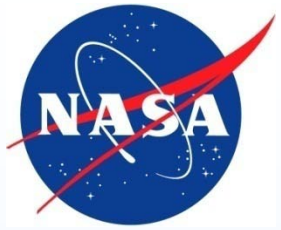
NEPP Electronic Technology Workshop
June 28-30, 2011

National Aeronautics
and Space Administration



Taming the SEU Beast - Approaches and Results for FPGA Devices and How To Apply Them

Melanie Berg, MEI Technologies/NASA GSFC

M. Friendlich, C. Perez, H. Kim: MEI Technologies/NASA GSFC

K. LaBel: NASA GSFC

To be presented by Melanie Berg at the NASA Electronic Parts and Packaging (NEPP) Program Electronic Technology Workshop, Greenbelt, Maryland, June 28-30, 2011, and published on nepp.nasa.gov.

Introduction

- **Field Programmable Gate Array(FPGA) Single Event Effect (SEE) Models have been developed by NASA/GSFC Radiation Effects and Analysis Group (REAG)**
  - **Compartmentalize SEEs to enhance analysis**
  - **Uses a top-down approach**
- **Details of SEE generation and other electrical properties are part of ongoing development**
- **Presented models are only expected to fit synchronous designs as per NASA design guidelines.**

Goal:

- **Application of the NASA REAG FPGA Single Event Upset (SEU) Cross Section (σ_{SEU}) Model to a variety of FPGA types**

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

$$P(fs)_{error} \propto P_{Configuraton} + P(fs)_{functionaLogic} + P_{SEFI}$$

Probability for Design Specific system SEE:

Probability for Configuration SEE

Probability for Functional logic SEE

Probability for Single Event functional Interrupt

Impact to Community

- **Provides a standard method for comparing various types of FPGAs**
- **Enhances analysis by providing a means for evaluating Single Event Upsets (SEUs) and Single Event Transients (SETs):**
  - **Generation**
  - **System Propagation**
  - **Evaluate effectiveness of mitigation strategies**
  - **Determine dominant SEE components**
  - **Eases the overall analysis process**
- **Analysis provides designers with dominant or insignificant susceptible components… enhanced design for radiation strategies**

Technical Highlights

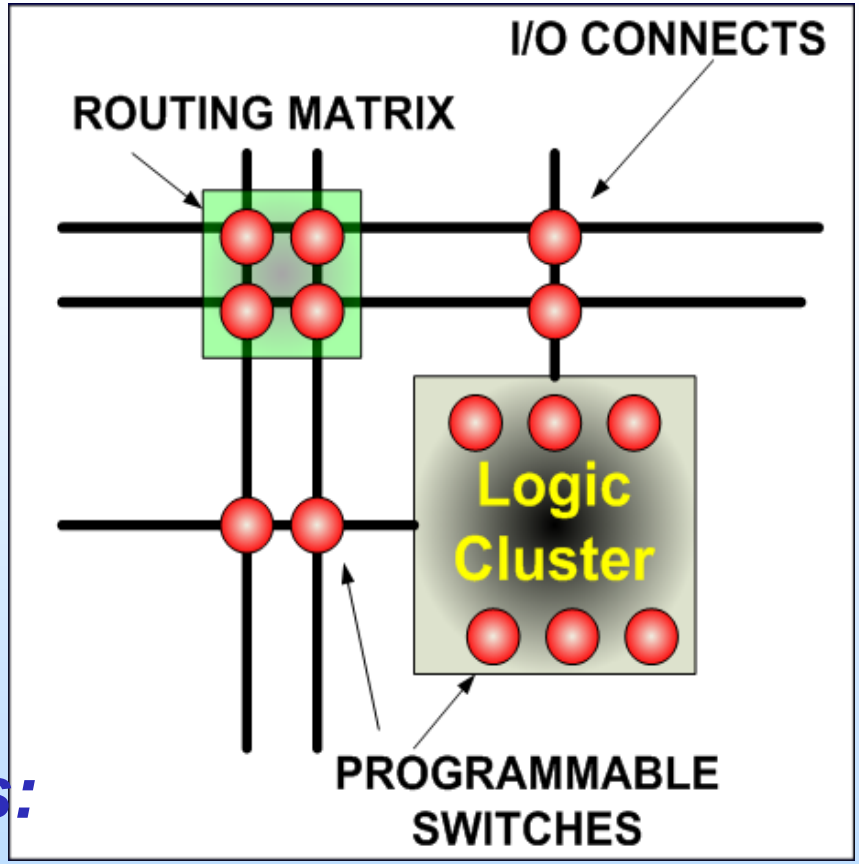
- **This presentation will focus on:**
  - **Configuration:** $P_{configuration}$
  - **Data Path functional logic:** $P_{functionalLogic}$
- **Model Derivation is presented**
- **Model application to Microsemi FPGAs:**
  - **RTAXs: Embedded Radiation Hardened by Design (RHBD)**
  - **ProASIC3: No embedded mitigation**

$$P(fs)_{error} \propto P_{Configuraton} + P(fs)_{functionaLogic} + P_{SEFI}$$

Configuration

$$P_{Configuraton}$$

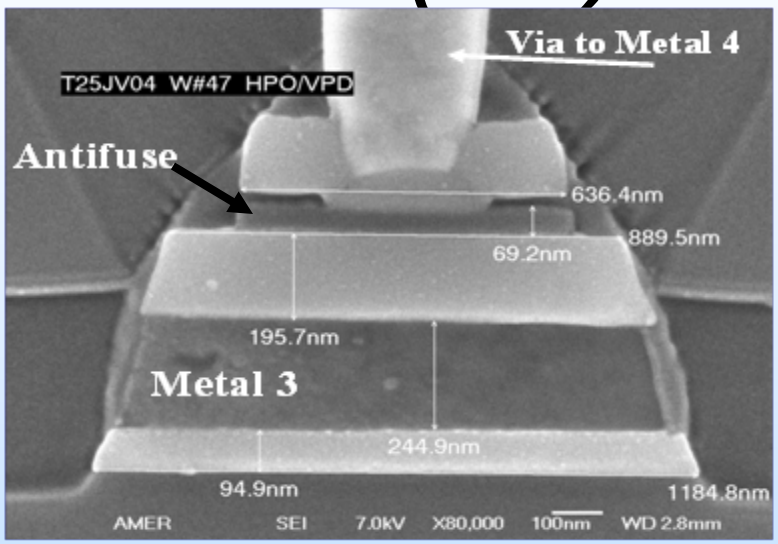
Place, Route, and Gate Utilization are Stored in the FPGA Configuration



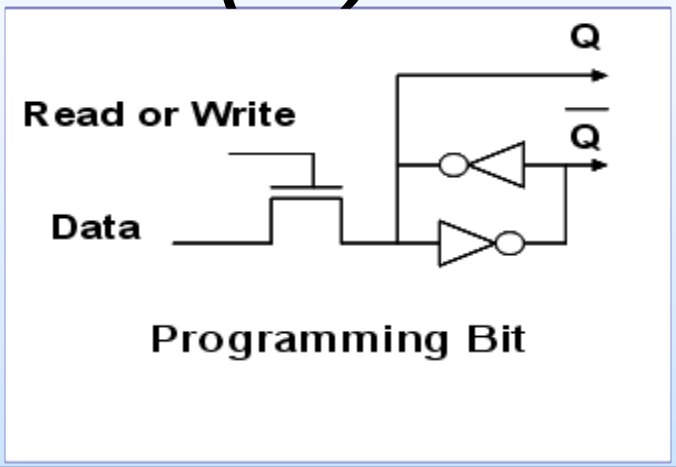
- ***Configuration Defines:***
  Arrangement of pre-existing logic via programmable switches
  - ***Functionality (logic cluster)***
  - ***Connectivity (routes)***
  - ***Placement***
- ***Programming Switch Types:***
  - ***Antifuse:*** ***One time Programmable (OTP)***
  - ***SRAM:*** ***Reprogrammable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

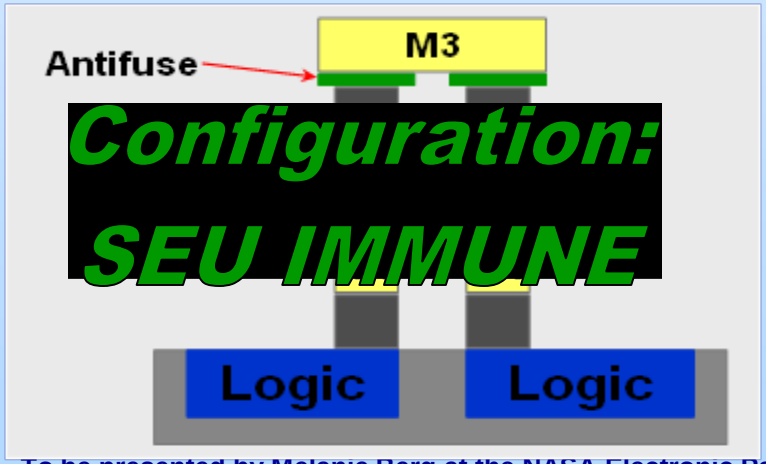
Programmable Switch Implementation and Single Event Upset (SEU) Susceptibility

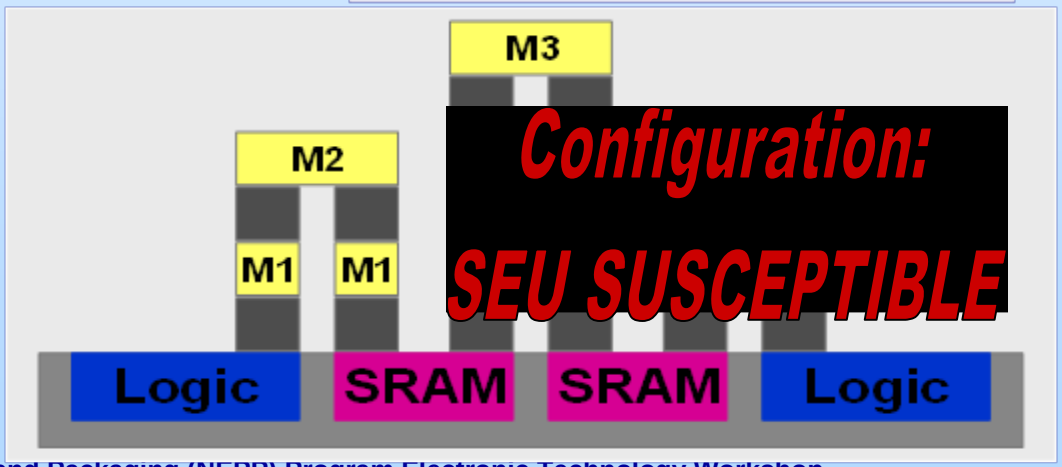
ANTIFUSE (OTP)



SRAM (RP)







Configuration Test and Analysis

- **Configuration is static during operation… hence we test and evaluate it statically**

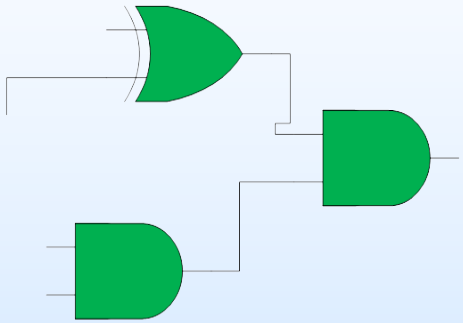
	Antifuse	SRAM	FLASH	SEU Hardened SRAM
Manufacturer	Microsemi Aeroflex	Xilinx Achronix	Microsemi	Xilinx, Acrhonix Atmel
Upset Signature	Fuse Resistivity	Bit State	Bit State or resistivity	Bit State
Configuration Test	Non-Specific	Read-back post-irradiation	Verify Post-irradiation	Read-back post-irradiation
Information from Configuration Test	N/A	Upset configuration bits	Pass/Fail	Upset configuration bits
Results	No upsets observed	Dominant upsets	Insignificant	Low significance
REAG Tested	Yes	Yes	Yes	Atmel, AchronixYes/ Xilinx No

Impact of Configuration Testing and Analysis to the REAG Model

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

Data Path Functional Logic and Concepts of Synchronous Design

Synchronous Design Basic Building Blocks: Combinatorial Logic and Flip-Flops (DFF's)

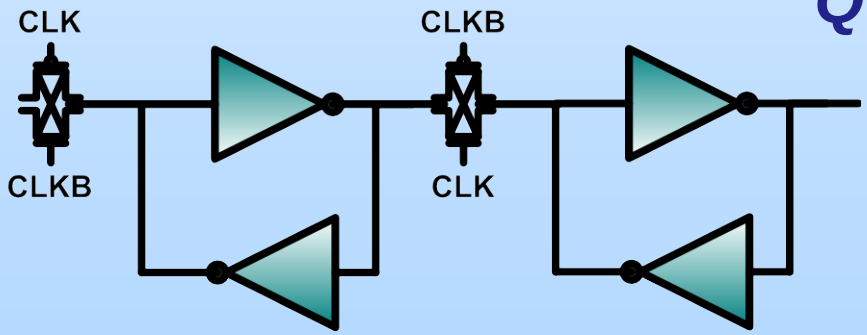


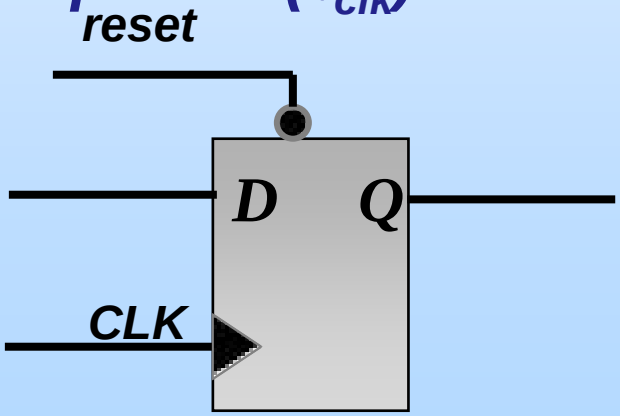
Combinatorial Logic: Output is a function of the inputs after some delay(τ_{dly})

