection Circuitry

REAG State Machine Driven Error Correction

- Does not use FRAME_ECC Syndrome (see block diagram)
- Scrubs entire Configuration upon CRC error using the Internal Configuration Access Port (ICAP)



Triplicated outputs of GTMR converge before ICAP and before FPGA outputs to NVM

GTMR'd State Machine Reads Bit stream from NVM and sends to ICAP

INIT_B
Indicates to external word that CRC error has occurred

V5 REAG State Machine Driven Performance – Proton Testing

- External
  - CRC Error invoked external scrubber
  - Was always able to correct and operation was not disrupted
- Internal
  - CRC Error invoked internal scrubber
  - Was always able to correct and operation was not disrupted
  - Worked just as well as external scrubber
  - Total circuitry only occupied 134 slices
    - GTMR: Clocks, DFFs, and LUTs
    - Non GTMR Logic: ICAP + internal read back blocks

V5 REAG State Machine Driven Performance – Heavy Ion Testing – Not As Good as Proton Results

- External
  - CRC Error invoked external scrubber
  - Was always able to correct and operation was not disrupted
- Internal
  - CRC Error invoked internal scrubber
  - readback SEFIs occurred – required a full reconfiguration. Upsets occurred at the lowest LET tested – 2.5 MeV*cm^2/mg

An Alternative is to build your own CRC checker (either externally or GTMR'd internally

Xilinx V4 and V5 Takeaway Points

- Can be used in non-critical missions without any mitigation
  - Upset rates in the order of days
  - Will need to be reconfigured periodically
  - Watchdog required
  - Great for non-critical data processing
- Can be used in a critical path (beware of SEFIs) with mitigation
  - Utilize mitigation tools from a proven vendor, otherwise:
    - Design may break after GTMR (XTMR) insertion
    - Mitigation may not be placed where expected
  - Upset rates are low
  - It is a complex process to make a commercial device perform at the required level for a critical mission
- Xilinx has come up with a solution: RHBD V5QV

Agenda

- Section I: Single Event Effects in Digital Logic
- Section II: Application of the NASA Goddard Radiation Effects and Analysis Group (REAG) FPGA SEU Model
- Section III: Reducing System Error: Common Mitigation Techniques

Break

- Section IV: When Your Mitigation Fails
- Section V: Xilinx V4 and Mitigation
- **Section VI: Fail-Safe Strategies**

How Safe is Your Design?

- Are SEU error modes addressed properly?
  - Did you mitigate where you expected to mitigate?
  - Have you taken into account device SEFIs?
- Are there lock-up conditions in my design?
- Does your strategy protect the entire critical path?
- Is the synthesized design fail-safe?
- Can your watch-dog catch failure?
- Will your recovery scheme work?
- What are the limitations of your verification strategy?

The list goes on… Focus will be on fail-safe concerns regarding SEUs

System Fail-Safe Concepts

Common FPGA Options for Space-Flight Missions:

