Intel® Stratix® 10 SEU Mitigation User Guide

Updated for Intel® Quartus® Prime Design Suite: 21.2
Contents

1. Intel® Stratix® 10 SEU Mitigation Overview ................................................................. 3
   1.1. SEU Mitigation Techniques for Intel Stratix® 10 Devices ........................................... 3
   1.2. Configuration RAM .................................................................................................. 4
   1.3. Memory Blocks Error Correction Code Support ......................................................... 4
   1.4. Triple-Module Redundancy ...................................................................................... 5
   1.5. Failure Rates .......................................................................................................... 6

2. Intel Stratix 10 Mitigation Techniques for CRAM .............................................................. 7
   2.1. CRAM Error Detection and Correction ...................................................................... 7
   2.1.1. Error Message Queue ...................................................................................... 7
   2.1.2. SEU_ERROR Pin Behavior .............................................................................. 8
   2.2. Internal Scrubbing and Priority Scrubbing ............................................................... 8
   2.2.1. Internal Scrubbing ....................................................................................... 9
   2.2.2. Priority Scrubbing ....................................................................................... 9
   2.3. SEU Sensitivity Processing ..................................................................................... 9
   2.3.1. Advanced SEU Detection IP Core .................................................................. 10
   2.4. Designating the Sensitivity of the Design Hierarchy ................................................. 15
   2.4.1. Hierarchy Tagging ....................................................................................... 15
   2.5. Evaluating a System’s Response to Functional Upsets ............................................ 16
   2.5.1. Intel Quartus Prime Fault Injection Debugger .................................................. 16

3. Intel Stratix 10 SEU Mitigation Implementation Guides .................................................. 18
   3.1. Setting SEU_ERROR Pin ....................................................................................... 18
   3.2. Intel Quartus Prime SEU Software Settings ............................................................ 18
   3.3. Enabling Priority Scrubbing ................................................................................... 19
   3.4. Performing Hierarchy Tagging ............................................................................. 19
   3.5. Programming Sensitivity Map Header File into Memory ........................................... 20
   3.6. Performing Lookup for Sensitivity Map Header ...................................................... 21
   3.7. Using the Fault Injection Debugger ....................................................................... 23
   3.7.1. Enabling the Fault Injection Debugger ............................................................ 23
   3.7.2. Launching and Setting Up the Fault Injection Debugger ....................................... 23
   3.7.3. Configuring Your Device and the Fault Injection Debugger ................................. 24
   3.7.4. Constraining Regions for Fault Injection ........................................................... 24
   3.7.5. Injecting Errors ............................................................................................ 25
   3.7.6. Injecting Double Adjacent Errors ..................................................................... 26

4. Advanced SEU Detection Intel FPGA IP References ..................................................... 27
   4.1. Advanced SEU Detection Intel FPGA IP Parameter Settings .................................... 27
   4.2. Advanced SEU Detection IP Core Ports .................................................................. 28

5. Intel Stratix 10 Fault Injection Debugger References ..................................................... 30
   5.1. Fault Injection Debugger Interface Parameters ....................................................... 30
   5.2. Fault Injection Debugger Command-Line Interface ................................................ 30

6. Intel Stratix 10 SEU Mitigation User Guide Archives .................................................... 33

7. Document Revision History for the Intel Stratix 10 SEU Mitigation User Guide .............. 34
1. Intel® Stratix® 10 SEU Mitigation Overview

Single event upsets (SEUs) are rare, unintended changes in the state of an FPGA's internal memory elements caused by cosmic radiation effects. The change in state is a soft error and the FPGA incurs no permanent damage. Because of the unintended memory state, the FPGA may operate erroneously until background scrubbing fixes the upset.

The Intel® Quartus® Prime software offers several features to detect and correct the effects of SEU, or soft errors, as well as to characterize the effects of SEU on your designs. Additionally, some Intel FPGAs contain dedicated circuitry to help detect and correct errors.

Intel FPGAs have memory in user logic (block memory and registers) and in Configuration Random Access Memory (CRAM). The Intel Quartus Prime Programmer loads the CRAM with a .sof file. Then, the CRAM configures all FPGA logic and routing. If an SEU strikes a CRAM bit, the effect can be harmless if the device does not use the CRAM bit. However, the effect can be severe if the SEU affects critical logic or internal signal routing.

Often, a design does not require SEU mitigation because of the low chance of occurrence. However, for highly complex systems, such as systems with multiple high-density components, the error rate may be a significant system design factor. If your system includes multiple FPGAs and requires very high reliability and availability, you should consider the implications of soft errors. Use the techniques in this chapter to detect and recover from these types of errors.

Related Information
- Introduction to Single-Event Upsets
- Understanding Single Event Functional Interrupts in FPGA Designs
- Intel Stratix 10 SEU Mitigation User Guide Archives on page 33

1.1. SEU Mitigation Techniques for Intel Stratix® 10 Devices

Intel Stratix® 10 SEU mitigation features can benefit the system by:
- Ensuring the system functions properly all the time
- Preventing a system malfunction caused by an SEU event.
- Handling the SEU event if it is critical to the system.
Table 1.  SEU Mitigation Areas and Approaches for Intel Stratix 10 Devices

<table>
<thead>
<tr>
<th>Area</th>
<th>SEU Mitigation Approach</th>
</tr>
</thead>
<tbody>
<tr>
<td>Error Detection and Correction</td>
<td>You can enable the error detection and correction (EDC) feature for detecting CRAM SEU events and automatic correction of CRAM contents.</td>
</tr>
<tr>
<td>Memory block error correction code</td>
<td>Intel Stratix 10 designs M20K memory blocks with special layout techniques and Error Correction Code (ECC) to reduce SEU Failures in time (FIT) rate to almost zero.</td>
</tr>
<tr>
<td>SEU Sensitivity processing</td>
<td>You can use sensitivity processing to identify if the SEU on a CRAM bit location is critical or not critical to the function of your compiled FPGA design bitstream file.</td>
</tr>
<tr>
<td>Fault injection</td>
<td>You can use fault injection feature to validate the system response to the SEU event by changing the CRAM state to trigger an error.</td>
</tr>
<tr>
<td>Hierarchical tagging</td>
<td>A complementary capability to sensitivity processing and fault injection for reporting SEU and constraining injection to specific portions of design logic.</td>
</tr>
<tr>
<td>Triple Modular Redundancy (TMR)</td>
<td>You can implement TMR technique on critical logic such as state machines.</td>
</tr>
</tbody>
</table>

1.2. Configuration RAM

FPGAs use memory both in user logic (bulk memory and registers) and in Configuration RAM (CRAM). CRAM is the memory loaded with the user's design. The CRAM configures all logic and routing in the device. If an SEU strikes a CRAM bit, the effect can be harmless if the CRAM bit is not in use. However, a functional error is possible if it affects critical internal signal routing or critical lookup table logic bits as part of the user's design.

Related Information
Intel Stratix 10 Configuration User Guide
Provides more information about CRAM and user design in Intel Stratix 10 devices.

1.3. Memory Blocks Error Correction Code Support

ECC detects and corrects data errors at the output of the memory.

Only M20K blocks and eSRAM blocks support the ECC feature.

If you engage the ECC feature, you cannot use the following features:
- Byte enable
- Coherent read

M20K Blocks
For M20K blocks, ECC performs single-error correction, double-adjacent-error correction, and triple-adjacent-error correction in a 32-bit word. However, ECC cannot guarantee detection or correction of non-adjacent two-bit or more errors.
The M20K blocks have built-in support for ECC when in ×32-wide simple dual-port mode.