Output=f(input, τ_{dly})

DFF: Captures data input at clock edge and is a function of the clock period (τ_{clk})

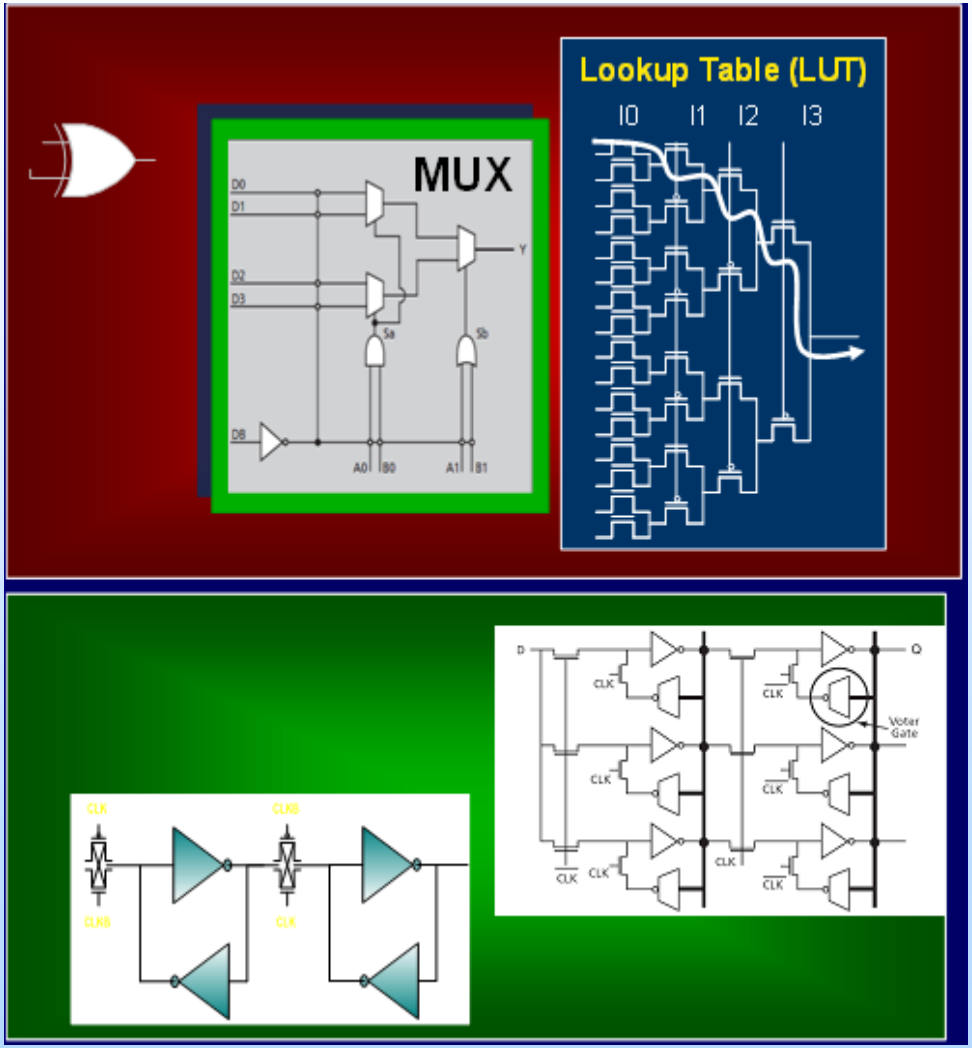
Q=f(D, τ_{clk})





Component Libraries: Basic Designer Building Blocks

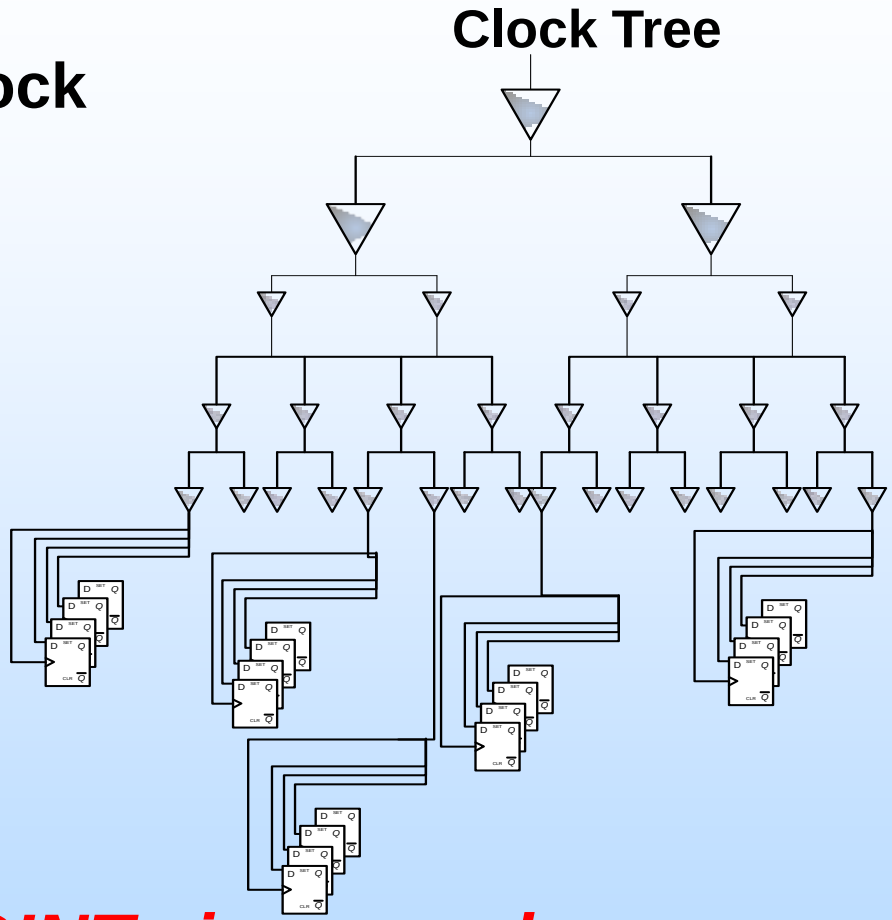
- **Combinatorial logic blocks**
  - ❑ Vary in complexity
  - ❑ Vary in I/O
- **Sequential Memory blocks (Flip-flops or DFFs)**
  - ❑ Uses global Clocks
  - ❑ Uses global Resets
  - ❑ May have mitigation



DFF's in a Synchronous Design

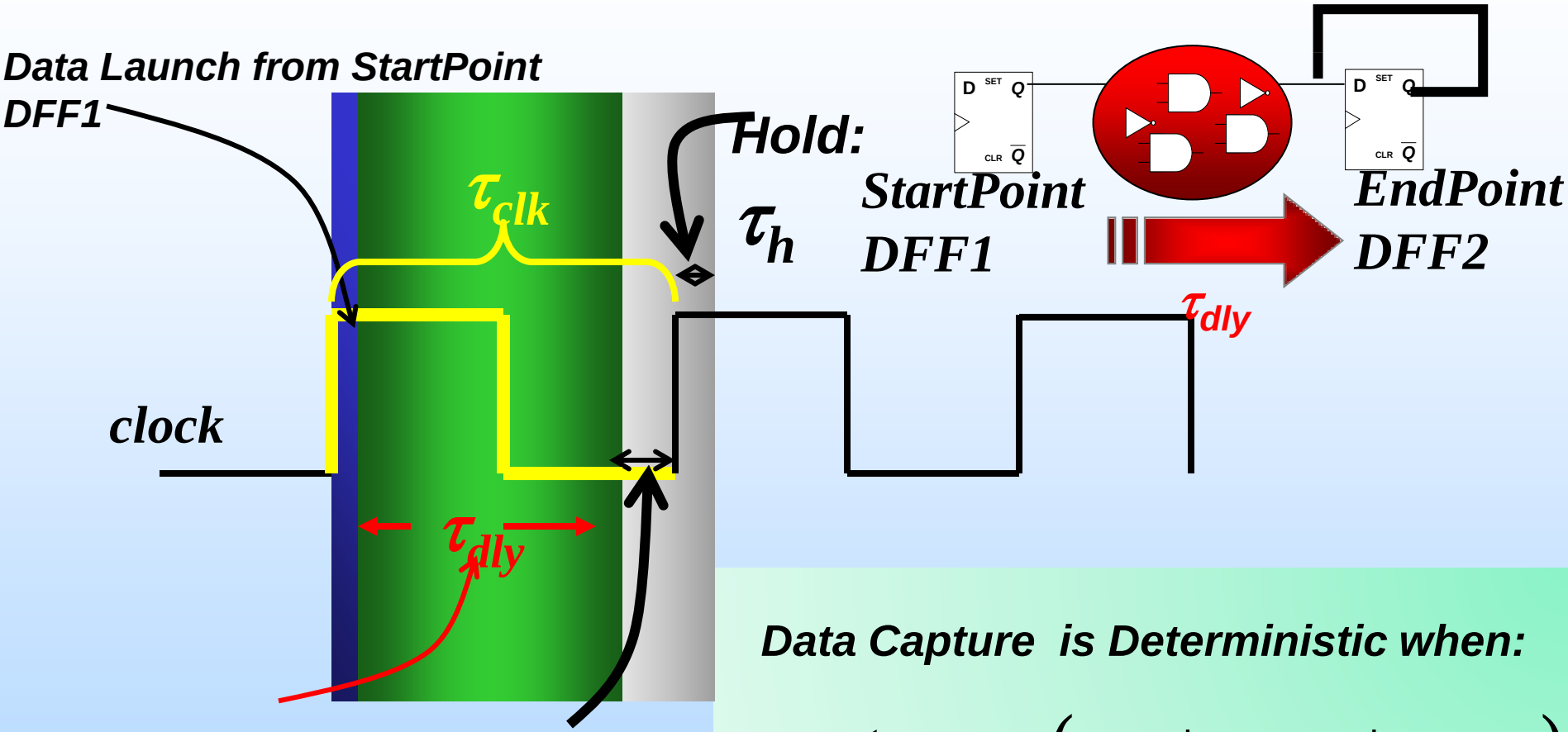
- All DFFs are connected to a clock
- Clock period: τ_{clk}
- Clock frequency: f_s

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



DFFs are BOUNDARY POINTs in a synchronous design

Deterministic Data Capture…Adhering to Setup and Hold Time for a DFF

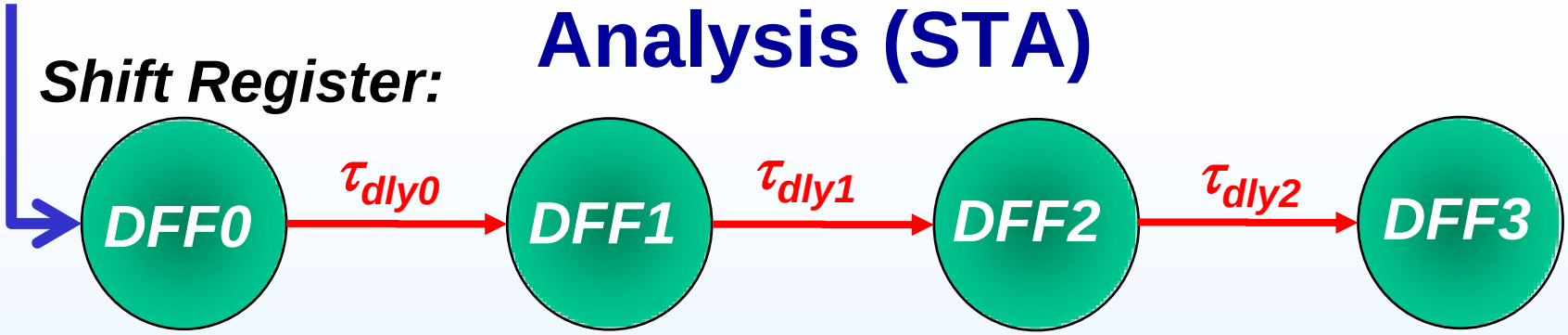


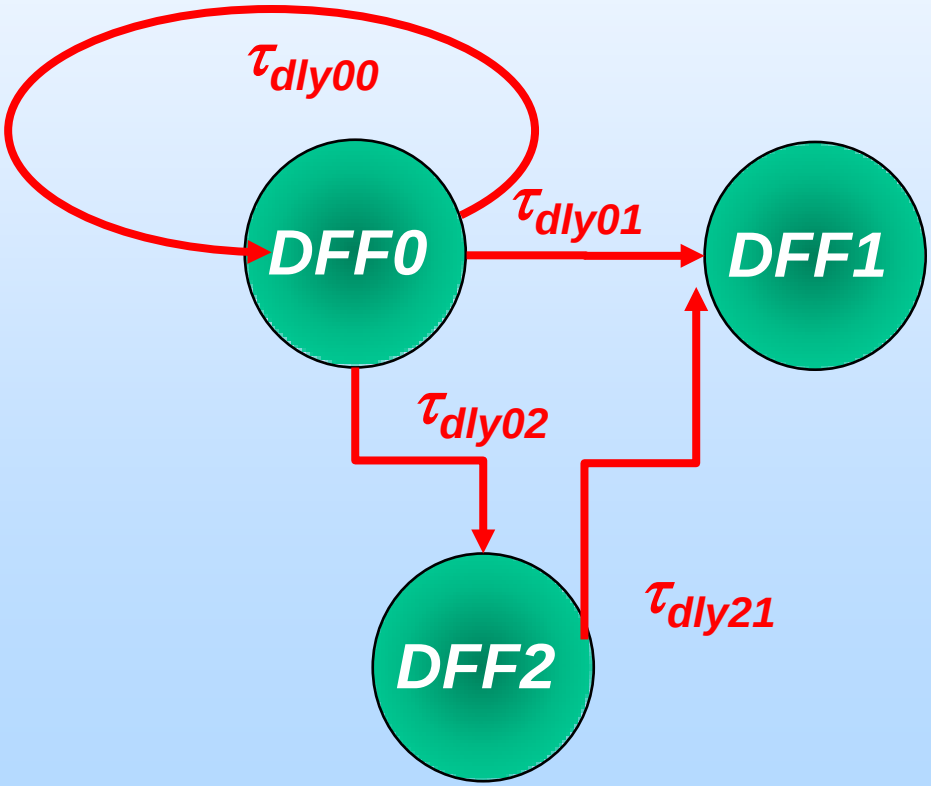
τ_{dly} : Data Delay through combinatorial logic and routes

