Intel® Stratix® 10 SEU Mitigation User Guide

Updated for Intel® Quartus® Prime Design Suite: **18.0**

UG-S10SEU | 2018.08.07
Latest document on the web: PDF | HTML

Contents

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

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

Area	SEU Mitigation Approach
Error Detection and Correction	You can enable the error detection and correction (EDC) feature for detecting CRAM SEU events and automatic correction of CRAM contents.
Memory block error correction code	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.
SEU Sensitivity processing	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.
Fault injection	You can use fault injection feature to validate the system response to the SEU event by changing the CRAM state to trigger an error.
Hierarchical tagging	A complementary capability to sensitivity processing and fault injection for reporting SEU and constraining injection to specific portions of design logic.
Triple Modular Redundancy (TMR)	You can implement TMR technique on critical logic such as state machines.

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. When you engage ECC, you cannot access two of the parity bits because the ECC status flag replaces them.

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 1 billion hours (or 8,333.33 years). Because SEU events are statistically independent, FIT is additive: if a single FPGA has 5,000 FIT, then 10 FPGAs have 50,000 FIT (or 50K failures in 8,333 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/billion hours, MTTF is:

  1 /(5,000/1Bh) =1 billion/5,000 = 200,000 hours = 22.83 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 JESD 89A 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

Error Type	Detection	Correction
Single bit error	Yes	Yes
Double adjacent errors(1)	Yes	Yes
Multiple bit errors(1)	Detect up to 8 CRAM bits that fit in a rectangular box of 8 CRAM bits (8x1, 4x2, 1x8 or 2x4 errors)	—

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 on error count in the queue, sector address, error type, and the location of the error. You can retrieve the contents of the error message queue using:

- Fault Injection Debugger tool
- Advanced SEU Detection Intel Stratix 10 FPGA IP

(1) Supported by Intel Stratix 10 device, the Intel Quartus Prime support to enable this feature will be available in a future release. For more information, contact your local Intel FPGA representative.

Table 3. Error Message Queue Description

Name	Width	Bit	Description
Sector address	32	31:24	Reserved
		23:16	Address of sector with error
		15:4	Reserved
		3:0	Number of errors detected in the sector-1
Error location(2)	32	31:29	Bit 31:29—Error type: • 01=Single bit • 10=Multi-bit
		28	Correction Status: • 0=Not corrected • 1=Corrected
		27:24	Reserved
		23:12	Bit position within frame
		11:0	Combined of Row and Frame index

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 25
- Using the Fault Injection Debugger on page 21

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 17

2.2. 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.

(2) The error location provides the bit position for single bit errors only. For multiple bit errors, bit [23:0] returns 0.

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.

2.2.1. 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.

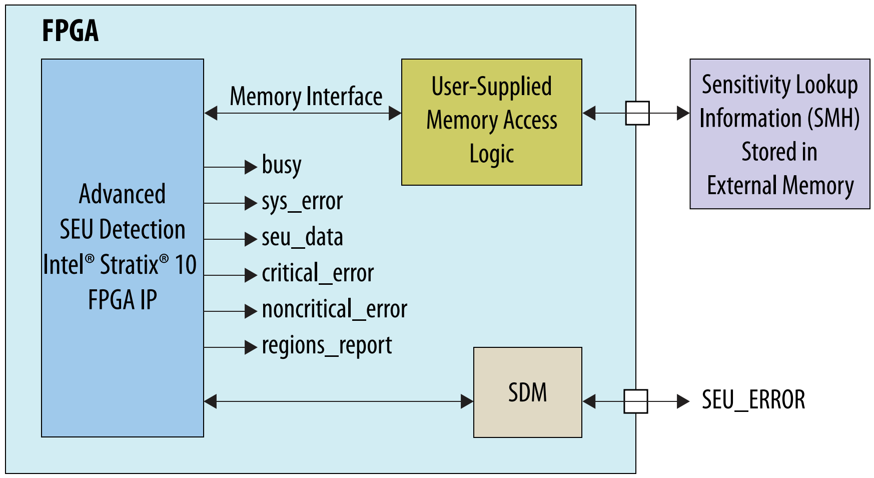
Related Information

- SMH Lookup on page 12
- Performing Lookup for Sensitivity Map Header on page 19
- Performing Hierarchy Tagging on page 18
- Advanced SEU Detection IP Core Parameter Settings on page 24
- Advanced SEU Detection IP Core Ports on page 25
- Intel Quartus Prime SEU Software Settings on page 17