- When you engage the ECC feature, the M20K runs slower than the non-ECC simple dual-port mode. However, you can enable optional ECC pipeline registers before the output decoder to achieve higher performance compared to non-pipeline ECC mode at the expense of one-cycle latency.
- Two ECC status flag signals—e (error) and ue (uncorrectable error) indicate the M20K ECC status. The status flags are part of the regular outputs from the memory block.

**eSRAM Blocks**

For eSRAM blocks, ECC performs single-error correction and double-error detection in a 64-bit word.

The eSRAM blocks have built-in support for ECC when in ×64-wide simple dual-port mode.

- Two ECC status flag signals—c{7:0}_error_correct_0 (error corrected) and c{7:0}_error_detect_0 (error detected) indicate the eSRAM ECC status.

**Related Information**

- Embedded Memory User Guide
  Provides more information about implementing ECC with Embedded Memory IP cores.
- Intel Stratix 10 Embedded Memory User Guide
  Provides more information about ECC in Intel Stratix 10 memory blocks.

### 1.4. Triple-Module Redundancy

Use Triple-Module Redundancy (TMR) if your system cannot suffer downtime due to SEU. TMR is an established SEU mitigation technique for improving hardware fault tolerance. A TMR design has three identical instances of hardware with voting hardware at the output. If an SEU affects one of the hardware instances, the voting logic notes the majority output. This operation masks malfunctioning hardware.

With TMR, your design does not suffer downtime in the case of a single SEU; if the system detects a faulty module, the system can scrub the error by reprogramming the module. The error detection and correction time is many orders of magnitude less than the MTBF of SEU events. Therefore, the system can repair a soft interrupt before another SEU affects another instance in the TMR application.

The disadvantage of TMR is its hardware resource cost: it requires three times as much hardware in addition to voting logic. You can minimize this hardware cost by implementing TMR for only the most critical parts of your design.

There are several automated ways to generate TMR designs by automatically replicating designated functions and synthesizing the required voting logic. Synopsys offers automated TMR synthesis.
1.5. Failure Rates

The Soft Error Rate (SER) or SEU reliability is expressed in Failure in Time (FIT) units. One FIT unit is one soft error occurrence per billion hours of operation.

- For example, a design with 5,000 FIT experiences a mean of 5,000 SEU events in one billion hours (or 114,155.25 years). Because SEU events are statistically independent, FIT is additive. If a single FPGA has 5,000 FIT, then ten FPGAs have 50,000 FIT (or 50K failures in 114,155.25 years).

Another reliability measurement is the mean time to failure (MTTF), which is the reciprocal of the FIT or 1/FIT.

- For a FIT of 5,000 in standard units of failures per billion hours, MTTF is:
  \[ \frac{1}{5,000} = \frac{1 \text{ billion}}{5,000} = 200,000 \text{ hours} = 22.83 \text{ years} \]

SEU events follow a Poisson distribution and the cumulative distribution function (CDF) for mean time between failures (MTBF) is an exponential distribution. For more information about failure rate calculation, refer to the Intel FPGA Reliability Report.

Neutron SEU incidence varies by altitude, latitude, and other environmental factors. The Intel Quartus Prime software provides SEU FIT reports based on compiles for sea level in Manhattan, New York. The JESD89A specification defines the test parameters.

**Tip:**

You can convert the data to other locations and altitudes using calculators, such as those at www.seutest.com. Additionally, you can adjust the SEU rates in your project by including the relative neutron flux (calculated at www.seutest.com) in your project's `.qsf` file.
2. Intel Stratix 10 Mitigation Techniques for CRAM

This chapter explains the SEU mitigation techniques for Intel Stratix 10 CRAM. For more information about the embedded memory ECC feature, refer to the *Intel Stratix 10 Embedded Memory User Guide*.

**Related Information**

- Embedded Memory User Guide
  Provides more information about implementing ECC with Embedded Memory IP cores.
- Intel Stratix 10 Embedded Memory User Guide
  Provides more information about ECC in Intel Stratix 10 memory blocks.

2.1. CRAM Error Detection and Correction

Intel Stratix 10 devices feature on-chip EDC circuitry to detect soft errors. If an error caused by SEU event is correctable, the Intel Stratix 10 FPGA corrects it if you enable the internal scrubbing feature.

**Table 2. Detection and Correction of Error Types**

<table>
<thead>
<tr>
<th>Error Type</th>
<th>Detection</th>
<th>Correction</th>
</tr>
</thead>
<tbody>
<tr>
<td>Single bit error</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Double adjacent errors</td>
<td>Yes(1)</td>
<td>—</td>
</tr>
<tr>
<td>Multiple bit errors</td>
<td>Detects up to 8 CRAM bits that fit in a rectangular box of 8 CRAM bits (8x1, 4x2, 1x8 or 2x4 errors)</td>
<td>—</td>
</tr>
</tbody>
</table>

2.1.1. Error Message Queue

The Intel Stratix 10 device error message queue stores the error messages when detecting an SEU error. The error message queue is capable of storing a maximum of four different messages. Each error message contains information about the sector address, error type, and error location. You can retrieve the contents of the error message queue using the following tools:

- Fault Injection Debugger tool
- Advanced SEU Detection Intel FPGA IP

---

(1) In Intel Stratix 10 devices, double adjacent errors are also detected as multiple bit errors.
2. Intel Stratix 10 Mitigation Techniques for CRAM

Table 3. Error Message Queue Description

<table>
<thead>
<tr>
<th>Name</th>
<th>Width</th>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sector address (Most significant 32-bit</td>
<td>32</td>
<td>31:24</td>
<td>Reserved</td>
</tr>
<tr>
<td>word in avst_seu_source_data signal</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>23:16</td>
<td>Address of sector with error</td>
</tr>
<tr>
<td></td>
<td></td>
<td>15:4</td>
<td>Reserved</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3:0</td>
<td>Number of errors detected in the sector minus one</td>
</tr>
<tr>
<td>Error location(2) (Least significant</td>
<td>32</td>
<td>31:29</td>
<td>Bit 31:29—Error type:</td>
</tr>
<tr>
<td>32-bit word in avst_seu_source_data signal)</td>
<td></td>
<td></td>
<td>• 001—single bit error</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 010—uncorrectable double adjacent bits error</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 011—uncorrectable multiple bits error</td>
</tr>
<tr>
<td></td>
<td></td>
<td>28</td>
<td>Correction Status:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 0=Not corrected</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 1=Corrected</td>
</tr>
<tr>
<td></td>
<td></td>
<td>23:12</td>
<td>Bit position within frame</td>
</tr>
<tr>
<td></td>
<td></td>
<td>11:0</td>
<td>Combined of Row and Frame index</td>
</tr>
</tbody>
</table>

Note: Intel recommends that you turn on the Internal Scrubbing feature. If an error is detected in a sector, and you did not enable the Internal Scrubbing option, the SEU feature for that particular sector is turned off. Additionally, subsequent SEU occurrence in the same sector either correctable or uncorrectable error, is not detected.

Related Information

- Advanced SEU Detection IP Core Ports on page 28
- Using the Fault Injection Debugger on page 23
- Has the error type definition of the error message queue been changed in Intel Stratix 10 device since Intel Quartus Prime Pro version 19.3?

2.1.2. SEU_ERROR Pin Behavior

The SEU_ERROR signal goes high whenever the error message queue contains one or more error messages. The signal stays high if there is an error message in the queue. The SEU_ERROR signal goes low only when the SEU error message queue is empty which happens after you shift out all the error messages.

You must set to the SEU_ERROR pin function to observe the SEU_ERROR pin behavior.