Data Capture is Deterministic when:

$$\tau_{dly} < \tau_{clk} - (\tau_{su} + \tau_{skew} + \tau_{jitter})$$

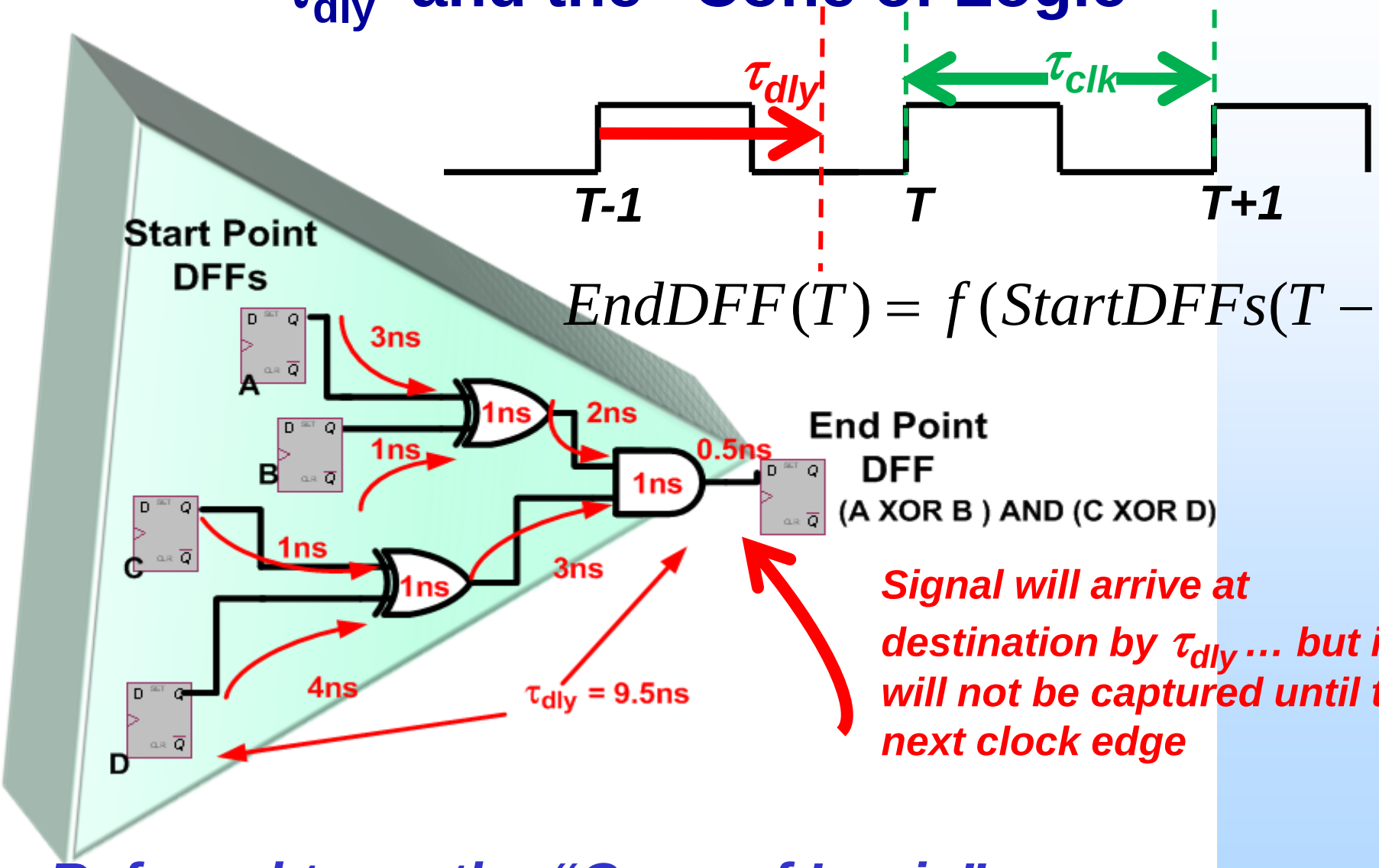
Making Setup Time: Static Timing Analysis (STA)





- *DFFs are boundary points*
- *Timing is performed from one DFF to the next DFF*
  - *For each DFF, data paths are traced backwards to their start-points*
- *Combinatorial logic and routes are part of the delay*

Start Point DFFs → End Point 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

Referred to as the "Cone of Logic"

System States

- System state is defined by the logic values within all DFFs
- In between clock edges (intermediate points)
  - Computations are occurring (combinatorial logic)
  - SETs or SEUs can occur
- System state is captured at each rising clock edge... after clock cycle computations are completed

Note: Upsets occur at intermediate points. They become part of the system state if they are captured into the next state

Synchronous Design Take Away Points

- **Basic Blocks: DFFs and Combinatorial logic**
- **DFFs are boundary points**
  - **For each DFF (end point) there is a backwards trace to start point DFFs**
  - **There is delay between start point DFFs and endpoint DFFs**
    - **Combinatorial logic**
    - **Routes**
- **SEE analysis is based on utilized DFFs in a design because an upset is not an upset unless it is captured by a DFF**

The question is… If an upset occurs will it reach an endpoint DFF?

Data Path Functional Logic

$$P_{functionalLogic}$$

Configuration versus Data Path (Functional Logic) SEE

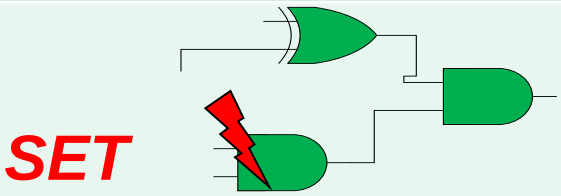
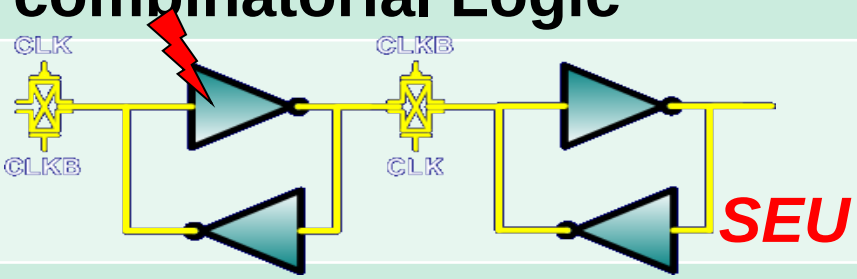
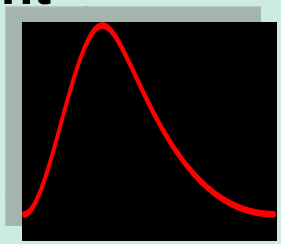
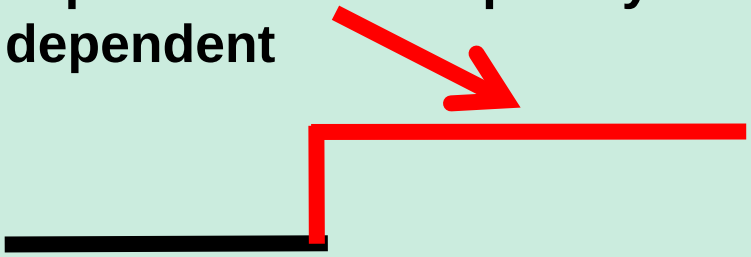
- **Configuration and Functional logic are separate logic**
- **Can be implemented with different technologies within one device (e.g. antifuse versus CMOS)**
- **Configuration is static and data paths are not. Requires a different test and analysis approach**

This explains why there are separate categories of error:

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

SEU and SET Background

Primary functional logic components can be classified into:

Combinatorial	Sequential
Logic function generation (computation)	**Captures and holds state of combinatorial Logic**
	
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** 

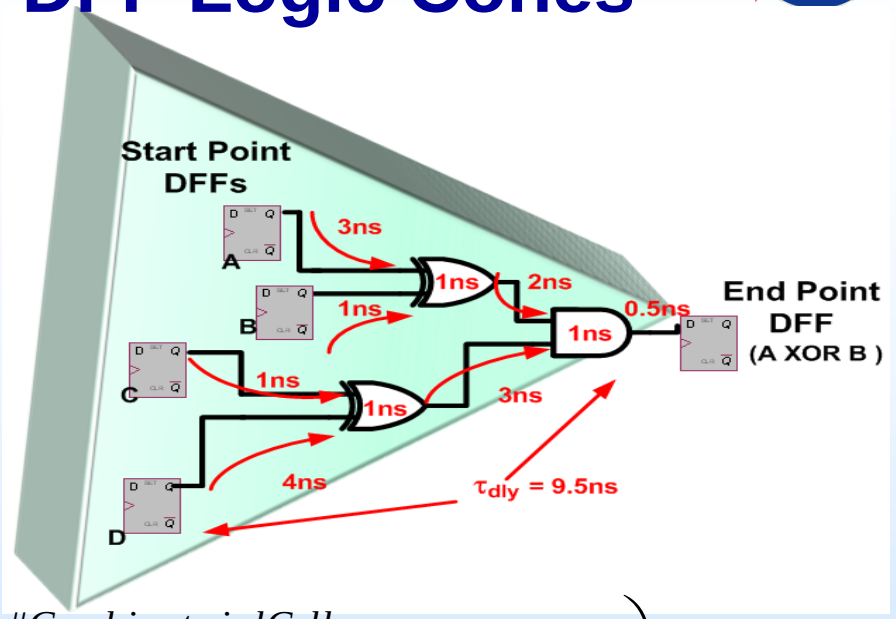
SET effects are nonlinear and are heavily design and state dependent.

Data Path Model and DFF Logic Cones

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

Probability for Functional logic SEE

Evaluate for Each DFF



$$\exists_{DFF}\left(\sum_{j=1}^{\#StartPointDFFs} P(fs)_{DFFSEU\rightarrow SEU(j)} + \sum_{i=1}^{\#CombinatorialCells} P(fs)_{SET\rightarrow SEU(i)} \right)$$

Probability for Captured DFF Events

Probability for Captured Combinatorial logic

DFF_k Cone of Logic

All Start Point DFFs and Combinatorial Logic gates that feed into End Point DFF under Evaluation (DFF_k):

Combinatorial Logic Contribution to System Error in a Synchronous System: Capturing a SET

$$P_{SET \rightarrow SEU}$$

SETs and a Synchronous System

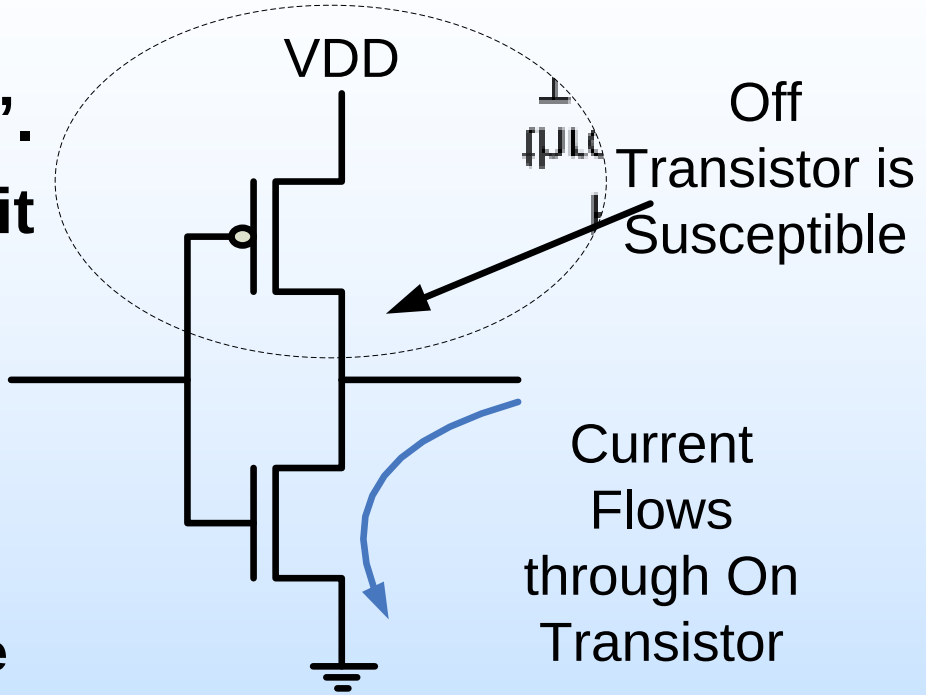
- Generation (P_{gen})
- Propagation (P_{prop})
- Logic Masking ($P_{logic.}$)
- Capture

All Components comprise:

$$P_{SET \rightarrow SEU}$$

SET Generation: P_{gen}

- **SET generation occurs due to an "off" gate turning "on".**
- **A transient in a CMOS circuit will be generated with an amplitude and width (τ_{width}) based on:**
  - **Amount of deposited charge (i.e. small Linear Energy Transfer (LET) values produce small transients)**
  - **The strength of the gate's load**
  - **The strength of its complimentary "ON" gate**
  - **The dissipation strength of the process.**



