

Field Programmable Gate Array (FPGA) Single Event Effect (SEE) Radiation Testing

Prepared by: Melanie Berg
MEI Technologies in support of NASA/Goddard Space Flight Center
Melanie.D.Berg@nasa.gov

For: NASA Electronic Parts and Packaging (NEPP); and Defense Threat
Reduction Agency Under IACRO #11-4395I

Date: 2/02/12

Acronym List:

CL: Combinatorial Logic

COTs: Commercial-off-the-shelf

DSP: Digital signal processor

DUT: Device under test

FF: Edge-triggered Master-Slave Flip-Flop

FIR: Finite impulse response filter

FPGA: Field Programmable Gate Array

 f_s : Operational System frequency

I/O: Input-Output

LET: Linear Energy Transfer ($\text{MeV}\cdot\text{cm}^2/\text{mg}$) LET_{th} : Linear Energy Transfer threshold ($\text{MeV}\cdot\text{cm}^2/\text{mg}$)

LUT: Look up table

MUX: multiplexer

 P_{DFESEU} : Probability that a flip-flop can change its state due to a single event upset that was generated internal to the flip-flop. P_{gen} : Probability that a gate can generate a single event transient. P_{logic} : Probability that the gates in the forward path of the node being analyzed will logically mask the node's upset from being captured by the system. Logical masking is in reference to a cone of logic. P_{prop} : Probability that a single event transient can propagate to a capture node (flip-flop). Also referred to as electrical masking and is in reference to a cone of logic.

SEE: Single Event Effect

SEU: Single Event Upset

 σ_{SEU} : SEU cross-sections (cm^2/bit or $\text{cm}^2/\text{device}$)

STA: Static Timing Analysis

 τ_{clk} : Clock period = inverse of the operational frequency. τ_{dly} : synchronous data path temporal delay measured from flip-flop to flip-flop τ_{jitter} : system clock jitter τ_{HOLD} : Flip-flop hold-time τ_{setup} : Flip-flop setup-time τ_{skew} : system clock skew τ_{width} : single event transient width**1 INTRODUCTION**

Field Programmable Gate Arrays (FPGAs) are widely used in critical space-flight applications as controllers and data processors. Due to their significant roles throughout a system, the integrity of FPGA operation can compromise the success of a mission. Consequently, a significant amount of effort is given to hardness assurance [1].

It has been shown that FPGA devices are susceptible to the radiation effects of ionizing particles routes [2]- [27]. When operating in such environments, critical space-applications require a significantly low number of temporary upsets, a high percentage of device availability, and virtually no risk of device damage during a complete mission.

Exposing a Device-Under-Test (DUT) to an accelerated radiation source and monitoring the DUT's response is the primary method for on-ground Single Event Upset (SEU) evaluation [10]-[23]. Radiation test data are processed and are used to estimate the potential for device degradation, damage, and functional-error rates.

The NASA Goddard Radiation Effects and Analysis Group (REAG) has developed a robust test and analysis methodology for evaluating FPGA SEU data. This document describes REAG's process for test development and data analysis. Included are guidelines and recommendations for test implementation.

1.1 FPGA Basics

Field Programmable Gate Arrays (FPGAs) are packaged integrated circuits (ICs) containing groups of logic, interconnects, and I/O referenced as blocks or cells [1]-[9]. The blocks have the ability to be configured (programmed) into variety of small functions. The premise of device usage is to map a specified digital design into an FPGA's configurable cells. Design mapping is feasible because each block type within an FPGA can be configured as a piecemeal implementation of the full design. Subsequently, each configurable cell of an FPGA device can be thought of as a building block.

There are four primary categories of structures that exist in an FPGA: configuration, functional logic data path, I/O and global routes [2]-[8]. Table 1 is a description of the FPGA categories. Each category has a unique contribution to the overall SEU cross section (σ_{SEU}) of an FPGA design. Subsequently, each category should have specific SEU tests that will assist in the evaluation of their susceptibility.

Table 1: Categorization of basic FPGA Structures. Proper radiation testing requires performing specific SEE tests that target each category. Each category will have a its own corresponding susceptibility.

FPGA Category	Description
Configuration	A static definition of the function. It consists of elements that hold information regarding: <ul style="list-style-type: none"> The identification of selected FPGA logic blocks The mapped function of the selected FPGA logic blocks Interconnects between logic blocks (local and global routes) that support the desired function The I/O definitions that support the targeted function
Functional Logic	The logic cells that perform operation: Combinatorial logic blocks, routes, sequential logic blocks. Functional logic form the internal data path of a design.
I/O	Input and Output blocks. Although I/O are part of a functional data path, they are placed in their own category because they are created using a different geometry transistor logic with different threshold voltages than internal functional logic.
Global Routes	Clock trees, resets, and high fan-out nets

1.2 Complex FPGA Devices

More complex FPGA devices have embedded blocks of logic that perform high-speed multifaceted functions such as: digital signal processors, general processors, SRAM, memory controllers, analog logic (with analog-to-digital converters and digital-to-analog converters), and clock synthesizers (such phase locked loops and digital clock managers). As with the categorized blocks in Section 1.1, each embedded block has its own susceptibility; and if used within a design, their unique susceptibility will add to the overall upset rate.

This document focuses on SEE testing regarding the four major categories described in Section 1.1.

1.3 Establishing A Design Methodology

In order to evaluate the susceptibilities of the various elements within an FPGA, designs using these elements must be created and mapped into the DUT-FPGA fabric. Test designs (test structures) must be reliably implemented such that radiation data obtained during SEE testing can characterize operational or performance susceptibility during exposure. Subsequently, a methodology for developing DUT-designs must be established. The most common methodology used in critical applications is synchronous design. In accordance and in order to control the volume of information within this documentation, the document's scope is limited to synchronous design methodology.

2 EXECUTIVE SUMMARY: SINGLE EVENT UPSET TESTING TARGETING FPGA DEVICES

This section highlights some key points regarding SEE testing. It begins with answers to some basic SEE questions; followed by general considerations for SEE test plan development.

2.1 SEE Q&A

How are FPGA SEU data generally processed and analyzed?

SEU testing requires counting the number of upsets that occur while exposing a DUT to a given number of ionizing particles. These test metrics are SEU cross-sections (σ_{SEU}) [9]. A σ_{SEU} unit is in respect to area and is generally expressed in cm^2/bit , $cm^2/design$, or $cm^2/device$. Calculating σ_{SEU} is the process of counting the number of error events during irradiating the DUT and dividing by the number of ionizing particles per unit area (fluence) of exposure. The simplest form of the equation used for calculating σ_{SEU} is shown in Eq. (1).

(1)

What type of particles are used during SEE testing?

When testing with heavy ions, a σ_{SEU} is calculated per particle LET. When testing with protons, neutrons, or alpha particles, a σ_{SEU} is calculated per particle energy.

What purpose do σ_{SEUS} serve?

σ_{SEUS} are used to calculate error rates by integrating the σ_{SEUS} across particle LET or particle energy. Hence, one aspect of achieving sensible error rates is to obtain σ_{SEUS} for at least 5 different LET values.

Another use of σ_{SEUS} is to analyze error signature trends. Such an analysis is performed to study a variety of effects due to variations in: operational frequency (f_s), data switching (i.e., data pattern)

rates, design complexity, and component susceptibility.

How much fluence is enough?

Not every particle will cause an error. Hence, in order to increase the integrity of σ_{SEUS} , it is best to expose the DUT to enough radiation particles to generate a significant number of observable events. As a rule of thumb, a significant number of upsets is considered ≥ 100 events for most LETs. However, when testing with near event-threshold LET or near event-threshold energy particles, the limit of events approaches zero. In this case, during irradiation fluence is increased (fluence $\geq 1 \times 10^7$ particle/cm² if possible); and a significant number of upsets is considered > 4 .

2.2 General Considerations when Preparing an FPGA SEE Test Plan

Performing radiation testing and calculating σ_{SEU} for FPGAs is a challenging process mostly because FPGAs are complex devices containing thousands-to-millions of components that implement complex designs. Different testing approaches are taken depending on the FPGA device type, design methodology, speed of operation, and type of radiation evaluation. The following is a synopsis of recommended procedures that constitute a process for FPGA SEU test and analysis:

1. Evaluate the DUT-FPGA fabric: A comprehensive study of the FPGA's fabric must be performed prior to testing. The evaluation involves understanding the FPGA's elements and how designs are mapped into its elements. From this information, specific radiation tests and test structures can be developed to target the DUT's various components.
2. Consider the goal of radiation testing prior to creating the test plan. The intention of testing will drive the test structures implemented in the DUT. The following are two goals that will require different test plans:
 - a. Evaluation of Flip-Flop (FF) susceptibility: If the FF's are radiation hardened by design (RHBD), then a goal of SEU testing should be to analyze the effectiveness of the mitigation strategies. Simple test structures such as shift registers are optimal for evaluating FF mitigation. The reason is the possibility for a shift register gate to logically block an upset is minimal. Alternatively, complex test structures have a significant number of gates that can logically block upsets; e.g., if at least one of an AND gate's inputs is set to a logic '0', upset feeding the AND gate's other inputs will be logically-masked. In order to optimize visibility of FF susceptibility, test structures should be selected that have minimal data-path logic-masking.
 - b. Extrapolation of σ_{SEU} data to calculate error rates for real designs: Characterizing SEU effects for designs is a different process than studying individual elements such as FFs. Usually the mission's final design is not tested in the radiation beam. Subsequently, test structures are developed and then radiation tested to evaluate trends. The trends are then used to facilitate the extrapolation of σ_{SEU} data to calculate error rates for the mission's final design.
3. Create DUT test structures: FPGA test structures should have the following characteristics:
 - a. Similar topologies that utilize the same basic elements as real designs,
 - b. Repetition of design to increase statistics
 - c. Functional visibility such that all upsets can be identified and recorded,
 - d. A state space that can be traversed within minutes; i.e., a traversable state-space.
4. Develop a test vehicle: The test vehicle connects to the DUT, provides stimuli to the DUT, monitors the DUT during radiation testing, and records DUT failures during radiation testing.
 - a. The test vehicle should be robust such that DUT stimuli (e.g., data patterns and operational frequency) can be varied.
 - b. The test vehicle must be robust such that it can monitor and capture a majority of failures; i.e. The test vehicle is expected to reliably capture DUT data and be fast enough to handle DUT upset events in an accelerated radiation environment.
5. Perform detailed Test and Analysis: Tests performed will be based on the type of FPGA. Table 2 is a short list of tests and considerations based on FPGA device type. Selected tests that are run for SEU evaluation should optimally expose upset events and concentrate on various aspects of the FPGA fabric. The analysis phase is also heavily dependent on FPGA type. Additional information regarding test and analysis is provided throughout this document.

Table 2: General considerations and recommendations for SEU tests based on FPGA type

FPGA Type	Configuration Considerations	Data path Considerations
Unhardened SRAM-Based	Configuration tests can be performed statically. This is generally accomplished by irradiating the device and then reading back the configuration	Will need a scrubber; Data path logic mitigation should be considered; Dynamic tests are recommended

	bits; It is not recommended to scrub when testing configuration;	
Antifuse-Based	Verify that the configuration fuses are intact after each irradiation run. This can be accomplished by verifying no stuck faults or degraded timing paths exist post-irradiation	Data path logic mitigation should be considered especially if no internal mitigation exists; Dynamic tests are recommended
Flash-Based	Configuration tests can be performed statically This is generally accomplished by irradiating the device and then reading back the configuration bits; Flash configuration is relatively hard, consequently, scrubbing is not necessary. In addition, manufacturers have disabled the ability to scrub flash configuration FPGA devices.	Data path logic mitigation should be considered especially if no internal mitigation exists; Dynamic tests are recommended

The following sections cover the procedures and considerations provided in the Executive summary in more detail.

3 DESIGN CONCEPTS AND BASIC FPGA ELEMENTS

How the FPGA building blocks are configured to form a design (design topology) governs the functional susceptibility. Hence developing an understanding of test structure design concepts and how the various types of elements within the design topology can affect susceptibility is essential.

3.1 Configuration Technology

During the design phase, the design is mapped into the FPGA device. The mapping process includes:

- Logic block function definition
- Logic block selection (i.e., placement)
- Logic block connection (i.e., routing)

Each FPGA element (combinatorial logic (CL), Flip-flop (FF), clock, route, etc...) has distinct switches that are used to form a specified function [2]-[8]. A design is implemented by selecting a switch state (e.g., on or off) to build logic and connectivity as illustrated in Figure 1 and Figure 2.

. Switch values (on or off) are determined during the design implementation and mapping process and are static thereafter. The static state of the switches defines the design and is referred to as the configuration.

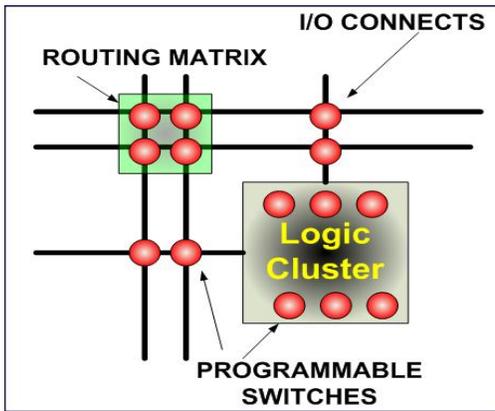


Figure 1: Programmable switches are selected to define the configuration of a design

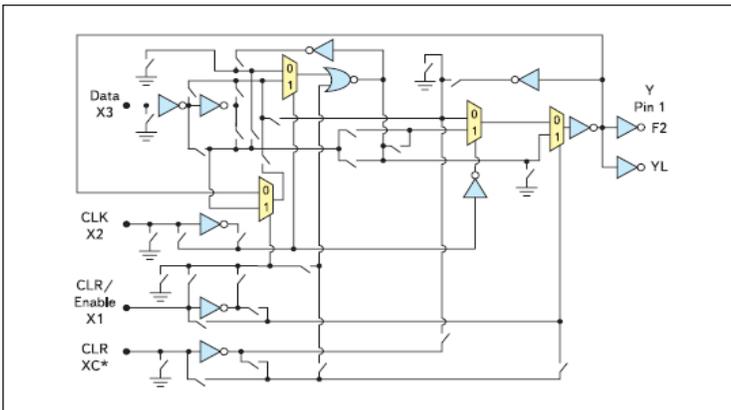


Figure 2: One functional building block in the Microsemi ProASIC3 FPGA [5]. Each open switch is a programmable node controlled by a configuration cell. The configuration cells are flash memory. Block function is created by fixing the state of the switch (configuration cell). The state of each switch is determined during the design phase; and remains static there after.

The technology of the configuration switch is manufacturer specific. There are three major types of FPGA configuration technology:

- SRAM memory routes [2][6]-[8]: Reprogrammable (RP): each SRAM bit sets a static state of the program switch. SRAM configurations are re-programmable meaning that designs can be changed. A couple of benefits for using SRAM as configuration are that problems (bugs) found in designs can be fixed by re-configuring (re-programming) the FPGA and FPGAs can be reused for a variety of functions. A con for using SRAM based FPGAs is that they are volatile such that the FPGA needs to be re-configured during every power cycle. In order to support the device re-programming, the system needs an external non-volatile memory (additional system component) that can store the design's configuration during power down and that can write the stored configuration into the FPGAs internal SRAM configuration bits during power up.
- Antifuse [3][4]: One Time Programmable (OTP): a programmable-switch is turned on by creating an electrically conductive path in the metallization layer of the FPGA IC. A benefit is that the device does not require an additional non-volatile memory component to store configuration because the configuration is permanently set within a metallization layer and does not get disturbed during power down. A con is that because the configuration is permanent, the FPGA cannot be re-programmed. Hence, if a bug is found, the device will need to be discarded and a new device will need to be anti-fused.
- Flash memory [5]: Reprogrammable (RP): A flashed based configuration uses reprogrammable flash type memory cells to store the state of the programmable switch. Because Flash is non-volatile no additional memory is required to store the configuration during power down. Subsequently, the benefit of using this technology is that it is reprogrammable and does not require additional components.

3.2 Basic Concepts of Synchronous Design

The complexity of FPGA designs is exponentially growing. The difficulty is centered on managing higher speeds, larger gate counts, and communicating across clock domains. The following is a list of challenges with creating working digital

designs:

- Obtaining deterministic behavior. The following are benefits of creating designs with deterministic behavior
 - Predictability
 - Facilitates Verification
 - Tool vendors are better able to optimize performance
- Managing capture mechanics. The following are some of the issues with data capture within a design
 - If data changes during the setup and hold window of a FF, the FF capture will not be deterministic. As a result, it is unknown whether the FF will contain a logic-0 or a logic-1 after input capture. In the worse case scenario, the FF capture can result in an oscillatory state between a logic-0 or a logic-1, i.e., the FF can reach a metastable state. The act of a FF oscillating between a logic-0 and logic-1 due to its input pin changing during its setup and hold time window is called metastability.
 - With the increase in clock speeds, clock domain crossings (CDCs) have become a significant contribution to non-deterministic behavior. CDCs have been reported as the number one bug (error) source. The reason is if CDC's are not managed correctly, a significant number of input signals can change within a FF's setup and hold window and unpredictable FF data capture can affect system behavior.
- Identifying and avoiding bad design practice –
 - Abstract methods of design implementation can cause problems with verification and reusability
 - It is understood that there are many ways that a design can be constructed. However, the method that strictly follows the specified design methodology

The goal of following synchronous methodology is to achieve deterministic circuit behavior in the most simplified manner. The following is a synopsis of concepts that compose synchronous design methodology and create determinism:

- Edge triggered flip-flops (FFs) are used to define system state. FFs can only change values at clock edges or resets, hence, state transitions occur at deterministic points in time.
- Reset starts the design in a well-defined and deterministic state. Consequently, state transitions can be traced.
- All data path signals launch at a clock edge and must become stable prior to the set-up time and must remain stable after the hold time of a FF. Ensuring data paths are stable during the setup and hold windows enables predictable capture behavior (avoids metastable or erroneous capture events).
- FF outputs are expected to be active for a complete clock cycle unless reset

As previously mentioned, it is essential for a test methodology to take into account the elements that comprise the DUT. This includes the building blocks contained in the FPGA and the topology of block connection for design creation. The following sections discuss Synchronous design concepts and how they relate to FPGA elements.

3.3 Global Routes: Synchronous Design Concepts

3.3.1 Clocks

A clock is the heartbeat of a synchronous design. At each (specified) clock edge (either rising or falling clock), the state of the design is defined. The state of the design is held in its FFs. Synchronous circuits require that all FFs on the same clock tree simultaneously capture its data input at a specified clock edge, usually the rising clock edge. In order to achieve synchronous data capture, clock trees must be low-skew global routes. FPGA manufactures provide low-skew global routes by using high-drive buffers in a balanced network (tree). The FPGA manufacturer achieves tree balance by buffer sizing and buffer route length control.

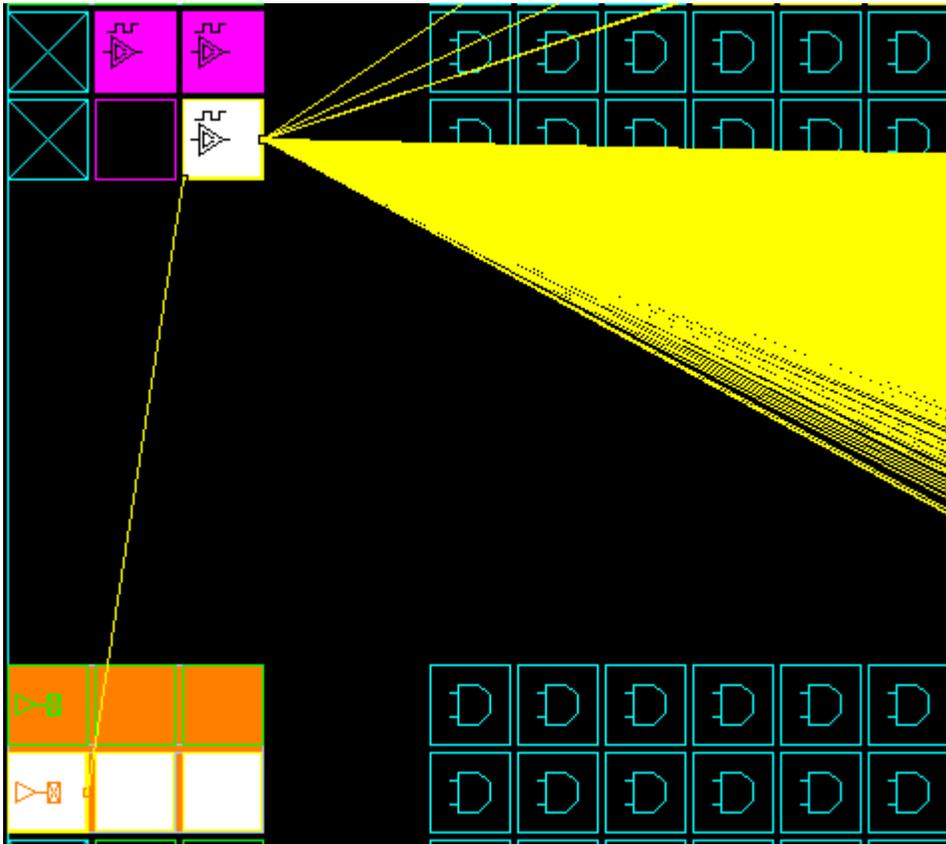


Figure 3: Microsemi RTAXs2000s input buffer connection to clock buffer and clock tree distribution.

Balanced clock trees are available in all modern day FPGA devices. Figure 3 is an example of a clock tree in the Microsemi RTAXs family of FPGA devices.

It is the designer's responsibility to avoid corrupting tree (global route) balance. The following are designer guidelines that will maintain balance and therefore adhere to the synchronous requirement of using minimally skewed clocks.

- Avoid introducing unacceptable noise levels by validating that the clock input pin (or other clock source) is in close electrical proximity the clock buffer.
 - If the pins are too far apart, the net will be too long. Long nets can cause issues with capacitance, crosstalk, and transmission line effects.
 - Designers should consult the manufacturer's data sheet.
- If a clock tree buffer is connected to the clock pin of FFs, then it cannot connect to any other type of logic or pin.
- Clock gating must be done prior to the clock tree buffer and in a glitch free implementation:
 - Clock gating is not recommended. However, if necessary, create a glitch-free circuit that switches clocks such that clocks end/start on the same edge. If implemented, the best practice is to switch clocks while circuitry is in reset.
 - A favorable alternative to clock gating is to use FF enables when possible, though it depends on the circuit and required fan-out.

For future reference throughout this document, clock period (τ_{clk}) is the inverse of frequency (f_s) as in Eq. (2).