Related Information

Intel Quartus Prime SEU Software Settings on page 18

2.2. Internal Scrubbing and Priority Scrubbing

Intel Stratix 10 devices support automatic CRAM error correction without reloading the original CRAM contents from an external copy of the original programming bit-stream.

(2) For single bit error with internal scrubbing, the error location provides the error bit position. For multiple bits error or single bit error without internal scrubbing, bit [23:0] returns 0.
2.2.1. Internal Scrubbing

The internal scrubbing feature automatically corrects single-bit errors.

Intel recommends that you turn on internal scrubbing. If you do not enable internal scrubbing, the device turns off the SEU mitigation feature for a sector after an error occurs in the sector. Subsequently, the device stops detection of correctable or uncorrectable SEU occurrence in the affected sector.

If you enable the internal scrubbing feature, you must still plan your recovery sequence. Although the scrubbing feature can restore the CRAM array to the intended configuration, a latency period exists between detection and correction of the soft error. During this latency period, the Intel Stratix 10 device may be operating with errors.

For uncorrectable errors, the SDM periodically inserts an error message to the error message queue. The insertion reasserts the SEU_ERROR pin to alert you about the error.

2.2.2. Priority Scrubbing

You can assign portions of the design as high priority sectors for internal scrubbing; the unassigned portions become low priority sectors. The Intel Stratix 10 EDC circuitry detects and corrects errors that occur in the high priority sectors more frequently than the other sectors.

Related Information
Enabling Priority Scrubbing on page 19

2.3. SEU Sensitivity Processing

Reconfiguring a running FPGA has a significant impact on the system using the FPGA. When planning for SEU recovery, account for the time required to bring the FPGA to a state consistent with the current state of the system. For example, if an internal state machine is in an illegal state, it may require reset. In addition, the surrounding logic may need to account for this unexpected operation.

Often an SEU impacts CRAM bits not used by the implemented design. Even in a fully utilized FPGA design, many configuration bits are not used because they control logic and routing wires that are not used in a design. Depending on the implementation, only 40% of all CRAM bits can be used even in the most heavily utilized devices. This means that only 40% of SEU events require intervention, and you can ignore 60% of SEU events. The utilized bits are considered as critical bits while the non-utilized bits are considered as non-critical bits.

Additionally, there may be portions of the implemented design are not utilized in the FPGA’s function. Examples may include test circuitry implemented but not important to the operation of the device, or other non-critical functions that may be logged but do not need to be reprogrammed or reset.

You can perform SEU sensitivity processing using the Advanced SEU Detection IP core.
2.3.1. Advanced SEU Detection IP Core

The Advanced SEU Detection IP core does the following:

- Communicates with the Secure Device Manager (SDM) to detect SEU event, send or receive commands or responses from SDM for reporting SEU error.
- Read Sensitivity Map Header (.smh) Revision 4 file to allow On-Chip or Off-Chip Lookup Sensitivity Processing, and report criticality of SEU error occurred in device based on region specified in the .smh file.

The Advanced SEU Detection IP core allows you to perform sensitivity processing for SEU errors at runtime. The Advanced SEU Detection IP core supports the following implementations:

- On-Chip Lookup Sensitivity Processing—The sensitivity processing soft IP provides error location reporting and lookup.
- Off-Chip Lookup Sensitivity Processing—An external unit (such as a microprocessor) performs error location lookup using the error message queue information.

Related Information

- SMH Lookup on page 13
- Performing Lookup for Sensitivity Map Header on page 21
- Performing Hierarchy Tagging on page 19
- Advanced SEU Detection Intel FPGA IP Parameter Settings on page 27
- Advanced SEU Detection IP Core Ports on page 28
- Intel Quartus Prime SEU Software Settings on page 18

2.3.1.1. Release Information for Advanced SEU Detection Intel FPGA IP


The Intel FPGA IP version (X.Y.Z) number can change with each Intel Quartus Prime software version. A change in:

- X indicates a major revision of the IP. If you update the Intel Quartus Prime software, you must regenerate the IP.
- Y indicates the IP includes new features. Regenerate your IP to include these new features.
- Z indicates the IP includes minor changes. Regenerate your IP to include these changes.

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IP Version</td>
<td>19.1.0</td>
</tr>
<tr>
<td>Intel Quartus Prime Version</td>
<td>21.2</td>
</tr>
<tr>
<td>Release Date</td>
<td>2021.06.23</td>
</tr>
</tbody>
</table>
2.3.1.2. On-Chip Lookup Sensitivity Processing

The Advanced SEU Detection IP core reads the error message queue content and then compares single-bit error locations with a sensitivity map. This check determines whether or not the failure affects the device operation.

Figure 1. System Overview for On-Chip Lookup Sensitivity Processing with Advanced SEU Detection IP Core

The on-chip lookup sensitivity processing is as follows:

1. The SEU_ERROR is asserted when there is an SEU error.
2. The Advanced SEU Detection IP core retrieves the SEU error message from SDM. 
   *Note:* The Advanced SEU Detection IP core asserts sys_error signal if error occurs in system while retrieving the SEU error message.
3. The Advanced SEU Detection IP core starts performing sensitivity processing. During this process:
   - The Advanced SEU Detection IP core asserts the busy signal.
   - The Advanced SEU Detection IP core reads the .smh file. You must provide the information for the memory access logic and external memory.
4. The Advanced SEU Detection IP core deasserts the busy signal to indicate completion of sensitivity processing and reports the criticality of the SEU error through the following signals:
   - critical_error
   - noncritical_error
   - regions_report
   - seu_data (optional)

2.3.1.3. Off-Chip Lookup Sensitivity Processing

The Advanced SEU Detection IP core reads the error message queue content and presents information to a system processor. The processor determines whether the failure affects the device operation. The system processor implements the algorithm to perform a lookup against the .smh.
The off-chip lookup sensitivity processing is as follows:

1. The `SEU_ERROR` is asserted when there is an SEU error.
2. The Advanced SEU Detection IP core retrieves the error message from SDM and stores it in the internal FIFO.
   
   *Note*: The Advanced SEU Detection IP core asserts `sys_error` signal if error occurs in system while retrieving the error message.

3. The Advanced SEU Detection IP core asserts the `avst_seu_source_valid` signal to indicate an error message is available.
4. The external sensitivity processor must monitor the `avst_seu_source_valid` signal of the Advanced SEU Detection IP core. If there is an error message available, the processor can start to read the SEU error through the Avalon® streaming interface and perform lookup against the sensitivity map to determine the criticality of the SEU error.
2.3.1.3.1. Off-Chip Lookup Sensitivity Processing Operation Flow

Figure 3. Off-Chip Lookup Sensitivity Processing Operation Flow

2.3.1.4. SMH Lookup

The .smh file represents a hash of the CRAM bit settings on a design. Related groups of CRAM are mapped to a signal bit in the sensitivity array. During an SEU event, the application can perform a lookup against the .smh to determine if a bit is used. By using the information about the location of a bit, you can reduce the effective soft error rate in a running system.
The following criteria determine the criticality of a CRAM location in your design:

- **Routing**—All bits that control a utilized routing line.
- **Adaptive logic modules (ALMs)**—If you configure an ALM, the IP core considers all CRAM bits related to that ALM sensitive.
- **Logic array block (LAB) control lines**—If you use an ALM in a LAB, the IP core considers all bits related to the control signals feeding that LAB sensitive.
- **M20K memory blocks and digital signal processing (DSP) blocks**—If you use a block, the IP core considers all CRAM bits related to that block sensitive.