- *Xilinx Virtex 4*
- *Microsemi RTProASIC3*
- *Microsemi RTAX2000S*

Know The SEU Susceptibility: FPGA Types and the REAG FPGA SEU Model

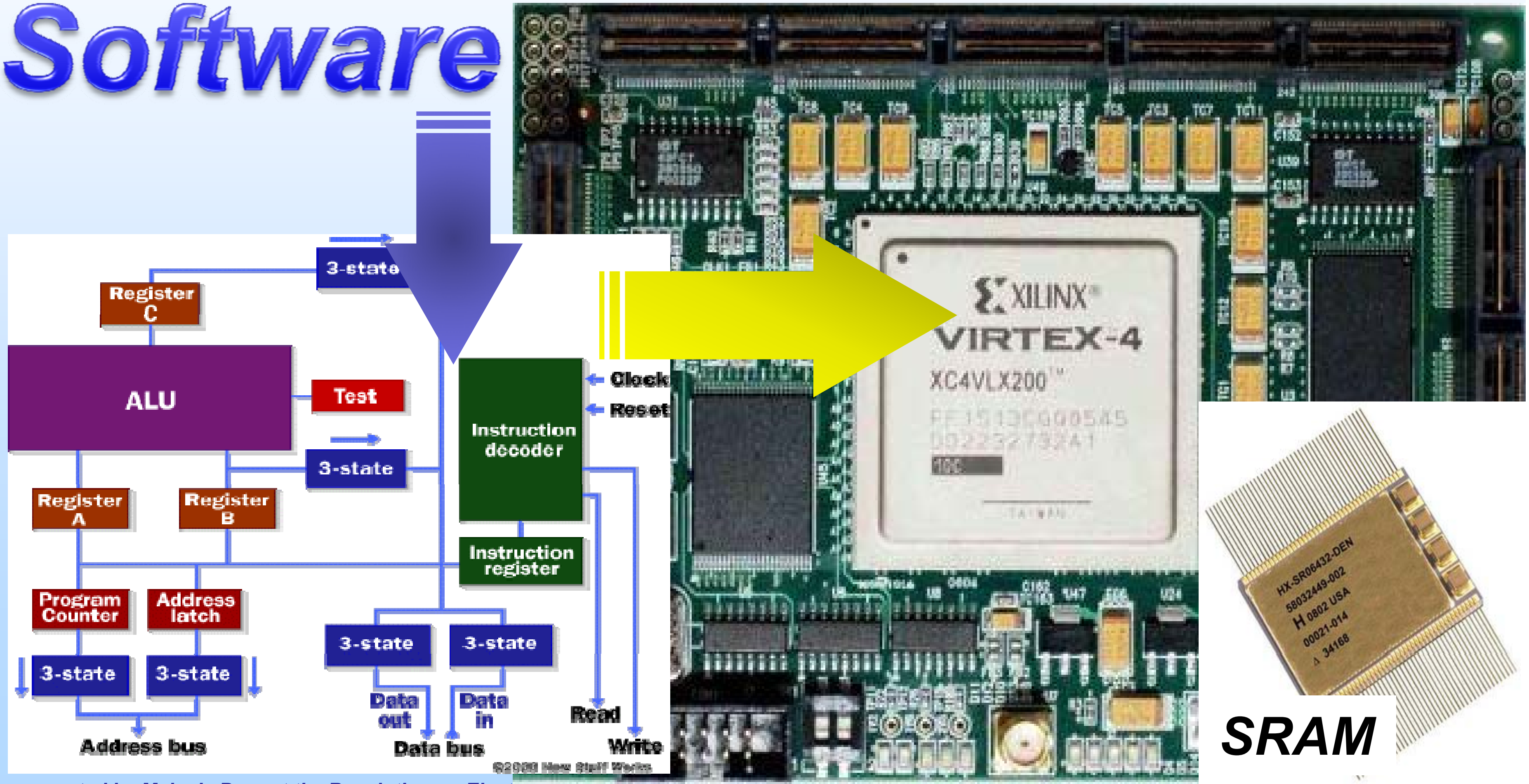
FPGA	REAG Model
RTAX2000s	$P(fs)_{error} \propto P(fs)_{SET \rightarrow SEU} + P_{SEFI}$
Virtex 4	$P(fs)_{error} \propto P_{Configuration}$
RTProASIC3	$P(fs)_{error} \propto P(fs)_{DFFSEU \rightarrow SEU} + P(fs)_{SET \rightarrow SEU} + P_{SEFI}$

Virtex 4 susceptibility is proportional to configuration upsets (in the order of days).

Example Has A Processor Designed into the FPGA

Software is required to run in the processor

External memory may also be required



Hardened Software… How Fail-Safe Is It?

- Depends on how susceptible the hardware is
- Upsets do not occur in software
  - Hardware gets upset and can disrupt software
- Software can be used to reduce the necessity of resets
  - Best for register or memory type SEUs
  - Can also be used in pipelined architectures where upsets can be flushed out
- Software will not help with hardware upsets such as:
  - Stuck states
  - Broken routes
  - Broken functionality

CMOS Microprocessors in a Heavy Ion Rich Radiation Environment

- Critical mircoprocessor designs in heavy ion environments require some form of hardened hardware before hardened software can be effective