(2)

3.3.2 Resets

Reset and Set pins are not differentiated in this document and are both referenced as resets. Resets are control signals used to force the design into a defined state (i.e., initial state). Resets are commonly utilized in critical designs. Because a reset signal connects to a large number of elements (FFs), it has a high fan-out. In a critical design, the reset will be expected to be placed onto a global route (i.e., high-drive, high fan-out net) for two reasons:

1. Meet timing requirements

2. Reduce the effects of reset Single Event Transients (SETs). Because global nets are created out of high-drive buffers, they have a lower susceptibility than other internal circuitry

3.4 Functional Data Path Topology: Synchronous Design Concepts

A synchronous design is a compute-and-capture system. The basic building blocks of a data path are CL and FF cells. CL are used to perform computations and to route data. FFs are used to capture the CL computations and to hold the state of the system. The following sections provide brief descriptions of the two FPGA element types.

3.4.1 Combinatorial logic in a functional data path

There are no hold states in combinatorial logic within a synchronous design. The output of a CL gate has the potential to change its logic state if one of its inputs changes its logic state. There is a temporal delay from a CL input to the CL output. Hence changes in CL output do not occur simultaneously with changes in its input.

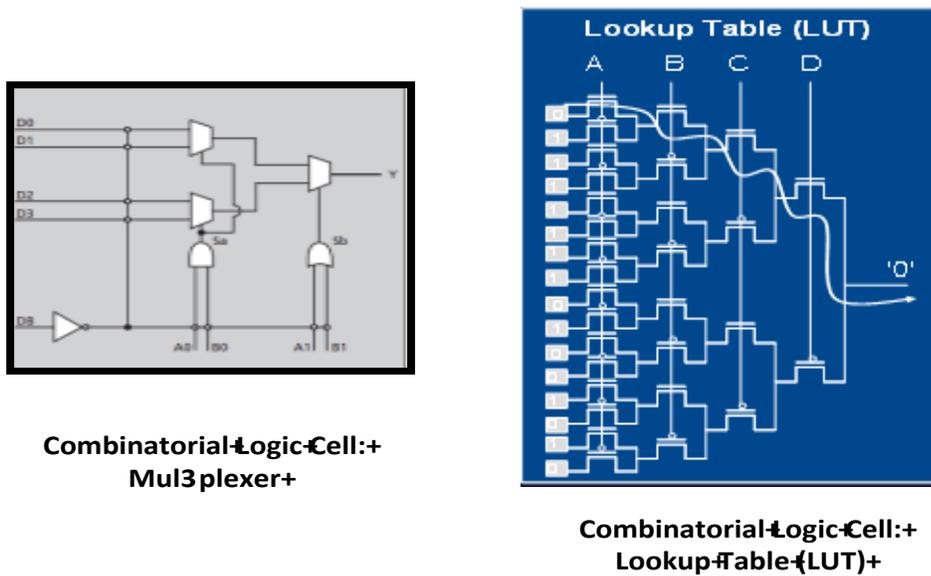


Figure 4: Two types of Combinatorial Logic blocks. As an example: Microsemi FPGA devices tend to design their combinatorial logic cells as multiplexers. Xilinx tends to design their combinatorial logic blocks as Look-up-Tables (LUTs).

In an FPGA a CL cell is a collection of CL gates. It is usually in the form of a multiplexer (MUX) [5] or a Look-Up-Table (LUT) [2]-[8] as illustrated in Figure 4. The goal of an FPGA manufacturer is to provide enough flexibility within each CL cell so that a variety of functionality can be mapped into any of the cell's gates.

3.4.2 Edge Triggered FFs (Sequential Logic) in a functional data

FF's store the state of the system. Because they are storage elements, they are also referred to as sequential elements. Every FF is connected to a clock, has a data input pin (D), and has an output pin (Q) as illustrated in Figure 5. At each clock edge, all FFs that are enabled capture the state of their data input. A clock period (τ_{clk}) is defined as the time between clock edges (rising to rising or falling to falling) as illustrated in **Error! Reference source not found.** As previously mentioned, the clock frequency (f_s) of operation is defined as the inverse of the clock period and is expressed in Eq.(2).

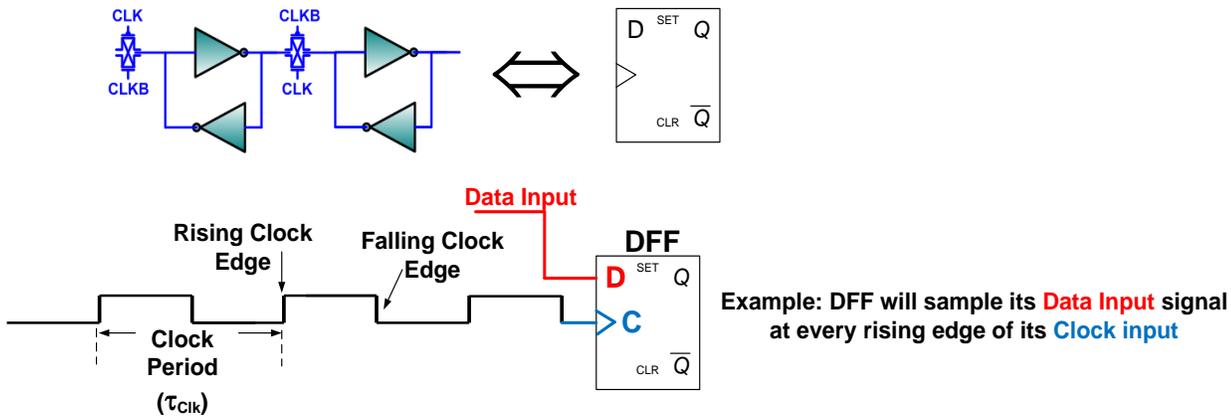


Figure 5: An ideal clock is a square wave. A clock period (τ_{clk}) is defined to be the time between a rising edge to the next rising edge or a falling edge to the next falling edge

As defined by synchronous design rules, all FFs on the same clock tree simultaneously sample their data inputs. In turn, it is important that each FF encounters their clock edge at virtually the same moment in time, i.e., the clock must have minimal skew between FFs.

The following is a synopsis of FF synchronous design methodology concepts that ultimately create reliably predictable systems:

- A synchronous design is a compute-and-capture system:
 - Edge-triggered flip-flops (FFs) feed combinatorial logic for computation
 - A FF captures its data input logic value at a specified clock edge
- FFs are used to define system state. FFs can only change values at clock edges or resets, hence, state transitions occur at deterministic points in time.
- Resets should be connected to all FFs. As previously mentioned, resets initializes the design in a well-defined and deterministic state. Consequently, state transitions can be traced.
- During a clock capture window (setup and hold), all FF data inputs that do not cross clock domains have completed their computation phase and are logically stable. Static Timing Analysis (STA) is performed to verify that all data paths meet this criterion. This enables predictable capture behavior and avoids metastable or erroneous capture events.
- FF outputs are expected to be active for a complete clock cycle unless a reset is administered.

3.5 Putting it all together: Synchronous Data Path Analysis

Within a clock domain, all data are launched from one FF to another. This is referred to as a data path. Each data path will have FFs, routes, and may contain combinatorial logic. A data path is defined to begin from an input or a FF and always end at a FF or an output. The FFs and I/O are considered boundary points. Accordingly, synchronous data paths do not contain multiple stages of FFs. Associated with each FF-FF data path is a temporal delay (τ_{dly}); i.e., the time it takes for the output of one FF to get to the input of a following FF. τ_{dly} dictates how fast a system can operate – e.g., the delay (τ_{dly}) of every data path must be less than its clock period (τ_{clk}). The following sections provide additional information on the topology of data paths and their analysis.

3.5.1 Functional Data Path Cone-of-Logic

The topology of a synchronous design simplifies the process of determining when it is valid for a FF to sample its data path. Synchronous design capture fundamentals and the formulation of τ_{dly} are based on the following (see Figure 6 for illustration):

- Boundary elements (FFs) are deterministic timing points in a synchronous design
- Data is launched from a boundary point (Start-Point)
- Data is captured by a boundary point (End-Point)

- The data path that fans-into the End-Point's data pin is comprised of FFs, routes, and CL. The combination of the End-point and its fan-in data paths form a Cone-of-Logic as illustrated in Figure 6. For every path in the Cone-of-Logic, a logic delay (τ_{dly}) is calculated. τ_{dly} designates the time it takes for a signal to launch from a Start-Point FF, propagate through a path of CL and routes, and reach an End-Point. There is a unique τ_{dly} from every Start-Point to End-Point.

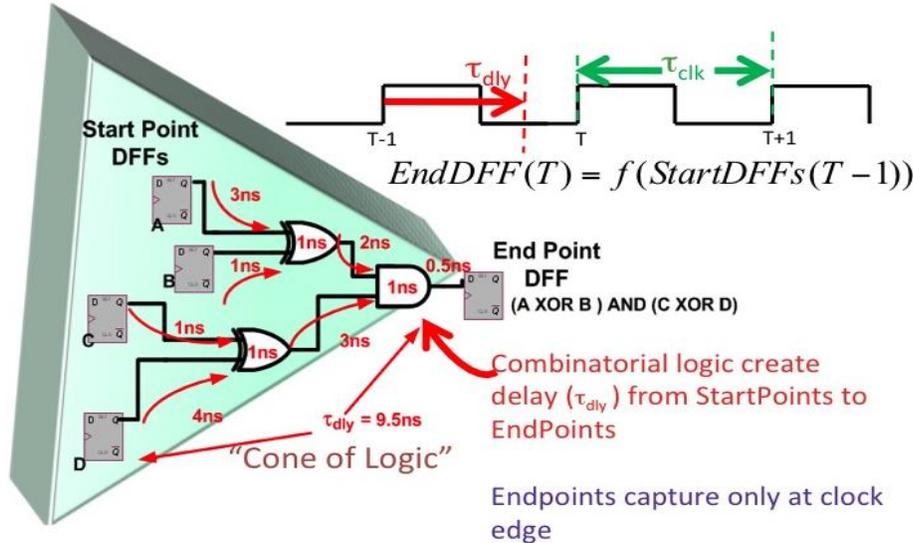


Figure 6: Cone-of-Logic. Signal delay from Start-Point flip-flops to their End-Point in a Cone-of-Logic. Static Timing Analysis (STA) is performed for each cone to determine path delays (τ_{dly}).

3.5.2 The Cone of Logic and Static Timing Analysis (STA)

Cone-of-Logic τ_{dly} (see Figure 6) calculation is referred to as Static Timing Analysis (STA). STA is an automated tool provided by FPGA manufacturers; and is a mandatory procedure in the synchronous design process to verify if the design can operate at a specified frequency.

The STA tool will take into account setup (τ_{setup}), hold time (τ_{HOLD}), clock skew (τ_{skew}) and clock jitter (τ_{jitter}). Worst case analysis requires that $\tau_{dly} < (\tau_{clk} - \tau_{setup} - \tau_{skew} - \tau_{jitter})$ for every data path expected to operate within one clock cycle. Best case analysis requires that $\tau_{dly} > (\tau_{HOLD} + \tau_{skew} + \tau_{jitter})$ for every data path.

The importance of STA will become more apparent. It will be shown later in the document how τ_{dly} and operational frequency directly affect SEU cross sections.

Synchronous design concepts have been described. The following sections illustrate how synchronous SEU test circuits are developed, their contributions to SEU characterization, and their specific disadvantages regarding SEU data collection.

4 FPGA FABRIC AND DESIGN METHODOLOGY

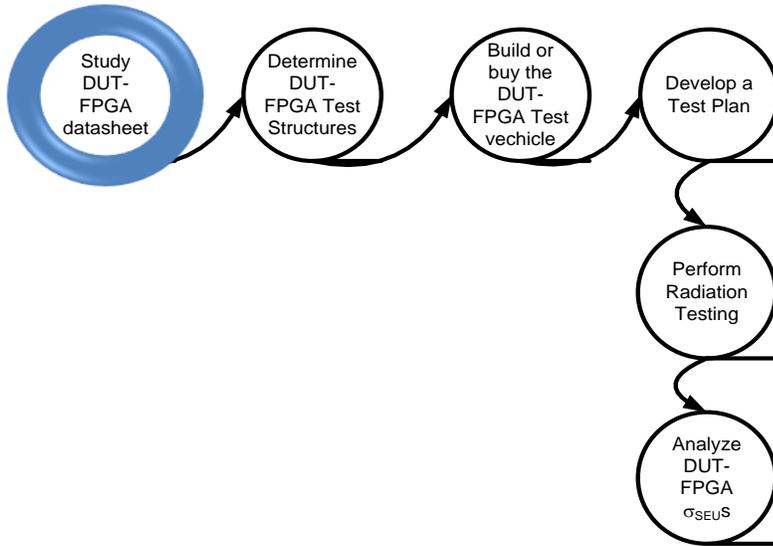


Figure 7: General flow for developing a SEU test strategy

Figure 7 is a diagram representing the recommended process flow of SEU test development. The first step of the flow is a study of device specifics and includes consulting the FPGA datasheet. There are a variety of FPGA types. The data sheet will provide information regarding the DUT’s internal elements, switching speeds, and power consumption. All must be taken into consideration prior to creating DUT test-structures. The following are common concerns that should be addressed prior to developing the DUT-FPGA test plan, test structures, and test vehicle:

- What types of elements exist in the DUT-FPGA to test? – e.g., type of flip-flops (FFs), types of combinatorial logic structures, type of configuration, types of global routing, hidden logic circuits, etc,...
- Are any of the elements mitigated or hardened?
- How much power does the DUT-FPGA consume? – e.g., is cooling equipment required?
- Does the DUT-FPGA require any special apparatus – e.g., are additional devices necessary to operate the DUT-FPGA such as a configuration manager or memory elements?
- Is special testing equipment required to operate at the maximum speeds of the DUT-FPGA?
- What are the switching characteristics of the I/O: e.g. – e.g. will the I/O speed limitations restrict maximum frequency test-structure evaluations? Or will the output switching characteristics cause signal integrity issues if not handled properly?

As highlighted in Figure 8, this section describes recommended considerations when developing test structures for SEU analysis.

5 CREATING TEST STRUCTURES FOR SEU ANALYSIS

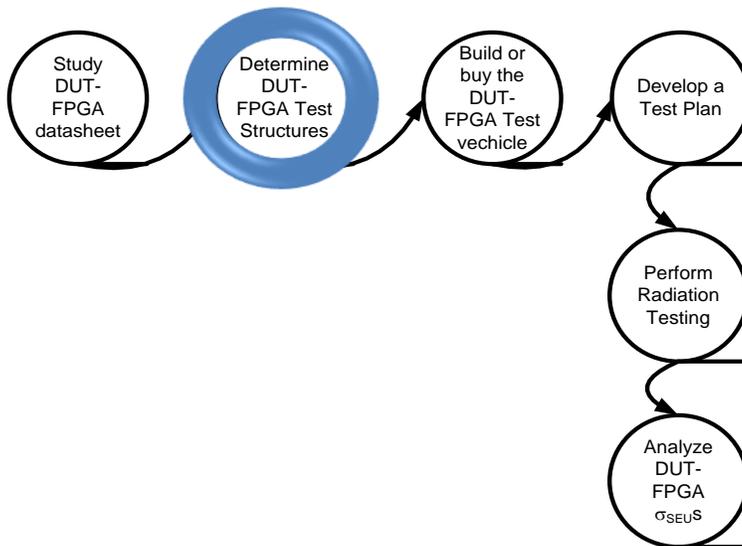


Figure 8: General flow for developing a SEU test strategy

Test structure development is the next step after being familiarized with the DUT-FPGA and understanding general FPGA design concepts. Test structures are designs implemented in the DUT-FPGA specifically for SEU analysis.

5.1 Recommendations for Test Structure Creation

Careful selection of test structures facilitates gathering radiation data that will sufficiently characterize the FPGA DUT susceptibilities. Attention should be given to the considerations listed in Table 3 while developing DUT-designs. Taking these considerations into account will maximize the integrity of SEU data.

Table 3: Best practice considerations for creating DUT test structures for SEU testing

	Recommendations For Creating Optimal SEE DUT Test Structures
1	Create a DUT design that has a large number of replicated logic structures in order to increase statistics.
2	Implement a DUT design that has a traversable state space that can be completed within one radiation test run
3	Develop a DUT design such that logic masking is minimized or is controllable
4	Create a DUT design such that all (or a significant percentage of) potential upsets are observable
5	Manage the I/O of the DUT design such that the DUT to tester interface is reliable. The following are DUT-Tester interface issues that can compromise test vehicle operation: Signal integrity, speed of I/O, number of I/O, data bandwidth, data control and capture.
6	Follow synchronous methodology guidelines in order to characterize topologies that match real designs

It is important to test circuits that reflect real designs when performing SEU characterization. Because critical designs are synchronous, it is therefore essential to test circuits that have synchronous architectures. The following are various events (i.e., error signatures) that can occur in a synchronous design if affected by an SET or SEU:

- Clock glitches are known to cause metastability and chaotic behavior. What is the rate of SET generation in Clock trees?
- Reset glitches are known to unexpectedly place the circuit into an initial state. What is the rate of SET generation in reset trees?
- Flip-flop upsets that are not logically masked, can cause the system to reach an incorrect state and consequently become nonfunctional. What is the rate of FF SEU generation and system capture? During analysis, it is important to make a distinction between FF SEU generation and SEU system capture – design topology will drive SEU system capture characteristics.
- Glitches in data path combinatorial logic can be captured by sequential elements. What is the rate of SET generation and capture? During analysis, it is important to make a distinction between SET generation and SET system capture – design topology will drive SET system capture characteristics

In addition to the test circuit being properly architected to reflect a real design, the test vehicle must be able to observe and identify the aforementioned error signatures. Therefore it is essential to keep the various types of error signatures in consideration during all stages of SEU test development.

Real-designs have complex topologies that can mask SEUs. As previously mentioned, it is important to investigate SEU effects in complex test structures, or real-designs. Hence, in order to determine if the test structure is a good candidate for testing, its propensity to mask SEUs must be well understood and managed.

5.2 Managing the Complexity of Test Structures

SEU test structures are developed to focus a study on the susceptibility of specific elements in the DUT-FPGA. As the complexity of the test structures increases, the ability to produce reliable σ_{SEU} data decreases. Hence, it is essential to manage the complexity of the test structure design. There are two primary concerns with complex designs that can compromise the integrity of SEU data:

- **Logic Masking:** Logic masking occurs when one or more of a gate’s inputs have logic values that block other input values from affecting the output of the gate. In this document, P_{logic} is the probability that the gates in the forward path, of the node being analyzed, will logically mask the node’s upset from being captured by the system. A gate’s logic masking is determined for each cone of logic that it affects. As illustrated in Figure 9 and Figure 10 any gate that has more than one input has the potential to logically mask an upset.
- **State Space Traversal:** Each test structure will have a state space based on the value of each of its FFs and their input conditions. The state space defines how many states the test structure can have. State space traversal, as illustrated in Figure 11, is the action of logically moving from one valid state to another.

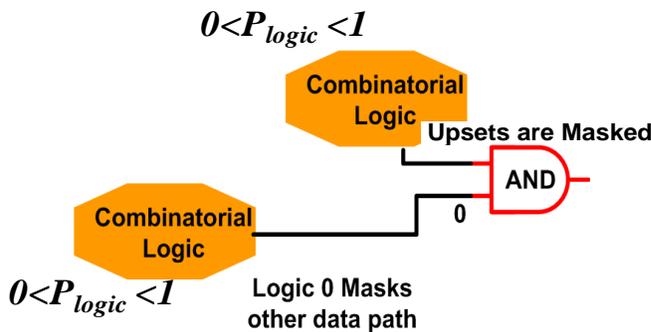


Figure 9: Logical Masking of SETs due to the state of the CL in the propagation path. Example uses an AND gate with one of its inputs in the ‘0’ state.

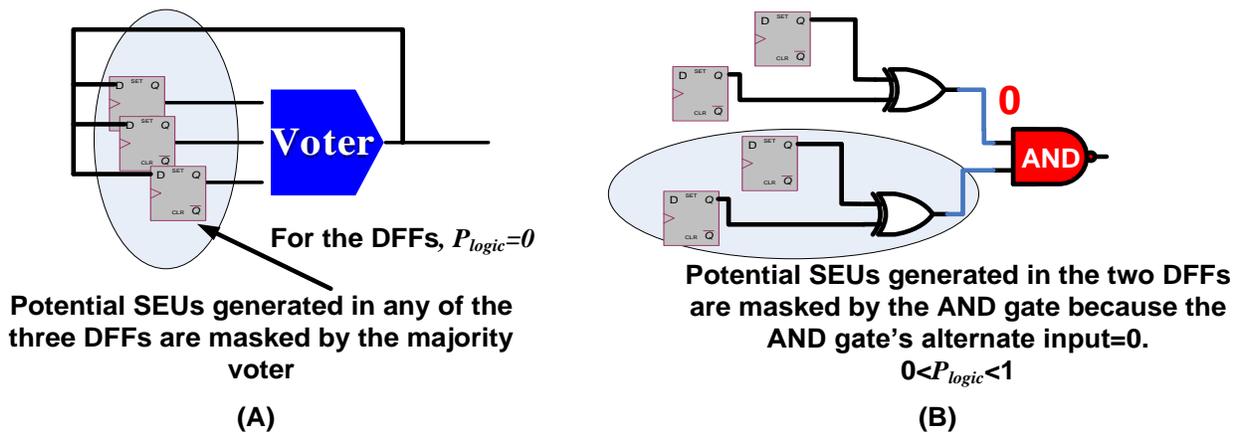


Figure 10: (A) An example of a Majority Voter masking potential SEUs from three FFs. $P_{logic}=0$ for all three FFs. Hence if one of the three FFs incurs an SEU, it will never manifest as a system upset. (B) An example of an AND gate that can mask upsets if one of its inputs is zero. Hence $0 < P_{logic} < 1$ for the FFs

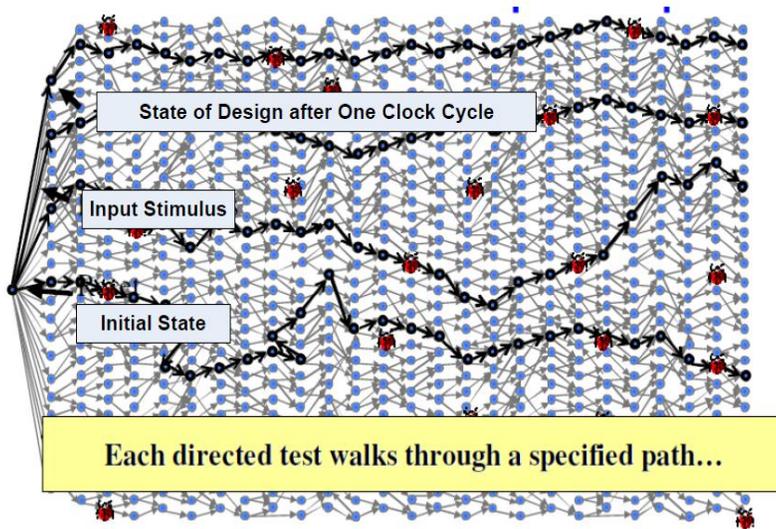


Figure 11: The complexity of state space traversal

People question whether to perform SEU testing on the mission's real designs or specialized test-circuits. Previously, recommendations were provided regarding the development of SEU test structures. Depending on the real-design, its complexity may violate most of them for the reasons listed in Table 4. Note the significance of logic masking and state-space traversal within the listed violations.