2.2.1.1. 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:

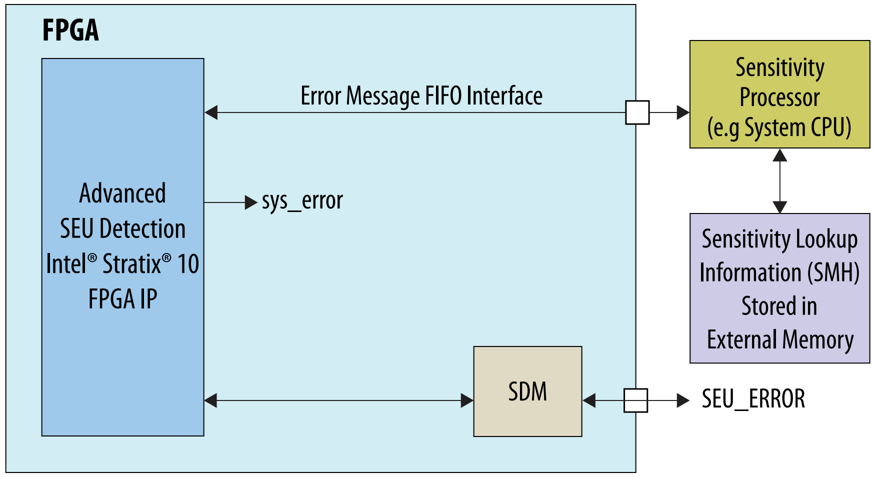
1. The SEU_ERROR is asserted when there is an SEU error.
2. The Advanced SEU Detection IP core retrieves the SEU error message from SDM.

   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)

2.2.1.2. 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`.

Figure 2. System Overview for Off-Chip Lookup Sensitivity Processing with Advanced SEU Detection IP Core



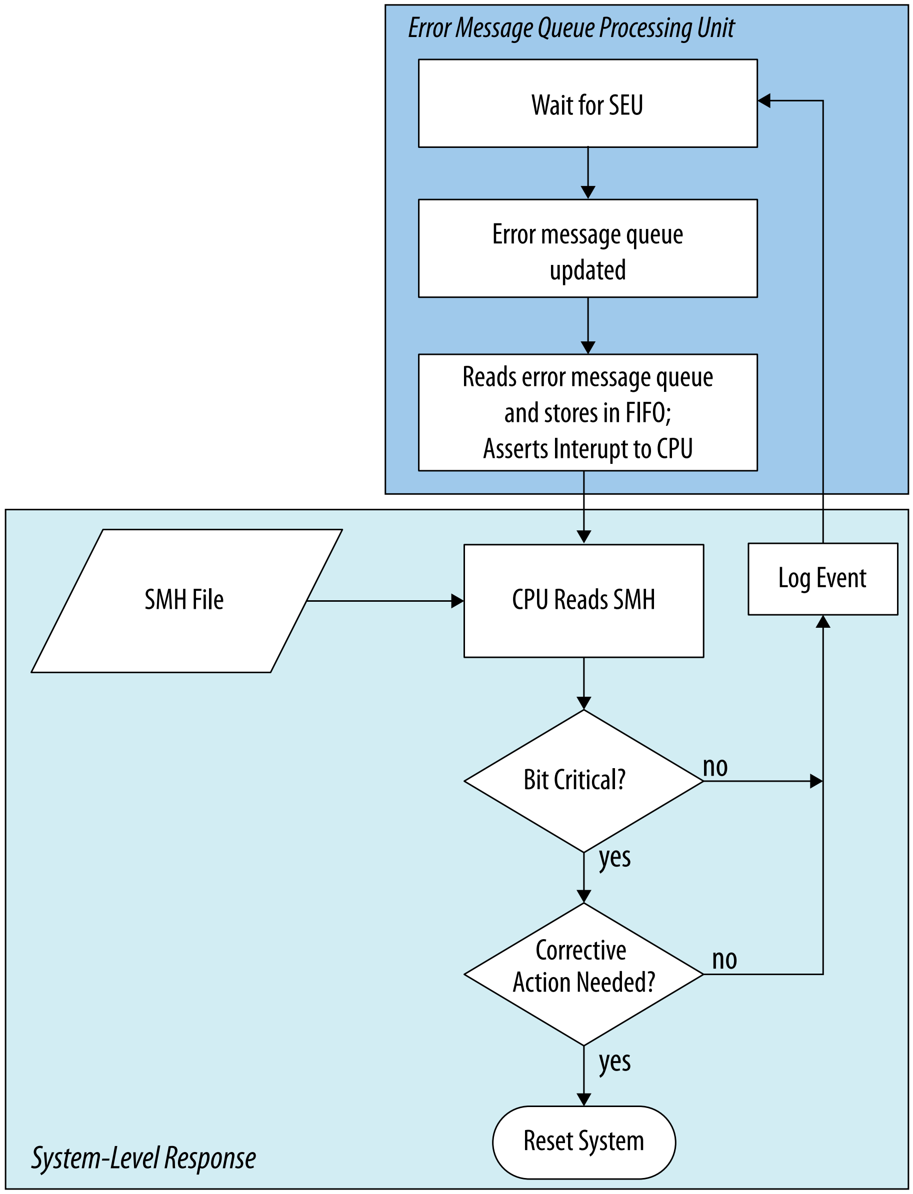
The off-chip lookup sensitivity processing is as follows:

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 `seu_avst_valid` signal to indicate an error message is available.
4. The external sensitivity processor must monitor the `seu_avst_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 Avalon-ST interface and perform lookup against the sensitivity map to determine the criticality of SEU error.

2.2.1.2.1. Off-Chip Lookup Sensitivity Processing Operation Flow

Figure 3. Off-Chip Lookup Sensitivity Processing Operation Flow



2.2.1.3. 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 9

2.2.1.3.1. SMH Revision 4 File Format

Table 4. SMH Revision 4 File Format for Intel Stratix 10 Devices

Block	Sub - block	32-bit Word	Bit	Description
Sensitivity map header	—	0	[31:0]	Identification word for SMH format and its version, 0xXE445341.
		1	[31:8]	Reserved
			[7:0]	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.
		2	[31:0]	Address of the Sector Information block.
Sectors Information block	Sector 0 Information	0	[31:0]	Address of the sector 0 encoding scheme.
		1	[31:0]	Address of the encoded sector 0 sensitivity data.
		2	[23:8]	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.
			[7:0]	The sector 0 SMH tag size in bits, can be 1, 2, 4, 8.
	...	...	...	...
	Sector N Information	N*3 .. N*3+2	...	...
Sectors Encoding block	Sector Encoding 0	0	[31:16]	Identification word 0xEEEE
			[15:0]	Size of a single frame encoding (i.e. bit->tag index) map in bytes.
		1	[31:0]	Address of the frame information (FADD).
		2	[31:0]	Address of the frame encoding map (EADD).
		FADD	[31:20]	Index of encoding map for frame 0
			[19:0]	Sensitivity data offset into sector sensitivity data for frame 0.
		...	...	...
		FAAD+K	[31:20]	Index of encoding map for last frame

continued...

Block	Sub - block	32-bit Word	Bit	Description
			[19:0]	Sensitivity data offset into sector sensitivity data for last frame.
		EADD		Frame encoding map 0. Contains the mapping of 'bit position' in frame to 16-bit 'bit group sensitivity tag index' into frame sensitivity data. For all the phantom bits in a frame 'bit group sensitivity index' is set to `0xFFFF` since no sensitivity data is needed.
		...	...	...
	...	...	...	...
	Sector Encoding M	...		
Sectors Sensitivity Data	Sector 0 Data	0	[31:16]	Sector data identification word (`0xDDDD`)
			[15:0]	Reserved
		1..L		Sector Regions Map: `L = ('ASD region bitmask size' * 'number of ASD region masks for sector'+31)/32`
		L+1		Encoded frames sensitivity data. Data for each frame is located at: `offset = L+1+frame sensitivity data offset * sector SMH tag size`
	...	...	...	...
	Sector N Data	...		

