SEU System Analysis: Not Just the Sum of All Parts

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

Melanie.D.Berg@NASA.gov

List of Acronyms

- Analog-to-Digital Converter (ADC)
- Application specific integrated circuit (ASIC)
- Block random access memory (BRAM)
- Combinatorial logic (CL)
- Device Under Test (DUT)
- Digital clock manager (DCM)
- Digital signal processor (DSP)
- Edge-triggered flip-flop (DFF)
- Error rate (dE/dt)
- Field programmable gate array (FPGA)
- Linear energy transfer (LET)
- Localized triple modular redundancy (LTMR)
- Look up table (LUT)
- Single event effects (SEEs)
- Single event functional interrupt (SEFI)
- Single event transient (SET)
- Single event upset (SEU)
- Single event upset cross section ($\sigma_{SEU}$)
- Static random access memory (SRAM)
- System frequency ($f_s$)
- Triple modular redundancy (TMR)
- Windowed shift register (WSR)
Acknowledgements

• Defense Threat Reduction Agency (DTRA)
• NASA Electronic Parts and Packaging (NEPP)
• Radiation Effects and Analysis Group (REAG) led by Kenneth LaBel and Jonathan Pellish.
Motivation

• SEU analysis of a system is complex.
• Currently, system SEU analysis is performed by component level partitioning and then:
  – Use the most dominant $\sigma_{SEU}$s for system error rate calculations, or
  – Sum component $\sigma_{SEU}$s for system error rate calculations.
• In many cases, system error rates are overestimated.
• Overestimation can cause overdesign:
  – Cost, schedule, functionality, and validation/verification can be compromised.
• The scope of this presentation is to discuss the risks involved with our current method of SEU analysis for complex systems.
Scope of Systems Regarding This Presentation

- Board or box level group of components:
  - FPGA, ASIC, ADC, microprocessor, microcontroller, memory, oscillator, voltage regulator, operational amplifier, etc…,

- Network of components within a digital design implemented in an ASIC or FPGA
  - DFFs, combinatorial logic, clock managers (DCMs), look up tables (LUTs), etc…,
Complex System SEU Evaluation

• Challenges of evaluating complex systems:
  – Fitting the entire system in an accelerated beam,
  – Having the entire system accessible for testing,
  – Enhancing the visibility of SEU-induced system errors,
  – Controlling and monitoring the system during accelerated testing, and
  – Performing SEU data analysis.

• Hence, SEU testing is generally performed using system partitions.
  – Partitioned component co-dependencies within the system should be determined and taken into account when performing SEU analysis.
  – Generally, there should not be just one SEU error rate for a system. Completely independent applications should have unique SEU error rates calculated.
Component Level Error Rates versus Error Responses

• SEU error rates: How often a component reaches an erroneous-state due to induced noise from ionization (SET or SEU).
• SEU error response: What happens when a component incurs an SET or SEU.
• Component Error rates are generally obtained from accelerated testing and $\sigma_{SEU}$ extrapolation.
• Other fault injection techniques exist, however, they are generally used for error-response studies.
Several Factors That Are Generally Not Taken Into Account during Component Level SEU Testing

• How often is the component used in the system?
• Is the component masked?
• Will the system be affected if the component incurs an SEU?
  – Can the SET dissipate prior to causing a system error?
  – Will the SET or SEU be captured by the system?
  – Is the SEU masked or is the system not communicating with the component while the SEU exists?
• If several of the same components exist, are they all equally likely to cause a system upset?
  • Can the analysis be considered linear, i.e., can we sum the component SEU error rates?
When Dominant Component Error Rates Can Be Used as the System Error Rate

- The easiest system to evaluate is one where a dominant component error rate can be applied.
  - For example, a design implemented in a commercial SRAM-based FPGA. The configuration upset rates dominate all others.

- However, this is not always straightforward:
  - If components are SEU tested separately, co-dependencies are not taken into account. This can change error rates significantly.
  - If components are co-dependent, it is important to either test as a system (sub-system) or evaluate how the co-dependencies can affect error rates.
    - For example, testing DFFs test structures versus DFFs in a system design.
Characterizing SEUs: Radiation Testing and SEU Cross Sections

SEU Cross Sections ($\sigma_{seu}$) characterize how many upsets will occur based on the number of ionizing particles the device is exposed to.