### Related Information

Advanced SEU Detection IP Core on page 10

#### 2.3.1.4.1. SMH Revision 4 File Format

<table>
<thead>
<tr>
<th>Block</th>
<th>Sub-block</th>
<th>32-bit Word</th>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sensitivity map header</td>
<td>—</td>
<td>0</td>
<td>[31:0]</td>
<td>Identification word for SMH format and its version, 0xEE45341.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1</td>
<td>[31:8]</td>
<td>Reserved</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>[7:0]</td>
<td>ASD Region bitmask size. ASD region bitmask size is the upper bound power of 2 for maximal ASD Region ID in design, can be 1,2,4,8,16 or 32.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2</td>
<td>[31:0]</td>
<td>Address of the Sector Information block.</td>
</tr>
<tr>
<td>Sectors Information block</td>
<td>Sector 0 Information</td>
<td>0</td>
<td>[31:0]</td>
<td>Address of the sector 0 encoding scheme.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1</td>
<td>[31:0]</td>
<td>Address of the encoded sector 0 sensitivity data.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2</td>
<td>[23:8]</td>
<td>The number of ASD region bitmasks used by sector 0 (i.e. number of SMH tags). Value of 0 indicates that there are no sensitive bits in a sector.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>[7:0]</td>
<td>The sector 0 SMH tag size in bits, can be 1, 2, 4, 8.</td>
</tr>
<tr>
<td></td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td></td>
<td>Sector N Information</td>
<td>N<em>3 .. N</em>3+2</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>Sectors Encoding block</td>
<td>Sector Encoding 0</td>
<td>0</td>
<td>[31:16]</td>
<td>Identification word 0xEEEE</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>[15:0]</td>
<td>Size of a single frame encoding (i.e. bit-&gt;tag index) map in bytes.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1</td>
<td>[31:0]</td>
<td>Address of the frame information (FADD).</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2</td>
<td>[31:0]</td>
<td>Address of the frame encoding map (EADD).</td>
</tr>
<tr>
<td></td>
<td>FADD</td>
<td></td>
<td>[31:20]</td>
<td>Index of encoding map for frame 0</td>
</tr>
<tr>
<td></td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td></td>
<td>FAAD+K</td>
<td></td>
<td>[31:20]</td>
<td>Index of encoding map for last frame</td>
</tr>
</tbody>
</table>

*continued...*
### 2. Stratix 10 Mitigation Techniques for CRAM

#### 2.1. Designating the Sensitivity of the Design Hierarchy

In the Intel Quartus Prime software, you indicate the criticality of each logic block by generating partitions, and assigning a sensitivity ID tag to each partition. The Intel Quartus Prime software stores this information in a Sensitivity Map Header File (.smh).

When an error occurs during system operation, the system determines the impact of the error by looking up the classification in the .smh file. The system can then take corrective action based on the classification.

*Note:* You must have a licensed version of Intel Quartus Prime software to generate .smh files.

To access the .smh file, you must add an instance of the Advanced SEU Detection IP core to your design.

#### 2.1.1. Hierarchy Tagging

The Intel Quartus Prime hierarchy tagging feature allows you to improve design-effective FIT rate by tagging only the critical logic for device operation.
You can also define the system recovery procedure based on knowledge of logic impaired by SEU. This technique reduces downtime for the FPGA and the system in which the FPGA resides. Other advantages of hierarchy tagging are:

- Increases system stability by avoiding disruptive recovery procedures for inconsequential errors.
- Allows diverse corrective action for different design logic.

The .smh file contains a mask for design sensitive bits in a compressed format. The Intel Quartus Prime software generates the sensitivity mask for the entire design.

### 2.5. Evaluating a System's Response to Functional Upsets

SEUs can strike any memory element, so you must test the system to ensure a comprehensive recovery response. The Intel Quartus Prime software includes the Fault Injection Debugger to aid in SEU recovery. You can use the Fault Injection Debugger graphically with the GUI, or you can use command line assignments.

#### 2.5.1. Intel Quartus Prime Fault Injection Debugger

You can detect and debug single event upset (SEU) using the Fault Injection Debugger in the Intel Quartus Prime software. Use the debugger to inject errors into the configuration RAM (CRAM) of an Intel Stratix 10 FPGA device.

With the Fault Injection Debugger, you can operate the FPGA in the system and inject random CRAM bit flips. These simulated SEU strikes allow you to observe how the FPGA and the system detect and recover from SEUs. Depending on the results, you can refine the system's recovery sequence.

**Figure 4. Fault Injection Debugger Overview Block Diagram for Intel Stratix 10 Devices**

The Fault Injection Debugger allows you to perform the following:

- Inject single-bit error to either:
  - Random location
  - Specified region
- Report error information by reading the error message queue

**Note:** You must have a licensed version of the Intel Quartus Prime software to use the Fault Injection Debugger tool.
2. Intel Stratix 10 Mitigation Techniques for CRAM

UG-S10SEU | 2021.07.05

Related Information

- Using the Fault Injection Debugger on page 23
- Fault Injection Debugger Interface Parameters on page 30
- Fault Injection Debugger Command-Line Interface on page 30
- Injecting Errors on page 25
3. Intel Stratix 10 SEU Mitigation Implementation Guides

3.1. Setting SEU_ERROR Pin

To set the SEU_ERROR pin function in the Intel Quartus Prime software, perform the following steps:

1. On the Assignments menu, click Device.
2. In the Device and Pin Options select the Configuration category and click Configuration Pins Options.
3. In the Configuration Pin window, turn-on the USE SEU_ERROR output.
4. Select any unused SDM pin from the drop-down selection to implement the SEU_ERROR pin function.
5. Click OK to confirm and close the Configuration Pin window.

3.2. Intel Quartus Prime SEU Software Settings

Use the Error Detection CRC settings in the Device and Pin Options window to set the minimum SEU interval, enable error detection and internal scrubbing features, enable the generation of .smh file, and enable fault injection.

1. From the Intel Quartus Prime menu, click Assignments ➤ Device.
2. In the Device window, click Device and Pin Options.
3. In the Device and Pin Options window, select Error Detection CRC category and specify the following settings:

<table>
<thead>
<tr>
<th>Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enable error detection check</td>
<td>Turn on to enable the error detection feature. This option is required for sensitivity processing and fault injection, or if you want to observe the SEU_ERROR pin behavior.</td>
</tr>
<tr>
<td>Minimum SEU interval</td>
<td>Specify a value of 0 to 10000 milliseconds to set the minimum time between two checks of the same bit. To check as frequently as possible, specify 0.</td>
</tr>
<tr>
<td>Enable internal scrubbing</td>
<td>Turn on to enable the error correction feature. This option is required for sensitivity processing.</td>
</tr>
<tr>
<td>Generate SEU sensitivity map file (.smh)</td>
<td>Turn on to generate the .smh file. This option is required for sensitivity processing.</td>
</tr>
<tr>
<td>Allow SEU fault injection</td>
<td>Turn on to enable injecting fault using the Fault Injection Debugger.</td>
</tr>
</tbody>
</table>

4. Click OK.

Related Information
- SEU_ERROR Pin Behavior on page 8
- Advanced SEU Detection IP Core on page 10
3.3. Enabling Priority Scrubbing

Use Intel Quartus Prime Logic Lock region and design partition feature to specify the areas for high priority internal scrubbing.

1. In the Intel Quartus Prime software, select **Assignments ➤ Logic Lock Regions Window**.
2. In the **Logic Lock Regions Window**, create a region and place it within a design partition.
3. Add your critical design modules, entities, or group of logic to preserve and lock them to the region.
4. In the Intel Quartus Prime software, select **Assignments ➤ Assignment Editor**.
5. In the **Assignment Editor**, assign **Priority SEU Area** to the design partition where you place the Logic Lock region.