Table 4: Potential violations of best practices when performing SEU tests on complex designs.

Consideration Number	Best Practice characteristics of a DUT design	Description of how complex real-design test structures violate best-practice considerations
1	Should contain a large number of replicated logic structures in order to increase statistics.	Statistics are poor because there usually is not a significant amount of replication. In addition, trends for specific elements are not able to be clearly identified/established.
2	Its state space should be traversable such that it can be covered within one radiation test run	The state space of a complex design cannot be traversed within one radiation test run. Hence, a significant amount of circuitry and system states are not tested. The result is σ_{SEUS} that are uncharacteristic of the design.
3	Logic masking should be minimized or controllable.	Unintentional logic masking can hide upsets that would normally cause system malfunction.
4	All (or a significant percentage of) potential upsets should be observable during testing	A significant number of upsets in a complex design are generally not observable during radiation testing. This is true mostly because of logic masking, limitations in state space traversal, limitations in I/O count, or time of upset propagation to observable node.

In conclusion, the complexity of real-designs limits the ability to perform reliable SEU testing. Hence, test structures are generally test circuits geared for the specified SEU study. Currently, σ_{SEU} data obtained from evaluating variety of test-circuits are extrapolated in order to estimate mission specific SEU error rates.

The following sections describe some of the test-structures that have been used during SEU testing. It is not a complete list. Each structure presented is analyzed based on how they adhere to the above considerations plus other factors.

5.3 Original FPGA Testing Methodologies and Test Structures: Long Chain of Inverters

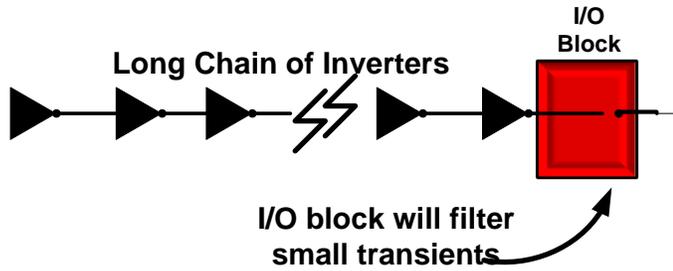


Figure 12: Test Structure used for SET characterization in FPGA devices. **NASA REAG does not recommend using this test structure.**

Figure 12 illustrates a commonly used test structure for measuring combinatorial logic SETs in FPGAs. The test structure is a long chain of serially cascaded inverters. The number of serial inverters is generally in the 100’s to 1000’s. NASA REAG does not recommend this test-structure for FPGA SEU testing for the following reasons:

- It has been proven that small SETs have the possibility to be attenuated at they propagate through the combination of combinatorial logic and routing. Because a large number of SETs can be generated but will not be observed in long chains of combinatorial logic and routing, this test structure will not provide an accurate study of SET measurement.
- The test structure is not indicative of a synchronous design. Synchronous designs must include FFs and combinatorial logic.
- SET error response is non-linear. Therefore, determining the SET cross section for one inverter will not be 1/10th the SET cross section for 10 inverters. The topology of the design will change capacitance, causes non-linear effects, and cannot be extrapolated from a long chain of inverters.
- I/O Block is slower than internal circuitry. An FPGA I/O block’s cutoff frequency is lower and will filter small transients. Hence small SETs will be unobservable.

The conclusion of using a long string of inverters as a SEU test-structure is that it will not provide SEU data indicative of a real design and hence should not be performed.

5.4 Original FPGA Testing Methodologies and Test Structures: Traditional Shift Register

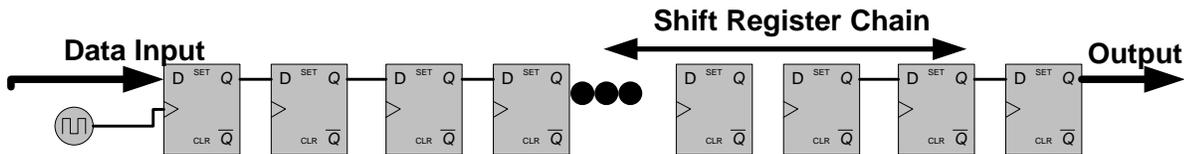


Figure 13: Traditional Shift Register only contains sequential logic.

Figure 13 illustrates a commonly used test structure for measuring sequential logic SEUs in FPGAs. The test structure is a long chain of FFs connected serially, otherwise referred to as a shift-register (SR). The number of FFs is generally in the 100’s to 1000’s. Original SEU testing evaluated SRs that were purely sequential logic, i.e., only FFs. Due to I/O signal integrity issues, the SRs were also tested at very low frequencies.

Table 5: The advantages and disadvantages of the original shift register test structures. The test structures only contained flip-flops; i.e., there were no combinatorial logic between flip-flop shift register stages.

Pro’s	Con’s
SRs are a reasonable method for measuring the susceptibility of FFs because there is no logic masking. An example of using an SR as a test structure is when mitigated FFs are built and evaluated. Placing the FFs in a SR structure is a method for analyzing the FFs SEU mitigation strength.	When attempting to calculate SEU error rates for a system, this method should not be the only test structure evaluated. The cone of logic for each FF contains only one Start-Point FF (i.e., each End-Point FF has a fan-in =1). The architectural topology on an SR is too simple as compared to a real design. Subsequently SR radiation data should not be the only source of data analysis when determining system SEU error rates.
Simple architecture – state space is traversable	High frequency testing is complicated because the output

	will switch at high speeds. This can cause signal integrity issues on the board (i.e., board level noise injection) and can consequently cause the test equipment to erroneously capture data.
Meets synchronous design requirements if each FF is connected to a balance clock tree	Testing SRs with FFs only at low frequencies can provide a fairly accurate characterization of the susceptibility of the FFs. However, low frequency testing, alone, will not be efficient to characterize the susceptibility of the system.

5.5 Evolution from Original Test Structures: Windowed Shift Registers

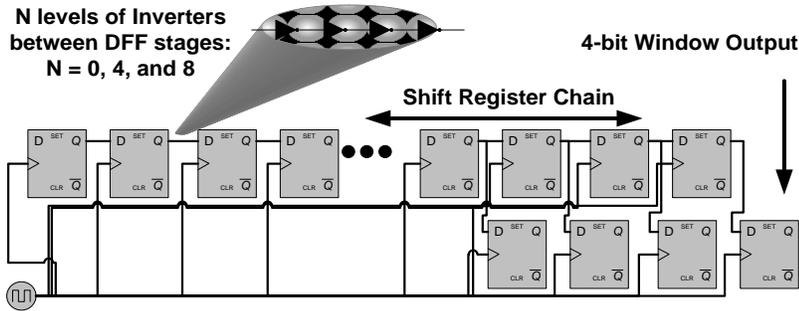


Figure 14: Windowed Shift Register (WSR). Output stays constant during testing. Simplifies data capture and gets depletes board-level signal integrity issues.

In order to improve signal integrity and facilitate high-speed SR output capture, SRs have evolved into Windowed Shift Registers (WSRs). WSRs are SRs with a serial-to-parallel output referred to as its output window. In addition to windowing the output, various levels of combinatorial logic have been inserted between each FF in a chain.

One WSR chain contains an equal number (N) of combinatorial logic blocks between each stage of FFs. Hence, if N denotes the number of combinatorial blocks between each state of FFs, then WSR_8 refers to a WSR chain with 8 combinatorial logic blocks between each FF stage and WSR_0 refers to a WSR chain with no inverters.

In order to optimize statistics by replicating circuitry, the number of FFs in a WSR chain, i.e., the number of stages in a WSR, is generally in the 100's to 1000's. The number of FF stages is dependent on the number of logic resources available in the DUT-FPGA. Refer to the DUT-FPGA data sheet for more information of resource availability and utilization.

5.5.1 WSR Data Input

Data input can be supplied to the WSR or SR by two methods:

- Data can be generated by a tester and then transferred to the DUT. This scheme requires that data be transferred from one device to another. This method is reasonable for low speed testing. However, for high speed testing it is challenging to manage the skew from device-to-device interface crossings.
- Data can be generated internally to the DUT. This scheme requires an input clock. The clock is distributed to the WSR and it is distributed to the circuitry that generates the data. Referring to the definition given by synchronous methodology, clock distribution requires that the input clock be connected to the clock pins of the data path FFs via a balanced clock tree. The usage of the input clock to generate the WSR data input guarantees that the data, WSR, and input clock are synchronous; and therefore the design will operate in a deterministic manner.

The most common data input patterns are:

- Static 0: data input is a constant logic '0'
- Static 1: data input is a constant logic '1'
- Checkerboard: data input changes its logic value every clock cycle ("10101...")
- Half-rate checkerboard: data input changes every other clock cycle ("1100110011001100..")
- Random: each logic state of the data pattern is randomly selected.

If the data pattern is generated inside of the DUT, and multiple data patterns can be selected, then there must be a data pattern selection scheme. Figure 15 is an example of using a MUX, internal to the DUT-FPGA, to select which data pattern to use during a test. In this example, there are 4 or less data patterns to select, hence, two bits are required to control the MUX. The two bits must be input to the DUT so that the user can have control of the data input pattern selection per

radiation test.

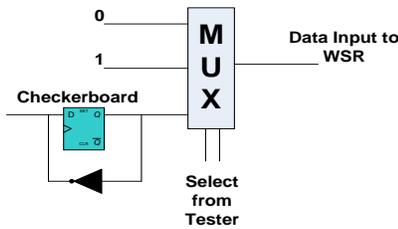


Figure 15: WSR Internal Data Input Circuit. Possible data patterns in this diagram are Static-0, Static-1, and Checkerboard.

Figure 16 illustrates connecting the WSR to an internal data generator.

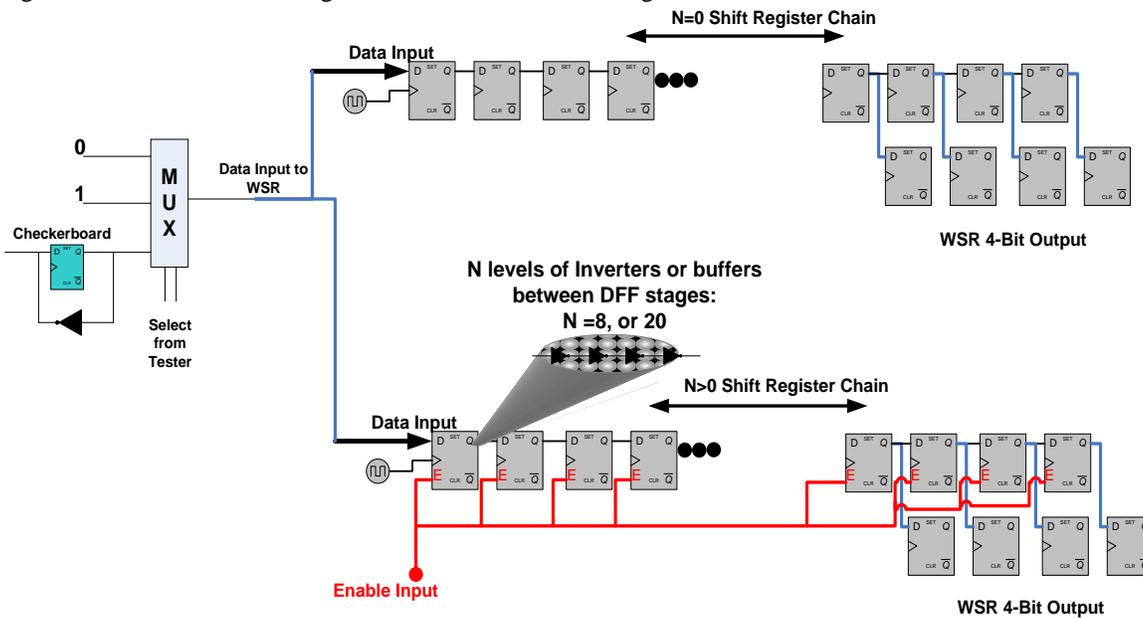


Figure 16: WSR Shift Register Strings with Optimal Combinatorial Logic. All FFs in one chain are connected to the same clock input and the same reset

5.5.2 WSR Functional Description

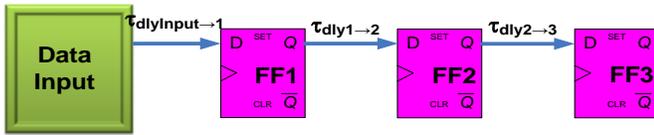
WSR's are created with the following considerations:

- Receive an input clock such that the test vehicle can vary WSR frequency.
- Have the ability to operate at the maximum frequency of the WSR chain in order to study the limits of SET capture.
- Simultaneously shift data through its chain of FFs every clock cycle
- Create a window from the DUT to the tester to minimize signal integrity issues. This is accomplished by capturing the last K bits of the shift register into a window of FFs, once every K clock cycles, where K is the size of the window. As an example, for a WSR with a 4-bit window (as illustrated in Figure 20), the last 4 bits will appear in the output window once every 4 clock cycles

The test vehicle is expected to monitor the output window for upsets. However, an alternative is to use an internal comparison circuit and have the test vehicle monitor the comparison outputs. If internal data checking is used, then it is essential to make the comparison circuitry redundant. In order to avoid single points of failure in the comparison circuit, mitigation of the redundant comparison circuits should be performed in the test vehicle.

5.5.3 WSRs and Frequency Control

End-Point	Start-Point	Data Path Delays
FF3	FF2	$\tau_{dly2 \rightarrow 3}$
FF2	FF1	$\tau_{dly1 \rightarrow 2}$
FF1	Input	$\tau_{dlyInput \rightarrow 1}$



Maximum frequency of operation depends on the maximum τ_{dly}

Figure 17: Shift Register cones-of-logic. Each flip-flop is treated as an End-Point with its Start-Point being its input flip-flop. All data paths have a unique τ_{dly} .

Depending on the FPGA's logic block structures, σ_{SEUS} may be frequency dependent. Subsequently, it is essential to evaluate σ_{SEUS} at a variety of frequencies to analyze trends. As illustrated in Figure 18 it has been shown that:

- Designs with well mitigated FFs will produce SEU cross sections that are directly proportional to frequency
- Designs with poorly mitigated FFs will produce SEU cross sections that are inversely proportional to frequency
- σ_{SEUS} can differ by decades based on the frequency of operation during testing. Hence, for error rate calculations, it is essential to use σ_{SEU} data that was obtained using a similar frequency as the target design.

Non Mitigated and Mitigated WSRs with the ProASIC3... Regard the Frequency Trends

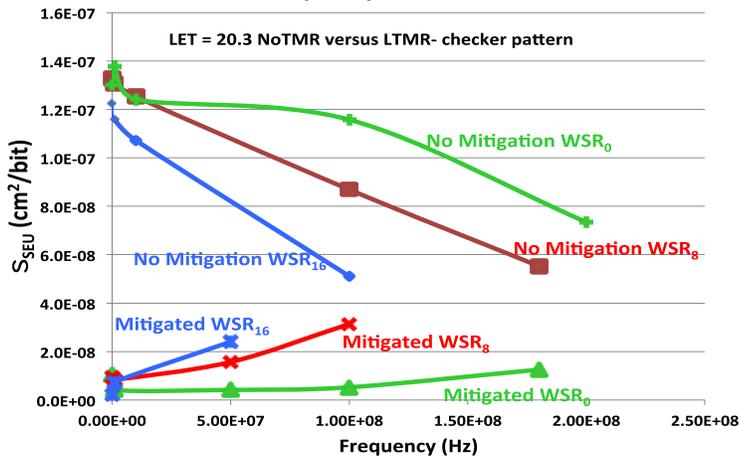


Figure 18: ProASIC3 Heavy Ion testing illustrates that WSR strings with non-mitigated FFs are inversely proportional to frequency. WSR strings with mitigated FFs are directly proportional to frequency. The mitigation strategy used is Localized Triple Modular Redundancy (TMR) [17].

Determining how fast a WSR operates, depends on the data path with the longest τ_{dly} . STA tools are used to provide the maximum τ_{dly} and hence the maximum operating frequency. As an example: a design with a reported maximum τ_{dly} equal to 9.8ns will operate at frequencies $< 1/9.8ns$. If time permits, the test plan should also require at least 4 frequencies to be tested spanning at least two decades. In this case, the test plan would incorporate irradiating the FPGA from 1MHz to 100MHz.

5.5.4 WSRs and Routing Control

As previously stated in Section **Error! Reference source not found.** and illustrated in Figure 18, each FF in the WSR chain is treated as a cone-of-logic End-Point. Its Start-Point is the previous FF in the chain. There is a delay between each Start-Point to End-Point (τ_{dly}). τ_{dly} will determine the maximum frequency the entire WSR chain can operate – i.e., the slowest path (greatest (τ_{dly})) dictates the clock speed of the WSR chain. τ_{dly} and clock speed (τ_{clk}) influence SEU data by the following:

- SETs: It has been shown [11] that the ratio of transient width (τ_{width}) to clock speed (τ_{clk}) will affect the probability of SET capture as follows: as the ratio of τ_{width} to τ_{clk} approaches 1, the probability of capturing an SET is increased.

Hence, in order to measure upper bound σ_{SEU} , it is essential to test with the smallest τ_{clk} – i.e., the maximum frequency. Statistically, it is also important that each data-path have approximately the same τ_{dly} .

- SEUs: Due to the τ_{dly} from a Start-Point to an End-Point, the probability of the End-Point being affected by a Start-Point SEU is decreased as τ_{dly} approached τ_{clk} . When studying SEU capture, it is also essential to keep τ_{dly} consistent across test structure data paths in order to maximize the integrity of statistics.

Taking the above information into account, keeping τ_{dly} consistent between each stage of a WSR chain will increase the integrity of SEU data. This is primarily because controlling τ_{dly} facilitates each WSR stage to have similar probabilities of SEU or SET capture

It is important to note that the routing of element to element will affect τ_{dly} . Long routes produce longer τ_{dly} . The normal FPGA design flow process utilizes an automated tool to place the WSR gates into the FPGA element cells. However, the automated tool will not place the cells in such a way where τ_{dly} is consistent for each stage. In order to increase SEU data integrity, it is best practice to manually-place the stages in a WSR such that the τ_{dly} from stage to stage is approximately equal.

5.5.5 WSR Output

The WSR output is the 4-bit window of the shift register. For a data pattern of all 0's, the WSR window output will be all 0's. For a data pattern of all 1's, the output will be all 1's. For a checkerboard pattern, the last 4 bits of the shift register change every clock cycle. Because the WSR window is a snapshot of the last 4 shift-register bits every 4-clock cycles, the window stays static. Table 6 lists data input patterns with expected window output for a WSR. The operation is illustrated in Figure 19.

Table 6: Data Input pattern and expected Window Output

Data Input	4-bit Window Logic Output Value
Static 0	“0000”
Static 1	“1111”
Checker board	“1010” or “0101”... depends on when reset is released
Half-rate checker board	“1100” or “0011” or “0110” or “1001”... depends on when reset is released
Random	Output is not static – has not been used with WSR testing

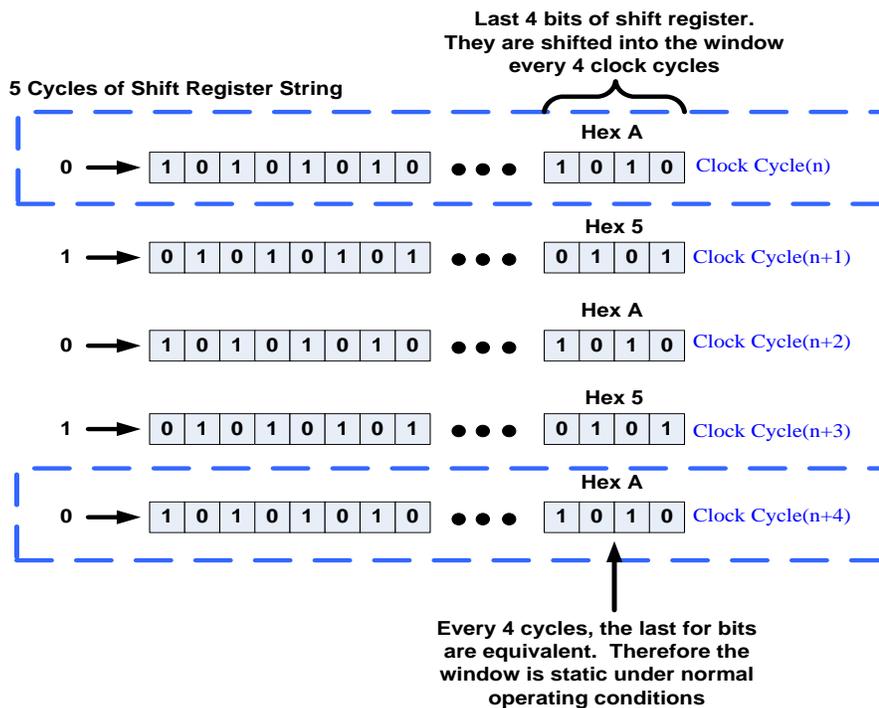


Figure 19: WSR shift register operation for a checkerboard input. Every 4-clock cycles the last 4 shift register bits are equivalent. Every 4-clock cycles the window gets a snap shot of the last 4 bits of the shift register. Consequently, the window is static under normal operating conditions

If there are enough I/O available, it is best to have the last bits of the shift register fan-out to two windows instead of one. In this case, it is easier to detect a bit flip in the window versus a bit flip in the shift register.

5.5.6 WSR Expected Upsets

Because of the WSR structure, the string outputs are expected to be constant after the length of the string cycles following reset de-assertion. Therefore, an error is easily detected by monitoring any change within the WSR outputs as illustrated in Figure 20: Example of WSR SEE DUT output to tester.

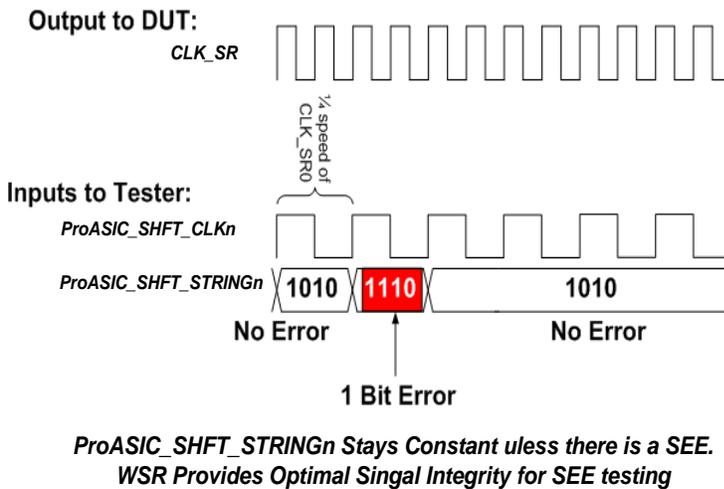


Figure 20: Example of WSR SEE DUT output to tester

Primary Expected WSR SEUs:

- Bit flip in shift register: Will be observed in the WSR window for 4 clock cycles (because window can only change once every 4 cycles).
- Bit flip in window: Upset will be observed for less than 4 clock cycles
- Output transient: May not be able to distinguish from bit flip in window.
- Global routes: An upset can occur in the clock or reset circuitry or enable circuitry (4 out of the 6 strings have enables).