FPGA Type	Mitigation?	Hardened Software Effectiveness
Virtex 4	None	Low
Virtex 4	User inserted DTMR or GTMR	Good
RTAXs	Embedded LTMR+Configuration+hardened globals	Good
RTProASIC3	Configuration	Medium
RTProASIC3	Configuration + User inserted LTMR	Good

↗ *Vague description – really depends on design and actual radiation environment*

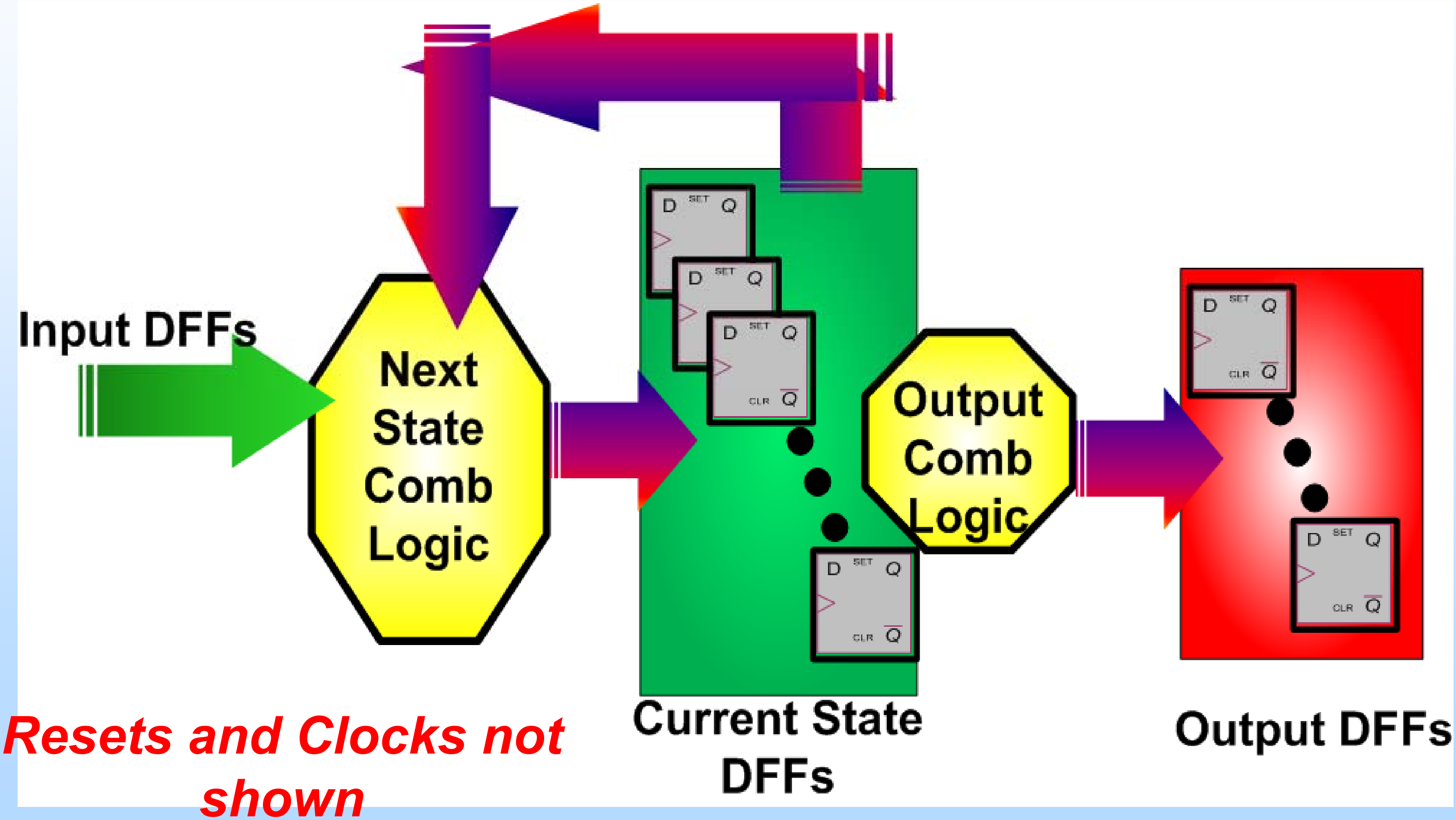
CMOS Microprocessors in a Proton Rich Radiation Environment