Instead of using the **Assignment Editor**, you can also include the following instruction in the project's Quartus Settings File (.qsf):

```
set_instance_assignment -name PRIORITY_SEU_AREA ON -to <partition name>
```

The Intel Quartus Prime software sets the internal scrubbing schedule of the priority sectors to "as fast as possible". The internal scrubbing schedule for other sectors follows the project's **Minimum SEU interval** global assignment.

**Related Information**


Provides more information about creating and using Logic Lock regions.

3.4. Performing Hierarchy Tagging

You define the FPGA regions for tagging by assigning an ASD Region to the location. You can specify an ASD Region value for any portion of your design hierarchy using the Design Partitions Window.

1. In the Intel Quartus Prime software, choose **Assignments ➤ Design Partitions Window**.
2. Right-click anywhere in the header row and turn on ASD Region to display the **ASD Region** column (if it is not already displayed).
3. Enter the logic sensitivity ID value from 0 to 32 for any partition to assign it to a specific ASD Region.

The Logic Sensitivity ID represents the sensitivity tag associated with the partition:

- A sensitivity tag of 1 is the same as no assignment and indicates a basic sensitivity level, which is "region used in design".
- A sensitivity tag of 0 is reserved and indicates unused CRAM bits. You can explicitly set a partition to 0 to indicate that the partition is not critical. This setting excludes the partition from sensitivity mapping.
Note: You can use the same sensitivity tag for multiple design partitions.

Figure 5. ASD Region Column in the Design Partitions Window

Compile the design and the Intel Quartus Prime software generates sensitivity data as a standard Intel hex (big endian) .smh file during .sof file generation.

Related Information
- Advanced SEU Detection IP Core on page 10
- Intel Quartus Prime SEU Software Settings on page 18

3.5. Programming Sensitivity Map Header File into Memory

You can program the .smh into any type of memory. The following example procedure generates the programming file for quad serial peripheral interface (SPI) flash memory.

1. Rename the .smh to <file_name>.hex. If required, convert the file to a little-endian .hex file.
3. In the Device family box, select Stratix 10.
4. In the Configuration mode box, select Active Serial x4.

Figure 6. Programming File Generator Window

5. In the Output Files tab:
a. Specify the **Output directory** and **Name** for the output file.
   
   *Note*: The output directory you specify must already exist in the file system.

b. Select **JTAG Indirect Configuration File (.jic)**.

c. Select **Memory Map File (.map)**.

6. In the **Input Files** tab:

   a. Click **Add Raw Data**.

   b. Navigate your file system and select the `.hex` file and click **Open**.

   c. Click to select the file in the list and then click **Properties**.

   d. In the **Input File Properties** window, if you want to use the Intel Quartus Prime Programmer to configure your device, select **On** in the **Bit swap** box and click **OK**.

   *Note*: Turning on **Bitswap** generates big-endian programming files, as required by the Intel Quartus Prime Programmer. If you use a different programming tool, you can keep **Bitswap** on or off, as required by your tool.

7. In the **Configuration Device** tab:

   a. Click **Add Device**.

   b. In the **Configuration Device** window, click to select your configuration device and click **OK**.

   c. Click to select the configuration device in the list and then click **Add Partition**.

   d. In the **Add Partition** windows, select the file in the **Input file** box, select **Start** in the **Address Mode** box, and then click **OK**.

   e. Click **Select**.

   f. In the **Select Devices** window, click **Stratix 10** in the **Device family** list, select your flash loader device in the **Device name** list, and then click **OK**.

8. Click **Generate**.

---

### 3.6. Performing Lookup for Sensitivity Map Header

You must enable the following options in the Intel Quartus Prime software before performing SMH lookup using the Advanced SEU Detection Intel FPGA IP:

- Error detection CRC
- Generate SEU sensitivity map file (`.smh`)

To perform a lookup into the sensitivity map header for Intel Stratix 10 devices, perform the following steps:

1. Read `.smh` file header to obtain generic `.smh` information:
   - **Address** = 0
   - **Word 0** = SMH_signature
   - **Word 1** = (reserved, region_mask_size)
   - **Word 2** = sector_info_base_address

2. Read three 32-bit words of sector information entry for:
a. Sector encoding scheme 32-bit address
b. Sector .smh data 32-bits address
c. 8 bits of sector .smh tag size (can be 1,2,4, or 8 bits)
d. 16 bits of ASD region map size that is the number of ASD region bitmasks used by sector

- Address = sector_info_base_address + (sector_index*3)
- Word 0 = encoding_scheme_address
- Word 1 = sector_data_address
- Word 2 = (reserved, regions_map_size, smh_tag_size)

3. Read the following sector encoding scheme information for error location frame index and bit position within the frame:
   a. Read the first three words of sector encoding scheme header information to obtain the encoding scheme parameters.
      - Address = encoding_scheme_address
      - Word 0 = (reserved, frame_encoding_map_size)
      - Word 1 = frame_info_base_offset
      - Word 2 = frame_encoding_base_offset
   b. Read the 32-bit frame information string for the frame number.
      - Address = encoding_scheme_address + frame_info_base_offset + frame_index
      - Word 0 = (frame_encoding_index, frame_data_offset)
   c. Get 16-bit index into frame sensitivity data for a bit position.
      ```
      int16* frame_encoding_map = encoding_scheme_address + frame_encoding_base_offset + (frame_encoding_map_size * frame_encoding_index)/4;
      int16 tag_index = frame_encoding_map[bit_position];
      ```

4. Read the following data from sector .smh data to establish affected ASD regions:
   a. The smh_tag_size bit length .smh for frame_data_offset and tag_index from 2 on page 21.
      ```
      int8* frame_data = (sector_data_address + 1 + (regions_map_size*region_mask_size+31)/32 + frame_data_offset*smh_tag_size);
      int8 sensitivity_byte = frame_data[tag_index*smh_tag_size/8];
      int8 smh_tag = (sensitivity_byte >> (tag_index*smh_tag_size %8)) & ((0x1<<smh_tag_size)-1);
      ```
   b. A zero SMH tag indicates that the bit error location is not critical for any region; a non-zero tag indicates an index in the region map. To get a region mask for SMH tag:
      ```
      int32* region_masks = sector_data_address+1;
      ```
int32 region_mask_offset = (smh_tag-1)*region_mask_size;
int32 region_mask_word = region_masks[region_mask_offset/32];
int32 region_mask = (region_mask_word >> region_mask_offset %32) & ((0x1<<(region_mask_size)-1);

Related Information
Advanced SEU Detection IP Core on page 10

3.7. Using the Fault Injection Debugger

To enable the fault injection feature, you must first enable it in your design. After the feature is enabled, launch and set up the Fault Injection Debugger tool.

Note: To use the tool, the Fault Injection Debugger license is required.

Related Information
• Intel Quartus Prime Fault Injection Debugger on page 16
• Error Message Queue on page 7
• Has the error type definition of the error message queue been changed in Intel Stratix 10 device since Intel Quartus Prime Pro version 19.3?
• Intel Quartus Prime SEU Software Settings on page 18

3.7.1. Enabling the Fault Injection Debugger

1. Open your design project.
2. From the menu, select Assignments ➤ Device.
3. Click Device and Pin Options.
4. Navigate to Error Detection CRC.
5. Turn on Enable error detection check and Allow SEU fault injection.
6. Click OK.

3.7.2. Launching and Setting Up the Fault Injection Debugger

To use the Fault Injection Debugger, connect the tool to the device via the JTAG interface and configure the JTAG chain.