5.5.7 WSR Pros and Cons

Table 7 lists the pros and cons of using WSR strings. WSRs have alleviated a most of the cons from inverter chains and traditional SRs. WSRs prove to be a formidable method for testing FF mitigation strength and to analyze combinatorial logic effects. However, in order to achieve a more comprehensive study regarding the susceptibility of actual FPGA design operation, trends in σ_{SEUS} across design complexity should be evaluated. Subsequently, it is recommended that WSR SEE testing be complimented with additional testing with more complex designs.

Table 7: WSR Pros and Cons

Pro's	Con's
SRs with and without combinatorial logic between FF stages are a reasonable method for measuring the susceptibility of FFs because there is no logic masking.	The WSRs with combinatorial logic between FF stages have more complexity than purely sequential SR's, due to the addition of combinatorial logic. However, the cones-of-logic are still very simple compared to actual designs. Test structures with fan-in and fan-out should also be evaluated.
Simple architecture – state space is traversable	
Meets synchronous design requirements if each FF is connected to a balance clock tree	
High Frequency testing can be performed without the SR	

causing signal integrity issues.	
High Frequency testing can be performed facilitating reliable DUT shift-register capture	
All nodes are observable by the tester	
The inclusion of strings with combinatorial logic facilitates evaluation of combinatorial logic effects, i.e., SET capture	
Meets synchronous design requirements if each FF is connected to a balanced clock tree	

5.6 Evolution from Original Test Structures: Complex Test Structures

As previously mentioned, WSRs are an efficient method of testing FF SEU behavior because they have no logic masking. However, due to their linear topology (fan-in=fan-out=1), WSRs lack the complexity of a real design. Increasing SEU test-structure complexity requires increasing the design's cone-of-logic while retaining a traversable state space. Complex test structures that have been successfully tested and meet SEU test requirements are:

- Counters: Counters are circuits that increment each clock cycle. They are built out of FFs and Combinatorial logic. Each bit (FF) of a counter has a unique cone of logic. All bits except for the least significant bit have cone's of logic with more than one Start-Point (i.e., fan-in >1). In order to comply with statistics, 100's of counters should be designed into the test-structure. Due to the limitation of the number of available outputs in an FPGA, each counter cannot be directly output to a tester simultaneously. The challenge becomes how to have visibility into each of the counters; i.e., how to detect if a counter has become upset. There are two schemes that support counter test structures:
 - Cascade the counters serially such that one counter feeds the next. This is similar to a shift register. The difference is the each FF is replaced with a counter. In this case, the counter becomes an accumulator.
 - Create a parallel bank of counter and devise a mechanism to output each counter one at a time to the tester.
- Digital Signal Processing (DSP) blocks: DSP blocks are complex circuits that perform data operations such as: adders, accumulators, multipliers, dividers, filters, etc.... Counters can be categorized as a DSP. However, they have been separated in this document because of complexity. Counters are created with less complex circuits as compared to the various DSPs listed.

When the goal of SEU testing is to extrapolate test data for "real-design" error rate calculations, the focus is on determining the probability that an SET or SEU will affect the value of a cone-of-logic End-Point. End-Point errors manifest as system errors when one of the following occur:

- The End-Point flips its state, i.e. End-Point SEU,
- The End-Point captures an incorrect computation from one of its erroneous Start-Points, i.e. Start-Point SEU, or
- The End-Point captures a combinatorial logic Single Event Transient (SET).

It is best practice to evaluate a variety of test structures that differ by cone-of-logic sizing. Comparing σ_{SEU} data from the differing test structures develops trends that can be used to guide data extrapolation. The affects of varying cones-of-logic are the following:

- Adding more Start-Points, i.e., increasing End-Point fan-in. Increasing an End-Point's cone-of-logic can produce more observable upsets at lower LETs. This will depend on the potential logic masking per cone.
- Adding more combinatorial logic. Studying combinatorial logic gates and their susceptibility affects in a cone-of-logic is an evaluation of SET capture in a design.

5.7 Test Structures with Built-In-Self-Test

High frequency SEU studies can be challenging because of signal integrity and data capture issues; each of which are due to DUT-tester interface signal crossings. One approach is to avoid passing signals from DUTs to their tester; e.g., placing error detection and/or DUT control inside the DUT test structure. Examples of preferable circuits to place internal to a DUT versus the tester are clocks and data inputs (e.g., a WSR data input). SEE Error detection circuits internal to DUTs are referred to as Built -In-Self-Test (BIST).

The following two sections are examples of common BIST SEE test structures. Both BIST examples use internal circuits to compare DUT internal values. A mismatch between the values signifies an SEU and it is a trigger that is sent to the test vehicle.

5.7.1 Circuit for Radiation Effects Self Test (CREST)

A common technique of shift register SEU BIST is the CREST test structure [34]. The CREST DUT structure consists of 5 primary blocks. The blocks are illustrated in Figure 21 and Figure 22 and are as follows:

1. Data Source: The data source block generates the data that is fed to the shift register. Data is generated pseudo-randomly using a Linear Feedback Shift Register (LFSR). The LFSR must have $\text{Log}_2(\#\text{FFs in shift register}+1)$; e.g., for a shift register that has 127 stages, the LFSR must have 8 stages.
2. Test Structure: The test structure is a shift register.
3. Data Saving FIFO: The data saving FIFO is not actually a FIFO structure. It is another shift register. During error capture, it is expected to contain the last 8 values in the test structure prior to error.
4. Error Detection and Latch Circuitry (EDLC): The EDLC compares the last stage of the shift register to the output of the LFSR. Based on this architecture scheme, the two should always be equivalent except for an error event. Upon an error event, the compare circuit triggers a self-clearing Error flag.
5. Clock Control: Clock control is illustrated in Figure 22. In order to increase statistics, a CREST circuit will have several blocks of logic listed in Figure 21. Each will contain its own error flag. All of the error flags coalesce into a clock control circuit. Upon an error, the fast clock is turned off. Hence operation ceases. A slow clock is turned on so that the information in the data-saving FIFO and test structure can be shifted into the tester at a low frequency. Subsequently, the slow clock is used for reliable data capture and reduces signal integrity issues.

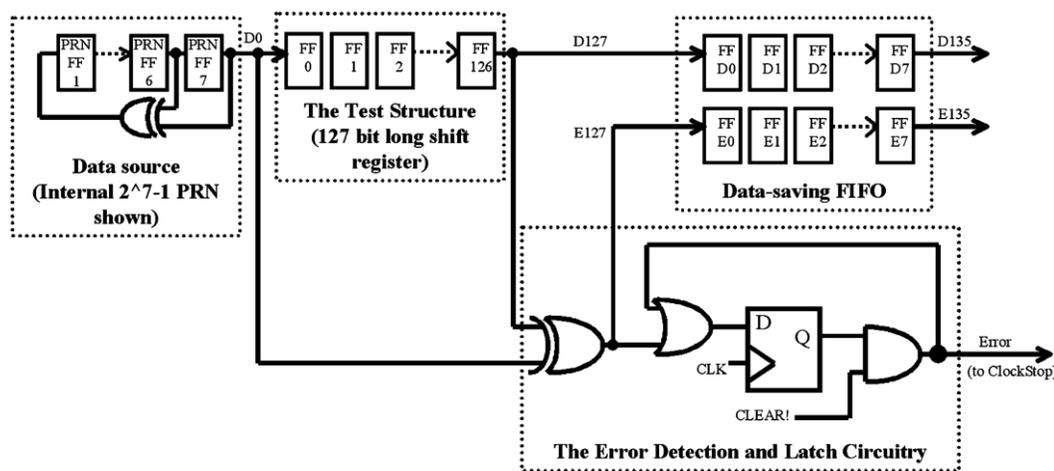


Figure 21: CREST Data Generator, Shift Register, and Error Detection Circuitry

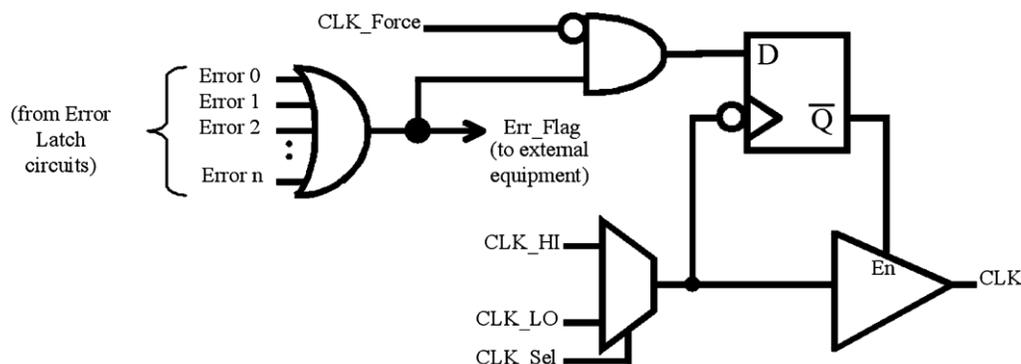


Figure 22: Error Flags generated Error Detection Circuitry control clock selection. High-speed clock is used during normal operation while low speed clock is used during

Table 8: CREST Pros and Cons

Pro's	Con's
Minimizes signal integrity concerns because all number of I/O is reduced and interface to tester is low frequency	Clock jitter: The design requires the user to create an internal high-speed clock. This can be a challenging task – and is the reason that oscillators are purchased. User created clocks generally have clock jitter and poor duty cycles.

	These characteristics will impact system operation at high frequency.
Simple architecture – state space is traversable	<p>Stop operation: upon each upset, normal shift operation must cease. Clocks are exchanged so that the internals of the DUT can slowly be shifted to the tester. The user has two choices:</p> <ul style="list-style-type: none"> • Completely stop each test upon upset – i.e., turn the beam off. This is the easiest solution, but will reduce statistics; i.e., it is optimal for the tester to keep recording upsets during each run – a goal is to reach 10’s-100’s of upsets for proper statistical event handling (data processing). • Keep the beam on after each upset but adjust the flux to accommodate time that upsets are being shifted to the tester. During the shift out period, SEUs in the shift register will not be properly recorded. This scenario can be challenging to manage
Meets synchronous design requirements if each FF is connected to a balance clock tree	A data source upset in the LSFR can look like a noisy burst upset (difficult to differentiate from a clock upset)
	Flexibility and control is minimized because all control is internal to the DUT
	Visibility is limited to the user and the tester

5.7.2 Built in Dual Redundant Test Structures

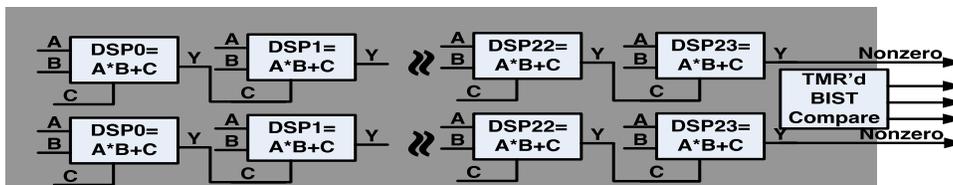


Figure 23: Example of a Dual Redundant BIST

BIST dual redundant test structures are DUTs that contain redundant circuits. The redundant circuits are internally compared inside of the DUT. Figure 23 is an example of a Dual redundant BIST structure. It illustrates two cascaded strings of multiply-accumulate logic blocks. Such strings are commonly used to implement finite impulse response (FIR) filters. The final stages of the redundant FIRs are compared. Upon error, the compare circuit signals the tester. Best practice when implementing a dual redundant BIST is to mitigate the compare logic so that reported upsets can be isolated to the test circuits. Regarding the example illustrated in Figure 23, the compare circuitry is triplicated; i.e., there are three compare circuits. The outputs of each compare are sent to the tester so that the tester can differentiate if there is an upset in the test circuits versus a compare circuit.

Table 9: Dual Redundant Pros and Cons

Pro's	Con's
Minimizes signal integrity concerns because all number of I/O is reduced and interface to tester is low frequency	Synchronization: Careful consideration must be made to insure that the dual circuits are synchronized so that compares are reliable.
Simple architecture – state space is traversable	Re-synchronization post upset: Depending on the architectures and depending on the error response, the redundant circuits can become unsynchronized. In this case, the compare will not operate correctly. Consequently it will appear as a string of events are occurring – while this is not

	the case. The problem is that the redundant strings are unsynchronized and the compare isn't operating correctly. This is not an issue for all architectures. However, if it is, a re-synchronization solution can be implemented (e.g., resets).
Meets synchronous design requirements if each FF is connected to a balance clock tree	
	Visibility is limited to the user and the tester

5.8 Test Structures with Inserted Mitigation

There are commercial FPGA devices that are not made for critical applications and do not contain RHBD Circuits [5]-[7]. Hence, a design without mitigation will have substantial upsets in its data path and potentially in its configuration (depending on the configuration technology).

Commercial devices used as-is may not satisfy the requirements for critical applications. However, if mitigation is designed into the circuitry, via logical masking and correction circuits, then the commercial device may be a candidate for system use [11][27]. In this case, it is essential to investigate how much mitigation is required and what is the effectiveness of adding mitigation.

Error correction codes such as single error correction double error detection (SECDED) are popular correction schemes for reading and writing memories. However, SECDED and other correction codes are not effective methods for protecting circuit data path upsets. The most common user-applied data path mitigation is Triple Modular Redundancy (TMR). TMR has many forms of application. The various types of TMR techniques are differentiated by the portions of circuitry that are replicated and how they are mitigated. The TMR Mitigation strategies are:

- No-TMR: no additional circuitry is added to the design pertaining to SEU mitigation
- LTMR: Localized Triple Modular Redundancy. Only FFs are triplicated. Combinatorial logic paths, Clocks, and resets are shared and consequently single sources of failure. With this mitigation strategy, only the effects of *FF SEUs* are reduced (because they are masked). However, susceptible circuitry remain as such: transients in data path combinatorial logic can be captured by End-Point FFs and Global routes can cause Single Event Functional Interrupts (SEFI). Figure 24 is an illustration of applied LTMR.
- DTMR: Distributed Triple Modular Redundancy. The entire design is triplicated except for global routes (clocks, resets, and high fanout enables). This mitigation strategy reduces data path upsets. However, since the global routes are not mitigated, then transients on global routes can still disrupt the system. Figure 25 is an illustration of applied DTMR.
- GTMR: Global Triple Modular Redundancy. The entire design is triplicated including global routes (clocks, resets, and high fanout enables). This strategy mitigates most upsets. However, some FPGA have additional logic outside of the data path that cannot be mitigated. In this case, GTMR will effectively reduce the upset rate, but will still have some points of failure.
- BTMR: Block level Triple Modular Redundancy. The entire design is triplicated. The outputs of the replicated blocks are voted. The inputs may or may not come from a common source. However, if the I/O are not fanned out to the replicated blocks from a common source, voting will be unreliable due to synchronization issues. This scheme only provides masking capability and does not correct errors. Consequently, this technique is only practical for a design that can regularly be reset. In this case, upsets are regularly flushed and the design can be forced to reach a deterministic state.

Regarding SEU susceptibility, GTMR is attractive because it has the highest level of TMR mitigation. However, in most FPGA devices, applying GTMR is infeasible. This could be due to the lack of clock trees in the FPGA or due too much skew between the clock trees. In addition, the amount of power and area required to implement GTMR can restrict the FPGA from being considered for system implementation.

The trade-off per TMR mitigation strategy is the reduction of susceptibility versus resource utilization and power consumption. Hence it is essential to test and evaluate a variety of mitigation schemes so that the design team can perform a proper trade of which mitigation strategy to implement.

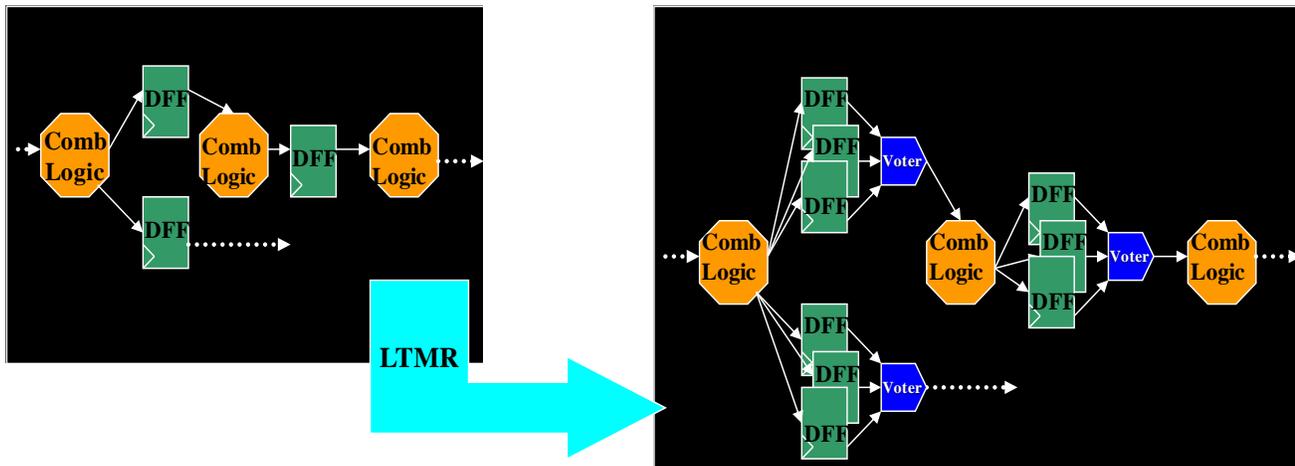


Figure 24: Applied LTMR ... Only the FFs are triplicated. Consequently data inputs to each FF are shared and are single points of failure

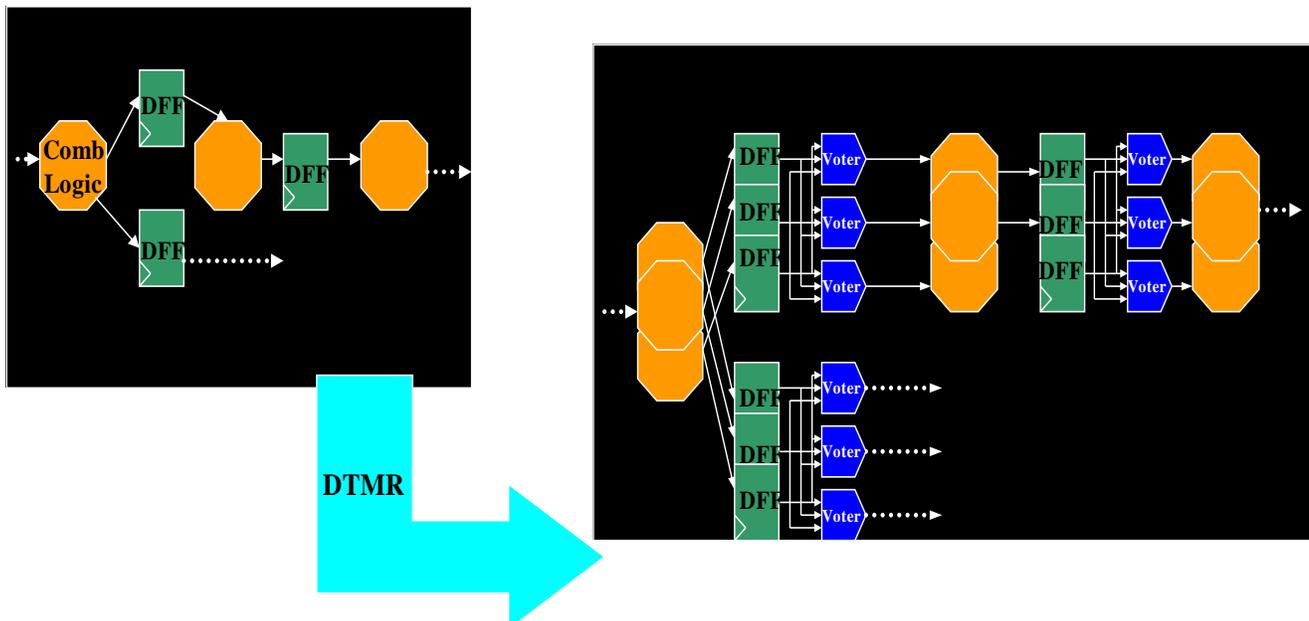


Figure 25: Application of DTMR. All functional logic is triplicated except global routes (Clocks and Resets are not triplicated)

5.9 Summary of Presented SEU Test-Structures

The proposed test procedures will depend on the intended target of FPGA SEU characterization. Hence, it is essential to clearly define the goal of the SEU study. For example:

- Is the plan to evaluate individual components?
 - FF mitigation strength: shift registers have no logic masking and are good test structures for studying FF susceptibility
 - Configuration elements: test procedures will depend on the accessibility and the sensitivity of the configuration
- Is the plan to study system susceptibility?
 - System evaluation: A combination of test-structures is best for design studies. Shift registers are a good reference point because they have no logic masking. Complex test-structures are a good method to study trends regarding system topology due to their larger cones-of-logic.
 - Error rate calculations: In order to calculate error rates, SEU data is extrapolated. A combination of test structures assists in evaluating trends and hence assists in data extrapolation.

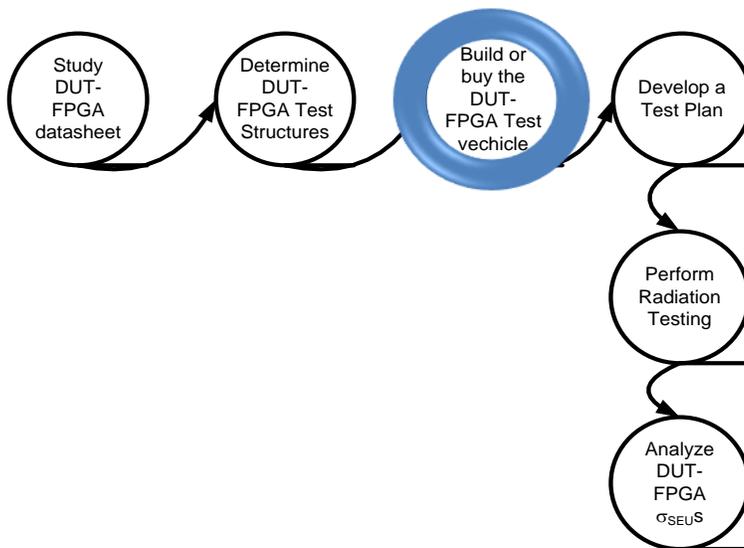
Table 10 lists the test structures described in this section and includes a summary of their advantages and disadvantages regarding SEU characterization.