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

Terminology:

- **Flux**: Particles/(s·cm$^2$)
- **Fluence**: Particles/cm$^2$
- $\sigma_{seu}$ is calculated at several LET values (particle spectrum)
Characterizing SEUs: LET vs. SEU Cross Section Graph and How They Relate to Error Rates

\[ \sigma_{\text{seu}} = \frac{\# \text{errors}}{\text{fluence}} \]

\( dE/dt \) is calculated by integrating \( \sigma_{\text{SEU}} \) over the LET spectrum using a Weibull fit.

\( \text{LET}_{\text{SAT}} = \text{Saturated LET} \)
\( \text{LET}_{\text{TH}} = \text{Threshold LET} \)
\( \sigma_{\text{SAT}} = \text{Saturated SEU Cross Section} \)

**GEO Upset Rate:**

\[ \frac{dE}{dt} \approx \frac{C \times \sigma_{\text{sat}}}{\text{LET}^{0.25}} \]

After Ed Petterson’s figure of merit.

C varies based on the orbit. For GEO, values between 200 and 400 are common.

Example of Dominant $\sigma_{SEU}$

- If the co-dependency between components is insignificant, then component error-rates can be summed; e.g., FPGA high-level internal structures:

$$SEU \text{ Cross-Sections } (\sigma_{SEU}) = \#\text{upsets/particle/cm}^2$$

\[
P(f_s)_{error} \propto P_{configuration} + P(f_s)_{functional\ Logic} + P_{SEFI}
\]

With hardened configuration and hardened global routes (e.g., Microsemi RTAX2000s)

Taking into Account The Non-Linearity of Systems during the Extrapolation Process

How do we extrapolate $\sigma_{SEU}$s to complex designs?
What Forces Non-Linear $\sigma_{\text{SEU}}$ Extrapolation

- **System Block SEUs**
  - How often is the component active?
  - Is the component masked?
  - Are global route SETs taken into account?

- **SETs**
  - Dissipation during propagation
  - Elongation during propagation
  - Masking via logic components
  - Ringing/oscillation due to metastability (e.g., transistor push-pull during transient creation or clock tree SETs).

Each capacitance has its own $f_c$

$\quad f_c = \frac{1}{2\pi RC}$

- Cutoff frequency ($f_c$)
- Resistance (R)
- Capacitance (C)
SET Characterization via Long Inverter Chains

- Common method for testing SET behavior is to use a long chain of inverters.
- Inverter SET cross sections are calculated by counting the number of SETs and dividing by the number of inverters.
- Problem: This method assumes all inverters have the same probability of upset as seen from the observation point (I/O).
SEU Cross Sections and Error Rates – How We Apply Them to FPGA Designs

• A goal of SEU testing is to provide error rate \( \frac{dE(fs)}{dt} \) predictions to critical missions.
• \( \sigma_{SEU} \)'s from SEU testing are used to calculate \( \frac{dE(fs)}{dt} \).

\[ \frac{dE(fs)}{dt} \] for FPGA and ASIC devices are calculated using:

\[
\frac{dE(fs)}{dt} < \frac{dE_{bit}(fs)}{dt} \times (#UsedDFFs)
\]

• Assumes linearity – all DFFs are used every cycle and that they have the same probability of upset.
Background: Synchronous Design Data Path – Sample and Hold

- Synchronous design components:
  - Edge Triggered Flip-Flops (DFFs),
  - Clocks and resets (global routes), and
  - Combinatorial Logic (CL).
- All DFFs are connected to a clock.
- DFFs sample their input at the rising edge of the clock.
- CL compute between clock edges.

**Designs are complex – We modularize for simplicity**
Background: Synchronous Data Paths:

Datapath defined as StartPoint via CL to EndPoint.

Every data path has a unique delay ($\tau_{dly}$).

Delay is calculated using Static Timing Analysis (STA) design tools.

Modularization: Every DFF has a unique cone of logic

Every DFF has a function that determines its state

$$\text{EndPoint}(T) = f(\text{StartPoint}(T-1), CL)$$

- Datapath defined as StartPoint via CL to EndPoint.
- CL and routes create delay ($\tau_{dly}$) from StartPoints to EndPoints.
- Every data path has a unique $\tau_{dly}$. 
- $\tau_{dly}$ is calculated using Static Timing Analysis (STA) design tools.
How can a DFF Contain an Incorrect State from a SEU?

- DFFs have various modes of reaching a bad state due to SEUs.
- Attribute some modes to EndPoints and some to StartPoints.

We make a clear distinction between DFF SEUs based on Clock state and Capture.

EndPoint DFF SEUs + StartPoint DFF SEUs + CL SETs

DFF upsets that occur at the clock edge.

DFF upsets that occur between clock edges and are captured by EndPoints.