- Xilinx embedded processor has not proven to be highly susceptible to protons
  - International Space Station (ISS) experiments such as Space cube
  - Embedded processor uses minimal amount of configuration bits
- Xilinx user designed processor (or soft core) will be highly susceptible in a proton environment
  - User designed processor will use a significant number of configuration bits
  - Virtex configuration bits are highly susceptible to protons
- Microprocessors designed into Microsemi FPGAs will have low susceptibility to protons

SRAM Configuration: Susceptible to Protons

CMOS functional Data Path: Low susceptibility to Protons

State Machines Drive Most Synchronous Designs: Basic State Machine

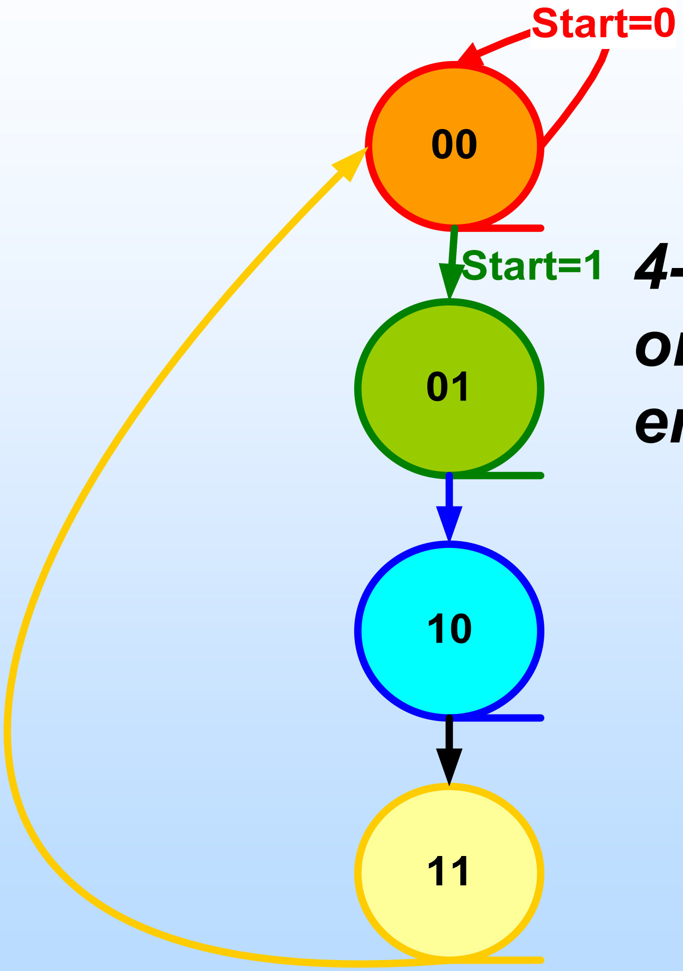


Resets and Clocks not shown

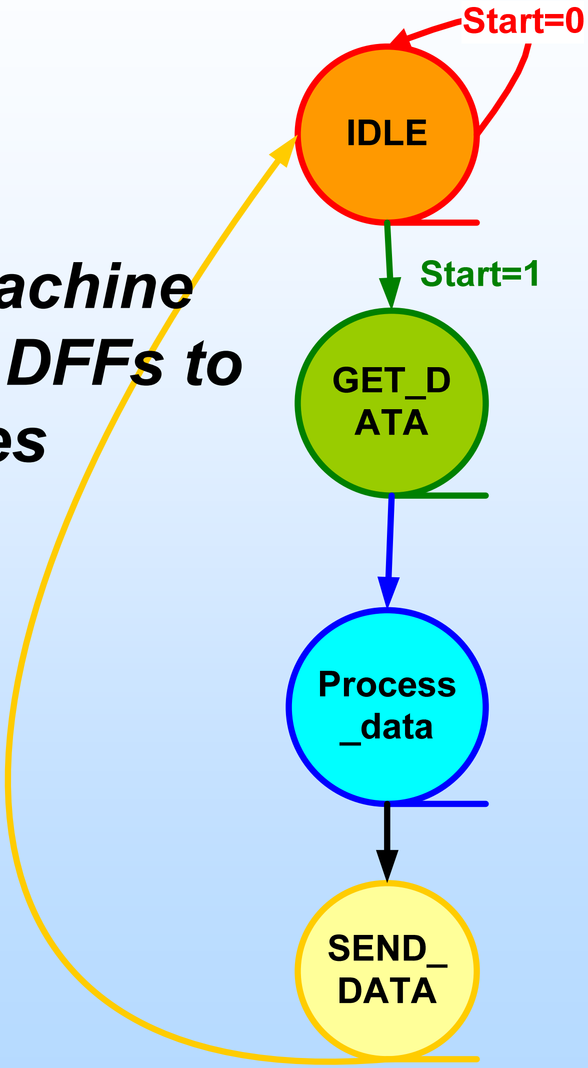
Hardware Fail-Safe Concepts: Locked-up State Machine versus A Locked-up System