Table 10: Summary of Presented, Evolved SEU Test Structures

DUT Test-Structure	Primary Advantage	Primary Disadvantage
Windowed Shift Register (Widowed Shift Register)	<ul style="list-style-type: none"> No Logic masking. Best method to measure the mitigation strength of FFs. Testing WSRs that differ by the number of combinatorial logic between FFs facilitates SET analysis. 	Lacks cone-of-logic complexity. Can inaccurately characterize system level SEU susceptibility
Complex Test Structures	<ul style="list-style-type: none"> Adds complexity to the cone-of-logic. Increasing an End-Point's fan-in can increase the visibility of events at lower LETs. Evaluating σ_{SEUS} from a variety of complex designs facilitates the development of trends. The trends are used to facilitate extrapolation of data for error rate calculations 	Logic Masking is significantly higher than a WSR. Not an efficient method to measure FF SEU susceptibility.
Test structures with user applied mitigation strategies	<ul style="list-style-type: none"> Measures the effectiveness of a variety of mitigation strategies Facilitates performing a trade between mitigation schemes for critical design applications 	

Once the test-structures have been determined, they need a test vehicle that will supply input stimulus and monitor the DUT outputs during radiation testing. The following section provides guidelines and considerations for building FPGA test vehicles.

6 SEU TEST VEHICLE DEVELOPMENT

**Figure 26: General flow for developing a SEU test strategy**

As previously mentioned, the SEU test vehicle is responsible for applying DUT stimulus and for monitoring DUT outputs. Monitoring DUT outputs requires the test vehicle to be able to identify and report upsets. The following is a synopsis of the responsibilities of a test vehicle:

- Provide input stimuli to the DUT:
 - Functional control:
 - Types of DUT functional input control: Clocks, resets, data inputs
 - Devices that can perform functional control: Functional generators, Computers, (semi)custom FPGA test boards
 - Concerns:
 - Managing frequencies of operation: high-speed control can be challenging. Determine whether the selected test vehicle can supply inputs as required; e.g., can the test vehicle provide the full range of frequency as stipulated by the test requirements
 - Synchronizing inputs and managing skew between inputs.
 - Operating the device in a realistic manner:
 - Do not over load the device with unrealistic stimulus during radiation testing. If the device is operating in states that would never occur, then radiation data will not be characteristic
 - Do not under load the device during radiation testing. If the device is underperforming, this means that a large amount of circuitry is not operating. This produces operational states with a large amount of logic masking; and consequently, radiation data will not be characteristic.
 - Power control:
 - Types of voltage controllers: power supplies and special on-board voltage regulation circuitry.
 - Concerns:
 - Device may draw a larger amount of current than originally expected. Cooling apparatus may be necessary during operation
 - Power glitching or Single Event Latch-up (SEL) can cause the system to cease operation or be damaged. Hence it is best practice to separate test vehicle power from DUT power. It is also ideal to have current limiting circuitry for the test vehicle and the DUT.
- Monitor DUT outputs:
 - Functional error detection:
 - Types of error detection equipment: oscilloscopes, logic analyzers, computers or (semi)custom FPGA test boards
 - Error detection logistics:
 - Compare DUT outputs to expected values. This can be done:
 - Visually (not recommended); i.e., watching the error indication on the error detection equipment
 - Using equipment event triggers
 - Custom comparison circuitry
 - Differentiate upset types: e.g., clock tree SET, FF SEU, combinatorial logic captured SET, or configuration fault.
 - Count SEUs (upset statistics): After the upsets have been detected and differentiated, they need to be counted. The higher the number of upsets, the better the statistics.
 - Voltage and current monitoring.
 - Can be performed using power supply monitors or specialized on-board (tester) circuitry
 - As previously mentioned, the ability to automatically power down or limit current if the DUT current gets too high is beneficial

6.1 *Developing the DUT-FPGA Test Board*

As previously mentioned, the test vehicle is responsible for supplying the input stimulus and monitoring DUT outputs for potential upsets. In order for the DUT to be controlled and monitored, it must be mounted on a board. The following are several issues that should be taken into account while selecting a DUT-FPGA board:

- Socketing versus soldering the DUT-FPGA to its board: In the case of damaging a DUT during testing or having a large number of DUTs to test, it can be beneficial to socket the DUT-FPGA onto its board. In such cases DUTs can be replaced and boards can be reused. However, when testing at angle, it should be taken into account that sockets can shadow the DUT and hence limit heavy-ion angular tests.
- Using high-grade interfaces: It is beneficial to have high-grade DUT-FPGA interfaces for high frequency operation
- Accessing a large number of DUT-FPGA I/O: As the complexity of DUT-FPGA test structures increase, the

number of nodes that should be monitored during testing also increases. In order to obtain a reasonable amount of visibility into the internal state of DUT operation, it is beneficial to have a large number of DUT-FPGA I/O available to the test vehicle

- Obtaining board development expertise: Boards with a large number of I/O operating at high speeds will have signal integrity issues. Voltage regulation is another issue. Hence, if the plan is to build custom boards, it is essential to have the proper expertise in place.
- Testing a variety of angles with heavy-ions: Heavy-ion angular testing requires changing the angle of incidence of the DUT-FPGA to the heavy-ion beam. If other devices are near the DUT, the other devices can shadow and degrade beam penetration into the sensitive region of the DUT
- Separating other devices from the DUT for proton testing: If the DUT is surrounded by other devices on its board, the other devices should be shielded during proton testing. Otherwise their upsets due to proton and neutron scattering can affect DUT evaluation. As proton energy decreases, this becomes less of an issue.

While taking into account the issues that exist with DUT-board and SEE testing, the trade-off becomes whether to: (1) buy a commercial-off-the-shelf (COTs) board that already contains the DUT and potentially other control devices that can act as the test vehicle, (2) build a custom test board and DUT board, or (3) develop a semi-custom test system. Table 11 lists the Pros and Cons for each option.

Table 11: Various Options for DUT-FPGAs with their Pros and Cons.

DUT-FPGA Board acquirement scheme	Pro	Cons
Buy a commercially available evaluation board (COTs board) with the DUT already mounted.	<ul style="list-style-type: none"> • Good option for very simple tests that do not require a significant amount of control and do not require a significant amount of output monitoring. • Quickly available and does not require the expertise of a team to build a board • High frequency testing of test-structures is feasible. However, a significant amount of the monitoring must be performed internal to the DUT-FPGA 	<ul style="list-style-type: none"> • Limited I/O and control • In ability to socket the DUT-FPGA (Hence, may need to buy a large number of evaluation boards) • Limited angular access for heavy ion testing • Problem with other devices on the evaluation board having latch-up during proton testing – e.g., SRAM • DUT-FPGAs with internal monitoring have reduce visibility regarding the state of operation
Build a board containing the DUT-FPGA with expectations that the test vehicle will interface to the board.	<ul style="list-style-type: none"> • The option of socketing the DUT-FPGA is available • Heavy ion angular access can be maximized • The test vehicle is on a different board, hence it is easier to shield everything excluding the DUT-FPGA during proton testing • Reusability of the test vehicle is an option • The test vehicle can be custom or COTs • High frequency testing of complex test-structures is feasible 	Requires board development
Build one board that contains the test vehicle and the DUT.	<ul style="list-style-type: none"> • The option of socketing the DUT-FPGA is available • Heavy ion angular access can be maximized • High frequency testing of complex test-structures is feasible 	<ul style="list-style-type: none"> • Requires board development • Reduced reusability of test vehicle • Problem with other devices on the board having latch-up during proton testing – e.g., SRAM

As a summary, the primary difference between DUT boards is flexibility versus ease of development. Once the DUT board design is determined, the DUT-FPGA to tester interface is assigned. The test vehicle is constructed based off of the interface, test-structures and SEU testing goals. The following sections describe a variety of test-vehicle options.

6.2 Original Test Vehicles

Because original test structures were shift registers, their interface and control were simple. Function generators were used to create DUT stimuli. Oscilloscopes and/or logic analyzers were used to capture DUT output.

When using function generators as input stimuli, care must be taken to guarantee that data-input is synchronized with the clock. Hence, it is best practice to use one function generator that generates multiple-synchronized signals –e.g., one for the clock and one for the data input. However, if it is necessary to use two function generators, the generators must be synchronized to keep their signals in sync. It is important to note that the granularity of synchronizing two generators is usually in 10's to 100's of ns. Because of this, high speed testing cannot be reliably performed with two generators. See the manufacturer datasheet on synchronizing function generator outputs.

Table 12: Original Shift Register Test Vehicle to DUT Interface

Signal	Interface Direction with respect to DUT	Device
Clock	Input	Function Generator
Reset (not mandatory)	Input	Function Generator
Data Input	Input	Function Generator
Data Output	Output	Oscilloscope or Logic Analyzer

For high frequency tests, it is best practice to use test structures that internally generates shift-register data so that the only necessary input is the clock. See section 5.5.1 for more detail regarding the generation of internal shift-register data.

The scheme for monitoring DUT-FPGA outputs in the original test vehicles is to set SEU event triggers in the logic analyzers and oscilloscopes. The limitation with using triggers is that they unreliably capture and report SEU information. Table 13 is a more detailed list of data monitoring and encapsulation limitations when using original test vehicles for SEU studies.

Table 13: Limitations with data monitoring and encapsulation with original test vehicles

Limitations using logic analyzers or oscilloscopes to capture data	Explanation
Limited I/O monitoring	Logic analyzers and oscilloscopes can manage capturing data output for simple test structures with a small number of I/O. (semi)custom testers are essential for designs with a large number of I/O
Limited time stamping capabilities	There is a substantial time delay with an unpredictable margin of error from when an error event occurs to when a logic analyzer or oscilloscope can capture and report the event.
Missed Events	Event triggers can be missed due to the time it takes the test equipment to download and record events. During the download and record interval, triggers are either disabled or limited. Because the goal of SEU testing is to count the number of upset events per number of particle exposure, for this case, the test equipment can reduce the integrity of test results.
Upset differentiation	Because when using logic analyzers and oscilloscopes there is a limited amount of information per event, it can be difficult to differentiate between upset types
Frequency	The original SEU test structures were shift registers (SRs). The board-level noise produced from SRs operating at high frequencies causes unreliable data capture by the test vehicle. Hence original test vehicles operated at low frequencies. The use of the WSR test structure minimizes signal integrity issues such that DUT output capture is reliable. However, contemporary logic analyzers and oscilloscopes can be constrained by the number of required I/O and by the inability to capture consecutive errors.

6.3 Evolution of Test Vehicles: (semi)Custom SEU Testers

As frequency, number of outputs, and DUT-FPGA functionality increase, using logic analyzers and oscilloscopes as the

source of data stimulus and capture become impractical. A common solution is to build a system specifically for testing a DUT. Automated Test Equipment (ATEs) have been built by manufacturers to perform reliability testing for a long time. Within the past decade, it has become popular to use ATE test vehicles for SEU testing.

The following are options for building an ATE:

- Fully custom testers: The test vehicle is designed from scratch to specifically meet the needs of the SEU test structure
- Semi-custom testers: The test vehicle is created by modifying an existing test set-up. The modifications are made to specifically meet the needs of the SEU test structure. This implies reusability and is best implemented with using reconfigurable FPGAs as the test vehicle controller. Hence, in this instance, an FPGA is testing the DUT-FPGA.

SEU ATEs have three primary components:

- A DUT-FPGA controller and data capture device,
- A DUT mounted on a board, and
- A host pc that is controlled by a user. The user provides commands to the host PC which in turns sends the commands to the ATE. This requires the ATE to have a command decoder.

A variety of DUT-FPGA ATE systems are illustrated in Figure 27 through Figure 29. Each has advantages and disadvantages. Physically, Figure 27 is the easiest ATE to develop because of its simplicity. However, for high-speed operation with this set-up, the skew between the DUTs may make data comparison unreliable.

Figure 28 alleviates the skew issue by placing the two circuits and compares within the DUT. The disadvantage in this scheme is that there is limited visibility of the current state of the design during the error event. As an example, if both designs become inoperable such that their outputs remains at a constant logic '0', the compare would not report an upset because, although erroneous, both designs are equivalently '0'.

Figure 29 is an optimal, intelligent test vehicle. However, it can be extremely complex and expensive to create and requires the appropriate expertise.

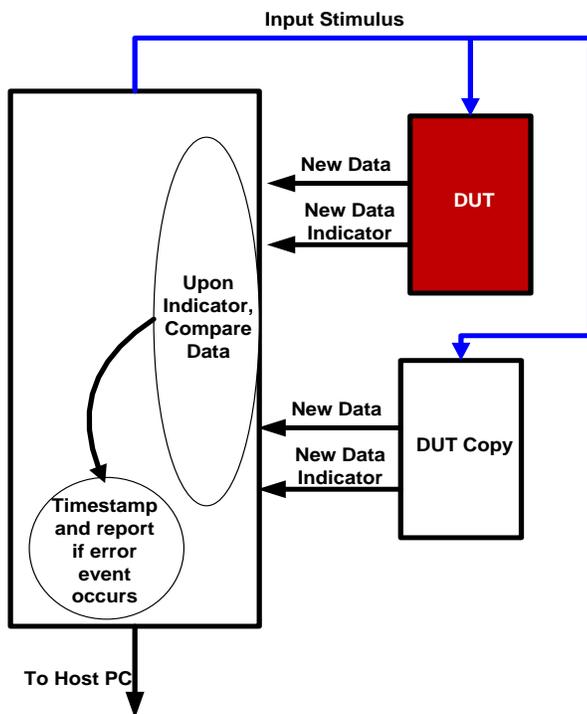


Figure 27: One test structure resides in the irradiated DUT and one test structure resides in another device. The test structures are equivalent and are controlled by the same input stimulus. Comparisons of the test structure outputs are performed in the test vehicle. This scheme is not recommended for high-speed operations because it is difficult to control the skew between the separate devices. This scheme can also be difficult to implement with complex test structures requiring a significant number of I/O.

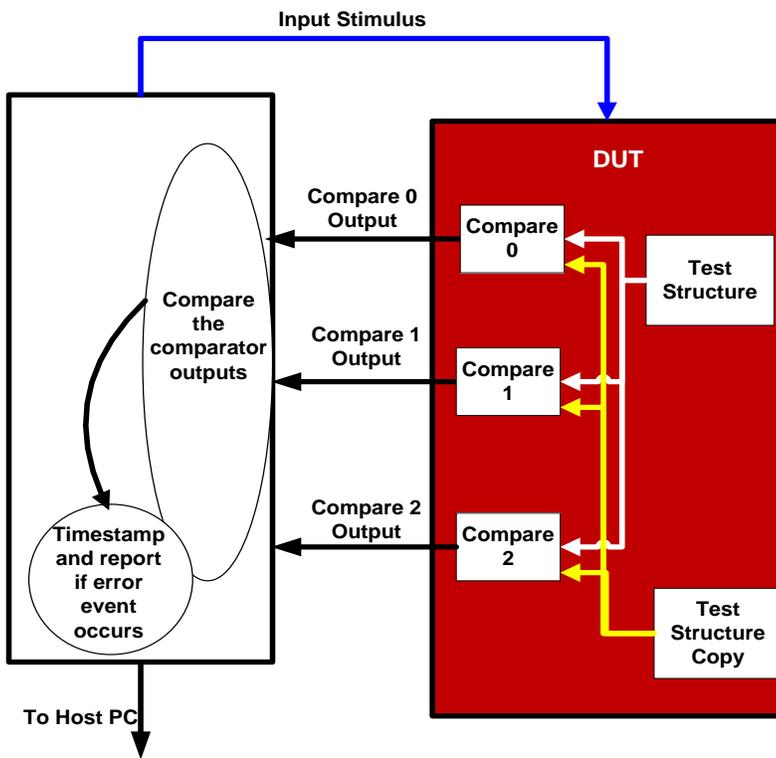


Figure 28: Two copies of the test structure reside in the irradiated DUT. The test structure states are compared in the DUT and the comparators flag upon mis-compare. The compare circuitry is triplicated to increase the integrity of the compare circuits. The compares are voted in the tester and checked for upset events. Good for high speed testing of complex test structures. However, due to the fact that the output is only a compare function, there is limited visibility into the state of the logic. Hence, this scheme is good for counting events, yet inefficient in differentiating the event.

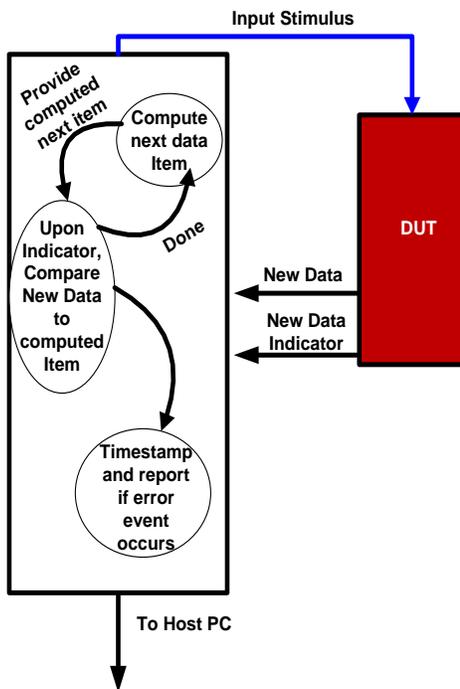


Figure 29: The tester contains circuitry that will capture the DUT outputs and compare to an expected value. Good for high speed data transmission.

The advantages of developing an ATE are the following:

- The capability of fine grain control over DUT-FPGA input stimulus and DUT-FPGA data capture. It is best practice to be able to observe the output of the DUT-FPGA such that all potential changes of state can be captured. This requires the ATE to operate at least as fast (sometimes faster) than the DUT-FPGA.
- The ability to customize command driven options such as:
 - Test set-up parameters
 - Input stimulus
 - Output masking
- The ability to maximize the amount of information associated with each error event (e.g. time stamping, and providing states of surrounding circuitry during the event)
- The ability to differentiate error events on –the-fly.
- The flexibility on how to store and report error information to the host PC.
- The ability to capture and associate significant amount of information to consecutive error events.

It is best practice to optimize the visibility of DUT operation. Visibility is accomplished by connecting test equipment to DUT outputs. As previously mentioned, developing an ATE is an enhancement to off-the-shelf logic analyzer equipment or oscilloscopes. However, combining test equipment, ATEs, logic analyzers, and oscilloscopes is the optimal approach. Figure 30 is an example of a combined-approach test vehicle that was used to test and evaluate SEU susceptibility in the Microsemi RTAX2000s and RTAX4000D FPGA devices.

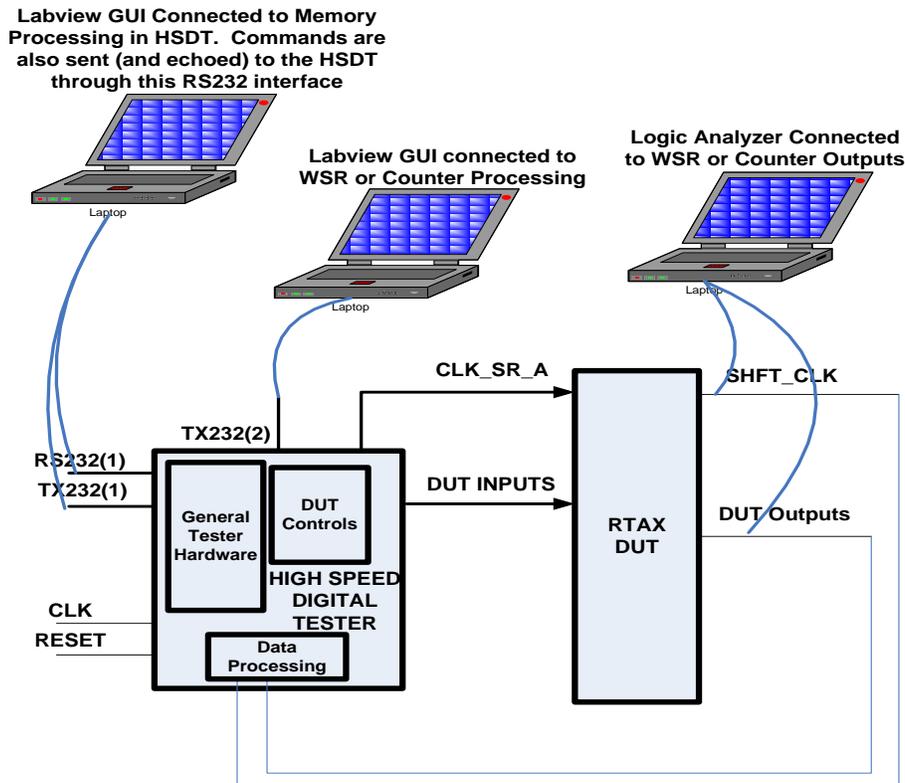


Figure 30: Example of Microsemi RTAXs ATE. Additional equipment is used to enhance real-time visibility of the DUT-FPGA operation.

7 SINGLE EVENT UPSET RADIATION TESTING

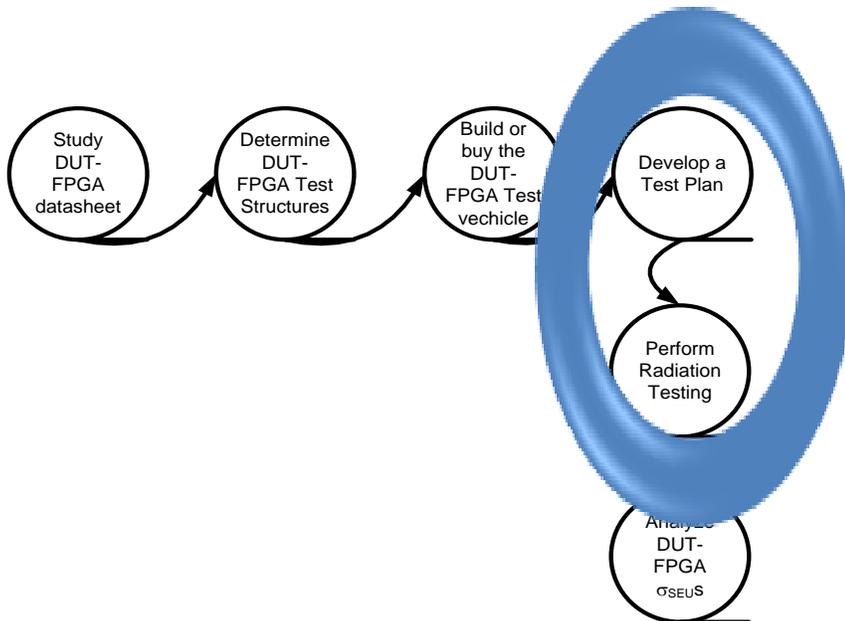


Figure 31: General flow for developing a SEU test strategy

During testing, the DUT is exposed to a radiation beam that generates a certain number of particles per area per second. The beam fluence is given in Eq. (3). The fluence per second is the flux and is provided in Eq.(4). SEE testing is relatively accelerated because the particle flux during irradiation is significantly faster than the particle flux in space.

(4)

An SEU occurs when an ionized particle interacts with the sensitive region of a device such that interaction changes the state of the device. The change of state can be permanent or temporary. As previously mentioned, σ_{SEU} s are calculated by counting the number of SEU events during irradiation per particle type.