Collected Charge ***Critical Charge***

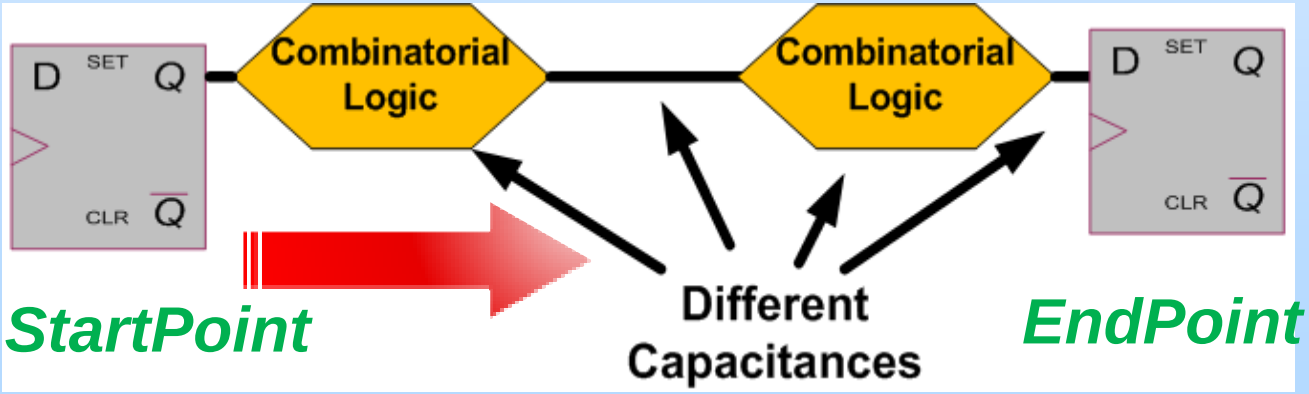
$$Q_{coll} > Qcrit$$

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

Node Capacitance ***Node Voltage***

SET Propagation to an EndPoint DFF: P_{prop}

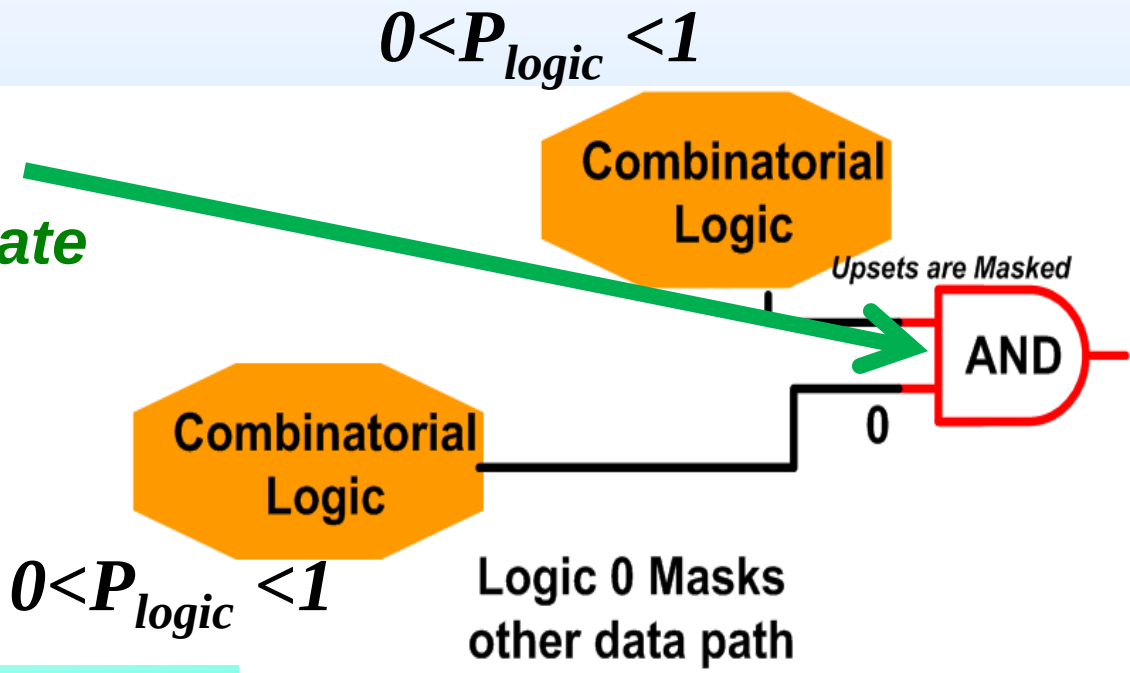
- In order for the data path SET to become an upset, it must propagate and be captured by its endpoint DFF
- P_{prop} only pertains to electrical medium (capacitance of path… combinatorial logic and routing)
  - Capacitive SET amplitude reshaping
  - Capacitive SET width reshaping
  - Small SETs or paths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



SET Logic Masking: P_{logic}

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

"AND" gate reduces probability that SET will logically propagate



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

SET Capture at Destination DFF

Each combinatorial element can generate a transient. The transient width will be a fraction of the clock period for a synchronous design in a CMOS process.

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

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

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

Data Path Model and Combinatorial Logic SETs

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

$$\exists_{DFF}\left(\sum_{j=1}^{\#StartPointDFFs} P(fs)_{DFFSEU\rightarrow SEU(j)} + \sum_{i=1}^{\#CombinatorialCells} P(fs)_{SET\rightarrow SEU(i)}\right)$$

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

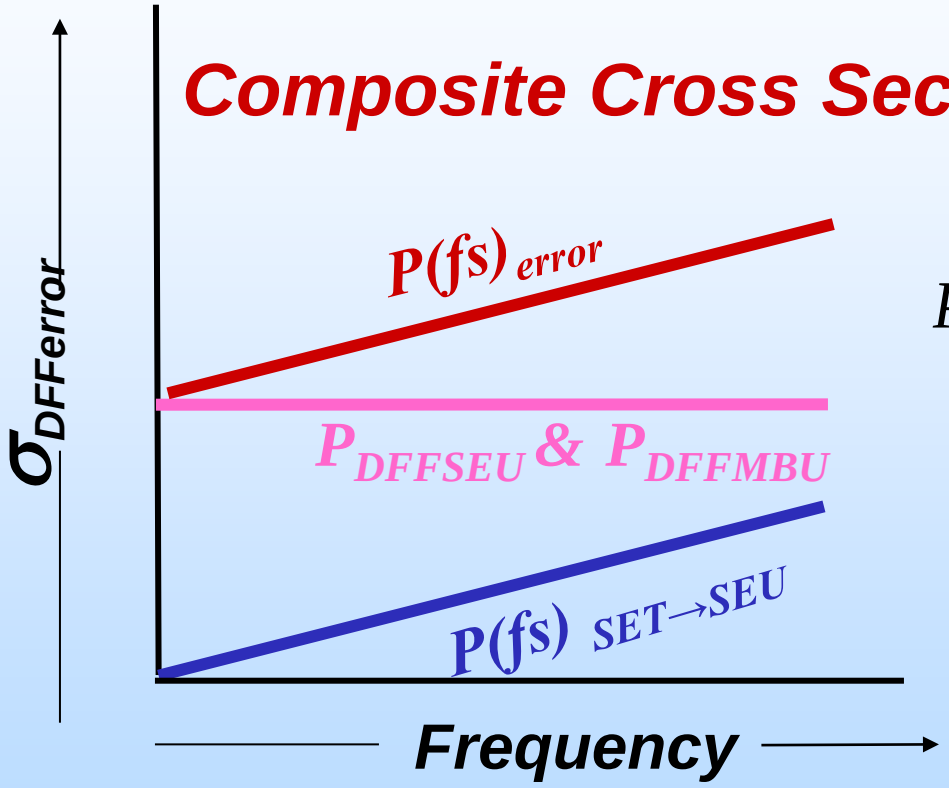
$$\sum_{i=1}^{\#CombinatorialCells} P_{gen(i)} P_{prop(i)} \tau_{width(i)} fs$$

Upper Bound SET $P_{logic}=1$

DFF Contribution to System Error in a Synchronous System

$$P_{DFFSEU \rightarrow SEU}$$

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



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

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

SEUs and a Synchronous System: New Stuff

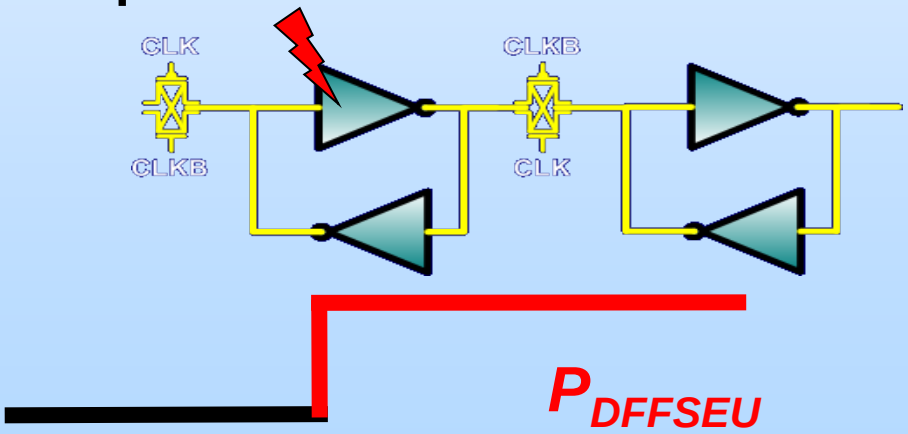
- Generation (P_{DFFSEU})
- P_{prop}=1 for hard state switch
- Logic Masking ($P_{logic.}$)
- Capture

All Components comprise:

$$P_{DFFSEU \rightarrow SEU}$$

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**



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

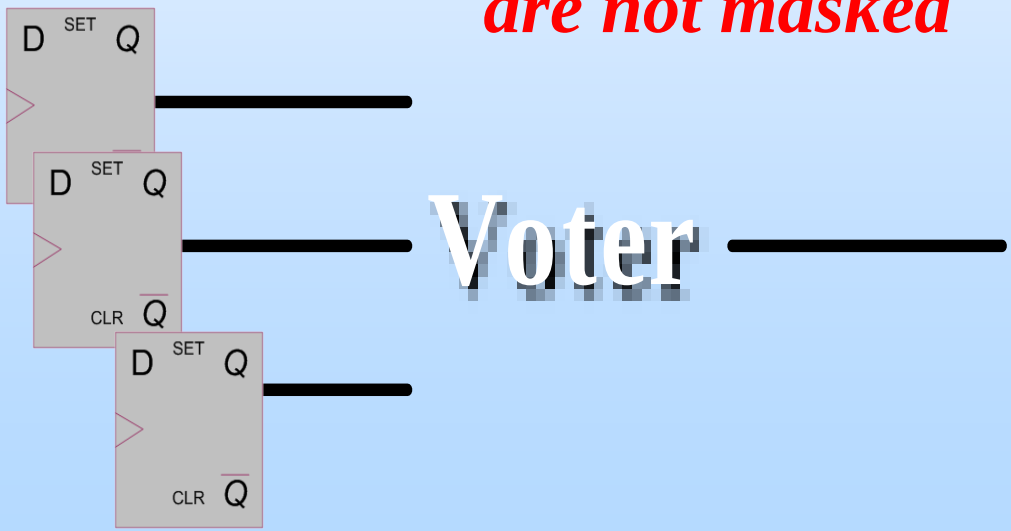
P_{DFFSEU}	$P(fs)_{DFFSEU \rightarrow SEU}$
Probability a Startpoint DFF becomes upset	Probability that the Startpoint upset is captured by the endpoint DFF
Occurs at some point in time within a clock period	Occurs at a clock edge (capture)
Not frequency dependent	Frequency dependent

Logic Masking DFFs… P_{logic}

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

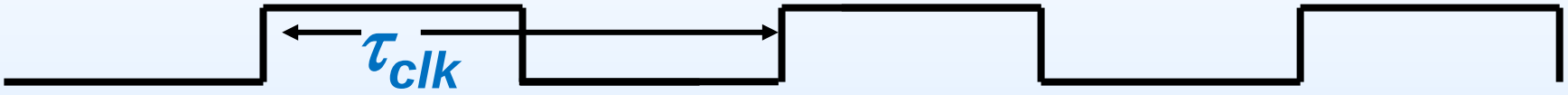
$P_{logic}>0$ *for Voter… its upsets are not masked*

$P_{logic}=0$ *for DFFs… their upsets are masked*

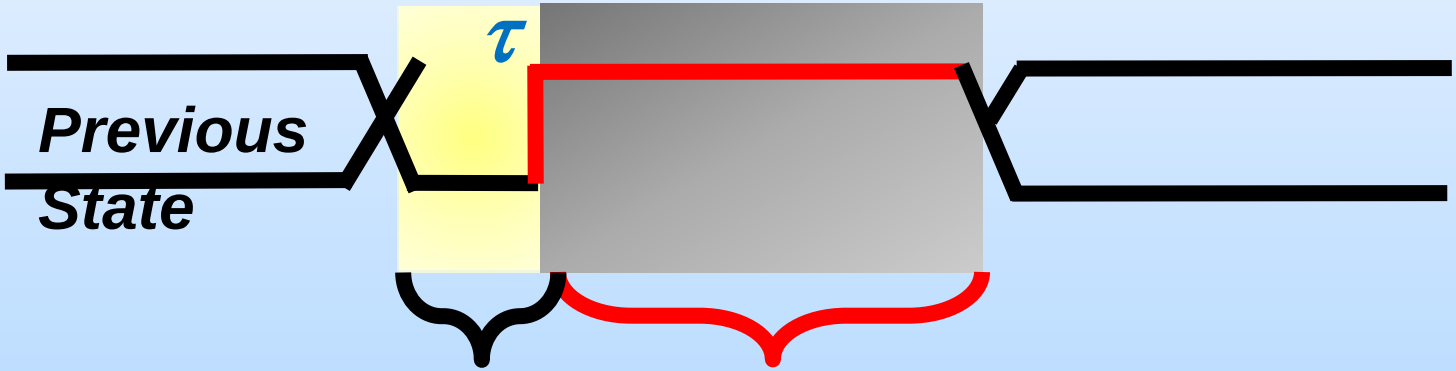


DFF Upsets (SEUs) and Next State Capture: P_{DFFSEU} and $P_{DFFSEU \rightarrow SEU}$