- A great deal of attention is given to fail-safe state machines
- Very little attention is given to consequence of implementation and system recovery
- First we'll discuss potential state machine lock-up conditions
- We will follow with system response and fail-safe considerations

State Machine Diagram… Binary Encoding

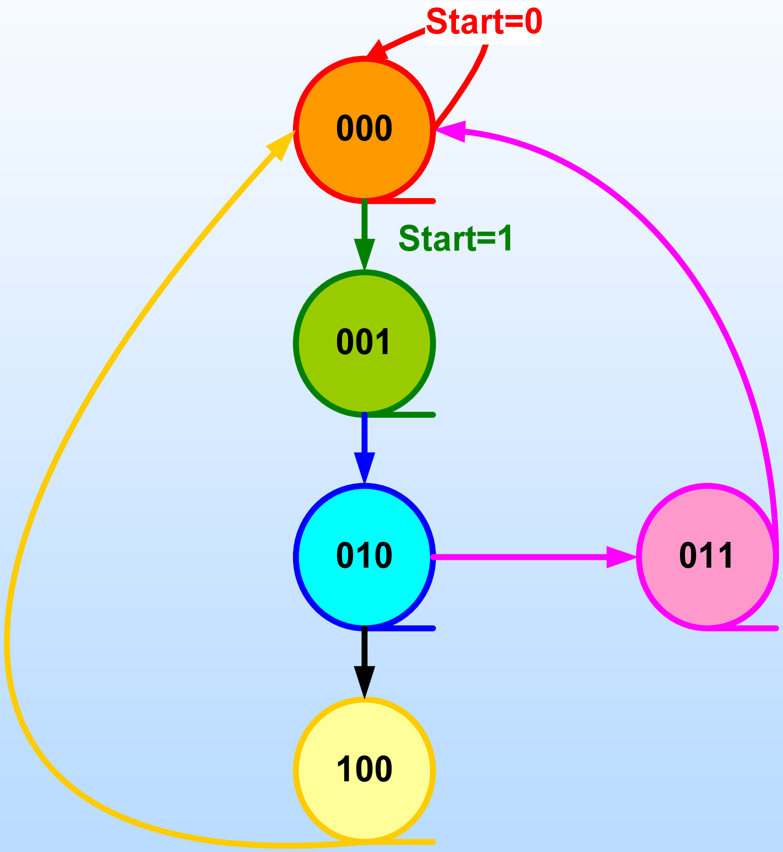


4-State State Machine only requires 2 DFFs to encode all states



All possible states of 2 bits are encoded

5 State-State Machine Requires 3 bits… Some states not assigned (State:101 State:110 State:111)



What happens when a bit flips?