2.3. 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.

2.3.1. 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.

2.4. Evaluating a System's Response to Functional Upsets

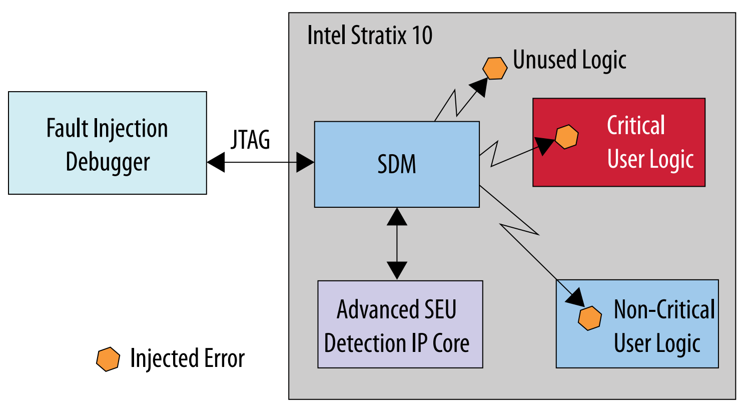
SEUs can strike any memory element, so you must test the system to ensure a comprehensive recovery response. The Intel Quartus Prime software includes the Fault Injection Debugger to aid in SEU recovery. You can use the Fault Injection Debugger graphically with the GUI, or you can use command line assignments.

2.4.1. 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.

Related Information

- Using the Fault Injection Debugger on page 21
- Fault Injection Debugger Interface Parameters on page 27
- Fault Injection Debugger Command-Line Interface on page 27
- Injecting Errors on page 22

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

To enable the error detection CRC, internal scrubbing, generate `.smh`, or set the minimum SEU interval, perform the following steps:

1. On the **Assignments** menu, click **Device**.
2. In the **Device and Pin Options** select the **Error Detection CRC** category.
3. Turn on the following settings:

Enable error detection CRC	Enables error detection feature.	Required for sensitivity processing and fault injection. Required if you want to observe the SEU_ERROR pin behavior.
Minimum SEU Interval	Sets the minimum time between two checks of the same bit. Possible values of 0 - 10000 (milliseconds) . 0 ms means check as frequent as possible.	—
Enable internal scrubbing	Enables error correction feature.	Required for sensitivity processing.
Generate SEU sensitivity map file (.smh)	Generates the `.smh` file	Required for sensitivity processing.

4. Click **OK**.

Related Information

- SEU_ERROR Pin Behavior on page 8
- Advanced SEU Detection IP Core on page 9
- Performing Hierarchy Tagging on page 18
- Using the Fault Injection Debugger on page 21

3.3. Performing Hierarchy Tagging

You define the FPGA regions for testing 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.

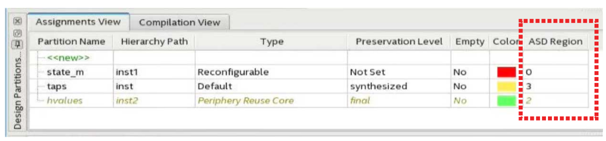
1. In the Intel Quartus Prime software, choose **Assignments ➤ Design Partitions Window**.
2. Right-click anywhere in the header row and turn on ASD Region to display the **ASD Region** column (if it is not already displayed).
3. Enter the logic sensitivity ID value from 0 to 32 for any partition to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