- **If a DFF is affected by an SEU it will change its state somewhere within a clock cycle at time τ**



$0 < \tau < \tau_{clk}$ ***τ Is defined to be within 1 clock period (τ_{clk})***



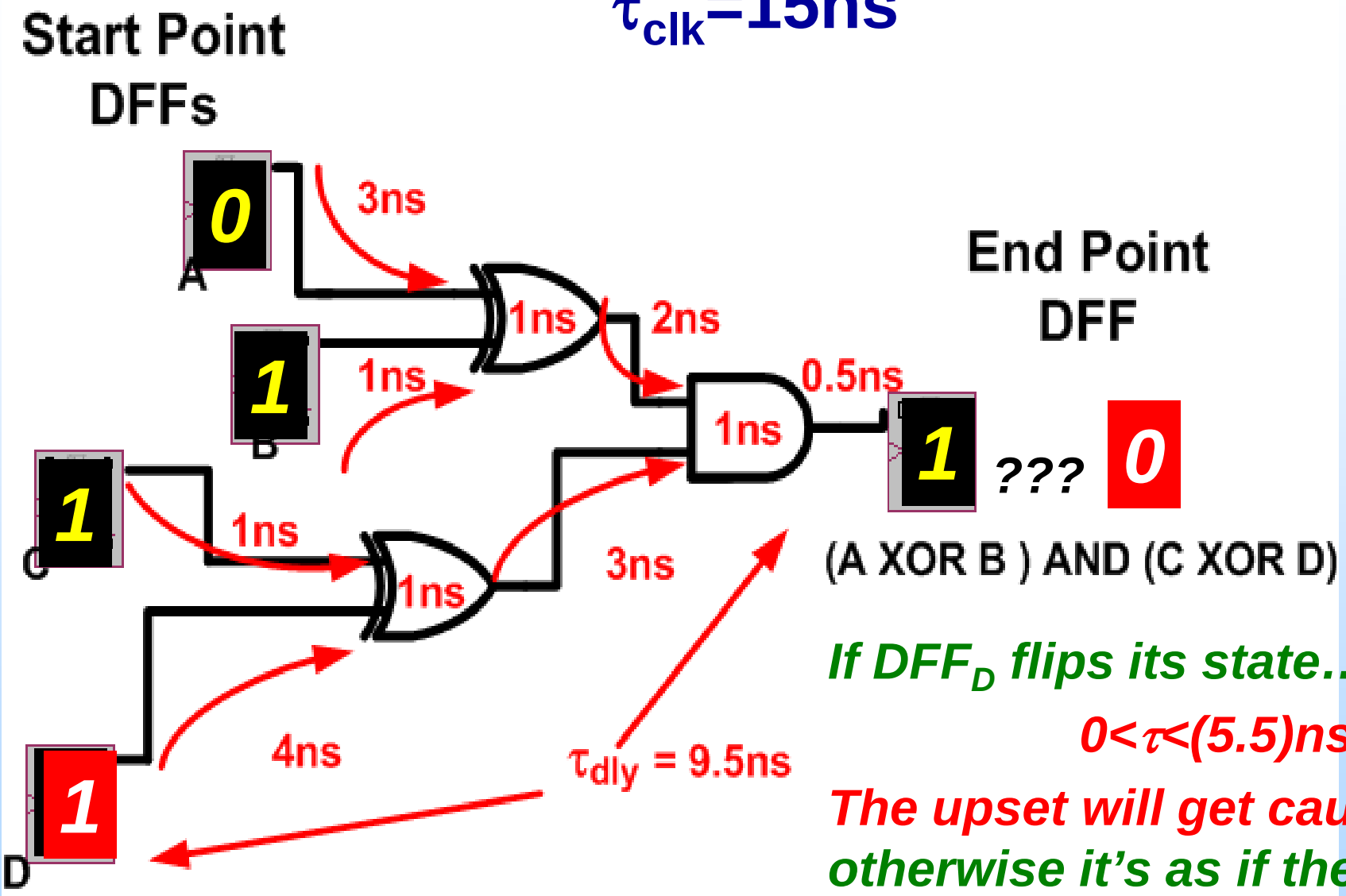
Will the endpoint DFF capture the correct value?

OR…

Will the endpoint DFF capture the upset?

To be presented by Melanie Berg at the NASA Electronic Parts and Packaging (NEPP) Program Electronic Technology Workshop, Greenbelt, Maryland, June 28-30, 2011, and published on nepp.nasa.gov.

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



If DFF_D flips its state…

$0<\tau<(5.5)$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

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

Fraction of clock period for upset capture

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

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

Fraction of clock period for upset capture wrt to frequency

Data Path Upsets and Start Point DFFs

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

$$\exists_{DFF}\left(\sum_{j=1}^{\#StartPointDFFs} P(fs)_{DFFSEU\rightarrow SEU(j)} + \sum_{i=1}^{\#CombinatorialCells} P(fs)_{SET\rightarrow SEU(i)}\right)$$

$$\sum_{j=1}^{\#StartPointDFFs} P_{DFFSEU(j)} P_{\mathrm{logic}(j)} (1-\tau_{dly(j)} fs)$$

Putting it all together:

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

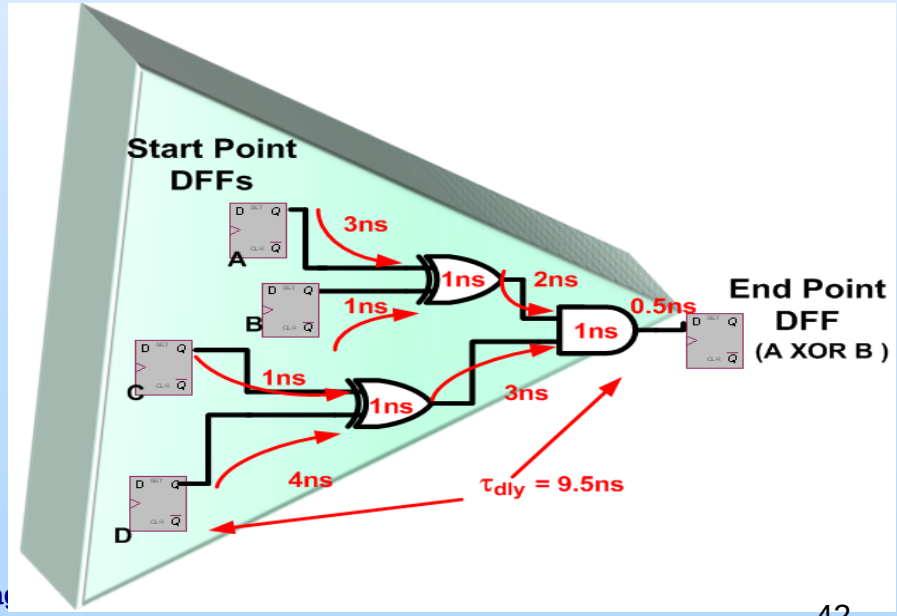
DFF SEU capture ***DFF SEU capture*** ***Combinatorial Logic SET capture***

NASA REAG FPGA Data Path Functional Logic Susceptibility Model

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

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

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



NASA REAG FPGA Upper Bound Susceptibility Model

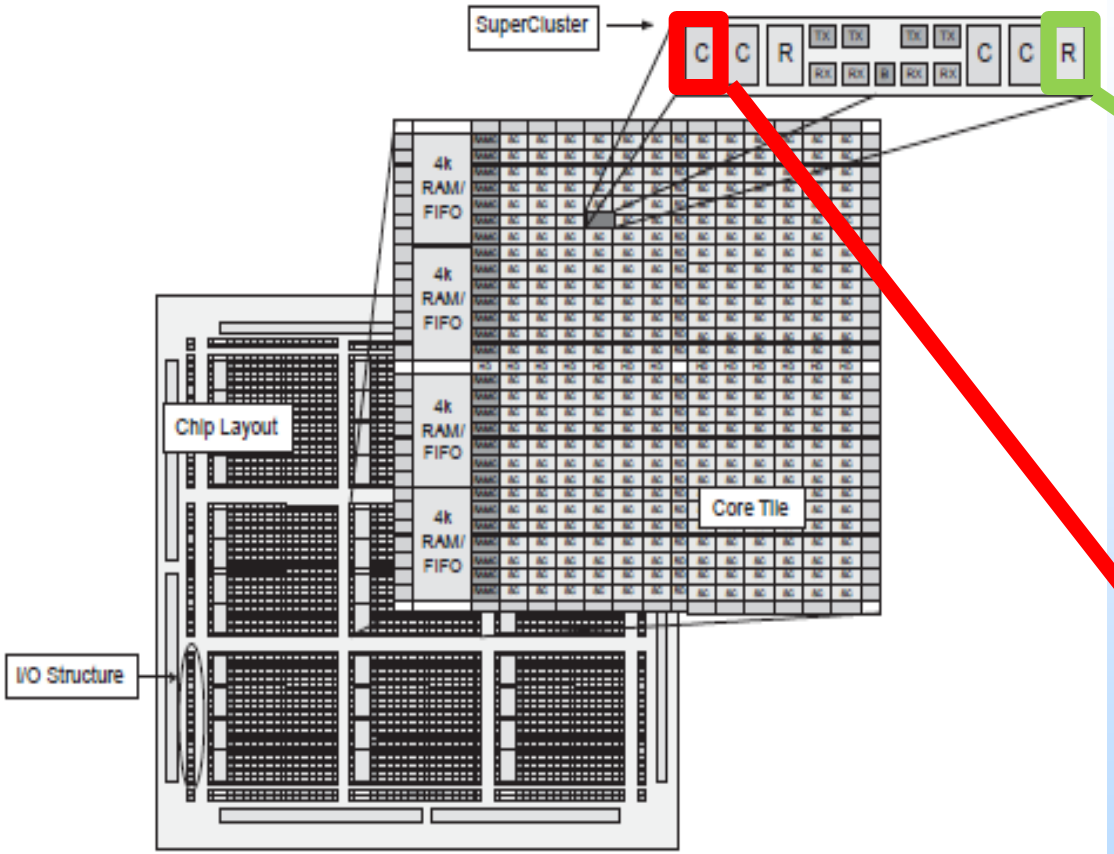
$$\exists_{DFF}\left(\left(\sum_{j=1}^{\#StartPointDFFs} P_{DFFSEU(j)}(1-\tau_{dly(j)}\,fs)P_{\log ic(j)} \right) + \sum_{i=1}^{\#CombinatorialCells} (P_{gen(i)}P_{prop(i)}P_{\log ic}\tau_{width(i)}\,fs) \right)$$

Upper-bound assumes Plogic=1 (no mitigation) and NO DFF frequency (fs) dependency

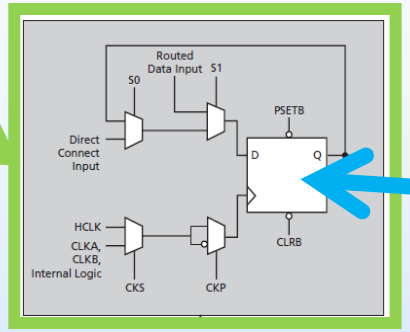
$$\exists_{DFF}\left(\sum_{j=1}^{\#StartPointDFFs} P_{DFFSEU} + \sum_{i=1}^{\#CombinatorialCells} (P_{gen(i)}P_{prop(i)}\tau_{width(i)}\,fs) \right)$$

NASA REAG Models + Heavy Ion Data: RTAXs

RTAXs FPGA Core Logic: Basic Building Blocks are R-CELLs and C-CELLs

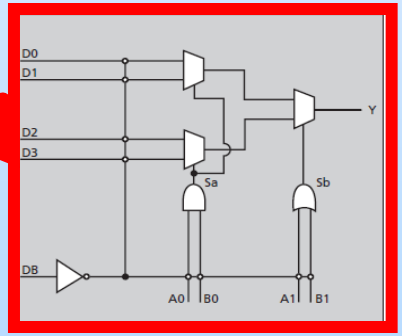


R-CELL: Sequential + Combinatorial



DFF

C-CELL: Combinatorial



Combinatorial Logic is susceptible to Single Event Transients (SETs)

(RCELLs, CCELLs, and buffers are susceptible to SETs)

Model Application to RTAXs: Embedded Localized Triple Modular Redundancy (LTMR)