Single Event Transients captured by EndPoints.

Wrong function = Wrong DFF State
**Edge Triggered DFFs... Creating Deterministic Boundary Points**

*D input must be settled by rising edge of clock.*

**Master:**
- Clock Low: Transparent
- Clock High: Hold

**Slave:**
- Clock Low: Hold
- Clock High: Transparent

In order to create precise boundary points of state capture, *latches are NOT allowed* in synchronous designs.

CLK = clock

CLKB = inverted clock

StartPoint and EndPoint DFF SEUs as a Function of Clock State \((P(fs)_{DFFSEU})\)

**Low: SEU generated in Slave**

- **High→Low:** Slave Captures its SET

**High: SEU generated in Master; or SET in Slave**

- **Low→High:** Master Captures its SET

Summary of Internal DFF SEUs

\[
P(f_s)_{\text{DFFSEU}} = \alpha P(f_s)_{\text{DFFSEU}} + \beta P(f_s)_{\text{DFFSEU}}
\]

**Percentage of SEUs that occur at rising clock edge**

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

**EndPoint SEU**

**Percentage of SEUs that occur between clock edges**

- Master or slave is in hold state or Slave captures its own SET during transition from transparent to hold state.
- This is not considered a definitive state change.
- Must be captured by an EndPoint to cause an incorrect change in system state.

**StartPoint SEU**

*By definition, EndPoint SEUs are already captured into the system. How do StartPoints get captured?*

How Does a StartPoint SEU get Captured by an EndPoint?

If DFF\(_D\) flips its state @ time=\(\tau\):

\[0 < \tau < \tau_{\text{clk}} \quad \text{or} \quad \tau + \tau_{\text{dly}} < \tau_{\text{clk}}\]

Probability of capture:

\[1 - \left(\frac{\tau_{\text{dly}}}{\tau_{\text{clk}}}\right) = 1 - \tau_{\text{dly}} f_S\]
Details of Capturing StartPoint DFFs

\[ \forall_{DFF} \left( \sum_{j=1}^{\#StartPoint \text{ DFFs}} \beta P(f_s)_{DFFSEU(j)} \left(1 - \tau_{dly(j)} f_s\right) P_{logic(j)} \right) \]

- SEU generation occurs in a StartPoint between rising clock edges (\(\beta P(f_s)_{DFFSEU}\)).
- StartPoint upsets can be logically masked by logic between the StartPoint and its EndPoint.
- Design topology and temporal effects:
  - Increase path delay (# of gates) – decrease probability of capture.
  - Increase frequency – decrease probability of capture.
Synchronous System: CL SET Capture

Start Point
DFFs

Set

End Point
DFF

\( \tau \text{ width} \)

\( \tau \text{ dly} = 9.5 \text{ ns} \)

(\( A \oplus B \) \text{ AND} (C \oplus D)\)

\[ 3\text{ns} \]

\[ 1\text{ns} \]

\[ 0.5\text{ns} \]

\[ 1\text{ns} \]

\[ 1\text{ns} \]

\[ 3\text{ns} \]

\[ 4\text{ns} \]
Details of CL SET Capture

- SET Generation \( (P_{gen}) \) occurs between clock edges.
- EndPoint DFF captures the SET at a clock edge.
  - Increase frequency – increase probability of capture.
  - Increase CL – increase probability of capture.

\[
\forall DFF \left( \sum_{i=1}^{\text{# Combinatorial Cells}} \left( P_{gen(i)} P_{prop(i)} P_{logic} \tau_{width(i)} f_S \right) \right)
\]

Propagation:
- Electrical Masking from routes and gate cut-off frequencies
- \( \tau_{width} \)

Width of SET relative to clock period \( \tau_{clk} \)
Putting it All Together – Analyzed Per Particle Linear Energy Transfer (LET)

\[
\sum_{k=1}^{\#\text{EndPoint}} \sum_{j=1}^{\#\text{StartPoint DFFs}} \left( \sum_{i=1}^{\#CL} \left( P_{\text{gen}_{(i)}} \cdot P_{\text{prop}_{(i)}} \cdot P_{\text{logic}_{(i)}} \cdot \tau_{\text{width}_{(i)} fs} \right) \right) + \alpha P(fS)_{\text{DFFSEU}(k)} + \beta P(fS)_{\text{DFFSEU}(j)} (1 - \tau_{\text{dly}_{(j)} fs}) \cdot P_{\text{logic}(j)} + \gamma P(fS)_{\text{DFFSEU}(l)} \cdot P_{\text{logic}(l)}
\]

StartPoints and CL need to be captured by an EndPoint… hence data path derating factors exist.