The tests and σ_{SEU} calculations are performed over a range of particles in order to emulate a space environment. Particle vs. σ_{SEU} graphs are generated and are generally fitted with a Weibull curve. Various particles are more significant in particular environments at given intervals of time. Accordingly, the σ_{SEU} s are integrated across the various particles (in a weighted form), in order to obtain error rates for a given environment.

There are three groups of particles that are used for SEU testing: Heavy-ion, Proton, and Neutron. This section addresses heavy-ion and proton testing.

7.1 Heavy Ion

The ability of a heavy-ion particle to interact with materials is a function of its linear energy transfer (LET) value. LET is essentially the measure of ionizing energy deposited in a material per distance traveled, generally rendered in millions of electron volts per square centimeter per milligram ($\text{MeV}\cdot\text{cm}^2/\text{mg}$). For particles in space, the range of LET varies primarily from a few hundredths to just under $100 \text{ MeV}\cdot\text{cm}^2/\text{mg}$. Particles with low LET values are far more abundant than particles with high LET.

The goal of heavy ion testing is:

- Determine the LET threshold (LET_{TH}): This is the lowest LET value where upsets are first observed. This is the most difficult goal to achieve. It will depend on:
 - Test structure: is there enough complexity of the test structure to observe upsets at small LETs? Or, is there too much complexity in the test structure where upsets are being masked and are unobservable?
 - Frequency: Regarding SETs, is the testing frequency fast enough to capture upsets? Regarding FF SEUs, is the frequency slow enough to observe upsets?
 - Data pattern: It has been shown that data paths that switch state every clock cycle are the most susceptible. Is the data pattern significantly switching states?
 - Test Vehicle: Does the test vehicle have enough granularity to capture all upsets?
- Calculate σ_{SEU} s for at least 5 LET points. For a proper Weibull fit, it is best to have as many σ_{SEU} -LET data points as test time permits
- Determine the LET saturation (LET_{sat}) point. As technology shrinks, LET_{sat} is not observable.

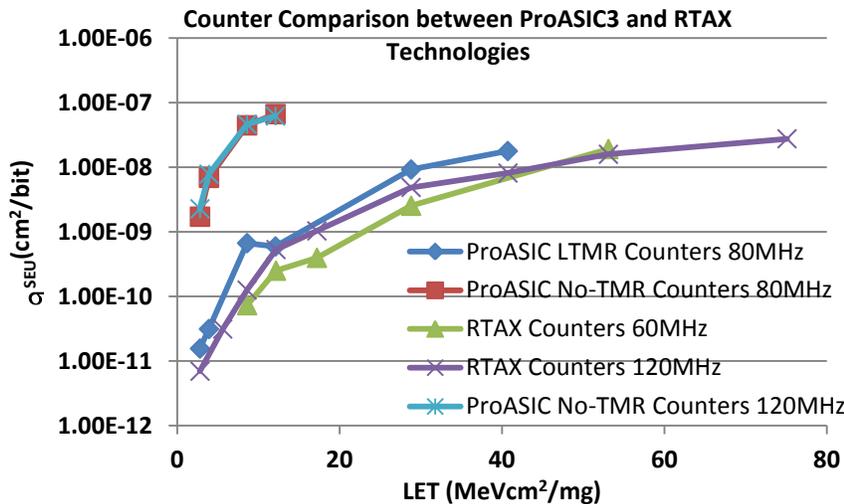


Figure 32: Example of σ_{SEU} -LET data for two separate FPGAs with counter test structures operating at a variety of frequencies. Regarding the graph, $\text{LET}_{\text{TH}} < 2.8 \text{ MeV}\cdot\text{cm}^2/\text{mg}$; and LET_{sat} is not definitive because all σ_{SEU} s are still increasing as LET increases [17].

7.2 Proton

Protons have relatively low LET values versus heavy ions. Protons can produce ionization by two primary methods:

- Direct ionization: The proton, itself generates the charge that interacts with the sensitive region of the device
- Indirect ionization: The products of a proton-nucleus collision generate scattered charge, e.g. secondary electrons, that can interact with the sensitive region of the device.

Either of these event types can induce an SET in combinatorial logic or an SEU in a FF. No FPGAs have shown susceptibility to proton direct ionization, because there is not enough generated charge.

FPGA data paths have shown low susceptibility to indirect ionization as a function of proton energy. Proton energies are generally rendered in millions of electron volts (MeV). As proton energy increases, the probability of an SEU increases.

Alternatively, SRAM configuration memories have shown to be highly susceptible to protons. Subsequently, a significant amount of proton testing should be performed to evaluate configuration σ_{SEUS} per proton energy.

The following are general rules of thumb for proton testing [35]:

- If SEE is not observed with heavy ions at $LET_{th} \leq 37$, then proton SEE testing is NOT required. – An LET of 34 is approximately the highest LET secondary possible from a reaction with a 500 MeV proton and modern semiconductor materials.
- If SEE is observed with a $LET_{th} \leq 20$, then proton SEE testing with $100 < MeV < E < 200$ MeV is required. – Additional margin on predicted proton SEE rate should be included. – A factor of 10X is sufficient.
- It is best practice to obtain σ_{SEUS} for at least three proton energies.

The following sections combine the information provided in this document to form guidelines and recommendations for testing DUT-FPGA configuration and DUT-FPGA functional data paths.

7.3 Configuration Radiation Testing

The configuration technology type and its accessibility dictate how the configuration SEU susceptibility can be evaluated. For flash and SRAM based FPGAs, testing the configuration requires a separate procedure versus testing the functional logic. The additional procedure entails reading back the configuration memory elements. Table 14 lists how various configuration technologies can be tested.

Table 14: Configuration SEU Test Recommendations listed per Configuration Type

Configuration Type	Configuration Node Visibility during testing	Flux and fluence Considerations	Recommended Test Procedure
Antifuse	None	None	Indirect testing of the Antifuse via dynamic testing of designs. Permanent malfunction of the design can be an artifact of a damaged fuse node (i.e., configuration node).
SRAM	Full visibility of each configuration node via read-back.	<u>Flux</u> : If performing read-back of the configuration during irradiation, flux must be kept low. High flux with read-back can produce false multiple bit upsets during testing. If read-back is done after irradiation, flux is not an issue <u>Fluence</u> : Determined at testing. Too many particles can produce false multiple bit upsets during static testing	<ul style="list-style-type: none"> • Configure the DUT • Irradiate the DUT: There is a choice to read-back during irradiation or do nothing during irradiation. No real difference in upset rates has been observed when reading back versus not doing anything except when the read-back path gets corrupted. It is recommended to try both methods: <ul style="list-style-type: none"> ○ Read-back: if done properly, it may help in differentiating multiple bit upsets ○ Do nothing: flux can be higher and consequently tests can be run faster • After irradiation is complete, read-back the DUT configuration
FLASH	Currently, manufacturers do not allow direct access to configuration flash-memory bits. Consequently, Read-back will not provide the	<u>Flux</u> : Not yet determined if flux is a concern. If so, it would only be a concern during Heavy-ion, high Linear Energy Transfer (LET) ions. <u>Fluence</u> : Due to potential dose	<ul style="list-style-type: none"> • Configure the DUT • Irradiate the DUT • Verify (i.e., read-back) the DUT configuration

	value of each configuration bit. However, read-back, also known as verify, will provide a pass/fail. Failure indicates that one or more flash bits do not contain a correct value.	issues fluence could be a problem. However, dose problems have not yet been observed with heavy ions.	
--	--	---	--

7.4 Functional Data Path SEU Testing

The previous section concentrated on SEU testing for a variety of DUT-FPGA configuration technologies. This section focuses on developing tests to evaluate the DUT-FPGA functional data path. Table 15 is meant to provide information on procedures that will ultimately achieve optimal SEU characterization of FPGA devices. However, due to time and financial restrictions, it is understood that there will be a tradeoff regarding the ability to implement some of the guidelines.

Table 15: Functional Data Path SEU Test Guidelines and Recommendations listed by: test structures, test vehicles, and test procedures

Test Category and Section	Guideline	Recommendations
Test Structures	Determine the goal of testing: e.g. FF susceptibility or real-design data extrapolation prior to selecting the test structures	<ul style="list-style-type: none"> • <u>FF susceptibility</u>: use a variety of WSR chains • <u>Real-design data extrapolation</u>: use a variety of WSR chains and other more complex test structures. Note that the complexity of the test structure should be limited in order to sustain the integrity of the SEU data.
	Test structures should follow the design topology of real-design implementation	Synchronous design Methodology is the recommended design scheme
	Control the routing of the WSR strings such that each path has approximately the same τ_{dly} and that τ_{dly} is minimized	Manually place WSR elements to guarantee approximately equal τ_{dly} between each WSR stage.
	Optimize the integrity of the σ_{SEU} data.	<p>Do the following when developing test structures:</p> <ol style="list-style-type: none"> 1. Create a DUT design that has a large number of replicated logic structures in order to increase statistics. 2. Create a DUT design that has a traversable state space that can be completed within one radiation test run 3. Create a DUT design such that logic masking is minimized or is controllable. 4. Create a DUT design such that all (or a significant percentage of) potential upsets are observable 5. Create a DUT design that follows synchronous methodology guidelines in order to characterize topologies that match real designs. 6. Consider any limitations regarding the interface to the test vehicle and the board that the DUT is mounted on.
	Apply mitigation strategy to commercial devices.	Test a commercial device with no mitigation as a reference. In addition, mitigation strategies should be applied and tested to determine the effectiveness of the scheme for reducing σ_{SEUs} .

	For FLASH or SRAM based FPGA's Utilize a large amount of configuration bits	Strive to achieve between 80% to 100% resource utilization in order to observe configuration effects to the data path
Test Vehicle	Test vehicle should be able to supply input stimulus at the maximum rates that the test-structures can operate.	A compressive timing analysis study should be performed for the test structures. This can be performed using an STA tool. It is then recommended that the test equipment be properly selected to handle the rates. For high frequency complex designs, ATEs may be required.
	Synchronize input signals when required; e.g., a clock and its data	<ul style="list-style-type: none"> • <u>When using one functional generator:</u> select the option of synchronous output of signals when necessary • <u>When using two functional generators:</u> use the synchronizer cable to synchronize the functional generators when necessary • <u>When using an ATE:</u> The designer has full control over the ATE behavior. The signals can be synchronized optimal precision.
	Test vehicle should be able to be controlled by the user in order to change input stimulus and test parameters	<ul style="list-style-type: none"> • <u>When using function generators for input stimulus:</u> Develop a graphical user interface (GUI) controller to automatically control the function generator (e.g. LabView GUI) • <u>When using an ATE:</u> Design a command decoder be into the ATE. The command decoder will take commands from the user that will control the test vehicle.
	Test vehicle should timestamp errors in order to enhance post-processing of data	Provide as much information per error event. This will facilitate identifying error sources: e.g. Differentiate whether the event was due to SET capture or an SEU flip
	Combine ATE and other off-the-shelf equipment	Combine an ATE with logic analyzers and oscilloscopes during SEU testing. This enhances the visibility of DUT operation and the integrity of data
	Have the ability to control the power supplied to the DUT	Use a power supply that can be controlled automatically. Software should be developed for automated power supply control during radiation testing
Test Procedures and Parameters	Develop tests to establish trends	Frequency: Strive to test at least 5 frequencies per test structure per effective heavy-ion-LET or proton-energy. Data pattern: The variation of data input will depend on the test structure.
	Test the DUT at different angles	Change the angle of incidence especially during heavy-ion testing. Common angles of incidence are: 0 , 45 , and 60 .
	Obtain σ_{SEUs} for a variety of particles	<ul style="list-style-type: none"> • Strive to obtain σ_{SEUs} for at least 5 heavy-ion LET values. Keep in mind that finding the LET_{th} is essential per test structure. However, finding LET_{th} can be a challenging processing • Strive to obtain σ_{SEUs} for at least 3 proton energies
	Test until a high enough fluence is	Determining the appropriate number of events is

	reached or a significant number of events have occurred	challenging because it is important to guarantee that the calculated number of events pertains to the same type of events; This requires event differentiation at the test site. It is recommended to observe > 10 events prior to stopping a test. Otherwise stop when reaching the recommended fluence. <ul style="list-style-type: none"> • Heavy-ion: It is recommended to test until a fluence of $1e^{-7}$ particles/cm² specifically at low LET values • Protons: It is recommended to test until a fluence of $1e^{-7}$ particles/cm² specifically at low energy values
	Control flux such that tests can be run as fast as possible. However, the flux cannot be too high such that unrealistic error events are occurring.	It is recommend to determine the proper flux on site. Flux is dependent of the LET of the heavy-ion or the energy of the proton. Flux can be higher at low LETs or low energies. WSRs have shown to withstand the highest flux due to their linear architecture.
	Replicate tests to increase integrity of results	It is recommended to run at least each test twice

Once the test-structures undergo radiation testing, the radiation data is analyzed. The following sections describe how SEUs are generated and captured in synchronous designs. Supporting mathematical models that have been developed by NASA Goddard REAG are also provided. The models are used to assist in radiation data evaluation.

8 ANALYZING RADIATION DATA – SEUs IN FPGA AND THE APPLICATION OF THE NASA RADIATION EFFECTS AND ANALYSIS FPGA SEU MODEL

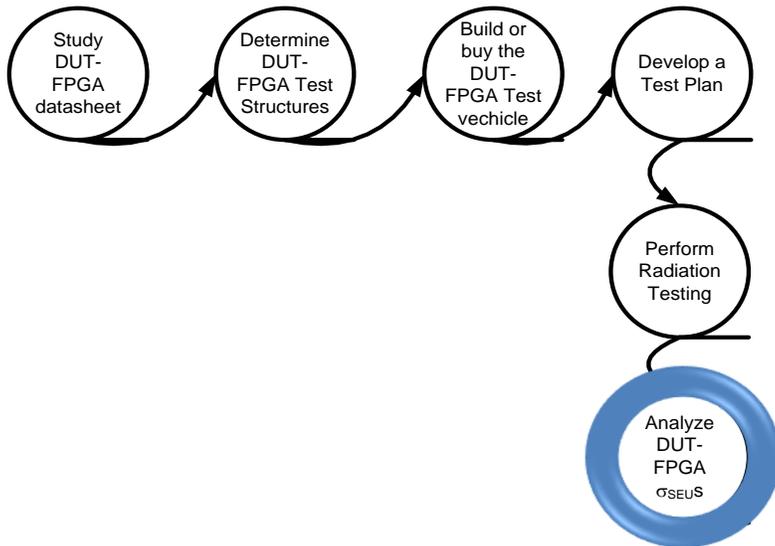


Figure 33: General flow for developing a SEU test strategy

FPGA devices and their design implementations are challenging to evaluate. Their mix of complex structures and functional states produce convoluted error signatures during SEU testing. Consequently, identifying SEU error sources or deconvolving error signatures is a difficult task during σ_{SEU} data analysis. In response, models have been developed and used during the data analysis process.

Usage of models developed by NASA Goddard Radiation Effects and Analysis Group (REAG) has successfully identified

SEU sources and correlated their trends. The purpose of such evaluation is to characterize error signatures so that FPGA designers can optimize their design based on criticality, general requirements (i.e. speed, area, and power), device type, and radiation environment.

8.1 Top Level FPGA SEU Model Development

In a synchronous design, it is understood that SEUs/SETs can occur in:

- Configuration
- Data path Flip-flops (FFs) and Combinatorial Logic (CL)
- Clock trees
- Reset trees
- Embedded memory
- Inputs or Outputs: (I/O)

The NASA REAG FPGA SEU probability ($P(fs)_{error}$) model is proportional to σ_{SEU} such that the probability events are with respect to ionizing particles. $P(fs)_{error}$ has three major categories:

- Configuration σ_{SEU} ($P_{configuration}$),
- Data path or functional logic σ_{SEU} ($P_{FunctionalLogic}$), and
- Single Event Functional Interrupt (SEFI) σ_{SEU} (P_{SEFI}).

$P(fs)_{error}$ is reflected in (5).

(5)

As previously mentioned, the SEU Probability model is used by REAG as a Single Event Effects (SEE) data analysis tool. Upsets that occur during radiation testing are differentiated and are categorized in order to enhance device evaluation. The model is a reflection of the SEU cross section (σ_{SEU}) for a synchronous digital system. As a reminder, operational frequency (fs) is understood to be the inverse of clock period (τ_{clk}) as in Eq.(1).

The following sections describe the three categories of $P(fs)_{error}$ in more detail.

8.2 $P_{configuration}$: Configuration SEU Susceptibility and Analysis

Section **Error! Reference source not found.** describes the most common FPGA configuration technologies. SEU susceptibility varies with configuration type. Figure 34 shows a physical comparison of potential susceptibility for an Antifuse configuration versus SRAM configuration. Because of the varying SEU susceptibilities, the SEU test methodology will depend on the type of configuration. Therefore, prior to test development, refer to the manufacturer data sheet to understand the configuration technology used in the DUT.

The following lists configuration technologies and their potential susceptibilities:

- **SRAM:** SRAM configuration transistors are implemented in the sensitive region of the device. Hence, SRAM cells are susceptible to SEUs. Due to the layout of the SRAM cells (bits), they tend to have a significantly higher σ_{SEU} than other elements within the FPGA.
- **Antifuse:** Antifuse configuration is formed in the metallization layers and is hence immune to SEUs.
- **Flash:** Flash configuration bits prove to have a fairly low SEU susceptibility in commercial flash memory devices; i.e., bit upsets exist but are rare. When used as a configuration, the FLASH structure remains static after programming. In addition, the flash charge pump (VPUMP) is tied to ground. This setup has proven to be beneficial because, during SEE testing, FPGA operation was not disrupted. However, more work needs to be done to improve the total ionizing dose (TID) effects. Flash devices are not recommended to be utilized in missions that will incur more than 10Krad.

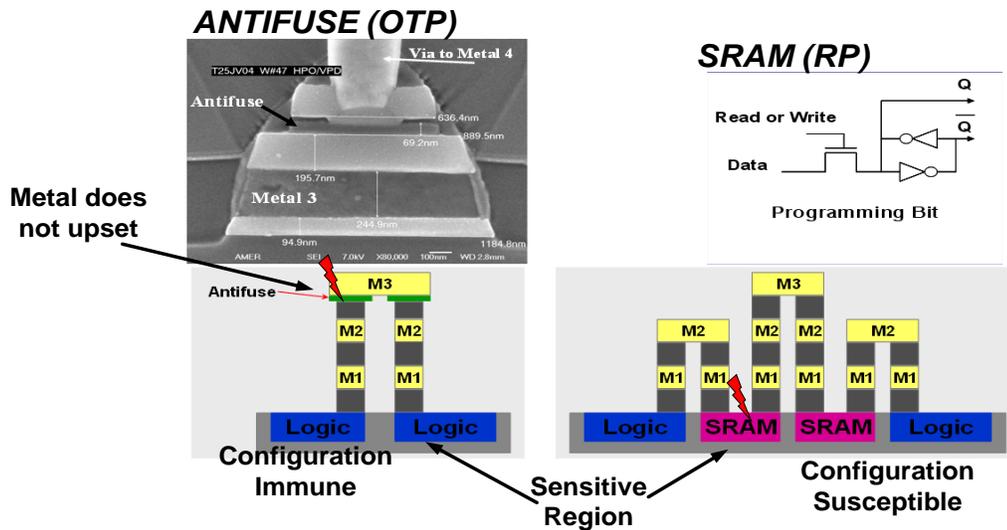


Figure 34: A comparison of Configuration Technology (Antifuse versus SRAM) and SEU

The evaluation of configuration σ_{SEUS} is technology dependent. The following is a list of configuration technologies and how their σ_{SEUS} are determined:

- **SRAM:** Because SRAM configurations can be read-back, their memory-bit values are accessible. Hence the σ_{SEUS} are based on the number of upsets in the configuration memory read-back data stream as shown in Eq. (6).

$$\sigma_{SEUS} = \frac{N_{upsets}}{F_{read-back} \cdot T_{exposure}} \quad (6)$$

- **Antifuse:** Antifuse configuration is not accessible. Hence, if a configuration failure is observed, it will be based off of the event normalized by the fluence during DUT exposure as shown in Eq. (7).

$$\sigma_{SEUS} = \frac{N_{failures}}{F_{exposure}} \quad (7)$$

- **Flash:** Although flash configuration can be read-back, currently the manufacturer has not made the read-back stream accessible to the user. A read-back is performed with a pass-fail result. Because information is limited, the σ_{SEUS} are calculated similar to the Antifuse: if a configuration failure is observed, it will be based off of the event normalized by the fluence during DUT exposure as shown in Eq. (7).

8.3 $P(fs)_{functionalLogic}$ SEU Susceptibility and Analysis

As previously mentioned, the functional logic data path of a synchronous design is comprised of: Combinatorial Logic (CL), Flip-Flops (FFs), and Routes. Table 16 illustrates upset types that can potentially occur in an FPGA data path. Routes are grouped into the CL category. Because FFs are master-slave edge-triggered-flip-flops, their internal structure uses both a global clock (CLK) and its logical inverse (CLKB), as shown in Table 16.

Although upsets can be generated in the individual components (CL and FFs) of a functional logic data path, it is not guaranteed that the upset will place the system in an erroneous state. In order to disrupt synchronous operation, the upset must manifest and change the system state. Consequently, the focus becomes the probability of capturing the upset into the next state of the system. If the upset is not captured, then the upset has no effect on the operation of the system.

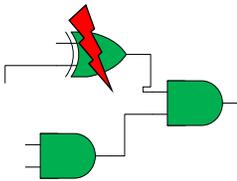
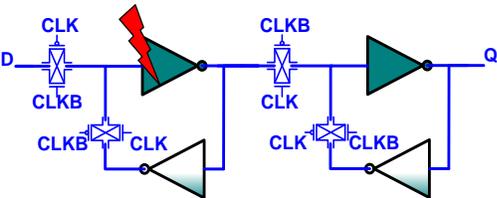
Because of the difference in error signatures (i.e. single sided versus double sided) between FFs and CL, upsets should be differentiated for proper characterization of SEUs. Subsequently, $P(fs)_{functionalLogic}$ has two major components:

- Captured upsets from combinatorial logic CL: $P(fs)_{SET \rightarrow SEU}$ and
- Captured upsets from flip-flops FFs: $P(fs)_{DFNSEU \rightarrow SEU}$.

The evaluation of $P(fs)_{SET \rightarrow SEU}$ and $P(fs)_{DFNSEU \rightarrow SEU}$ can be accomplished by logic cone analysis for each () End-Point FF. Each cone will have a collection of Start-Point FFs that can incur an SEU and a number of CL gates that can incur an SET as noted in (8). Because emphasis is on system operation, $P(fs)_{functionalLogic}$ is proportional to the CL SETs that can get captured by End-Point FFs ($P(fs)_{SET \rightarrow SEU}$) and the FF SEUs that can get captured by End-Point FFs ($P(fs)_{DFNSEU \rightarrow SEU}$). Taking this into account, the following sections discuss the capture of combinatorial logic SETs and FF SEUs from the perspective of a system.

(8)

Table 16: SEUs in Combinatorial Logic versus Sequential Logic.