1. From the main menu in the Intel Quartus Prime software, select Tools ➤ Fault Injection Debugger.
2. In the Programmer Fault Injection Debugger window, click Hardware Setup. The Hardware Setup window displays the programming hardware connected to your computer.
3. Select the programming hardware you want to use.
4. Click Close.
5. In the Programmer Fault Injection Debugger window, click Auto Detect. The command populates the device chain with the programmable devices found in the JTAG chain.
3.7.3. Configuring Your Device and the Fault Injection Debugger

The Fault Injection Debugger uses a Software Object File (.sof).

To specify a .sof:
1. Select the Intel Stratix 10 device you want to configure in the Device chain box.
2. Click Select File.
3. Navigate to the .sof and click OK. The Fault Injection Debugger reads the .sof.
4. Turn on Program/Configure.
5. Click Start.

3.7.4. Constraining Regions for Fault Injection

After loading an SMH file, you can direct the Fault Injection Debugger to operate on only specific ASD regions.

To specify the ASD region(s) in which to inject faults:
1. Right-click the FPGA in the Device chain box, and click Show Device Sensitivity Map.
2. Select the ASD region(s) for fault injection.

Figure 7. Device Sensitivity Map Viewer
3.7.5. Injecting Errors

You can inject errors using the following methods:

- Inject error on random location using options in the Fault Injection Debugger
- Inject error on specific location using the command-line interface

**Related Information**

Intel Quartus Prime Fault Injection Debugger on page 16

3.7.5.1. Injecting Errors to a Random Location

To inject errors to a random location using options in the Fault Injection Debugger, perform the following steps:

1. Turn on the **Inject Fault** option.
2. Choose whether you want to run error injection for a number of iterations or until stopped:
   - If you choose to run until stopped, the Fault Injection Debugger injects errors at the interval specified in the **Tools > Options** dialog box.
   - If you want to run error injection for a specific number of iterations, enter the number.
3. Click **Start**.

The Intel Quartus Prime Messages window shows messages about the errors that are injected. For additional information on the injected faults, click **Read EMR**. The Fault Injection Debugger reads the error message queue and displays the contents in the Messages window.

*Note:* **Read EMR** retrieves the content of error message queue.

3.7.5.2. Injecting Errors to a Specific Location

Use the following command to inject errors to a specific location through the command-line interface:

```
quartus_fid --cable=<cable_num> --index=@<device_num>=<sof_file> --number=<n> --user="@<device_num>=<sector_location> <frame_location> <bit_location>"
```

Example:

```
quartus_fid --cable=1 --index=@2=abc.sof --number=1 --user="@2=0x003c 0x000d 0x0269"
```

Or:

```
quartus_fid -c 1 -i @2=abc.sof -n 1 -u "@2=0x003c 0x000d 0x0269"
```

**Related Information**

Fault Injection Debugger Command-Line Interface on page 30

Provides more information about the command-line interface arguments.
3.7.6. Injecting Double Adjacent Errors

To inject double adjacent errors to specific locations, you can use the command-line interface or the GUI.

- Command-line interface—use the `quartus_fid` command.
- GUI—use the Intel Quartus Prime TCL console with the Fault Injection Debugger tool.

3.7.6.1. Injecting Double Adjacent Errors: Command-Line Interface

In the command-line interface, run the `quartus_fid` command with the following syntax:

```
quartus_fid --cable=<cable_num> --index=@<device_num>=<sof_file> --number=<n> --user="@<device_num>=<sector_location> <frame_location> <bit_location> + <frame_location> <bit_location>"
```

Example 1. Examples of the `quartus_fid` Command

- `quartus_fid --cable=1 --index=@2=abc.sof --number=2 --user="@1=0x19 0x192 0x10 + 0x192 0x11"
- `quartus_fid -c 1 -i @2=abc.sof -n 2 -u "@1=0x19 0x192 0x10 + 0x192 0x11"

3.7.6.2. Injecting Double Adjacent Errors: GUI

Before you begin, set up the Fault Injection Debugger tool and configure your device using a `.sof`.

1. From the Intel Quartus Prime menu, select View ➤ TCL Console.
2. In the TCL console, run the `injection` command with the following syntax:

```
injection -user "@1=<sector_location> <frame_location> <bit_location> + <frame_location> <bit_location>"
```

For example: `injection -user '@1=0x19 0x192 0x010 + 0x192 0x011`
3. In the Fault Injection Debugger window, click Read EMR to read the error message.
4. Advanced SEU Detection Intel FPGA IP References

You can set various parameter settings for the Advanced SEU Detection IP core to customize its behaviors, ports, and signals.

The Intel Quartus Prime software generates your customized Advanced SEU Detection IP core according to the parameter options that you set in the parameter editor.

4.1. Advanced SEU Detection Intel FPGA IP Parameter Settings

Table 6. Advanced SEU Detection IP Core Parameter Settings

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Value</th>
<th>Default Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use on-chip sensitivity processing</td>
<td>• On</td>
<td>• Off</td>
<td>Select to use external memory interface to access sensitivity data and perform SEU location look-up by the IP.</td>
</tr>
<tr>
<td>Largest ASD region ID used</td>
<td>1 to 32</td>
<td>1</td>
<td>Specifies the largest ASD region ID used in the design. This option is available if you turn on Use on-chip sensitivity processing. The maximum number of region IDs classifications you can use in a design is 16(^{(3)}).</td>
</tr>
<tr>
<td>Sensitivity data start address</td>
<td>0x0</td>
<td>0x0</td>
<td>Specifies a constant offset to add to all addresses generated by the external memory interface. This option is available if you turn on Use on-chip sensitivity processing.</td>
</tr>
<tr>
<td>Show raw SEU error message</td>
<td>• On</td>
<td>• Off</td>
<td>Select to show raw SEU error message. This option is available if you turn on Use on-chip sensitivity processing.</td>
</tr>
<tr>
<td>SEU error fifo depth</td>
<td>• 2</td>
<td>• 4</td>
<td>• 8</td>
</tr>
<tr>
<td>Use with Fault Injection Debugger Tool</td>
<td>• On</td>
<td>• Off</td>
<td>Turn on to use the IP with the Fault Injection Debugger tool.</td>
</tr>
</tbody>
</table>

Related Information

Advanced SEU Detection IP Core on page 10

\(^{(3)}\) Number of region ID in-use is limited by the region mask specified in the .smh.
4.2. Advanced SEU Detection IP Core Ports

Figure 8. Advanced SEU Detection IP Core On-Chip Sensitivity Processing Block Diagram