Component Contribution to \( \sigma_{\text{SEU}} \) across Frequency and Gate Count

<table>
<thead>
<tr>
<th></th>
<th>Frequency</th>
<th># of Gates in Path</th>
</tr>
</thead>
<tbody>
<tr>
<td>EndPoint</td>
<td>Directly Proportional</td>
<td>N/A</td>
</tr>
<tr>
<td>StartPoint</td>
<td>Inversely Proportional</td>
<td>Inversely Proportional</td>
</tr>
<tr>
<td>CL</td>
<td>Directly Proportional</td>
<td>Directly Proportional</td>
</tr>
</tbody>
</table>

Radiation Test Structures: Windowed Shift Registers (WSR) and Triple Modular Redundancy (TMR)

**NO-TMR**

\[ \tau_{dly_{WSR0}} > \tau_{dly_{WSR8}} \]

**Localized-TMR (LTMR)**

\[ \tau_{dly_{WSR0}} \]

\[ \tau_{dly_{WSR8}} \]

LTMR SEU Response

- Internal DFF upsets are 100% masked: StartPoint and EndPoint $P_{logic} = 0$;
- SETs from shared data path can propagate into all DFFs
- Voters can upset
Using the Model to Analyze Heavy Ion SEU Cross Sections

Microsemi ProASIC3 FPGA: LET = 20.3 MeV-cm²/mg
NoTMR versus LTMR- checker pattern

SEU Characterization of A Complex System: Microprocessor

Test-As-You-Fly versus Using Fest Structures and Extrapolation

Test Structures versus Final Designs

• Although error rates and error responses are design dependent, useful information can be extrapolated from test structures versus the final design.

• Why use test structures versus final designs?
  – By the time the final design is complete, it is usually too late to perform radiation testing on it.
  – Can be too difficult to apply input-stimuli to a final design.
  – Can be too difficult to monitor DUT responses.

The following slides give more insight into the benefits of using test structures versus full designs during radiation testing.
# Best Practice for Radiation Testing: Logic Replication for Statistics

<table>
<thead>
<tr>
<th>Best-Practice for DUT Test Structure Development</th>
<th>How Application-Specific Test Structures Violate Best-Practice Considerations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test structures should contain a large number of replicated logic in order to increase statistics: e.g., shift-registers with thousands of stages.</td>
<td>• Statistics are poor because usually there is not a significant amount of replication.</td>
</tr>
<tr>
<td></td>
<td>• In addition, trends for specific elements are not able to be clearly identified / established.</td>
</tr>
</tbody>
</table>
Best Practice for Radiation Testing: State Space Traversal

<table>
<thead>
<tr>
<th>Best-Practice for DUT Test Structure Development</th>
<th>How Application-Specific Test Structures Violate Best-Practice Considerations</th>
</tr>
</thead>
<tbody>
<tr>
<td>A test structure’s state space should be traversable such that it can be covered within one radiation test run.</td>
<td>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 SEU data that are uncharacteristic of the design.</td>
</tr>
</tbody>
</table>

Each directed test walks through a specified path...

# Best Practice for Radiation Testing: Logic Masking

<table>
<thead>
<tr>
<th>Best-Practice for DUT Test Structure Development</th>
<th>How Application-Specific Test Structures Violate Best-Practice Considerations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Logic masking should be minimized or controllable.</td>
<td>Application-specific test structures contain a significantly higher number of masked data paths than test structures.</td>
</tr>
</tbody>
</table>

**Best Practice**

- Logic masking should be minimized or controllable.

**Considerations**

- Application-specific test structures contain a significantly higher number of masked data paths than test structures.

![Diagram of logic masking](image)

\[ 0 < P_{\text{logic}} < 1 \]

\[ 0 < P_{\text{logic}} < 1 \]

# Best Practice for Radiation Testing: Avoiding Unrealistic SEU Accumulation

<table>
<thead>
<tr>
<th>Best Practice characteristics of a DUT design</th>
<th>How Application-Specific Test Structures Violate Best-Practice Considerations</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Avoid unrealistic SEU accumulation from accelerated testing:</strong></td>
<td>Application-specific test structures take up most of the DUT’s area. There are a lot of co-dependencies between logic.</td>
</tr>
<tr>
<td>• Flush through test structures; e.g., shift-registers.</td>
<td>Hence, it is difficult to control SEU accumulation in an accelerated test environment.</td>
</tr>
<tr>
<td>• Small number of gates per sub-test structure; e.g., testing hundreds of counters.</td>
<td></td>
</tr>
</tbody>
</table>