Combinatorial Logic	Flip-Flops
Synchronous function: Logic function generation; computation and routing	Synchronous function: Captures and holds state of its data input at a specified clock edge
	
SET: Glitch in the combinatorial logic. Must be captured to disrupt system behavior. Capture is frequency dependent	SEU: FF flips its state. Can occur at a clock edge or during the clock cycle. Depending on which part of the FF is upset and when the fault occurs, will determine if the capture is frequency dependent.
	
Double-sided function	Single-sided function

8.3.1 Capturing Combinatorial Logic Upsets (SETs) in a System ($P(fs)_{SET \rightarrow SEU}$)

An SET in a data path will only disrupt system operation if an End-Point FF captures it. SET capture is illustrated in Figure 35. As previously mentioned, $P(fs)_{SET \rightarrow SEU}$ is the probability that an SET is generated in a combinatorial logic gate and is captured by its cone’s End-Point. The following are factors that impact $P(fs)_{DFE \rightarrow SEU}$:

- The probability that an SET can be generated in a combinatorial gate (P_{gen})
- The ability for the generated SET to propagate to an End-Point FF (P_{prop})
- Probability that an SET can logically propagate through a cone of logic (P_{logic})
- Percentage of clock period for SET capture

The following sections describe the factors of $P(fs)_{DFE \rightarrow SEU}$ in more detail.

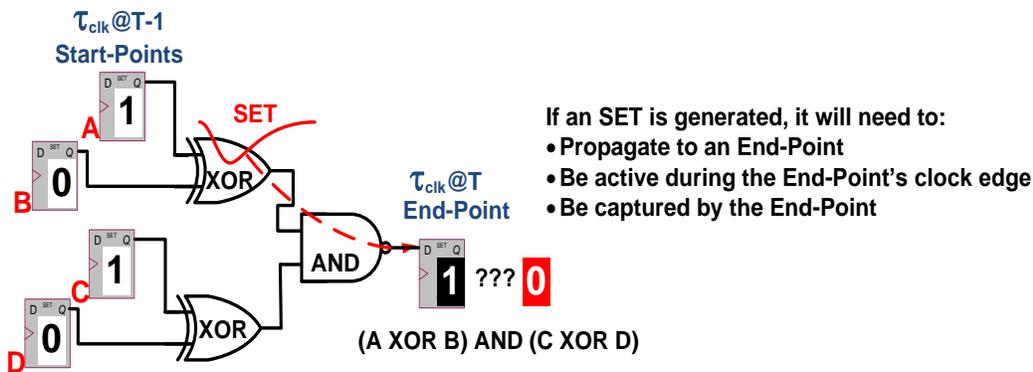


Figure 35: SET occurring in a combinatorial logic gate in between clock edges. Will it captured by its End-Points?

8.3.2 SET Generation (P_{gen})

For an SET to be generated in CMOS technology, an off-gate turns on [10]. In this case, the off-gate can only turn on if

the collected charge in its drain is greater than the critical voltage. As a result, the generated SET causes the direction of the current flow at the output of the transistor to temporarily change. Figure 36 is an illustration of the generation of a two-sided SET signal in CMOS.

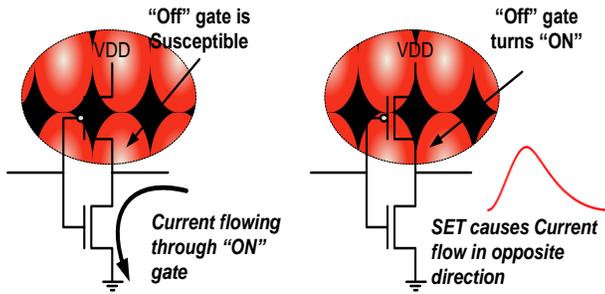


Figure 36: SET generation in a Complementary Oxide Semiconductor (CMOS) Gate

In a synchronous design, there are a variety of conditions that will affect the probability of generation and the size of the generated SET:

- Amount of collected charge: Particles with small LETs produce small SETs. A small SET is a two-sided signal with either a narrow width (τ_{width}) or low amplitude.
- The strength of the gate's load: As the capacitance of the load increases, the size of the generated SET decreases
- The strength of the complimentary "ON" gate: As the off-gate is turning "on", it must have enough drive strength to override the current flowing through the "ON" gate path. Subsequently, as the drive strength of the complimentary "ON" gate increases, the size of the SET decreases.
- The collection and recombination strength of the process

8.3.3 SET Propagation (P_{prop})

As previously mentioned, if an SET is generated in the data path, it must be captured by an End-Point in order to possibly disrupt synchronous system operation. The SET, which is generated in a CL gate, must propagate through CL and routes to reach an End-Point FF. The probability that the SET will not dissipate during propagation due to path capacitance is P_{prop} . Hence, $P_{prop} = 1$ for a given path suggests that the SET will always have enough energy to propagate through the cone of logic and reach the End-Point. The probability is based on the strength of the two-sided SET signal. As transistor geometries and critical voltages decrease, SETs are increasingly contributing to the overall σ_{SEU} . Consequently, a significant amount of research has been focused on SET propagation [18][30]-[32]

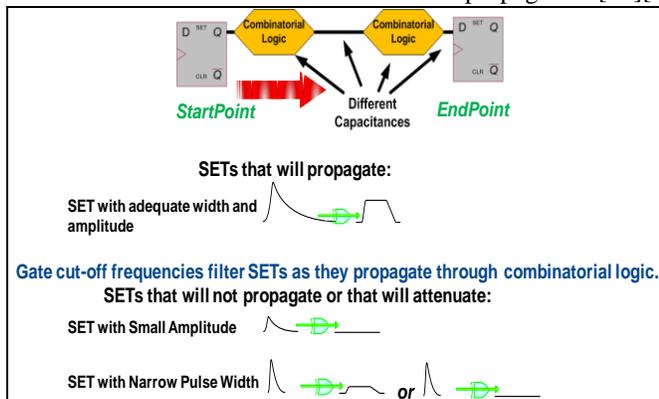


Figure 37: SET propagation through electrical medium can change the shape of the SET. Due to the unique capacitance in each propagation path, SET effects are non-linear

P_{prop} only pertains to electrical medium (capacitance of path due to combinatorial logic and routing). The capacitance within a propagation path can lead to SET amplitude and width reshaping. Small SETs may not have enough energy to withstand the capacitance of the propagation path and may subsequently get attenuated prior to reaching the End-Point FF. Capacitive effects are illustrated in Figure 37. The following are some key points that pertain to P_{prop} :

- 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

Because path capacitance can affect the propagation and size of an SET, SET effects are non-linear in a synchronous system. Subsequently, every combinatorial logic gate will have a unique P_{prop} due to its unique propagation path, load, drive

strength, and current-drive state during SET generation.

8.3.4 Logical Masking (P_{logic})

The difference between logical masking (P_{logic}) and propagation strength (P_{prop}) is that P_{logic} pertains to SET propagation through a logically turned-off/on data path due to the state of CL; while P_{prop} pertains to the potential attenuation of an SET due to the capacitance within its the propagation path.

P_{logic} is described in Section 5.2 and is illustrated in Figure 10. $P_{logic} = 1$ pertains to a path that can never be masked..

8.3.5 Percentage of Clock Period for SET Capture

Given that an SET is a two-sided function and that it can occur at any time within a clock period, the percentage of the clock period that the SET can be captured is the ratio of the width of the transient (τ_{width}) to the clock period (τ_{clk}). Hence, SET capture is directly proportional () to τ_{width} and the frequency of operation (fs) and is reflected in Eq.(9)

$$\text{---} \tag{9}$$

8.3.6 The Formulation of $P(fs)_{SET \rightarrow SEU}$ in the NASA REAG FPGA SEU Model

$P(fs)_{SET \rightarrow SEU}$ is the portion of $P(fs)_{functionalLogic}$ that strictly deals with combinatorial logic and SET effects in a synchronous design. As previously mentioned, an SET in the data path can only disrupt synchronous system operation if an End-Point FF captures it. Factors that affect SET system capture have been designated as: SET generation (P_{gen}), SET propagation (P_{prop}), Logical Masking (P_{logic}), and percentage of clock period for capture ($\tau_{width}fs$). In order to formulate $P(fs)_{SET \rightarrow SEU}$, each () FF in a design is evaluated as an End-Point with its cone of logic and SET factors of capture. Within the End-Point's cone of logic, the contribution of every CL gate must be taken into account. $P(fs)_{SET \rightarrow SEU}$ is expressed in (10) and is directly proportional to the number of combinatorial logic gates within a cone of logic and operational frequency (fs).

$$P(fs)_{SET \rightarrow SEU} \propto \exists_{DFE} \left(\sum_{i=1}^{\#Combinat\text{ialCells}} (P_{gen(i)} P_{prop(i)} P_{logic(i)} \tau_{width(i)} fs) \right) \tag{10}$$

8.3.7 Capturing FF Upsets in a System ($P(fs)_{DFE \rightarrow SEU}$)

Whereas the previous section discussed the generation and capture of SETs, the focus of this section is on the generation and capture of FF SEUs. In order for a system to be affected by a FF upset, the upset must first be generated in a FF and then manifest as an incorrect system state change at the rising edge of the system clock. In this document, FF SEU generation is differentiated by time of event as follows:

- SEU occurs at a clock edge. If not logically masked, the SEU is a disruption in system state
- SEU occurs at an intermediate point within a clock period. The time within a clock period of SEU occurrence is designated in this manuscript as time τ : $\tau < \tau_{clk}$. In this case, the SEU must be captured by an End-Point in order to disrupt the system state. The following are factors that impact whether a Start-Point FF SEU is captured by a End-Point FF ($P(fs)_{FFSEU \rightarrow SEU}$):
 - Probability of SEU Generation (P_{FFSEU})
 - Probability of Logic Masking (P_{logic})
 - Proportion of the clock period that the End-Point has to capture the Start-Point SEU

The following sections describe the factors of SEU generation and capture in more detail. The discussion refers to a synchronous design such that each FF is evaluated as an End-Point with its Start-Point FFs forming the base of its cone of logic. A unique τ_{dly} is calculated per Start-Point to End-Point path as illustrated in **Error! Reference source not found.**

8.3.7.1 Single Event Upsets Generated in FFs ($P(fs)_{FFSEU}$)

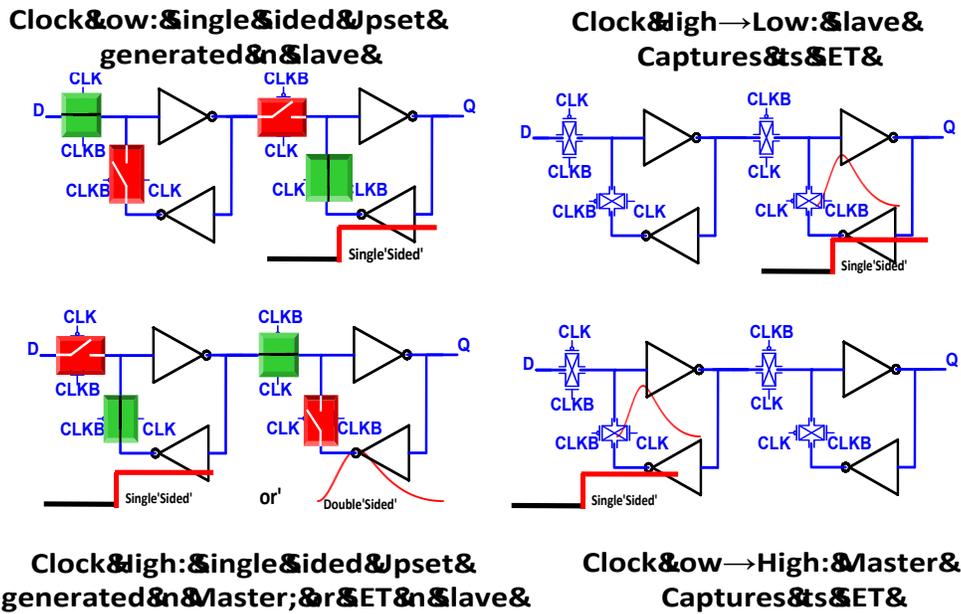


Figure 38: Clock state dependent modes of SEU error signatures in a master-slave FF

Error signatures resulting from an internal FF upset are determined by the state of the FF clock. The four clock states are defined to be: low, high, transitioning high to low (falling-edge), or transitioning low to high (rising-edge). Figure 38 illustrates the various modes of internal FF SEUs based on the clock state. As shown in Figure 38, FF SEU error signatures are either a single-sided signal representing an erroneous change in state; or a double-sided signal representing a transient. We break the probability of single-sided FF SEU occurrences ($P(fs)_{DFSEU}$) into two categories (10): (a) the percentage of $P(fs)_{DFSEU}$ that occur at the rising-edge ($\alpha P(fs)_{DFSEU}$), illustrated in Figure 38 (D); and (b) the percentage of $P(fs)_{DFSEU}$ that occur between rising edges of the clock ($\beta P(fs)_{DFSEU}$), Figure 38 (A-C).

(10)

Because the state of a synchronous design is defined at each rising-edge, the proportion of SEUs that occur at a rising-edge, $\alpha P(fs)_{DFSEU}$, force a definitive state change and thereby cause system error. Such upsets are attributed to End-Point FFs. Alternatively, the proportion of SEUs that occur between rising-edges, $\beta P(fs)_{DFSEU}$, may not affect the system state; i.e., the upsets must be captured by an End-Point at the next rising-edge to cause a system error. Such upsets are attributed to Start-Point FFs because they require End-Point capture. Summarizing: The proportion of rising-edge FF SEUs, $\alpha P(fs)_{DFSEU}$, are attributed to End-Points and the proportion of between-rising-edge FF SEUs, $\beta P(fs)_{DFSEU}$, refer Start-Points.

8.3.7.2 End-Point SEU Capture

An End-Point SEU occurs at a clock edge. It will cause a system state change if the forward data path of the erroneous FF is not logically masked from the system. An example of forward path logical mapping of a FF is a voter placed in front of the FF in a triple modular redundant (TMR) scheme.

8.3.7.3 Start-Point SEU Capture

The topology of synchronous design directly impacts SEU manifestation because of how and when data is captured and subsequently how and when system state is affected. As previously mentioned, the time a Start-Point SEU occurs relative to the beginning edge of a clock period is designated as τ in this manuscript. Because the upset is a single-sided function, attenuation during propagation is not an issue ($P_{prop}=1$). However, the delay of the routes and CL (τ_{dly}) between each Start-Point to its End-Point FF determines if the End-Point can capture the effect of the SEU. For instance, if the SEU is generated early in the clock period such that $\tau < \tau_{clk} - \tau_{dly}$ and $P_{logic} > 0$, the SEU will manifest as a system upset. In other words, the FF's flip in state has enough time to travel through the delay path and reach its End-Point by the next clock edge. Subsequently, data paths that have large delay (i.e., a large number of CL or long capacitive routes) relative to its clock period will reduce the probability that a Start-Point SEU is captured by an End-Point. An example of Start-Point SEU capture by an End-Point is illustrated in Figure 39: SEU occurring in a Start-Point FF in between clock edges. Will it manifest as a system upset?

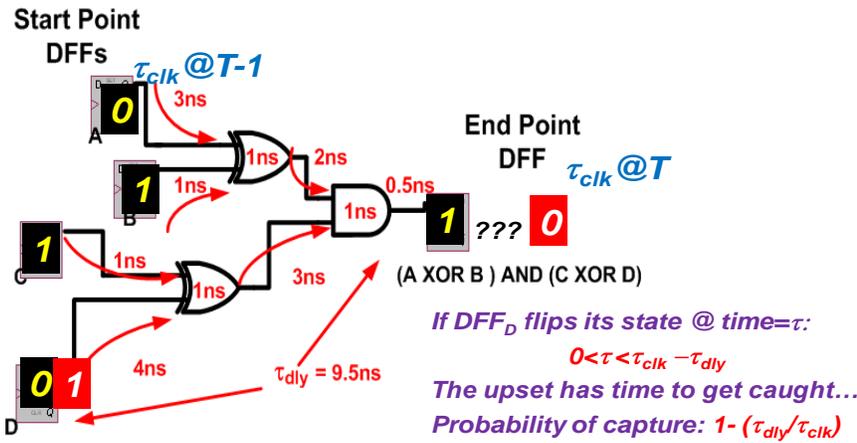


Figure 39: SEU occurring in a Start-Point FF in between clock edges. Will it manifest as a system upset?

The percentage of the clock period that an SEU can be captured is derived from $\tau < \tau_{clk} - \tau_{dly}$ to obtain (11).

$$\frac{\tau}{\tau_{clk}} < 1 - \frac{\tau_{dly}}{\tau_{clk}} = 1 - \tau_{dly} fs \quad (11)$$

8.3.7.4 FF Logical Masking

The probability of logic masking (P_{logic}) for FFs is the same for CL as previously described. $P_{logic} = 1$ means there will never be masking in a data path where $P_{logic} = 0$ means that an upset will always be masked.

Example 1: A Majority Voter is a three input CL gate. Its function is to output the following: if two or more inputs are equal to a logic '1', then output a logic '1'. Or if two or more inputs are equal to a logic '0', then output a logic '0'. Hence, if one of the FFs that feed the voter incurs an SEU, the SEU will be masked by the voter and the system will not be affected. Majority Voters are commonly used as the mitigation component in TMR [12][33] circuitry and is illustrated in Figure 10, Figure 24, and Figure 25.

8.3.8 The Formulation of $P(fs)_{DFFSEU \rightarrow SEU}$

$P(fs)_{DFFSEU \rightarrow SEU}$ pertains to the probability that a Start-Point or an End-Point flips its state and its upset is captured by an End-Point, i.e., manifested as a system error. It is expressed in Eq. (12)

$$P(fs)_{DFFSEU \rightarrow SEU} \propto \sum_{j=1}^{\#StartPointDFFs} P_{DFFSEU(j)} P_{logic(j)} (1 - \tau_{dly(j)} fs) \quad (12)$$

When evaluating FF susceptibility, it is important to reduce the amount of system level (or design specific) derating of upset manifestation. As previously mentioned, system level derating can occur from the proportion of data path delay to clock period and forward path logic masking. Because shift registers have no logic masking, they prove to be the optimal test structure for FF susceptibility analysis. During testing, attention should be given to the speed of test structure operation. Shift registers operating close to their maximum operational frequency will have a reduced FF SEU contribution because the value of τ_{dly} is too close to fs . Alternatively, high speed testing is essential for combinatorial logic SET evaluation.

8.3.9 Putting it all Together and the Formulation of $P(fs)_{functionalLogic}$

As previously mentioned, $P(fs)_{functionalLogic}$ pertains to captured SEUs in a synchronous data path. It has three categories of captured upsets:

- SETs generated by combinatorial logic ($P(fs)_{SET \rightarrow SEU}$). These upsets need to be captured by an unmasked End-Point to disrupt system behavior
- SEUs generated by Start-Point FFs ($\beta P(fs)_{DFFSEU \rightarrow SEU}$). These upsets need to be captured by an unmasked End-Point to disrupt system behavior

- SEUs generated in a FF at its clock edge ($\alpha P(fs)_{DFFSEU}$). These upsets will disrupt system behavior if the FF is unmasked

Table 17: Definition of Terms in the NASA REAG FPGA SEU Model

Term	Definition
$\alpha P(fs)_{DFFSEU}$	Probability that a flip-flop will flip its state at a clock edge. Upset is due to internal circuitry of FF – not due to capturing an incorrect data path signal
$\beta P(fs)_{DFFSEU}$	Probability that a flip-flop will flip its state in between clock edges. Upset is due to internal circuitry of FF – not due to capturing an incorrect data path signal
$P(fs)_{DFFSEU \rightarrow SEU}$	Probability a Start-point FF flips its state between clock edges and an End-Point will be affected by the Start-Point upset at the next clock edge.
$1 - \tau_{dly} / fs$	Portion of clock cycle that the End-Point FF can capture a Start-point FF SEU before the next clock edge. Assumes the SEU Start-point FF is always enabled and will have a valid value at the next clock edge
P_{logic}	Probability that the logic in the forward path of the element under evaluation will mask the element's upset
P_{gen}	Probability a combinatorial gate will incur a SET
P_{prop}	Probability the SET can propagate to an End-point FF
τ_{width} / fs	SET width to clock period ratio

Each term has been derived in previous sections, they are listed in Table 17, and their relationship to $P(fs)_{functionalLogic}$ is reflected in Eq(13):

(13)

The model in Eq 13 is used as an SEU data analysis tool. It assists in determining if FFs or combinatorial logic have the more dominant SEU cross-sections (σ_{SEU}). It also assists in evaluating the strength of the applied mitigation strategy.

8.4 Using the FPGA SEU Model for data path σ_{SEU} Evaluation

Table 18: The difference between SEU Capture Effects: $P(fs)_{DFFSEU \rightarrow SEU}$ versus $P(fs)_{SET \rightarrow SEU}$ and their corresponding system level trends

	$\alpha P(fs)_{DFFSEU(k)}$	$\beta P(fs)_{DFFSEU(i)}$	$P(fs)_{SET \rightarrow SEU(i)}$
Logic	End-Point flips to the wrong state at the clock edge. Upset is not due to a capture from its input data path. It is due to internal FF circuitry. Localized redundancy is taken into account in this term (e.g. LTMR or DICE)	Start-Point flips its state between clock edges. The upset is observed if an End-Point is affected by the flip. Hence, the upset is not an upset if not captured by the next clock edge. Logic and temporal masking can negate capturing the bit-flip event. Localized redundancy is taken into account in this term (e.g. LTMR or DICE). This term forms: $P(fs)_{DFFSEU \rightarrow SEU(i)}$	Combinatorial SET Capture
Capture percentage of clock period	? unknown. Depends on internal structure of flip-flops; e.g., how many gates consist in the FF, how are the transmission gates implemented?	$1 - \tau_{dly} fs = 1 - \tau_{dly} / \tau_{clk}$	$\tau_{width} / \tau_{clk}$
System Frequency Dependency	Increase in frequency increases $P(fs)_{DFFSEU}$	Increase in frequency decreases the ability to capture $\beta P(fs)_{DFFSEU(i)}$. Hence $P(fs)_{DFFSEU \rightarrow SEU}$ is inversely proportional to data path delay	Increase in frequency increases $P(fs)_{SET \rightarrow SEU}$
Data Path Combinatorial Logic Effect	N/A: The term intentionally does not take into account the data path	Increase in Combinatorial logic increases τ_{dly} which decreases the ability to capture $\beta P(fs)_{DFFSEU(i)}$. Hence $P(fs)_{DFFSEU \rightarrow SEU}$ is inversely proportional to data path delay	Increase in Combinatorial logic increases $P(fs)_{SET \rightarrow SEU}$

Trends across frequency and amount of combinatorial logic are studied to determine cell dominance and variable SEU effects. The trends are explained in Table 18. Regarding SEU effects, it has been shown that with non-mitigated synchronous designs (i.e., No-TMR), FFs are the dominant source of system upsets versus upsets from combinatorial logic. Hence the following discussion pertaining to No-TMR synchronous designs focuses on the NASA REAG FPGA SEU Model term:

$$P(fs)_{DFFSEU \rightarrow SEU} = \alpha P(fs)_{DFFSEU} + \beta P(fs)_{DFFSEU} (1 - \tau_{dly} fs)$$

Prior to the evaluation of SEU effects on system, it is important to emphasize the difference between the FF SEU terms $\alpha P(fs)_{DFFSEU}$ and $\beta P(fs)_{DFFSEU} (1 - \tau_{dly} fs)$:

- $\alpha P(fs)_{DFFSEU}$ is the probability of FF flipping its state ($P(fs)_{DFFSEU}$) at the FF's clock edge. α is the percentage of FFs flips that occur at the clock edge versus in between clock edges. SEUs associated with $\alpha P(fs)_{DFFSEU}$ directly disrupt system state and are directly proportional to frequency; i.e., the probability that a FF can flip its state at a clock edge increases as frequency increases.
- $\beta P(fs)_{DFFSEU}$ is the probability of FF flipping its state ($P(fs)_{DFFSEU}$) in between clock edges. β is the percentage of FF flips that occur in between clock edges versus at the clock edge. SEUs associated with $\beta P(fs)_{DFFSEU}$ in a synchronous design are not guaranteed to cause system disruption because they are generated between clock edges. They will disrupt system state if they are captured by an End-Point FF (see **Error! Reference source not found.**) and their capture rate is inversely proportional to frequency; i.e., the probability that a FF can flip its state in between clock edges and manifest into the next state will decrease as frequency increases.