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

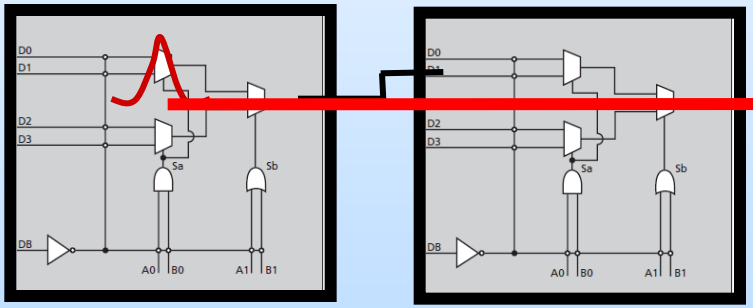
Design Specific SEE upset rate

Configuration SEE upset rate

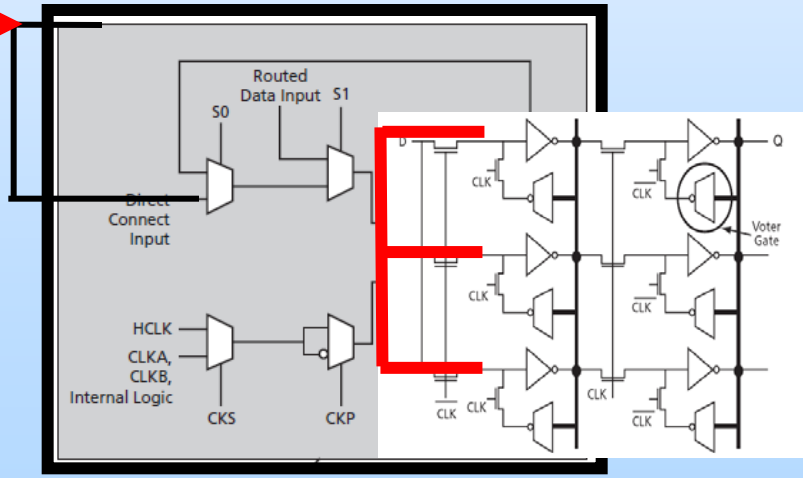
Functional logic SEE upset rate

Single Event functional Interrupt

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

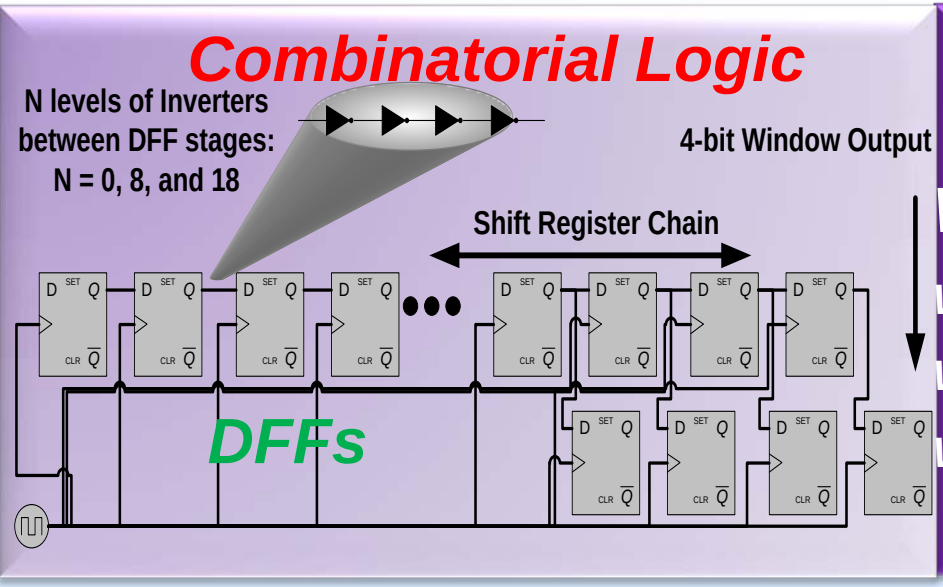


Combinatorial logic cells



(LTMR): DFF

Testing Combinatorial Logic Contributions to SEU Cross-Sections: WSRs with Inverters



Windowed Shift Register (WSR)

WSR_0: N=0 Chain … Only DFFs

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

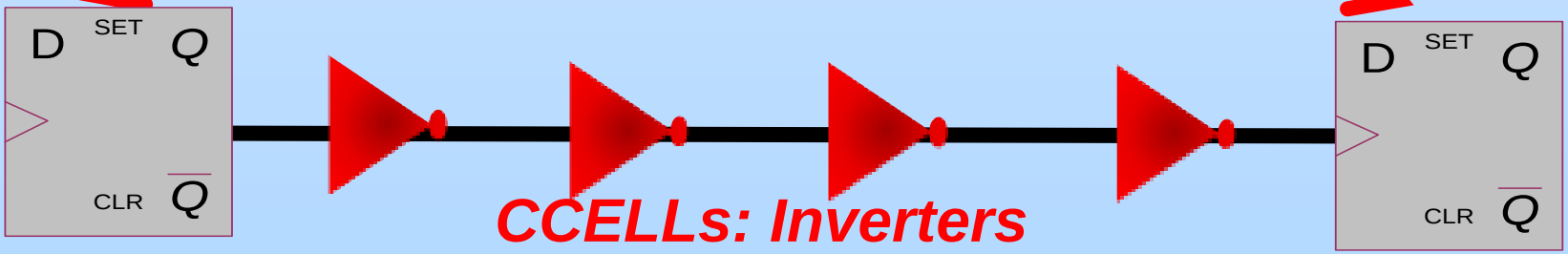
WSR_{16}: N=16 Chain… 16 Inverters per 1 DFF

Think of DFFs as your boundary points… They capture SETs

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

RCELL: Combinatorial

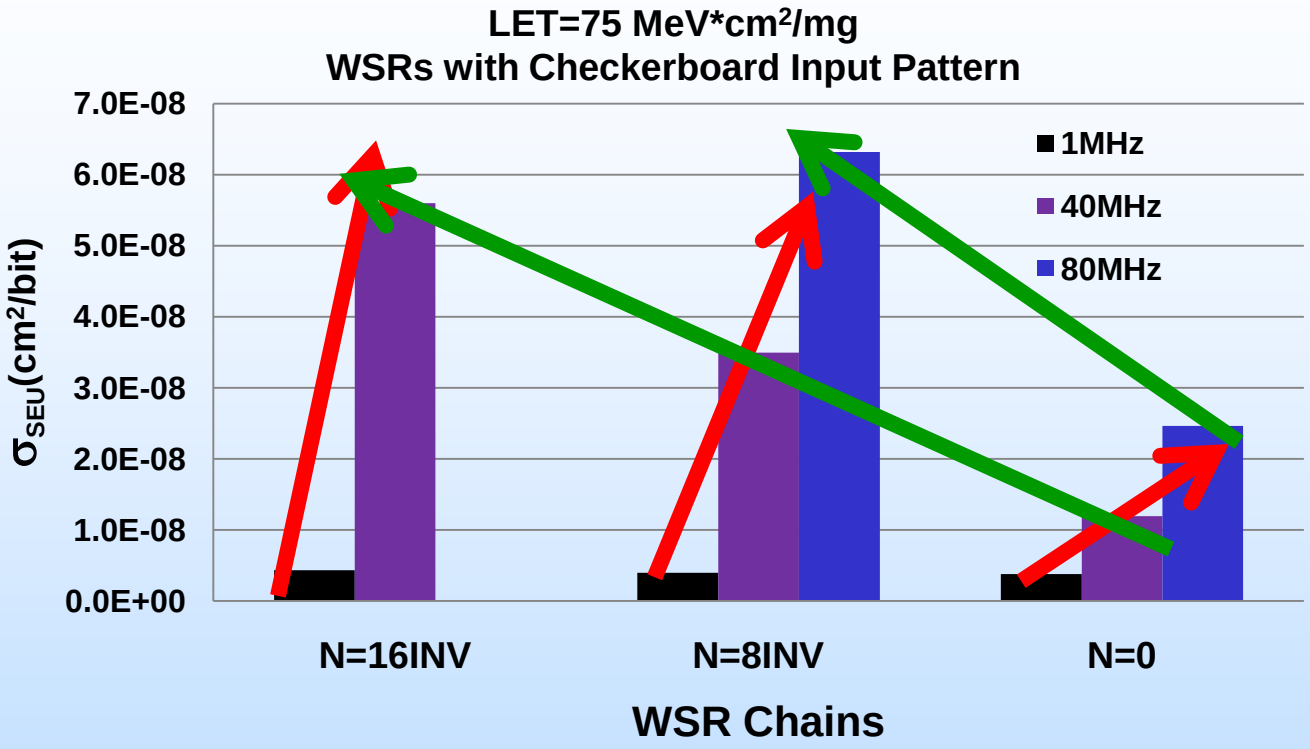
CCELL: Combinatorial



CCELLs: Inverters

To be presented by Melanie Berg at the NASA Electronic Parts and Packaging (NEPP) Program Electronic Technology Workshop, Greenbelt, Maryland, June 28-30, 2011, and published on nepp.nasa.gov.

RTAXs: SET Capture across LET affects WSR SEU Cross Sections



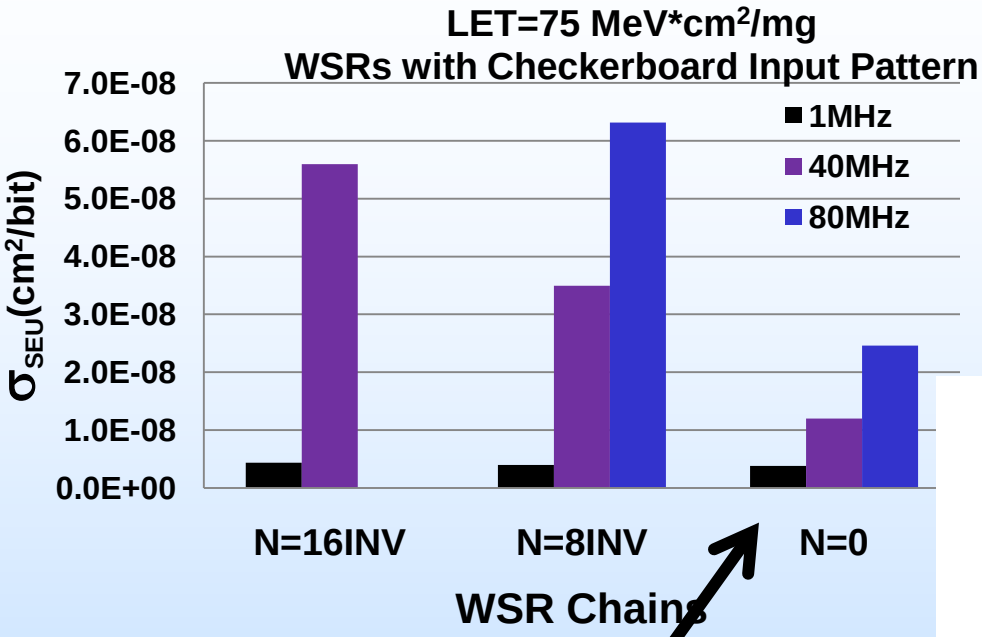
$$\left(\left(\sum_{j=1}^{1} \left(P_{gen(i)} P_{prop(i)} \tau_{width(i)} fs \right) \right) + \sum_{i=1}^{\#CombinatorialCells} \left(P_{gen(i)} P_{prop(i)} \tau_{width(i)} fs \right) \right)$$

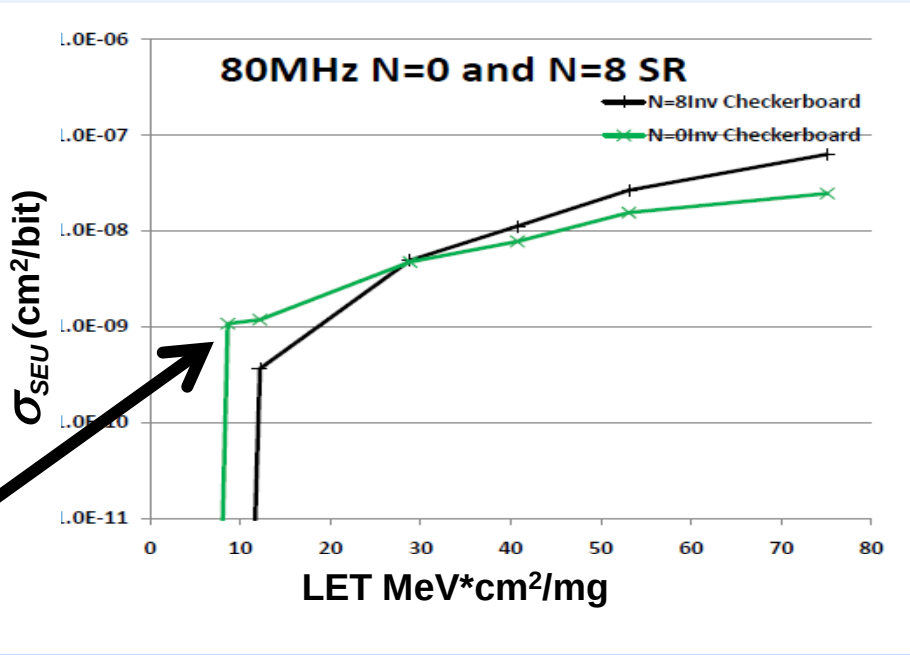
Per WSR stage

Increase Frequency→ Increase Cross section

Increase Combinatorial Logic → Increase Cross section

RTAXs: SET Capture across LET affects SEU Cross Sections (Non-Linearity)