**SRAM Based FPGAs: Scrubbing (correcting) configuration SEUs.** Extremely important during accelerated testing... must keep up with the particle flux to avoid accumulation.
**Best Practice for Radiation Testing: Increasing Visibility**

<table>
<thead>
<tr>
<th>Best Practice characteristics of a DUT design</th>
<th>How Application-Specific Test Structures Violate Best-Practice Considerations</th>
</tr>
</thead>
<tbody>
<tr>
<td>All (or a significant percentage of) potential upsets should be observable during testing.</td>
<td>A significant number of upsets in a complex design are generally not observable during radiation testing.</td>
</tr>
<tr>
<td>Test structures can easily be designed to enhance observable nodes; e.g., shift-registers and counters.</td>
<td>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.</td>
</tr>
</tbody>
</table>
Benefits of Testing Application Specific Designs

• Increase observation error responses specific to the application.

• However, the user must be aware of the following:
  – Unrealistic SEU accumulation in an accelerated environment.
  – Limited visibility due to masking and fractional state space traversal.
  – Poor statistics due to the variance in design circuits.

• $\sigma_{\text{SEU}}$s will most likely have a large variance if circuits are not able to be isolated and controlled.
CASE Study

• DUT is a Xilinx V5QV – radiation hardened FPGA.

• Application-specific test structure is an embedded microprocessor (Micro-blaze™).

• Goal is to determine error rates for using an embedded Micro-blaze™ processor in the Xilinx V5QV with and without cache.
  
  – Question: Does using cache in embedded memory increase the $\sigma_{SEU}$s such that the Micro-blaze™ will not meet project requirements?
Suggestions on How to Test the Application Specific Design

• Because the goal is to study caching SEU effects, test-plan should have a test design that contains cache and one that does not.

• Test basic structures such as shift-registers and counters to get an underlying understanding of device SEU characteristics.

• Basic test-structure analysis characterizes:
  – Sequential memory elements (DFFs),
  – Combinatorial logic (CL), and
  – Global routes.

• Increase visibility of the Micro-blaze™ during testing.
Processor and SRAM Communication

SRAM: Static random access memory
BRAM: Block random access memory

• **Processors talk to memory**

• Most processor radiation tests detect errors by erroneous SRAM memory writes.

• Visibility is significantly limited.

• **We increase visibility by replacing external SRAM with the REAG low-cost digital Tester (LCDT)**

---

**Micro-blaze™**

Cache

**SRAM Interface**

ALU

**Data Write**

Fetch Instruction
Or Data Read

**LCDT** using FPGA BRAM

More on Increasing Visibility with Microprocessor Testing (1)

• As previously stated, the embedded SRAM in the tester (BRAM) takes the place of normal memory accesses.
• In addition, each memory access is time stamped and logged in alternate bank of BRAM. Only the last 512 accesses are kept.
• After each test run, the time stamped logs are output to the user.
More on Increasing Visibility with Microprocessor Testing (2)

DUT: device under test

- Halted
- Error
- Trace Instruction
- Trace Valid Instruction
- Trace Exception Taken
- Trace Exception Kind
- Trace Register Write
- Trace Register Address
- Trace data cache Request
- Trace data cache Hit
- Trace Data cache Ready
- Trace Data cache Read
- Trace Instruction cache Request
- Trace Instruction cache Hit

TESTER

Watchdogs

Send watchdog errors to host computer

Summary of Case Study Test Enhancements

• Visibility was increased by isolating memory accesses as follows:
  – Moving the instruction and data storage to the LCDT for traffic observation.
  – Performing tests with and without cache to determine the influence cache has on upsets.

• Differentiating global upsets from the normal data set:
  – Helped to understand which upsets are prominent.
  – Gave insight to how the use of cache will affect $\sigma_{SEU}$s.

• Monitoring internal Micro-blaze™ signals
  – $\sigma_{SEU}$s are not reliant on detecting erroneous memory read and writes anymore. Data are too limited and uninformative with solely relying on memory reads and writes.
  – Can now determine when a processor crashes and how.
Comparing Micro-blaze™ $\sigma_{SEU}$s and Global Clock $\sigma_{SEUs}$

SEU Cross Sections:
Cache vs. No Cache with Global Routes

$\sigma_{SEU}$ (cm²/design) vs. LET (MeV·cm²/mg)

- Configuration 6: Cache
- Configuration 5: No Cache
- Global Routes
Floor Is Open To Discussion