0 1 0 ⇨ 0 1 1

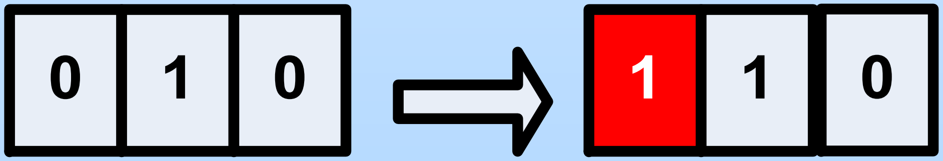
Flip into mapped state

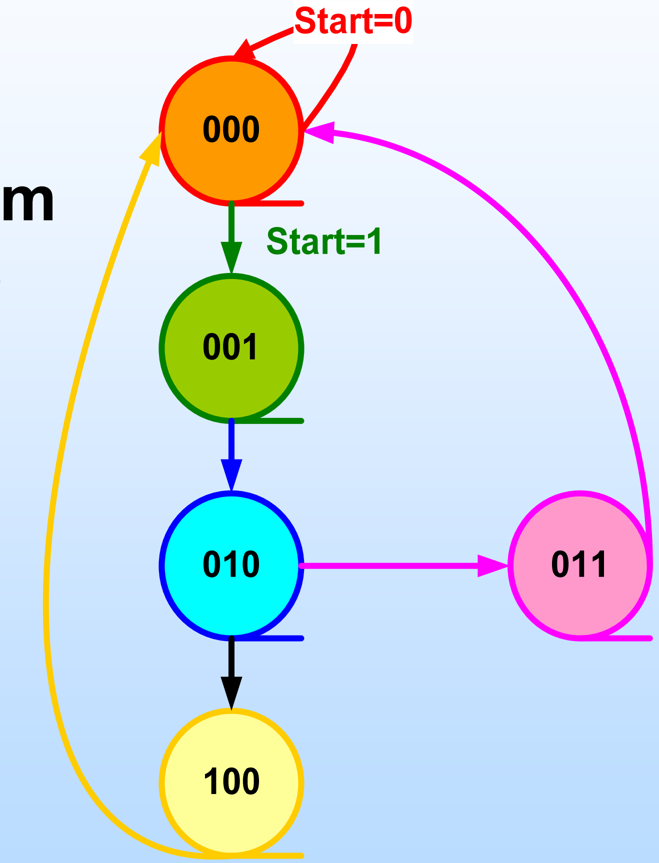
0 1 0 ⇨ 1 1 0

Flip into unmapped state

Commonly Used Definition of A Safe State Machine

- If a bit flips into an unmapped state, circuit automatically jumps to idle state
- Origin of a safe state machine is from a designer who suggests not to use resets on state machines… needs a scheme to get back to IDLE
- Warning… No resets on state machines is a violation of synchronous design rules



