The Quartus II software offers several features to enable the detection and correction of single event upsets (SEUs), or soft errors, as well as to characterize the effects of SEU on your designs.

Understanding SEU

SEU can affect any semiconductor device.

SEUs are rare, unintended changes in the state of internal memory elements, caused by cosmic radiation effects. The change in state results in a soft error, so the affected device can be reset to its original value and there is no permanent damage to the device itself. Because of the unintended memory state, the device may operate erroneously until this upset is fixed.

The Soft Error Rate (SER) is expressed as Failure-in-Time (FIT) units, defined as one soft error occurrence every billion hours of operation. Often SEU mitigation is not required because of the low chance of occurrence. However, for highly complex systems, such as with multiple high-density components, 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, and use the available techniques for detecting and recovering from these types of errors. If your system is requiring high reliability and availability, consider the implications of soft errors, and use the techniques in this document to detect and recover from these types of errors.

FPGAs use memory both in user logic (bulk memory and registers) and in Configuration Random Access Memory (CRAM). CRAM configures the FPGA; this is the memory loaded with the contents of a .sof file by the Quartus II Programmer. 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, the affect can be severe if it affects critical logic internal signal routing (such as a lookup table bit).

Related Information

Introduction to Single Event Upsets

Mitigating SEU Effects in Embedded User RAM

You can mitigate SEU effects for internal memories by using Error Correcting Codes (ECC) for internal SRAM blocks. This method is effective enough so that the FIT rate of ECC-protected memories is almost zero.
ECC use a Cyclic Redundancy Check (CRC) code for a given data word. These CRC bits provide redundancy on the data word which can detect the location of single-bit and double-bit flips in the data word. Since the location of the bit flips is known, the CRC can locate and correct the error. The CRC code can also account for bit flips in the CRC code itself. Altera FPGA support both Single Error Correction Double Error Detection (SECDED) and Double Error Correction Triple Error Detection (DECTED), depending on the device family.

Some Altera device memory blocks offer built-in CRC circuitry hardened in silicon. This is available in the M20K memory in Stratix V, and in the M144K block in Stratix IV, and Arria II devices. (Other device families can implement CRC functions using Altera IP cores). The CRC circuitry generates an EDCRC code at the data storage input of the RAM, and checks the CRC code at the output of the RAM. If an SEU affects any stored bits in the internal memory, the CRC automatically corrects the error when it is read from the memory. The ECC-enabled memory can report the occurrence of a single-bit flip, or adjacent double-bit and adjacent triple-bit flips, and will correct single- and double-bit flips. Adjacent triple-bit corruptions are detected and reported using a status bit, but not corrected.

**Configuring the ECCRAM**

You must configure the ECCRAM as a 2-port RAM (with independent read and write addresses). Use of these features does not reduce the amount of available logic.

While the CRC checking function results in some additional output delay, the hard ECC has a much higher $f_{\text{MAX}}$ compared with an equivalent soft ECC implemented in general logic. Additionally, the hard IP can be pipelined in the M20K block by configuring the ECCRAM to use an output register at the corrected data output port. This increases performance while adding latency.

For devices without dedicated circuitry, you can implement the ECC by instantiating the ECC generation and checking functions as the IP core ALTECC.

**Figure 15-1: Memory Storage**

![Memory Storage Diagram](image)

**Mitigating SEU Effects in Configuration RAM**

Use EDCRC to detect and correct soft errors in CRAM. These EDCRC blocks are similar to those that protect internal user memory.

CRAM is organized into frames. The size of the frame and the number of frames is device specific. CRAM frames are continually checked for errors by loading each frame into a data register. The EDCRC block checks the frame for errors. Soft errors found trigger the assertion of a CRC_ERROR pin on the device. Monitor this pin in your system. Take appropriate actions when this pin is asserted, indicating a soft error was detected in the configuration RAM.
Related Information

Single Event Upsets

Scanning CRAM Frames

To enable the Quartus II software to scan CRAM frames, turn on Enable Error Detection CRC_ERROR pin in the Device and Pin Options dialog box (Assignments > Device > Device and Pin Options).