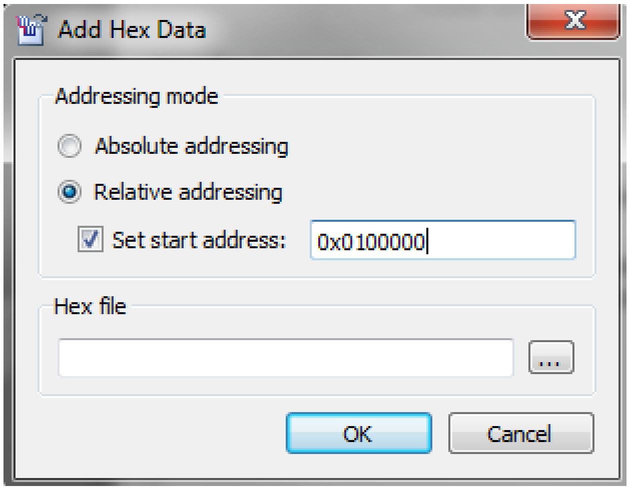
- Advanced SEU Detection IP Core on page 9
- Intel Quartus Prime SEU Software Settings on page 17

3.4. Programming Sensitivity Map Header File into Memory

You can program an `.smh` into any type of memory. For example, to use CFI flash memory, follow these steps:

1. Rename the `.smh` to *<file_name>***.hex**, or convert it to little-endian *<file_name>***.hex** if required.
2. In the Intel Quartus Prime software, click **File > Convert Programming Files**.
3. In the **Convert Programming Files** window under **Output programming file**, select the desired options.
4. To add hex data, follow these steps:
   a. Click **Add Hex Data**.
   b. In the **Add Hex Data** dialog box, turn on **Set start address** and enter a start address.
   c. In the **Hex file box**, click browse to select the `.hex` file, and click **OK**.

Figure 6. Add Hex Data Dialog Box



5. Click **Generate**.

3.5. 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 Stratix 10 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 tag for frame_data_offset and tag_index from 2 on page 19.

```
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 9

3.6. Using the Fault Injection Debugger

To enable the fault injection feature, your design must have the **Enable error detection CRC** option enabled.

Launch the Fault Injection Debugger from **Tools ➤ Fault Injection Debugger** in the Intel Quartus Prime software.

To use the Fault Injection Debugger, you connect from the tool to the device via the JTAG interface. Then, configure the JTAG chain. To configure your JTAG chain, perform the following steps:

1. Click **Hardware Setup**. The tool displays the programming hardware connected to your computer.
2. Select the programming hardware you want to use.
3. Click **Close**.
4. Click **Auto Detect**, which populates the device chain with the programmable devices found in the JTAG chain.

Related Information

• Intel Quartus Prime Fault Injection Debugger on page 15
• Error Message Queue on page 7
• Intel Quartus Prime SEU Software Settings on page 17

3.6.1. 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**.

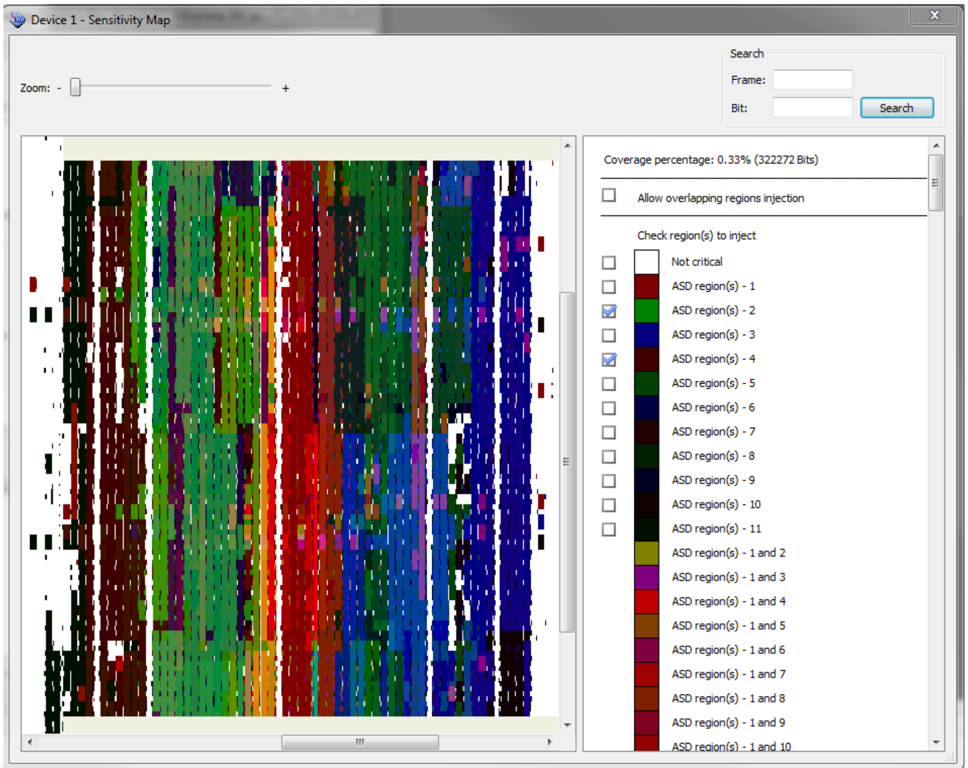
3.6.2. Constraining Regions for Fault Injection

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



3.6.3. Injecting Errors

You can inject error 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 15

3.6.3.1. Injecting Error on Random Location

To inject error on 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.

3.6.3.2. Injecting Error on Specific Location

Use the following argument to inject error on specific location using the command-line interface:

```
quartus_fid –cable=<cable_num> --index= “@<device_num>=<sof_file>” --user
“@<device_num>=<sector_location> <frame_location> <bit_location>”
```

For more information about the command-line interface arguments, refer to Fault Injection Debugger Command-Line Interface on page 27

4. Advanced SEU Detection Intel Stratix 10 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 IP Core Parameter Settings

Table 5. Advanced SEU Detection IP Core Parameter Settings

Parameters	Value	Default Value	Description
Use on-chip sensitivity processing	• **On** • **Off**	**On**	Select to use external memory interface to access sensitivity data and perform SEU location look-up by IP.
Largest ASD region ID used	**1** to **255**	**1**	Specifies the largest ASD region ID used in design. Available if **Use on-chip sensitivity processing** parameter is turned on. Maximum number of region IDs classification can be used in a design is 16(3).
Sensitivity data start address	**0x0**	**0x0**	Specifies a constant offset to be added to all addresses generated by the external memory interface. Available if **Use on-chip sensitivity processing** parameter is turned on.
Show raw SEU error message	• **On** • **Off**	**Off**	Select to show raw SEU error message. Available if **Use on-chip sensitivity processing** parameter is turned on.
SEU error fifo depth	• **2** • **4** • **8** • **16** • **32** • **64**	**4**	Specifies number of SEU errors to be stored. Available if **Use on-chip sensitivity processing** parameter is turned on.
Use with Fault Injection Debugger Tool	• **On** • **Off**	**Off**	Turn on to use with the Fault Injection Debugger Tool.

Related Information

Advanced SEU Detection IP Core on page 9

(3) Number of region ID in-use is limited by the region mask specified in SMH.

4.2. Advanced SEU Detection IP Core Ports

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

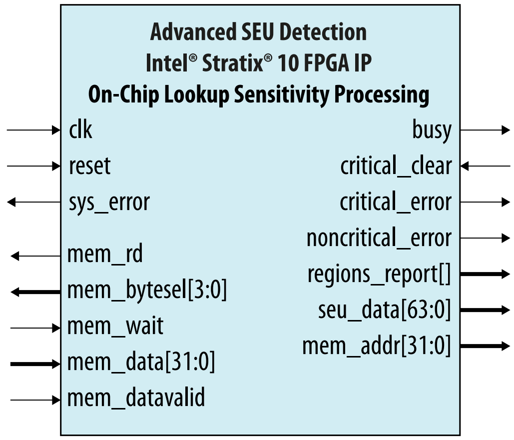


Table 6. Advanced SEU Detection IP Core On-Chip Sensitivity Processing Ports

Ports	Width	Direction	Description
`clk`	1	Input	User input clock. The maximum frequency is 250MHz.
`reset`	1	Input	Active high, synchronous reset signal.
`busy`	1	Output	Logic high indicates that Advanced SEU Detection IP core is busy processing SEU data. The signal goes low when processing completes with either the `critical_error` or `noncritical_error` signal is asserted.
`sys_error`	1	Output	Logic high indicates that there is an error in the system while retrieving SEU error.
`critical_clear`	1	Input	Assert high to clear error report (`critical_error`, `noncritical_error`, `regions_report` and `seu_data`) for the last processed SEU data input.
`critical_error`	1	Output	Logic high indicates that an SMH lookup determined that the SEU error is in a critical region.
`noncritical_error`	1	Output	Logic high indicates that an SMH lookup determined that the SEU error is in a non-critical region.
`regions_report`	1 - 32	Output	Indicates the region ID for the error as reported by the SMH lookup. The port width of this signal is set by **Largest ASD region ID used** parameter.