Table 7. Advanced SEU Detection IP Core On-Chip Sensitivity Processing Ports

<table>
<thead>
<tr>
<th>Ports</th>
<th>Width</th>
<th>Direction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>clk</td>
<td>1</td>
<td>Input</td>
<td>User input clock. The maximum frequency is 250 MHz.</td>
</tr>
<tr>
<td>reset</td>
<td>1</td>
<td>Input</td>
<td>Active high, synchronous reset signal. Note: For IP core instantiation guidelines, refer to the related information about the Reset Release Intel FPGA IP.</td>
</tr>
<tr>
<td>sys_error</td>
<td>1</td>
<td>Output</td>
<td>Logic high indicates that there is an error in the system while retrieving the SEU error.</td>
</tr>
<tr>
<td>busy</td>
<td>1</td>
<td>Output</td>
<td>Logic high indicates that the Advanced SEU Detection IP core is busy processing the SEU data. The signal goes low when processing completes with assertion of the critical_error or noncritical_error signal.</td>
</tr>
<tr>
<td>critical_clear</td>
<td>1</td>
<td>Input</td>
<td>Assert high to clear the error report (critical_error, noncritical_error, regions_report, and seu_data) for the last processed SEU data input.</td>
</tr>
<tr>
<td>critical_error</td>
<td>1</td>
<td>Output</td>
<td>Logic high indicates that a .smh lookup determined that the SEU error is in a critical region.</td>
</tr>
<tr>
<td>noncritical_error</td>
<td>1</td>
<td>Output</td>
<td>Logic high indicates that a .smh lookup determined that the SEU error is in a non-critical region.</td>
</tr>
<tr>
<td>seu_data</td>
<td>64</td>
<td>Output</td>
<td>Shows the SEU error message for the last processed SEU data input. The port is available if you turn on Show raw SEU error message. For more information, refer to the related information about the error message queue.</td>
</tr>
<tr>
<td>regions_report</td>
<td>1–32</td>
<td>Output</td>
<td>Indicates the region ID for the error as reported by the .smh lookup. The port width of this signal is set by the Largest ASD region ID used parameter.</td>
</tr>
<tr>
<td>address</td>
<td>32</td>
<td>Output</td>
<td>Avalon memory-mapped interface address bus in unit of Byte addressing.</td>
</tr>
<tr>
<td>read</td>
<td>1</td>
<td>Output</td>
<td>Avalon memory-mapped interface read control signal.</td>
</tr>
<tr>
<td>waitrequest</td>
<td>1</td>
<td>Input</td>
<td>Avalon memory-mapped interface wait request signal.</td>
</tr>
<tr>
<td>readdata</td>
<td>32</td>
<td>Input</td>
<td>Avalon memory-mapped interface data bus.</td>
</tr>
<tr>
<td>readdatavalid</td>
<td>1</td>
<td>Input</td>
<td>Avalon memory-mapped interface data valid signal.</td>
</tr>
</tbody>
</table>
Figure 9. Advanced SEU Detection IP Core Off-Chip Sensitivity Processing Block Diagram

![Advanced SEU Detection IP Core Off-Chip Sensitivity Processing Diagram](diag)

Table 8. Advanced SEU Detection IP Core Off-Chip Sensitivity Processing Ports

<table>
<thead>
<tr>
<th>Ports</th>
<th>Width</th>
<th>Direction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>clk</td>
<td>1</td>
<td>Input</td>
<td>User input clock. The maximum frequency is 250 MHz.</td>
</tr>
<tr>
<td>reset</td>
<td>1</td>
<td>Input</td>
<td>Active high, synchronous reset signal.</td>
</tr>
<tr>
<td>sys_error</td>
<td>1</td>
<td>Output</td>
<td>Logic high indicates that there is an error in the system while retrieving the SEU error.</td>
</tr>
<tr>
<td>data</td>
<td>64</td>
<td>Output</td>
<td>Avalon streaming interface data signal that provides SEU error message from the FIFO entry.</td>
</tr>
<tr>
<td>valid</td>
<td>1</td>
<td>Output</td>
<td>Avalon streaming interface data valid signal that indicates the avst_seu_source_data signal contains valid data.</td>
</tr>
<tr>
<td>ready</td>
<td>1</td>
<td>Input</td>
<td>Avalon streaming interface ready signal.</td>
</tr>
</tbody>
</table>

Related Information
- Advanced SEU Detection IP Core on page 10
- Error Message Queue on page 7
- Has the error type definition of the error message queue been changed in Intel Stratix 10 device since Intel Quartus Prime Pro version 19.3?
  Provides more information about Intel Stratix 10 Reset Release IP.
5. Intel Stratix 10 Fault Injection Debugger References

5.1. Fault Injection Debugger Interface Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hardware Setup</td>
<td>Opens <strong>Hardware Setup</strong> window</td>
</tr>
<tr>
<td>Start</td>
<td>Start program or configure the device.</td>
</tr>
<tr>
<td>Auto Detect</td>
<td>Scan the JTAG chain of the specified hardware and display the device chain in graphical way.</td>
</tr>
<tr>
<td>Select File</td>
<td>Select .sof file</td>
</tr>
<tr>
<td>Program/Configure</td>
<td>Call Programmer backend engine to program or configure the device.</td>
</tr>
<tr>
<td>Inject Fault</td>
<td>Inject fault (random location only)</td>
</tr>
<tr>
<td>Run For</td>
<td>Sets the number of fault injection iterations before the tool stop injecting errors.</td>
</tr>
<tr>
<td>Run until stopped</td>
<td>Tool keeps injecting faults until you click Stop.</td>
</tr>
<tr>
<td>Start</td>
<td>Start fault injection</td>
</tr>
<tr>
<td>Stop</td>
<td>Stop fault injection</td>
</tr>
<tr>
<td>Read EMR</td>
<td>Reads the error message queue</td>
</tr>
</tbody>
</table>

**Related Information**

Intel Quartus Prime Fault Injection Debugger on page 16

5.2. Fault Injection Debugger Command-Line Interface

You can run the Fault Injection Debugger at the command line with the `quartus_fid` executable, which is useful if you want to perform fault injection from a script.

**Table 9. Fault Injection Debugger Command-Line Interface Arguments for Intel Stratix 10 Devices**

<table>
<thead>
<tr>
<th>Short Argument</th>
<th>Long Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>l</td>
<td>list</td>
<td>Display all installed hardware.</td>
</tr>
<tr>
<td>c</td>
<td>cable</td>
<td>To select the cable number.</td>
</tr>
<tr>
<td>a</td>
<td>auto</td>
<td>For auto detect operation. You must select only one cable for this operation.</td>
</tr>
<tr>
<td>i</td>
<td>index</td>
<td>Option to specify the active device or devices to inject soft error.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Full syntax: @&lt;device_position&gt;=&lt;file_path&gt;#&lt;operation&gt;</td>
</tr>
</tbody>
</table>

continued...
**Short Argument** | **Long Argument** | **Description**
--- | --- | ---
|  |  | where:
  - device_position is the position of active device counting from nearest to TDI
  - file_path is the active device's programming file
  - operation is the operation you want to perform (P—Program/Configure, I—Inject fault)

Command example:
```
quartus_fid --cable=1 --index=2=abc.sof#P
```

| n | number | Option to specify the number of soft errors to inject. If you do not specify the number of errors, the Fault Injection Debugger executes the interactive mode. In the interactive mode, you can select to inject fault, read EMR, scrub errors, or quit. Note: You can inject up to four soft errors. Command examples:
  - To inject two errors:
    ```
quartus_fid --cable=1 --index=2=abd.sof --number=2
```
  - To execute interactive mode:
    ```
quartus_fid --cable=1 --index=2=abc.sof
```
  - To inject double adjacent errors in interactive mode:
    ```
quartus_fid --cable=1 --index=2=abc.sof
F <sector_location> <frame_location> <bit_location> + <frame_location> <bit_location>
```

| s | smh | Option to specify the sensitivity map header file. Full syntax:
  ```
@<device_position>=<file_path>#$<region_info>
  ```
where:
  - device_position is the position of active device counting from nearest to TDI
  - file_path is the active device's .smh file
  - region_info is the intended .smh region information with the following format:
    ```
    <targeted_regions><allow non critical><allow overlapping>
    ```
    - targeted_regions = binary representation of the regions
      ```
      • Region 1 = 1
      • Region 2 = 2
      • Region 3 = 4
      • Region 4 = 8
      • Region 1 and 2 = 3 (from 1 + 2)
      • Region 1 and 3 = 5 (from 1 + 4)
      ```
    - allow non critical—N = allow injecting to non-critical bit
    - allow overlapping—O = allow injecting to bits with overlapping regions

Examples:
  - To inject region 1 or 3 only: region_info = 5
  - To inject region 2 or non-critical bit, region_info = 4N
  - To inject any bit that has region 4 or non-critical bit, the region_info = 8NO

---

(4) If you do not specify any operation, the default operation is "inject fault".
## Command examples:

- **To inject one error in region 2:**
  ```bash
  quartus_fid --cable=1 --index=02=abc.sof --number=1 --smh=02=abc.smh#2
  ```

- **To execute interactive mode in a specific region:**
  ```bash
  quartus_fid --cable=1 --index=02=abc.sof --smh=02=abc.smh#2
  ```

### user

**Option to specify the user specific fault.**

**Full syntax:**

`@<device_position>=<sector-frame-bit-pair ?>#1 <sector-frame-bit-pair ?>#2 ... <sector-frame-bit?>#n`

**where:**

- `device_position` is the position of active device counting from nearest to TDI
- `sector-frame-bit-pair` is the frame bit and sector location where the error is injected.

**Command example:**