Figure 15-3: Enable Error Detection CRC_ERROR Pin

To enable the CRC_ERROR pin as an open drain output, turn on Enable open drain on CRC_ERROR pin.

To guarantee the availability of a clock, the EDCRC function operates on an independent clock generated internally on the FPGA itself. To enable EDCRC operation on a divided version of the clock select a value from the Divide error check frequency by value.
Internal Scrubbing

Arria V, Cyclone V (including SoC devices), Stratix V, and later device families support automatic CRAM error correction, without resorting to the original CRAM contents from an external copy of the original SRAM Object File.

Automatic correction is possible because EDCRC calculates and stores redundancy fields along with the configuration bits. This automatic correction is known as scrubbing.

To enable internal scrubbing, turn on Enable internal scrubbing option in the Device and Pin Options dialog box.

If the Quartus II software finds a CRC error in a CRAM frame, the frame is reconstructed from the error correcting code calculated for that frame, and then the corrected frame is re-written into the CRAM.

Note: If you enable internal scrubbing, you must still plan a recovery sequence. Although scrubbing can restore the CRAM array to intended configuration, latency occurs between the soft error detection and correction. Because of the large number of configuration bits to be scanned, this latency may be up to 100 milliseconds for large devices. Therefore, the FPGA may operate with errors during that period.

Related Information
Error Detection CRC Page

Understanding SEU Sensitivity

Reconfiguring a running FPGA typically 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. Also, the surrounding logic may need to account for this unexpected operation.

Often an SEU impacts CRAM bits not used by the implemented 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, 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.

You may determine that portions of the implemented design are not critical to 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.
The ratio of SEU strikes versus functional interrupts is the Single Event Functional Interrupt (SEFI) ratio. Minimizing this ratio improves SEU mitigation.

**Related Information**

Understanding Single Event Functional Interrupts in FPGA Designs

### Designating the Sensitivity of your Design Hierarchy

The design hierarchy sensitivity processing depends on the contents of the Sensitivity Map Header File (.smf). This file determines the correct (least disruptive) recovery sequence for any CRAM bit flip. The .smf designates the sensitivity of each portion of the FPGA’s logic design.

To generate the .smf, you must designate the sensitivity of the design from a functional logic view, using the hierarchy tagging procedure.

### Hierarchy Tagging

Hierarchy tagging is the process of classifying the sensitivity of the portions of your design.

The Quartus II software performs hierarchy tagging by creating a design partition, and then assigning the parameter **ASD Region** to that partition. The parameter can assume a value from 0 to 255, so there are 256 different classifications of system responses to the portions of your design. This sensitivity information is encoded into the .smf the running system uses to look-up the sensitivity of an SEU upset, and to perform the appropriate action to that CRAM location.

**Related Information**

Altera Advanced SEU Detection IP Core User Guide
Altera Advanced SEU Detection IP Core

You must instantiate the Altera Advanced SEU Detection IP core to enable SEU detection and correction features.

When the EDCRC function detects an SEU, the Altera Advanced SEU Detection IP core determines the designer-designated sensitivity of that CRAM bit by looking up the sensitivity in the .smf.

When an EDCRC block detects an SEU, a sensitivity processor looks up the sensitivity of the affected CRAM bit in the .smf.

The user determines which version of the IP core to instantiate: on-chip or external. If the Altera Advanced SEU Detection IP core is configured for on-chip sensitivity processing, the IP core performs the lookup with the user-supplied memory interface. If the Altera Advanced SEU Detection IP core is configured for off-chip sensitivity processing, it notifies external logic (typically via a system CPU interrupt request), and provides cached event message register values to the off-chip sensitivity processor. The SMH information is stored in the external sensitivity processor’s memory system.

Related Information
Altera Advanced SEU Detection IP Core User Guide

On-Chip Sensitivity Processor

You can use the Advanced SEU Detection IP core to implement an on-chip sensitivity processor. The IP core interacts with user-supplied external memory access logic to read the Sensitivity Map Header file, stored on external memory.

Once it determines the sensitivity of the affected CRAM bit, the IP core can assert a Critical Error signal so the system provides an appropriate response. If the SEU is not critical, the Critical Error signal may be left un-asserted.