`seu_data`	64	Output	Shows the SEU error message for the last processed SEU data input. The port is available if **Show raw SEU error message** parameter is turned on. Refer to the *Error Message Queue* for more information about the error messages.
`mem_addr`	32	Output	Avalon-MM address bus in the unit of Byte addressing.
`mem_rd`	1	Output	Avalon-MM read control signal.
`mem_wait`	1	Input	Avalon-MM waitrequest signal.
`mem_data`	32	Input	Avalon-MM data bus.
`mem_datavalid`	1	Input	Avalon-MM data valid signal.

Figure 9. Advanced SEU Detection IP Core Off-Chip Sensitivity Processing Block Diagram

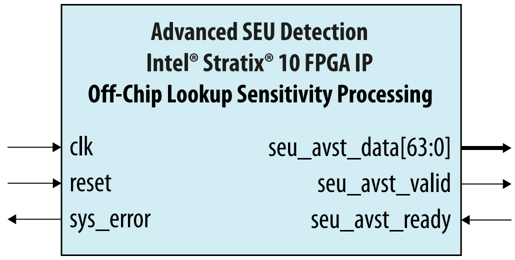


Table 7. Advanced SEU Detection IP Core Off-Chip Sensitivity Processing Ports

Ports	Width	Direction	Description
`clk`	1	Input	User input clock. The maximum frequency is 250MHz.
`reset`	1	Input	Active high, synchronous reset signal.
`sys_error`	1	Output	Logic high indicates that there is an error in the system while retrieving SEU error.
`seu_avst_data`	64	Output	Avalon-ST data signal that provides SEU error message from FIFO entry.
`seu_avst_valid`	1	Output	Avalon-ST data valid signal that indicates the `seu_avst_data` signal contains valid data.
`seu_avst_ready`	1	Input	Avalon-ST ready signal.

Related Information

- Advanced SEU Detection IP Core on page 9
- Error Message Queue on page 7

5. Intel Stratix 10 Fault Injection Debugger References

5.1. Fault Injection Debugger Interface Parameters

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

Related Information

Intel Quartus Prime Fault Injection Debugger on page 15

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 8. Fault Injection Debugger Command-Line Interface Arguments for Intel Stratix 10 Devices

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

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(4) — P—Program/Configure — I—Inject fault
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
u	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.(5)
n	number	Option to specify the number of soft error to inject.
t	time	Option to specify the interval time between injections.

Related Information

Intel Quartus Prime Fault Injection Debugger on page 15

(4) If you do not specify any operation, inject fault is the default operation

(5) The maximum pair of frame-bit depends on argument n

6. Document Revision History for Intel Stratix 10 SEU Mitigation User Guide

Document Version	Intel Quartus Prime Version	Changes
2018.08.07	18.0	• Removed correction support for multiple bit errors. • Corrected the signal names in the topic about off-chip lookup sensitivity processing from `seu_avst_ready` to `seu_avst_valid`. • Updated IP core name from "Intel FPGA Stratix 10 Advanced SEU Detection IP" to "Advanced SEU Detection Intel FPGA Stratix 10 IP".
2018.05.07	18.0	• Added `smh` argument to the Fault Injection command-line interface command. • Updated the `user` command in Fault Injection command-line interface description. • Added *Failure Rates* section. • Added *Constraining Regions for Fault Injection* section. • Updated argument to inject error on specific location. • Updated the ECC status flag signals for eSRAM blocks in the *Memory Blocks Error Correction Code Support* topic.

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