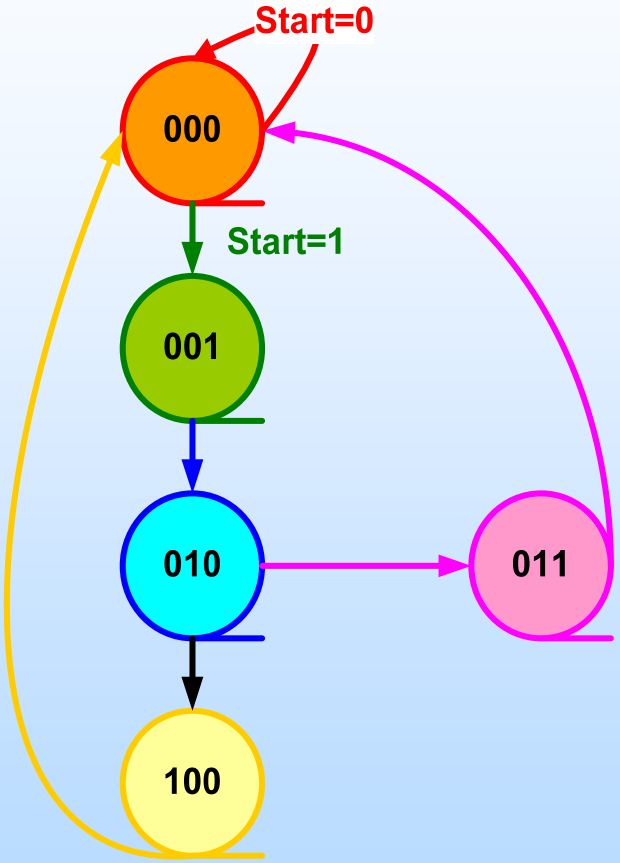


Unused

State:101 State:110 State:111

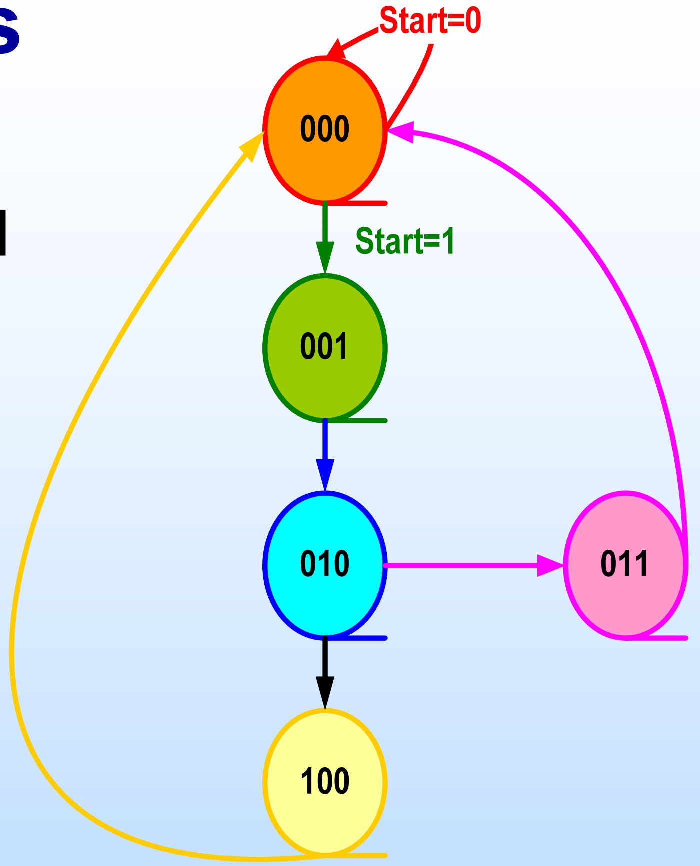
Safe State Machine Concerns

- This is a detection/recovery scheme,
  - there is no redundancy/mitigation and no correction
  - SEU Rate is increased
- Does not account for the state machine jumping into a mapped state
- Gives a false sense of "safety"



Binary Coded States: Jumping into Mapped States

- It is a much higher probability that the machine will jump into a mapped state than unmapped (binary coded)
- In a binary state machine, the complete state is changed, can be dangerous to jump into a mapped state
- During a design review… a designer is responsible to be aware of all possible upsets and the functional response
  - When the designer is asked what do you do if you jump into a mapped state
  - His answer is, use a reset



Unused

State:101 State:110 State:111

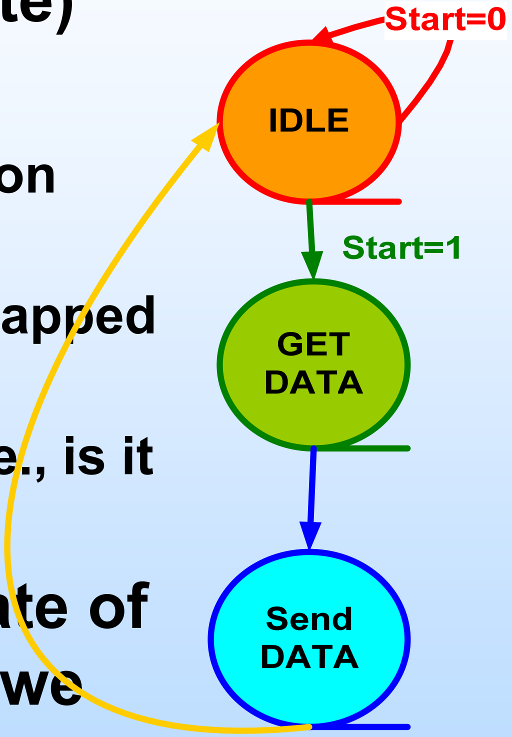
To be presented by Melanie Berg at the Revolutionary Electronics in Space (ReSpace) / Military and Aerospace Programmable Logic Devices (MAPLD) 2011 Conference, Albuquerque, NM, August 22-25, 2011, and to be published on nepp.nasa.gov web site