On-chip sensitivity processing is autonomous: the FPGA device determines whether it is affected by an SEU, without the need for external logic. However, this requires part of the FPGA’s logic resources for the external memory interface.

Related Information
Altera Advanced SEU Detection IP Core User Guide

External Sensitivity Processor

You can configure the Advanced SEU Detection IP core for use with an external sensitivity processor. In this case an external CPU, such as the ARM processor in Altera’s SoC devices, receives an interrupt request when the FPGA detects an SEU. The CPU then reads the Error Message Register, and performs the sensitivity lookup by referring to the Sensitivity Map Header file (.smf) stored in the CPU’s memory space.

External sensitivity processing does not require on-board memory dedicated to the SMH storage function. Also, this technique relieves the FPGA of external memory interface requirements, along with the memory storage requirements for the sensitivity map itself. If a CPU is already present in the system, external sensitivity processing may be the more hardware-efficient way to implement sensitivity lookup.

Related Information
Altera Advanced SEU Detection IP Core User Guide
Triple-Module Redundancy

If your system must suffer no downtime due to SEUs, consider Triple Module Redundancy as an SEU mitigation strategy.

Triple-Module-Redundancy (TMR) is an established technique for improving hardware fault tolerance. In TMR, three identical instances of hardware are supplied, along with voting hardware at the output of the hardware. If an SEU affects one of the instances, the voting logic notes the majority in a vote of the separate instances of the module to mask out any malfunctioning module.

The advantage of TMR is that there is no downtime in the case of a single SEU; if a module is found to be in faulty operation, that module can be scrubbed of its error by reprogramming it. The error detection and correction time is many orders of magnitude less than the MTBF due to SEU events. Therefore, you can repair a soft interrupt before another SEU affects another instance in the TMR triple.

The disadvantage of TMR is its extreme cost in hardware resources: it requires three times as much hardware, in addition to voting logic. This hardware cost can be minimized by judiciously implementing TMR only for the most critical part of the design.

There are several automated ways to generate TMR designs by automatically replicating designated functions and synthesizing the required voting logic. Synthesis vendors offering automated TMR synthesis include Synopsys and Mentor Graphics.

Recovering from a Single-Event Upset

After correcting a bit flip in CRAM, the device is in its original configuration with respect to logic and routing. However, the internal state of the FPGA may be illegal.

The state of the device may be invalid because it may have been operating while SEUs corrupted its configuration. The errors from faulty operation may have propagated elsewhere within the FPGA or to the system outside the FPGA.

Forcing the FPGA into a known state is system dependent. Determining the possible outcomes from SEU, and designing a recovery response to SEU should be part of the FPGA and system design process.

Evaluating Your System's Response to Functional Upsets

Because SEUs can randomly strike any memory element, system testing is especially important to ensure a comprehensive recovery response.

The Quartus II software includes the Fault Injection Debugger to aid in SEU recovery response. This feature is available for the Arria V, Cyclone V, and Stratix V device families.

The feature is available from the Quartus II GUI or at the command line. You must instantiate the Altera Fault Injection IP core into your FPGA design to use this feature. The IP core flips a CRAM bit by dynamically reconfiguring the frame containing that CRAM bit, flipping it to its opposite state.

The Fault Injection Debugger allows you to operate the FPGA in your system and inject random CRAM bit flips to test the ability of the FPGA and the system to detect and recover fully from an SEU. You should be
able to observe your FPGA and your system recover from these simulated SEU strikes. You can then refine your FPGA and system recovery sequence by observing these strikes.

If you have recorded an SEU in the device’s Error Message Register, the Fault Injection Debugger also allows you to specify a targeted fault to be injected (rather than inject the fault in a random location). This feature is available only from the command line.

Related Information
Debugging Single Event Upsets Using the Fault Injection Debugger

---

### Document Revision History

<table>
<thead>
<tr>
<th>Date</th>
<th>Version</th>
<th>Changes</th>
</tr>
</thead>
</table>
| June 2014       | 2014.06.30 | • Updated formatting.  
• Added "Mitigating SEU Effects in Embedded User RAM" section.  
• Added "Altera Advanced SEU Detection IP Core" section. |
| November 2012   | 2012.11.01 | Preliminary release.                                                     |