```bash
quartus_fid --cable=1 --index=02=abc.sof --number=1 --user="@2=0x003c 0x000d 0x0269"
```

### time

**Option to specify the interval time between injections.**

### Related Information

Intel Quartus Prime Fault Injection Debugger on page 16

---

(5) The maximum pair of frame-bit depends on argument n.
## 6. Intel Stratix 10 SEU Mitigation User Guide Archives

If the table does not list a software version, the user guide for the previous software version applies.

<table>
<thead>
<tr>
<th>Intel Quartus Prime Version</th>
<th>User Guide</th>
</tr>
</thead>
<tbody>
<tr>
<td>21.1</td>
<td>Intel Stratix 10 SEU Mitigation User Guide</td>
</tr>
<tr>
<td>20.2</td>
<td>Intel Stratix 10 SEU Mitigation User Guide</td>
</tr>
<tr>
<td>19.3</td>
<td>Intel Stratix 10 SEU Mitigation User Guide</td>
</tr>
<tr>
<td>19.2</td>
<td>Intel Stratix 10 SEU Mitigation User Guide</td>
</tr>
<tr>
<td>19.1</td>
<td>Intel Stratix 10 SEU Mitigation User Guide</td>
</tr>
<tr>
<td>18.1</td>
<td>Intel Stratix 10 SEU Mitigation User Guide</td>
</tr>
<tr>
<td>18.0</td>
<td>Intel Stratix 10 SEU Mitigation User Guide</td>
</tr>
</tbody>
</table>
# Document Revision History for the Intel Stratix 10 SEU Mitigation User Guide

<table>
<thead>
<tr>
<th>Document Version</th>
<th>Intel Quartus Prime Version</th>
<th>Changes</th>
</tr>
</thead>
</table>
| 2021.07.05       | 21.2                       | • Added support for injecting double adjacent errors.  
|                  |                            | • Updated the Advanced SEU Detection IP core ports.  |
| 2021.04.15       | 21.1                       | • Updated the error message queue table to update the description for bit 31:29.  
|                  |                            | • Updated the topic about internal scrubbing to improve clarity and add information about uncorrectable errors.  
|                  |                            | • Updated the topic about priority scrubbing to improve clarity.  
|                  |                            | • Updated the `quartus_fid` command and arguments, and added examples.  
|                  |                            | • Updated the Intel Quartus Prime SEU software settings to add the option to turn on **Allow SEU fault injection**.  
|                  |                            | • Updated the off-chip sensitivity processing ports information of the Advanced SEU Detection IP.  
|                  |                            | • Updated the following signal names:  
|                  |                            | — `seu_avst_data` to `avst_seu_source_data`  
|                  |                            | — `seu_avst_valid` to `avst_seu_source_valid`  
|                  |                            | — `seu_avst_ready` to `avst_seu_source_ready`  
|                  |                            | • Updated "Avalon-MM" and "Avalon-ST" to "Avalon memory-mapped interface" and "Avalon streaming interface".  |
| 2020.09.24       | 20.2                       | Updated the procedures for using the Fault Injection Debugger to improve clarity.  |
| 2019.10.16       | 19.3                       | • Updated the procedures in the **Programming Sensitivity Map Header File into Memory** topic from using the **Convert Programming File** tool to using the **Programming File Generator** tool.  
|                  |                            | • Updated the topic about failure rates to correct the number of years of one billion hours.  
|                  |                            | • Removed the footnote about future Intel Quartus Prime support for double adjacent errors and multiple bit errors. The features are now supported in the Intel Quartus Prime software.  
|                  |                            | • Updated the footnote to the "Error location" entry in the **Error Message Queue Description** table to clarify the different return values for single bit error with or without internal scrubbing.  
|                  |                            | • Updated the Advanced SEU Detection Intel FPGA IP to version 19.1.0:  
|                  |                            | — Changed the IP name from "Advanced SEU Detection Intel Stratix 10 FPGA IP" to "Advanced SEU Detection Intel FPGA IP".  
|                  |                            | — Updated the description for the **SEU error fifo depth** parameter to remove the condition that it is available only if you turn on **Use on-chip processing**. The **SEU error fifo depth** parameter is available for on-chip and off-chip sensitivity processing.  |
| 2019.07.01       | 19.2                       | Updated the table listing the error message queue description to clarify that the bit position of the sector address and error location fields on the `seu_avst_data` signal.  |

*Other names and brands may be claimed as the property of others.*
<table>
<thead>
<tr>
<th>Document Version</th>
<th>Intel Quartus Prime Version</th>
<th>Changes</th>
</tr>
</thead>
<tbody>
<tr>
<td>2019.05.17</td>
<td>19.1</td>
<td>Added a note to the reset port regarding IP core instantiation guidelines in the tables about Advanced SEU Detection IP core on-chip and off-chip sensitivity processing ports.</td>
</tr>
<tr>
<td>2018.10.10</td>
<td>18.1</td>
<td>• Removed correction support for double adjacent errors.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Updated the topic about the error message queue to remove mention of error count in the error message queue.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Added a topic about internal scrubbing.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Added a topic about priority scrubbing.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Updated the topic about setting the Intel Quartus Prime SEU settings to improve clarity.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Added procedures for enabling priority scrubbing.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Updated the allowed value of the Largest ASD region ID used parameter from &quot;1 to 255&quot; to &quot;1 to 32&quot;.</td>
</tr>
<tr>
<td>2018.08.07</td>
<td>18.0</td>
<td>• Removed correction support for multiple bit errors.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Corrected the signal names in the topic about off-chip lookup sensitivity processing from seu_avst_ready to seu_avst_valid.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Updated IP core name from &quot;Intel FPGA Stratix 10 Advanced SEU Detection IP&quot; to &quot;Advanced SEU Detection Intel FPGA Stratix 10 IP&quot;.</td>
</tr>
<tr>
<td>2018.05.07</td>
<td>18.0</td>
<td>• Added smh argument to the Fault Injection command-line interface command.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Updated the user command in Fault Injection command-line interface description.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Added Failure Rates section.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Added Constraining Regions for Fault Injection section.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Updated argument to inject error on specific location.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Updated the ECC status flag signals for eSRAM blocks in the Memory Blocks Error Correction Code Support topic.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Date</th>
<th>Version</th>
<th>Changes</th>
</tr>
</thead>
<tbody>
<tr>
<td>December 2017</td>
<td>2017.12.29</td>
<td>• Added Fault Injection tool information.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Added the Advanced SEU Detection IP core information.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Updated Implementation Guide to include Fault Injection tool and Advanced SEU Detection IP core implementations.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Restructured Overview chapter.</td>
</tr>
<tr>
<td>December 2016</td>
<td>2016.12.09</td>
<td>• Added SEU_ERROR Pin Settings.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Added Enabling Internal Scrubbing.</td>
</tr>
<tr>
<td>October 2016</td>
<td>2016.10.31</td>
<td>Initial release.</td>
</tr>
</tbody>
</table>