Binary Coded States: Jumping into Unmapped States

- It is a much lower probability that the machine will jump into an unmapped state
- However, a major amount of logic is added… why?
  - User is worried about getting locked into an unmapped state
  - However, same can be possible for jumping into a mapped state (except for very simple cyclic state machines)
- When the designer is asked what do you do if you jump into an unmapped state
  - His answer is, a safe state machine
  - Why not just use the same reset logic as if jumping into a mapped state? Keep it simple!!!!!!

When and How are “Safe State” Machines a Viable Option?

- For very simple state machines (cyclic without needing input stimulus to push to next state)
- Will need to verify:
  - No lockup conditions exist due to communication with other logic
  - No crucial events if state machine jumps into mapped state
  - No crucial events occur if jump to IDLE state (i.e., is it OK for the output to abruptly turn off)
- Not the safest solution because the full state of the system may not be deterministic (why we use resets)
- However, may reduce the need for soft resets
- When M. Berg is not your design reviewer!



Have You Looked at Your Synthesis Output!

- **Many designers are opting for cycling to IDLE instead of a direct assign to IDLE**
- **This option only works for:**
  - **When the state machine has jumped into an unmapped state**
  - **When the state machine has been synthesized as a binary encoded machine**
    - **Binary state machines synthesize into circuits that have counter-like control (cycle through all counter states)**
    - **One-hot state machines synthesize into shift register type logic… cycling concept doesn't apply**
    - **Don't assume encoding… you must check!**

Critical Paths

- We discussed fail-safe state machine concepts
- Don't forget that if a path is truly critical, then the designer will need to consider more than a pseudo safe state machine:
  - Should additional mitigation be inserted?
  - Are the inputs to the machine and other logic protected?
  - Can multiple outputs turn on at once
  - Can outputs turn off or on too soon
  - Lock up conditions
  - Other logic that is left in a particular state and not cleared

There is an entire system to consider – Resets can be your safest option

Fail Safe Memory Control

- Memory elements can be your most susceptible portion of your design
- There are various methods of protection:
  - None: data is not stored long enough to worry about it
  - Error Detection and correction (EDAC)
    - Pay attention to MBUs and ability of EDAC
    - How susceptible is the EDAC circuitry?
  - Scrubbing
    - Scrubbers should be hardened – can write a lot of bad information otherwise
    - Use of goldens instead of EDAC helps
  - TMR with read/write/modify cycles
- Be careful with FIFOs – if their address pointer become upset, you can lose all of your stored data

Watchdogs

- Monitors portions of circuit (error detection)
- Which portion do you monitor?
  - Common to monitor a heart beat
  - Generally does not give enough information regarding the state of the entire design
- How dependable is the watchdog?
- Where is it performed?
- Too often an after thought and not carefully evaluated during design reviews

Not Easy!!!!!!!!!

Recovery

- What happens if an error is detected?
- Usually an afterthought
- Sometimes recovery scheme is over-designed – too complex… KEEP IT SIMPLE
- Sometimes a reset is your best bet
- A list of all known error events should be made with a correspondent list of recovery modes
  - Sometimes not all error modes are known
  - The benefit of having a verification team (they think of how to break your design instead of how to create your design)
- Recovery should be verified (not easy)

Conclusion

- Understand the device's error signatures and upset rates before mitigation is implemented
- Slowing down the frequency does not necessarily mean you are reducing your SEU susceptibility
- Not all designs are critical and may not need mitigation
- Be aware when correction is necessary:
  - Make sure you are correcting your state
  - Masking without correction can incur error accumulation and eventually break

Conclusion

- Detection circuits don't generally have redundancy and can be susceptible – make sure they are not making your design more susceptible (e.g. "safe" state machines)
  - Perform trade … what is the detection circuitry buying the system
- Perform proper trade studies to determine the type of mitigation necessary to meet requirements:
  - Upset rates
  - Area+Power
  - Complexity… completion and verification within time specified
- Keep it simple – verification is the next and final step to a fail-safe system