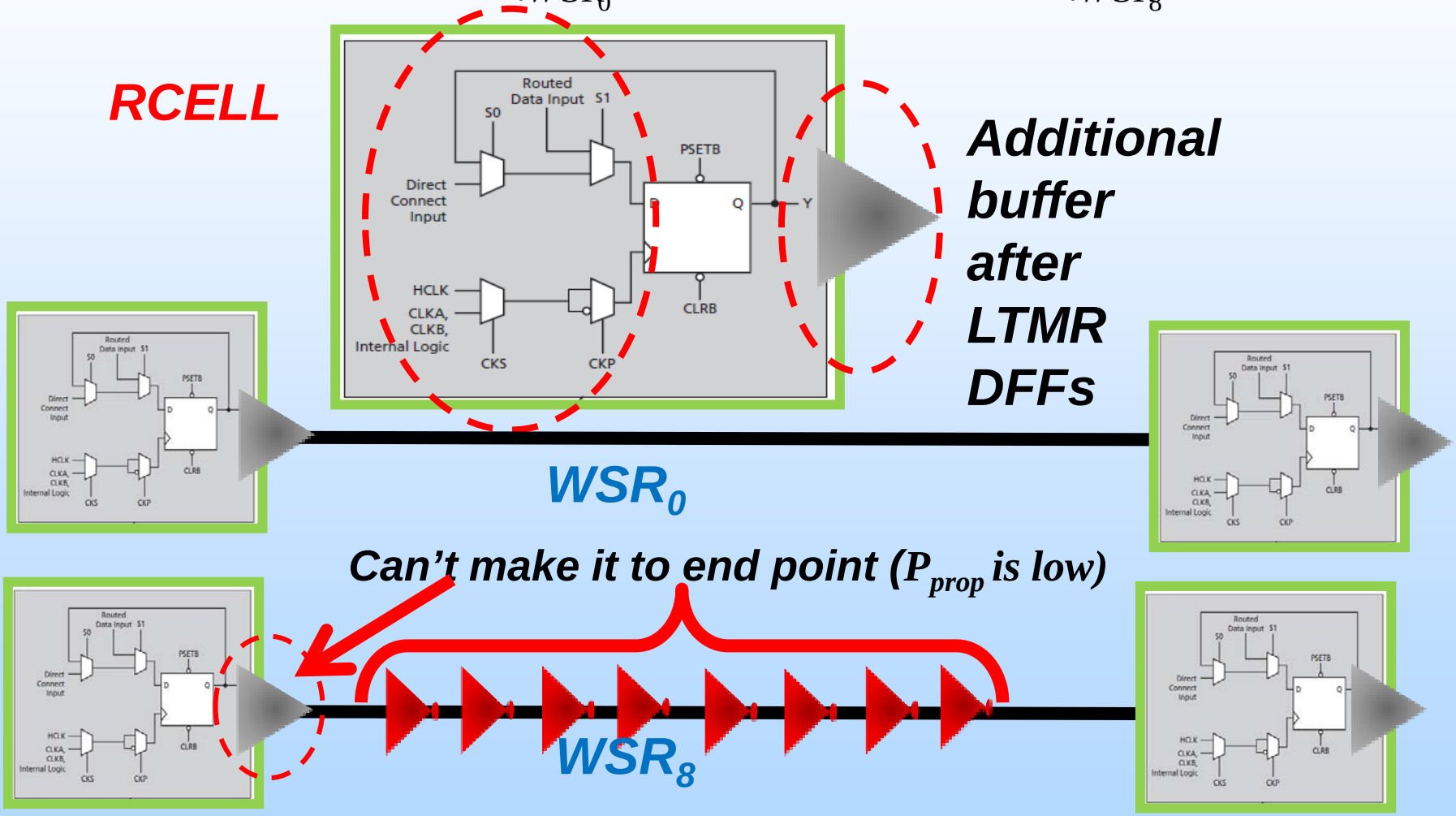
Non-Linear Effects:

- ***WSR_0 has the lowest σ_{SEU} at High LETs***
- ***WSR_0 σ_{SEU} > WSR_8 σ_{SEU} at Low LETs***

Low LETs: attenuation of Single Event Transients (SETs) at low LET values

Why is WSR_0 > WSR_8 for Low LET?

$$(P(fs)_{SET \rightarrow SEU})\big|_{WSR_0} > (P(fs)_{SET \rightarrow SEU})\big|_{WSR_8}$$



Why is $WSR_0 > WSR_8$ for Low LET? Proof using the REAG Model

WSR_8 CCELL P_{prop} is low

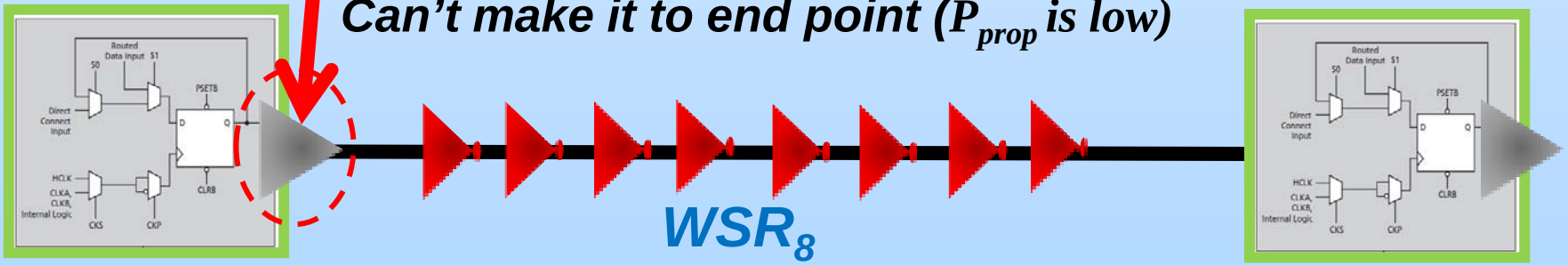
$$\sum_{j=1}^{1} P_{gen(j)} P_{prop(j)} \tau_{width(j)} fs + \sum_{i=1}^{\#CCELLCombinatorial} P_{gen(i)} P_{prop(i)} \tau_{width(i)} fs \Big|_{WSR_8} < \sum_{j=1}^{1} P_{gen(j)} P_{prop(j)} \tau_{width(j)} fs \Big|_{WSR_0}$$

RCELL — WSR_8 — RCELL — WSR_0

$$\sum_{j=1}^{1} P_{gen(j)} P_{prop(j)} \tau_{width(j)} fs \Big|_{WSR_8} < \sum_{j=1}^{1} P_{gen(j)} P_{prop(j)} \tau_{width(j)} fs \Big|_{WSR_0}$$

$RCELL\ P_{prop_{WSR8}} < RCELL\ P_{prop_{WSR0}}$

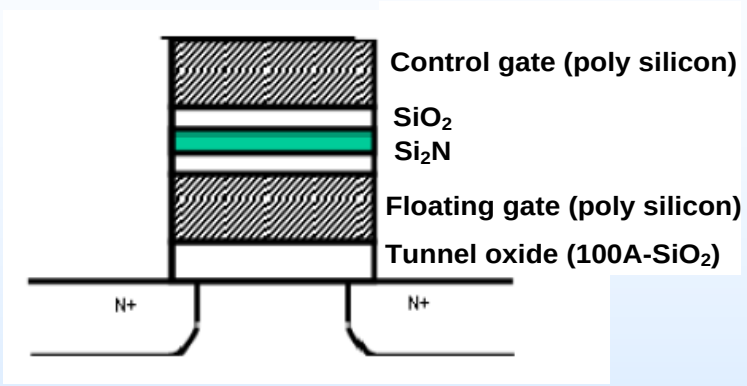
Can't make it to end point (P_{prop} is low)

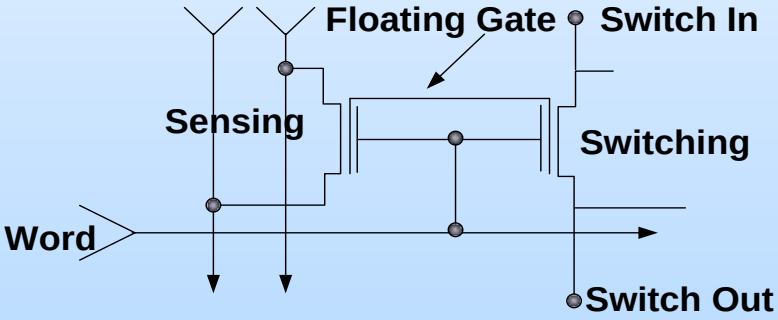


NASA REAG Models + Heavy Ion Data: ProASIC

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
- Evaluation of user mitigation insertion has been performed





Actel ProASIC3 Shift Register Study

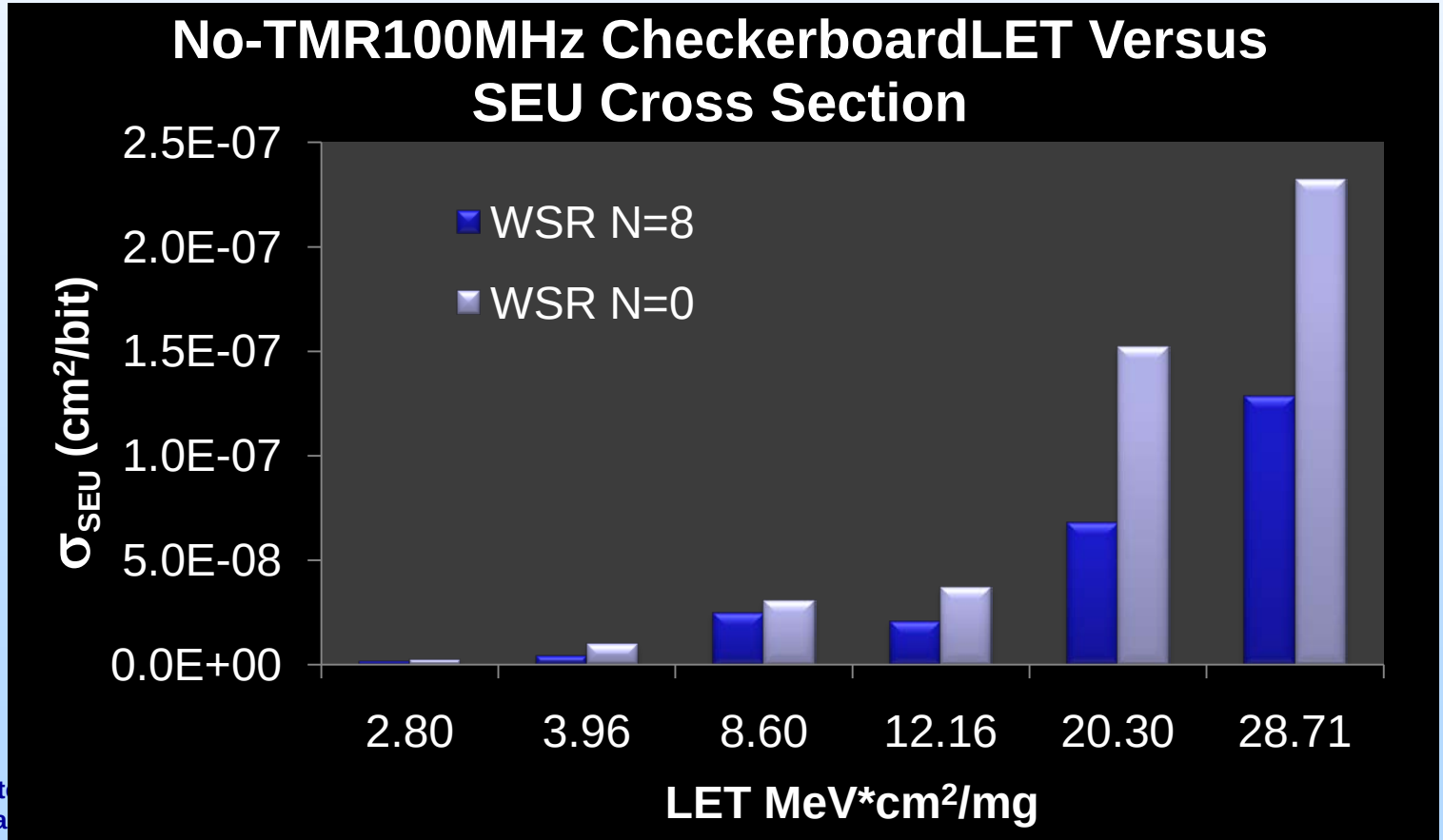
- **Shift Register Functional Logic Designs Under Test:**
  - **Six WSR strings with various levels of combinatorial logic**

$$P(fs)_{error} \propto P_{Configuraton} + P(fs)_{functionaLogic} + P_{SEFI}$$

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

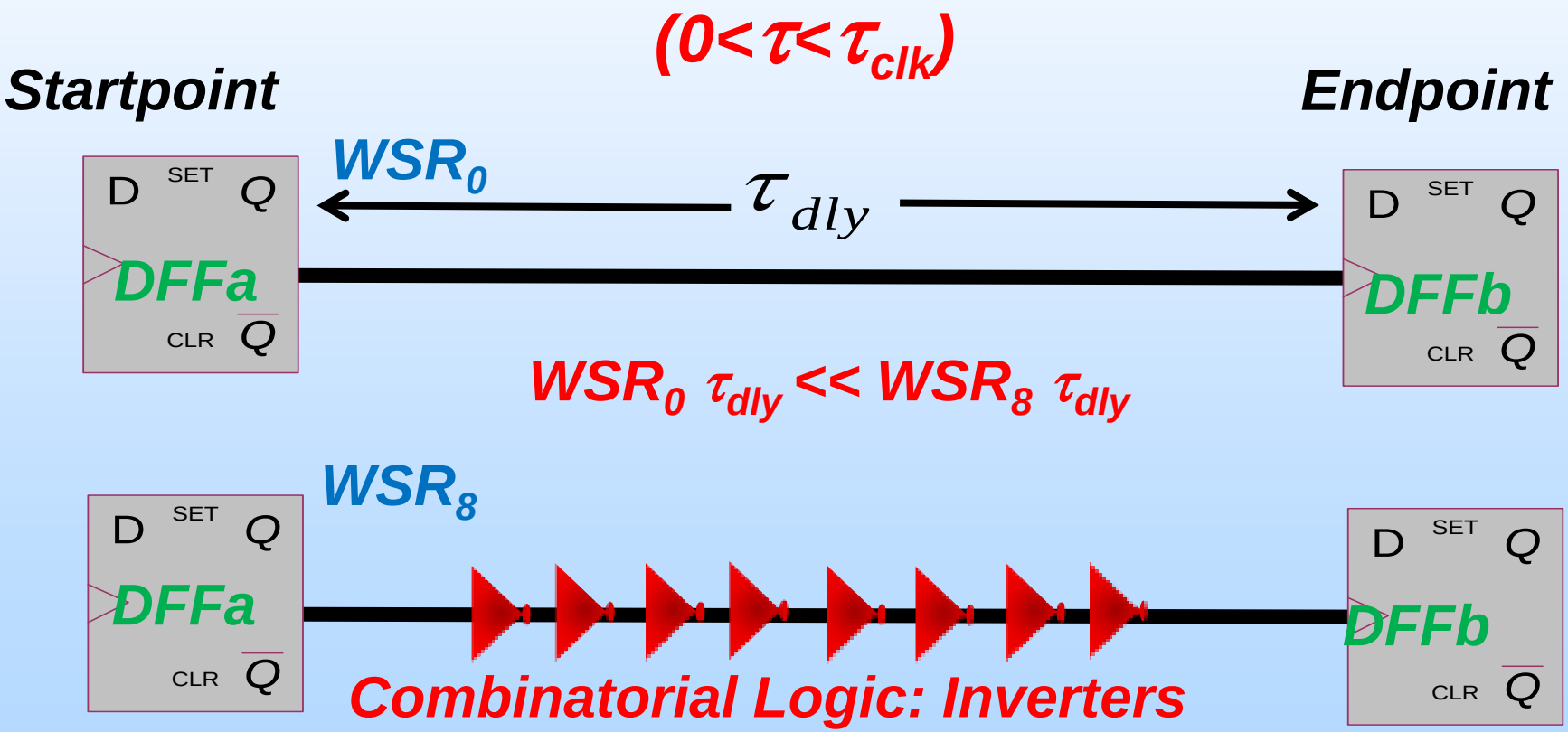
σ_{SEU} Test Results: Windowed Shift Registers (WSRs) No-TMR