Figure 40 illustrates WSR cross sections with checkerboard input pattern across operational frequency. The following is an analysis of frequency effects given the data in Figure 40:

General Frequency Trends:

- As frequency increases, more FFs can flip their state (i.e., $P(fs)_{DFESEU}$ increases with frequency)
- FFs are more dominant sources of upsets versus combinatorial logic in non-mitigated synchronous designs

Lower frequency Trends:

- A large percentage of Start-Point FF upsets can reach End-Points (i.e., the term $1 - \tau_{dly}fs$ approaches 1). Hence, in this frequency range the σ_{SEU} drop across frequency is not observable.
- Subsequently, the dominant trend in this frequency range stems from $P(fs)_{DFESEU}$ which increases with frequency: $P(fs)_{DFESEU} \rightarrow_{SEU} \alpha P(fs)_{DFESEU} + \beta P(fs)_{DFESEU}$

Higher frequency Trends:

- In this frequency range, a large percentage of Start-Point FF upsets cannot reach their End-Points. This is because the path delays start to approach the clock period and there is not enough time for the effects of the Start-Point upset to reach the End-Point. Hence, although more FFs are flipping their state ($P(fs)_{DFESEU}$ increases as frequency increases) – they cannot reach the End-Points and the σ_{SEU} drop across frequency is observable
- Subsequently, the dominant trend in this frequency range stems from $1 - \tau_{dly}fs$ and σ_{SEU} decreases as frequency increases: $P(fs)_{DFESEU} \rightarrow_{SEU} \alpha P(fs)_{DFESEU} + \beta P(fs)_{DFESEU}(1 - \tau_{dly}fs)$
 - The trend is controlled by the relationship of τ_{dly} to fs ; — ; if the delay takes up most of the clock period, then very few Start-Point FFs will not reach their End-Point
 - Increasing combinatorial logic in the path increases τ_{dly} and subsequently decreases $P(fs)_{DFESEU} \rightarrow_{SEU}$. This is apparent at high frequencies. However, when the frequency is very slow relative to the τ_{dly} , the effects are insignificant (i.e., the inverse relationship to frequency is not observable for data paths with small $\tau_{dly}fs$ term).

It is important to note that the trend is not simply dependent on frequency. It is dependent on the relationship of τ_{dly} to fs . When — is small, the drop-off of $P(fs)_{DFESEU} \rightarrow_{SEU}$ with respect to frequency is insignificant. During this portion of time where the drop-off due to design topology is insignificant, the σ_{SEU} will increase with frequency. However when — is large (e.g. a path with a large number of combinatorial logic stages between Start-Point FF to End-Point FFs), $P(fs)_{DFESEU} \rightarrow_{SEU}$ drop off is apparent across a significant amount of frequency range.

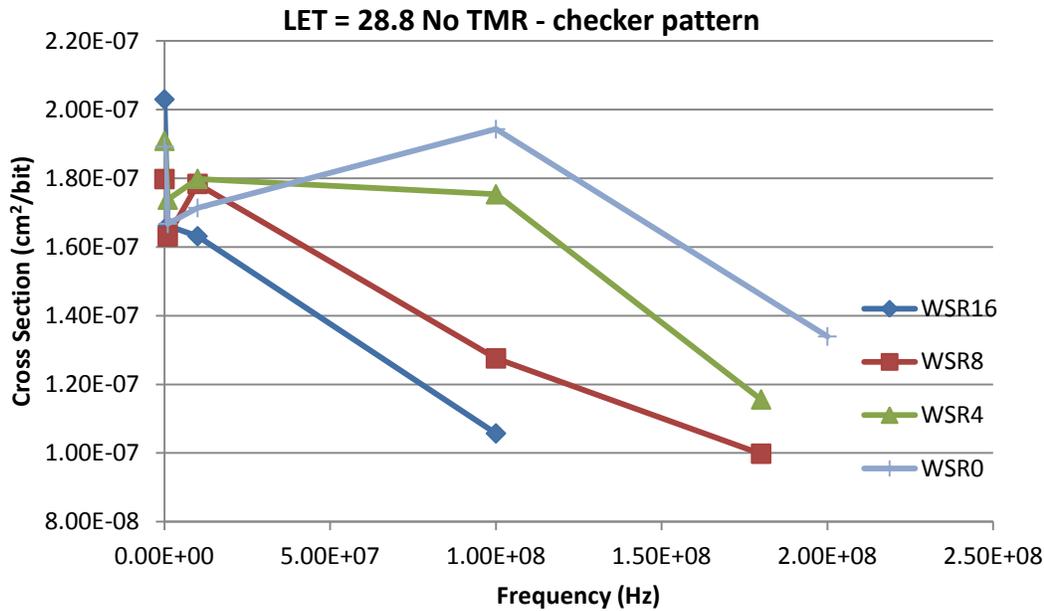


Figure 40: σ_{SEU} over frequency for $LET=28.8\text{MeVcm}^2/\text{mg}$. σ_{SEU} decreases as frequency increases. The addition of combinatorial logic within the path enhances the trend.

- $P(fs)_{DFSEU \rightarrow SEU}$ Dominance – Most SEUs stem from FFs. Figure 18 (non-mitigated σ_{SEUs}) and Figure 41 (Dual Interlock Cell mitigated FFs) illustrate σ_{SEUs} with FF dominance.
- If there is an increase in the number of combinatorial logic blocks or τ_{dly} and the $\sigma_{SEU} (P_{error})$ decreases in response
- If there is an increase in frequency and the $\sigma_{SEU} (P_{error})$ decreases in response
- $P(fs)_{SET \rightarrow SEU}$ Dominance – Most SEUs stem from Captured Combinatorial Logic SETs: . Figure 18 (mitigated σ_{SEUs}) and Figure 42 (LTMR FFs) illustrate σ_{SEUs} with mitigated FFs such that the combinatorial logic are the predominant contribution of upsets.
- If there is an increase in frequency and $\sigma_{SEU} (P_{error})$ increases in response
- If there is an increase in combinatorial logic and $\sigma_{SEU} (P_{error})$ increases in response

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

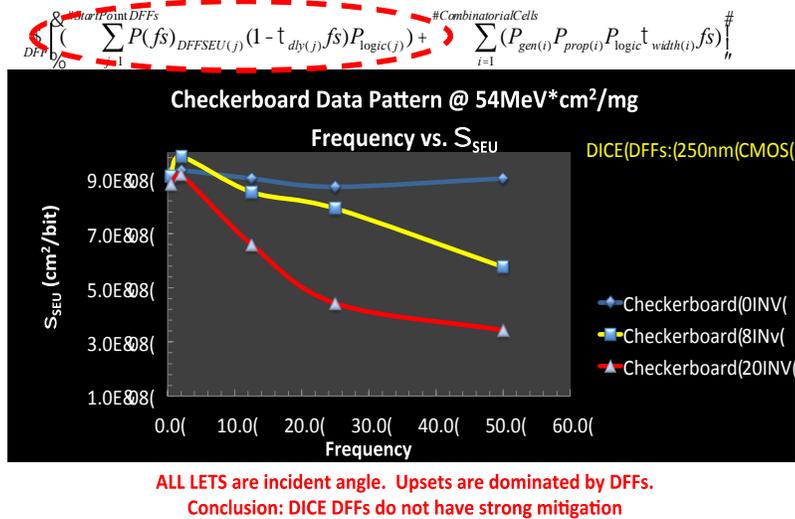


Figure 41: Aeroflex Eclipse has radiation hardened flip-flops. The hardening scheme is DICE. According to the trends in the WSR σ_{SEU} s the FFs predominantly contribute to upsets.

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

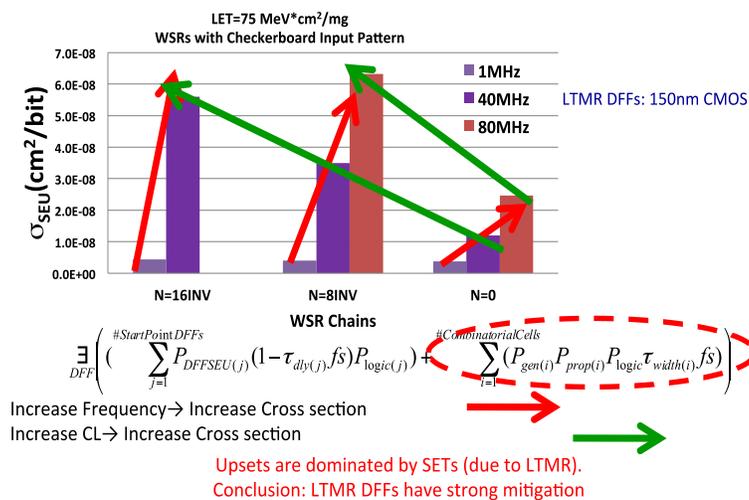


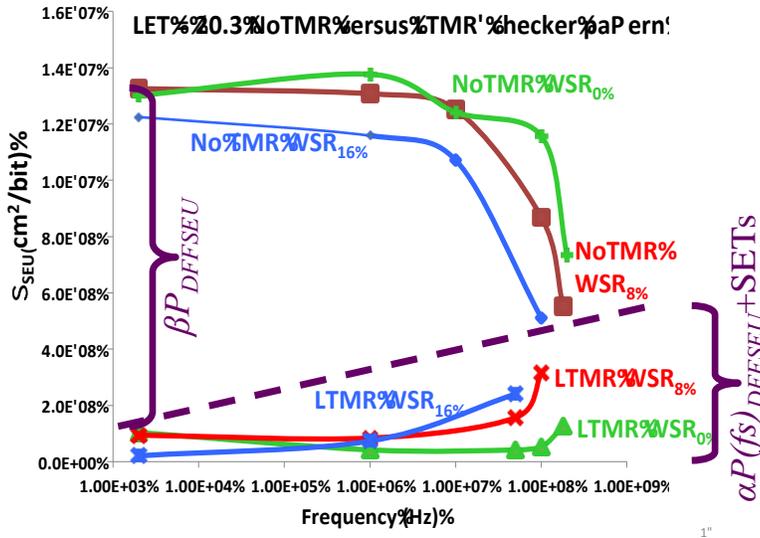
Figure 42: Microsemi RTAX2000s has radiation-hardened flip-flops. The hardening scheme is LTMR. According to the trends in the WSR σ_{SEU} s the FFs are completely mitigated and the combinatorial logic predominantly contribute to upsets.

An analysis of $\alpha P(fs)_{DFFSEU}$ versus $\beta P(fs)_{DFFSEU}$ was performed using the ProASIC3 FPGA device. It included a comparison of no-mitigation WSRs versus LTMR'd WSRs. It is understood that because all of the FF's are mitigated in an LTMR'd design, all upsets stem from either the combinatorial logic or from the global routes (clock or reset tree). Clock or reset upsets generally cause multiple upsets in a row (bursts). Hence, the error signature is used to differentiate global route upsets versus data path SETs. Alternatively, as previously mentioned, for non-mitigated ProASIC3 designs the FFs dominate the upsets as compared to SETs.

SEU cross section (σ_{SEU}) radiation data across frequency is illustrated in Figure 43. At lower frequencies, the σ_{SEU} s are dominated by the frequency independent components of $\beta P(fs)_{DFFSEU}$. As a result, σ_{SEU} s calculated in the KHz range can provide an estimate of $\beta P(fs)_{DFFSEU}$. For operational frequencies where τ_{dly} approaches $1/fs$, the term $(1 - \tau_{dly} fs)$ approaches 0. In this frequency range, the σ_{SEU} s are dominated by End-Point upsets ($\alpha P(fs)_{DFFSEU}$) and captured combinatorial logic SET contributions as illustrated in Regarding Figure 43. Data demonstrates that as τ_{dly} and fs increase, $\beta P(fs)_{DFFSEU}$ is temporally mask and the σ_{SEU} values start to sharply drop.

In order to further analyze $\alpha P(fs)_{DFFSEU}$ and SET contributions, the WSRs were tested with localized triple modular redundancy (LTMR) inserted at each FF. This evaluation masks all upsets that occur in FFs and hence produces cross

sections that reflect SET contributions. The SET contribution to the σ_{SEU} s are directly proportional to frequency and are characterized by (5). This data (LTMR σ_{SEU} s) are also illustrated in Figure 43. Key points from this data are that SETs have a relatively insignificant contribution when a design has non-mitigated FFs. In addition, given that τ_{dly} and f_s are known quantities, subtracting the SET σ_{SEU} s from the term $\beta P(f_s)_{DFE_{SEU}}(1-\tau_{dly}f_s)$ can provide a rough estimate of $\alpha P(f_s)_{DFE_{SEU}}$. ProASIC3 SEU data illustrates that $\alpha P(f_s)_{DFE_{SEU}}$ has an insignificant contribution to the overall σ_{SEU} for this device.



$$SEUs \approx \alpha P(f_s)_{DFE_{SEU}} + \beta P_{DFE_{SEU}}(1 - \tau_{dly}f_s) \quad SETs \approx P_{gen} P_{prop} t_{width} f_s$$

Figure 43: SEU cross section versus frequency for non-mitigated and mitigated designs

Using the model to differentiate errors proves beneficial. Application of the model to the σ_{SEU} s shows that the DICE mitigation strategy is not as effective as LTMR. DICE FFs have a similar trend as non-mitigated FFs showing that the FFs are still the dominating factor.

9 REFERENCES

- [1] L. Barth, et. al., "Radiation assurance for the space environment," International Conference on Integrated Circuit Design and Technology, pp. 323-333, 2004
- [2] Atmel Document: "Rad Hard Reprogrammable FPGA ATF280F Advance Information" http://www.atmel.fi/dyn/resources/prod_documents/doc7750.pdf, 2007
- [3] Aeroflex datasheet: "UT6325 RadTol Eclipse FPGA", <http://www.aeroflex.com/ams/pagesproduct/datasheets/RadTolEclipseFPGA.pdf>
- [4] Microsemi Datasheet: "RTAX-S/SL RadTolerant FPGAs" http://www.actel.com/documents/RTAXS_DS.pdf, V5.2, October 2007.
- [5] Microsemi Datasheet: "ProASIC3 Flash Family FPGAs", http://www.actel.com/documents/PA3_DS.pdf, (v5.3) May 17, 2010
- [6] Xilinx document, "Virtex 4 FPGA User Guide", http://www.xilinx.com/support/documentation/user_guides/ug070.pdf, v2.6 December 1, 2008
- [7] Xilinx document, "Virtex 5 FPGA User Guide", http://www.xilinx.com/support/documentation/user_guides/ug190.pdf,
- [8] Xilinx Document, "Radiation-Hardened, Space-Grade Virtex-5QV Device Overview", http://www.xilinx.com/support/documentation/data_sheets/ds192.pdf, v1.2, July 11, 2011
- [9] R. Baumann, "CMOS Single-Event Effects in Advanced CMOS Technology" IEEE NSREC Short Course, Section II, P. 329-332, July 2005, Seattle, WA.
- [10] M. Berg, J.-J Wang, R. Ladbury, S. Buchner, H. Kim, J. Howard, K. LaBel, A. Phan, T. Irwin, M. Friendlich, "An Analysis of Single Event Upset Dependencies on High Frequency and Architectural Implementations within Actel RTAX-S Family Field Programmable Gate Arrays," *IEEE Trans. Nucl. Sci.*, vol. 53, n° 6, Dec. 2006."

- [11] M. Berg "Trading Application Specific Integrated Circuit (ASIC) and Field Programmable Gate Array (FPGA) Considerations for System Insertion", NSREC Short Course, Quebec City, CN, July 2009
- [12] M. P. Baze, S. P. Buchner, W. G. Bartholet, and T. A. Dao "An SEU analysis approach for error propagation in digital VLSI CMOS ASICs", IEEE Trans. Nucl. Sci., Vol. 42, No. 6, 1863 (1995).
- [13] S.E. Diehl-Nagle, J.E. Vinson, and E.L. Petersen, "Single Event Upset Rate Predictions for Complex Logic Systems," IEEE Trans. Nucl. Sci., Vol-NS-31, 1132 (1984).
- [14] M. Berg, H. Kim, M. Friendlich, C. Perez, C. Seidleck, K. LaBel, R. Ladbury "SEU Analysis of Complex Circuits Implemented in Actel RTAX-S FPGA Devices", *IEEE Trans. Nucl. Sci.*, vol.58, no.3, pp.1015-1022, June 2011
- [15] J. George, R. Koga, G. Swift, G. Allen, C. Carlmichael, C. Tseng, "Single Event Upsets in Xilinx Virtex-4 FPGA Devices", IEEE Radiation Effects Data Workshop, 2006, p. 109-114.
- [16] N. Battezzati, S. Gerardin, A. Manuzzato, D. Merodio, A. Paccagnella, C. Poivey, L. Sterpone, M. Violante, "Methodologies to Study Frequency-Dependent Single Event Effects Sensitivity in Flash-Based FPGAs," IEEE Transactions on Nucl. Sci. Vol. 56, Issue 6, Dec 2009 pp 3534 – 3541
- [17] NASA Goddard Radiation Effects and Analysis Group test information website, <http://radhome.gsfc.nasa.gov/>
- [18] G. Allen, G. Swift, "Single Event Effects Test Results for Advanced Field Programmable Gate Arrays," Radiation Effects Data Workshop, July 2006, pp 115-120
- [19] G. M. Swift "Xilinx Single Event Effects 1st Consortium Report Virtex-II Static SEU Characterization." January 2004. Available: http://parts.jpl.nasa.gov/docs/swift/virtex2_0104.pdf
- [20] J. J. Wang, B Cronquist, J. McCollum, R. Katz, I. Kleyner, and R. Koga. "Single Event Effects of a FLASH-based FPGA." 2002 Single Event Effects Symposium, Manhattan Beach, CA, 2002. Available: http://klabs.org/richcontent/presentations/see_symp/see02_flash.pdf
- [21] Poivey, C.; Grandjean, M.; Guerre, F. X.; Radiation Effects Data Workshop (REDW), 2011 IEEE Digital Object Identifier: [10.1109/REDW.2010.6062510](https://doi.org/10.1109/REDW.2010.6062510) Publication Year: 2011 , Page(s): 1 - 5
- [22] C. Carmichael, E. Fuller, J. Fabula, F. De Lima. "Proton Testing of SEU Methods for the Virtex FPGA." 2001 Military and Aerospace Programmable Logic Device Conference, Washington D.C., 2001. Available: http://klabs.org/richcontent/MAPLDC01/Presentations/P/P6_Carmichael_S.pdf
- [23] J. George, S. Rezgui, G. Swift, C. Carmichael. "Initial Single Event Effects Testing and Mitigation in the Xilinx Virtex-II Pro FPGA." 2005 Military and Aerospace Programmable Logic Device Conference, Washington D.C., 2005. Available: http://klabs.org/mapld05/presento/211_george_p.pdf
- [24] V. Ferlet-Cavrois, P. Paillet, D. McMorrow, N. Fel, J. Baggio, S. Girard O. Duhamel, J.S. Melinger, M. Gaillardin, J.R. Schwank., P.E. Dodd, M.R. Shaneyfelt, J.A. Felix, "New Insights Into Single Event Transient Propagation in Chains of Inverters—Evidence for Propagation-Induced Pulse Broadening" IEEE Trans. Nucl. Sci., vol. 54, n° 6, Dec. 2007
- [25] P.E. Dodd , M.R. Shaneyfelt , J.A. Felix and J.R. Schwank "Production and propagation of single-event transients in high-speed digital logic ICs", IEEE Trans. Nucl. Sci., vol. 51, pp.3278 2004 .
- [26] S. Buchner, M. Baze, D. Brown, D. McMarrow, J. Melinger, "Comparison of Error Rates in Combinatorial Logic and Sequential logic", IEEE Transactions on Nucl. Sci. Vol. 35, Issue 6, Dec 1988 pp 1517-1522
- [27] F. L. Kastensmidt, "SEE mitigation strategies for digital circuit design applicable to ASIC and FPGAs," in *2007 IEEE NSREC Short Course Notebook*. Porto Alegre, Brazil: UFRGS, unpublished.
- [28] C. Maxfield, *The Design Warrior's Guide to FPGAs*. Burlington, MA Elsevier, 2004
- [29] B. Zeidman, *Designing with FPGAs & CPLDs*, Lawrence, KS, CMP Books, 2002
- [30] P. Chambers, The Ten Commandments of Excellent Design, http://www.asic-world.com/code/verilog_tutorial/peter_chambers_10_commandments.pdf

- [31] Atmel Document: “ASIC Design Guidelines”, http://www.atmel.com/dyn/resources/prod_documents/doc1205.pdf, 1999
- [32] Altera Document: “Recommended Design Practices”, http://www.altera.com/literature/hb/qts/qts_qii51006.pdf, May 2011
- [33] E. Petersen, “Single Event Effects in Aerospace”, Hoboken NJ; Wiley, 2011
- [34] P. W. Marshall, M. Carts, S. Currie, R. Reed, B. Randall, K. Fritz, K. Kennedy, M. Berg, R. Krithivasan, C. Seidleck, R. Ladbury, C. Marshall, J. Cressler, G. Niu, K. LaBel, and B. Gilbert, “Autonomous bit error rate testing in a 5AM SiGe circuit for radiation effects self test (CREST),” *IEEE Trans. Nucl. Sci.*, vol. 52, no. 6, pp. 2446–2454, Dec. 2005.
- [35] K. Label, “Considerations for a Proton Single Event Effects Guideline”http://ntrs.nasa.gov/archive/nasa/casi.ntrs.nasa.gov/20100021120_2010020334.pdf