- **N=0: WSR with only DFFs**
- **N=8: WSR with 8 inverters between each DFF stage**
- ***No Mitigation: σ_{SEU} WSR_0> σ_{SEU} WSR_8 For every LET***



Why is WSR0 > WSR8 for Non-Mitigated ProASIC: τ_{dly}

For a clock period = τ_{clk}, if DFFa flips @ time $\tau > (\tau_{clk} - \tau_{dly})$ then DFFb will never capture the upset.



Why is WSR_0 σ_{SEU} > WSR_8 σ_{SEU} for The Non-Mitigated ProASIC3 Design?

New Research!

$$(P(fs)_{DFFSEU\rightarrow SEU}\big|_{WSR_0} > (P(fs)_{DFFSEU\rightarrow SEU} + P(fs)_{SET\rightarrow SEU})\big|_{WSR_8}$$

DFF Capture ($P(fs)_{DFFSEU\rightarrow SEU}$) for WSR_0 is not the same as WSR_8 because of τ_{dly}

$$P_{DFFSEU}(1-\frac{\tau_{dly}\big|_{WSR_0}}{\tau_{clk}}) > P_{DFFSEU}(1-\frac{\tau_{dly}\big|_{WSR_8}}{\tau_{clk}}) + \sum_{i=1}^{8} P(fs)_{SET\rightarrow SEU(i)}$$

$\tau_{dly}|_{WSR0} < \tau_{dly}|_{WSR8} < \tau_{clk}$

$$P_{DFFSEU} > \frac{\tau_{clk}}{\tau_{dly}\big|_{WSR_8} - \tau_{dly}\big|_{WSR_0}} \sum_{i=1}^{8} P(fs)_{SET\rightarrow SEU(i)}$$

DFF σ_{SEU} > combinatorial logic σ_{SET}

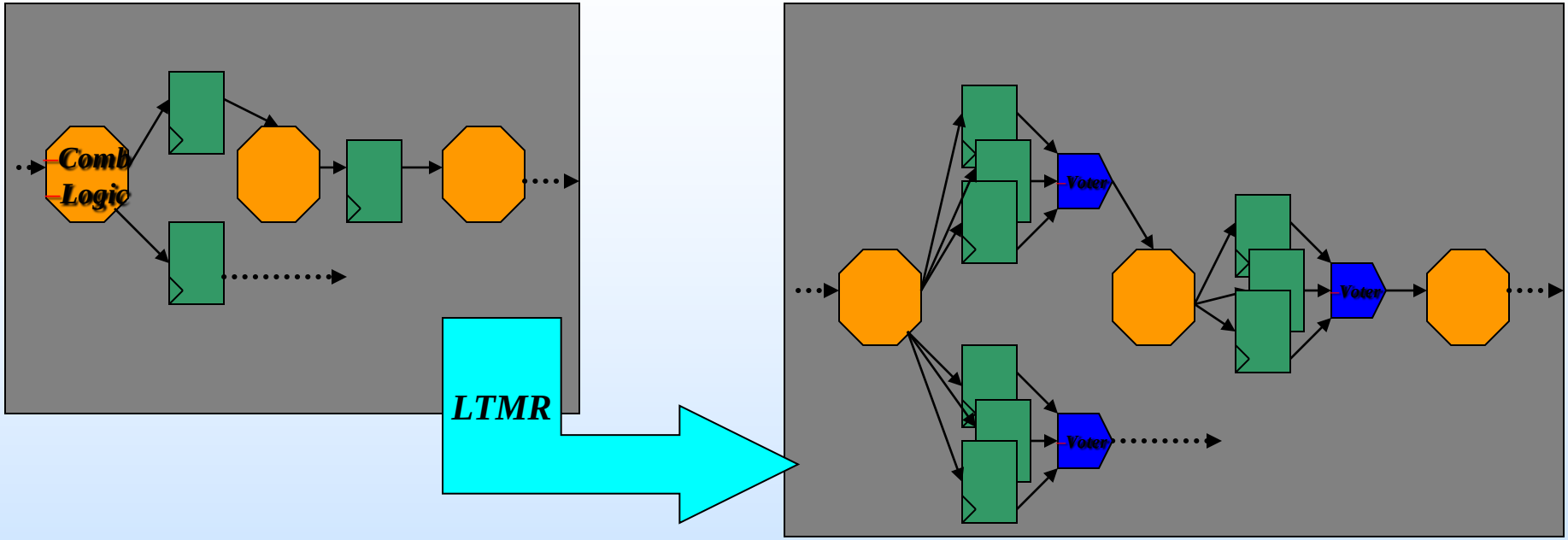
$P_{funtionalLogic}$: Comparison of $P_{DFFSEU \rightarrow SEU}$ and $P_{SET \rightarrow SEU}$ … How it Impacts System Susceptibility

	$P_{DFFSEU \rightarrow SEU}$	$P_{SET \rightarrow SEU}$
Logic	**Startpoint DFF Capture by Endpoint DFF**	**Combinatorial SET Capture by Endpoint DFF**
Capture percentage of clock period (τ_{clk})	$(1-\frac{\tau_{dly}}{\tau_{clk}}) = (1-\tau_{dly} fs)$	$\frac{\tau_{width}}{\tau_{clk}} = \tau_{width} fs$
Frequency Dependency	**As frequency increases, $P_{DFFSEU \rightarrow SEU}$ decreases**	**As frequency increases, $P_{SET \rightarrow SEU}$ Increases**
Combinatorial Logic Effects	**Increase Combinatorial logic increases τ_{dly} and decreases $P_{DFFSEU \rightarrow SEU}$**	**Increase in combinatorial logic increases P_{gen} and increases $P_{DFFSEU \rightarrow SEU}$**

Another Application of the $P(fs)_{functionalLogic}$ Model Components

- If the DFFs are mitigated and the σ_{SEU} is decreasing over frequency, how do you analyze this?
  - Combinatorial logic effects are directly proportional to system frequency $P_{gen(j)} P_{prop(j)} \tau_{width(j)} fs$
  - Hence, most likely not due to combinatorial logic… i.e not $P_{SET \rightarrow SEU}$.
- More than likely, the DFFs are not as mitigated as you expected them to be
  - As system frequency increases cross section decreases $(1 - \tau_{dly} fs)$
  - $P_{logic} \neq 0$

Local Triple Modular Redundancy (LTMR): Triple DFFs Only



- Triple Each DFF + Vote+ Feedback Correct at DFF
- Unprotected:
  - Clocks and Resets… SEFI
  - Transients (SET->SEU)
  - Internal/hidden device logic: SEFI

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

ProASIC LTMR Shift Register Data Path Model

Evaluate for Each DFF

$$\exists_{DFF}\left(\sum_{j=1}^{\#StartPointDFFs} P(fs)_{DFFSEU\rightarrow SEU(j)} + \sum_{i=1}^{\#CombinatorialLogicGates} P(fs)_{SET\rightarrow SEU(i)}\right)$$

LTMR: P_{logic}=0

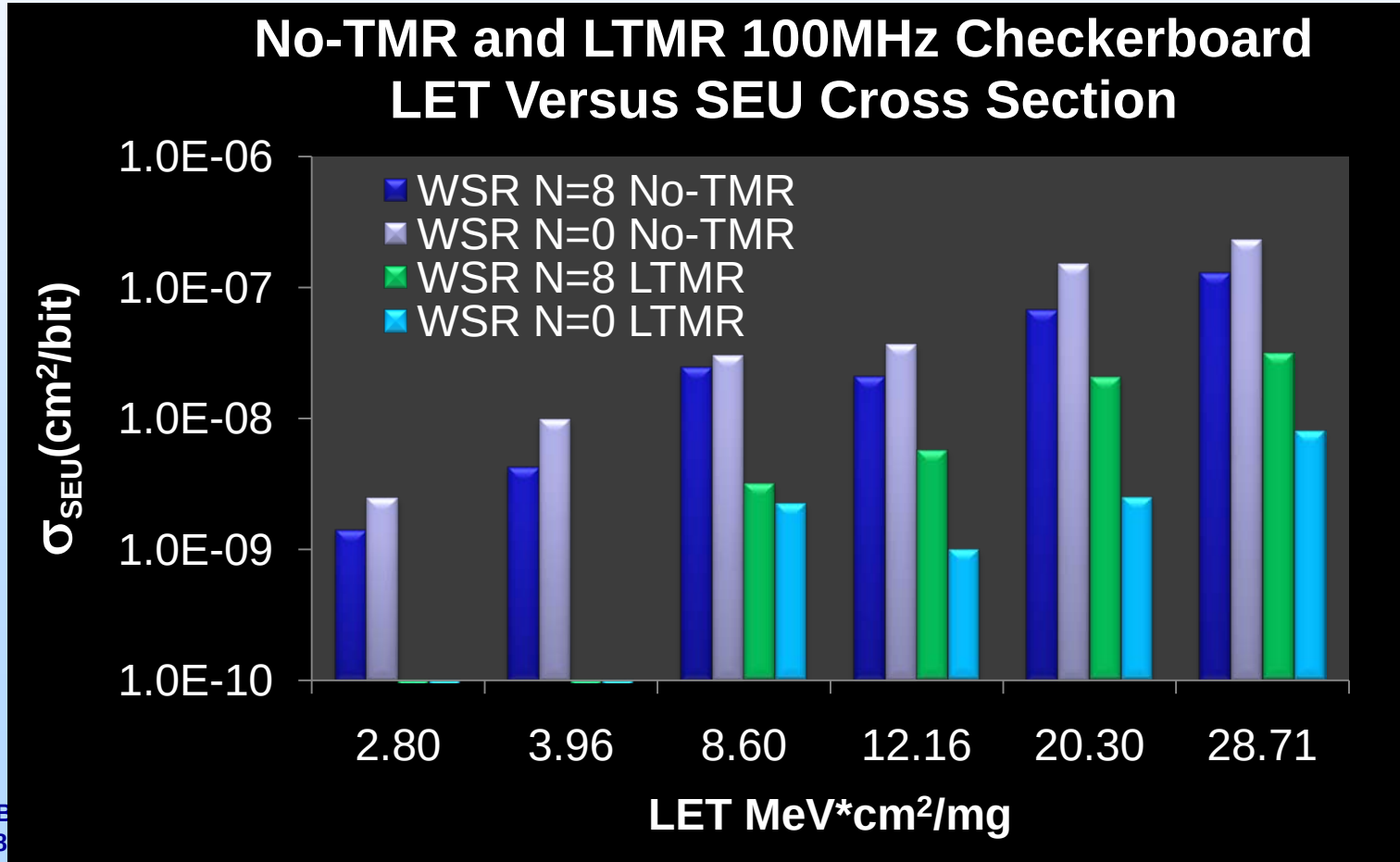
$$\exists_{DFF}\left(P(fs)_{DFFSEU\rightarrow SEU} + \sum_{i=1}^{\#CombinatorialLogicGates} P(fs)_{SET\rightarrow SEU(i)}\right)$$

$$\exists_{DFF}\left(\sum_{i=1}^{\#CombinatorialLogicGates} P(fs)_{SET\rightarrow SEU(i)}\right)$$

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

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

- **LTMR is effective and has reduced P_{DFFSEU}**
- ***LTMR: SEU cross Sections $WSR_0 < WSR_8$ For every LET***



Deliverables

- Further develop components of model for all FPGAs of concern
- Apply the model to an Application Specific Integrated Circuit (ASIC) Design
- Utilize the models to develop:
  - Develop design guidelines for radiation effects per FPGA
  - Evaluate strength of a variety of mitigation strategies per FPGA type

Summary

- **REAG has developed a FPGA SEE model:**
  - **Specifically for Synchronous designs**
  - **Categorizes SEE upsets to assist analysis and test structure development**
  - **Successfully applied across a variety of FPGA types**
  - **Great method for comparing different device types**
  - **Upper-bound version is mostly utilized**

$$P(fs)_{error} \propto P(fs)_{Configuraton} + P(fs)_{functionaLogic} + P(fs)_{SEFI}$$

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

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