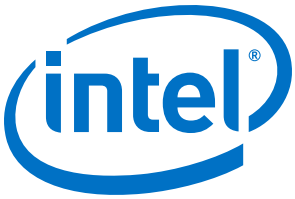


Fault Injection Intel® FPGA IP Core User Guide

Updated for Intel® Quartus® Prime Design Suite: **18.1**


Subscribe


Send Feedback

UG-01173 | 2019.07.09
Latest document on the web: **PDF** | **HTML**

Contents

1. Fault Injection Intel® FPGA IP Core User Guide

The Fault Injection Intel® FPGA IP core injects errors into the configuration RAM (CRAM) of an FPGA device.

This procedure simulates soft errors that can occur during normal operation due to single event upsets (SEUs). SEUs are rare events, and are therefore difficult to test. After you instantiate the Fault Injection IP core into your design and configure your device, you can use the Intel Quartus® Prime Fault Injection Debugger tool to induce intentional errors in the FPGA to test the system's response to these errors.

Related Information

- Single Event Upsets
- AN 737: SEU Detection and Recovery in Intel Arria 10 Devices

1.1. Features

- Allows you to evaluate system response for mitigating single event functional interrupts (SEFI).
- Allows you to perform SEFI characterization in-house, eliminating the need for entire system beam testing. Instead, you can limit the beam testing to failures in time (FIT)/Mb measurement at the device level.
- Scale FIT rates according to the SEFI characterization that is relevant to your design architecture. You can randomly distribute fault injections throughout the entire device, or constrain them to specific functional areas to speed up testing.
- Optimize your design to reduce disruption caused by a single event upsets (SEU).

1.2. Device Support

The Fault Injection IP core supports Intel Arria® 10, Intel Cyclone® 10 GX and Stratix® V family devices. The Cyclone V family supports Fault Injection on devices with the -SC suffix in the ordering code. Contact your local sales representative for ordering information on -SC suffix Cyclone V devices.

1.3. Resource Utilization and Performance

The Intel Quartus Prime software generates the following resource estimate for the Stratix V A7 FPGA. Results for other devices are similar.

Table 1. Fault Injection IP Core FPGA Performance and Resource Utilization

Device	ALMs	Logic Registers		M20K
		Primary	Secondary	
Stratix V A7	3,821	5,179	0	0

1.4. Installing and Licensing Intel FPGA IP Cores

The Intel Quartus Prime software installation includes the Intel FPGA IP library. This library provides many useful IP cores for your production use without the need for an additional license. Some Intel FPGA IP cores require purchase of a separate license for production use. The Intel FPGA IP Evaluation Mode allows you to evaluate these licensed Intel FPGA IP cores in simulation and hardware, before deciding to purchase a full production IP core license. You only need to purchase a full production license for licensed Intel IP cores after you complete hardware testing and are ready to use the IP in production.

The Intel Quartus Prime software installs IP cores in the following locations by default:

Figure 1. IP Core Installation Path

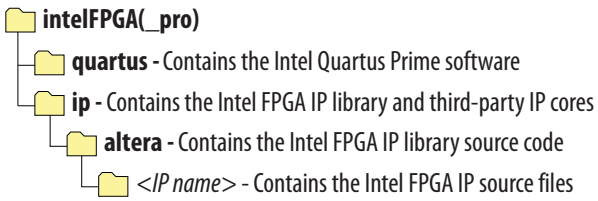


Table 2. IP Core Installation Locations

Location	Software	Platform
`<drive>:\intelFPGA_pro\quartus\ip\altera`	Intel Quartus Prime Pro Edition	Windows*
`<drive>:\intelFPGA\quartus\ip\altera`	Intel Quartus Prime Standard Edition	Windows
`<home directory>:/intelFPGA_pro/quartus/ip/altera`	Intel Quartus Prime Pro Edition	Linux*
`<home directory>:/intelFPGA/quartus/ip/altera`	Intel Quartus Prime Standard Edition	Linux

Note: The Intel Quartus Prime software does not support spaces in the installation path.

1.5. Customizing and Generating IP Cores

You can customize IP cores to support a wide variety of applications. The Intel Quartus Prime IP Catalog and parameter editor allow you to quickly select and configure IP core ports, features, and output files.

1.5.1. IP Catalog and Parameter Editor

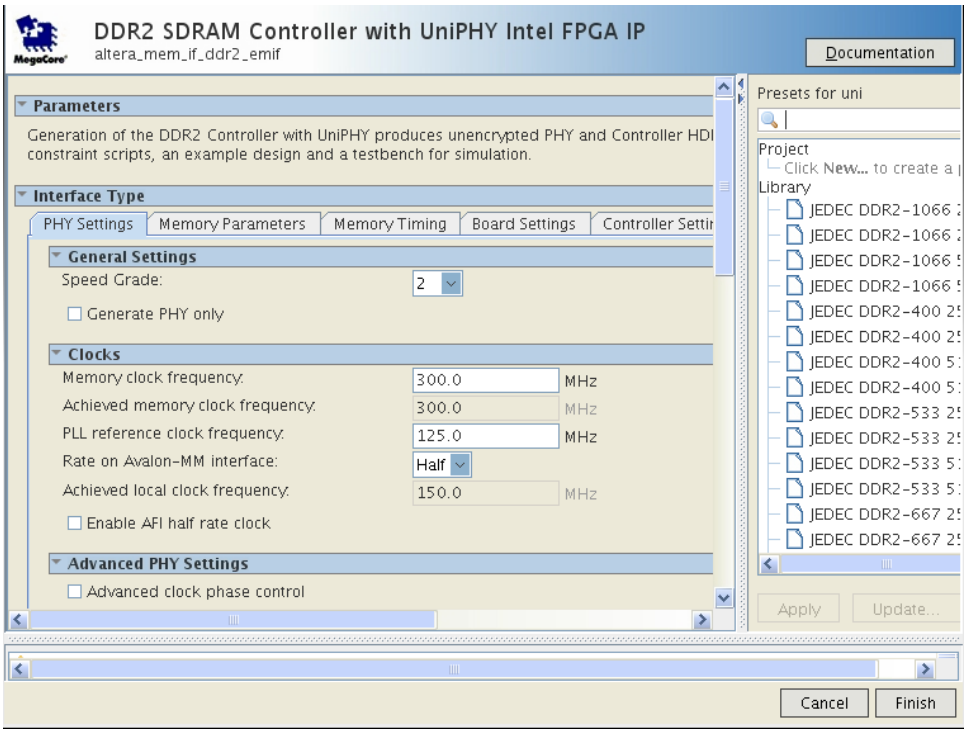
The IP Catalog displays the IP cores available for your project, including Intel FPGA IP and other IP that you add to the IP Catalog search path.. Use the following features of the IP Catalog to locate and customize an IP core:

- Filter IP Catalog to **Show IP for active device family** or **Show IP for all device families**. If you have no project open, select the **Device Family** in IP Catalog.
- Type in the Search field to locate any full or partial IP core name in IP Catalog.
- Right-click an IP core name in IP Catalog to display details about supported devices, to open the IP core's installation folder, and for links to IP documentation.
- Click **Search for Partner IP** to access partner IP information on the web.

The parameter editor prompts you to specify an IP variation name, optional ports, and output file generation options. The parameter editor generates a top-level Intel Quartus Prime IP file (`.ip`) for an IP variation in Intel Quartus Prime Pro Edition projects.

The parameter editor generates a top-level Quartus IP file (`.qip`) for an IP variation in Intel Quartus Prime Standard Edition projects. These files represent the IP variation in the project, and store parameterization information.

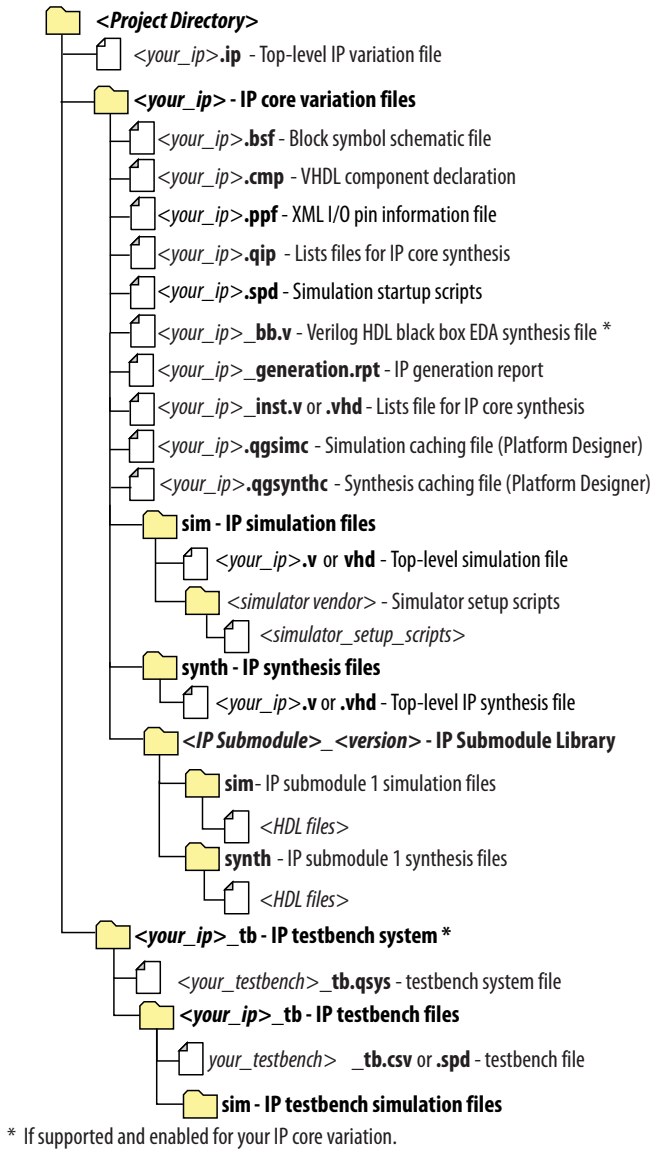
Figure 2. IP Parameter Editor (Intel Quartus Prime Standard Edition)



1.5.2. IP Core Generation Output (Intel Quartus Prime Pro Edition)

The Intel Quartus Prime software generates the following output file structure for individual IP cores that are not part of a Platform Designer system.

Figure 3. Individual IP Core Generation Output (Intel Quartus Prime Pro Edition)



* If supported and enabled for your IP core variation.

Table 3. Output Files of Intel FPGA IP Generation

File Name	Description
<your_ip>.ip	Top-level IP variation file that contains the parameterization of an IP core in your project. If the IP variation is part of a Platform Designer system, the parameter editor also generates a .qsys file.
<your_ip>.cmp	The VHDL Component Declaration (.cmp) file is a text file that contains local generic and port definitions that you use in VHDL design files.
<your_ip>_generation.rpt	IP or Platform Designer generation log file. Displays a summary of the messages during IP generation.

continued...

File Name	Description
<*your_ip*>.qgsimc (Platform Designer systems only)	Simulation caching file that compares the .qsys and .ip files with the current parameterization of the Platform Designer system and IP core. This comparison determines if Platform Designer can skip regeneration of the HDL.
<*your_ip*>.qgsynth (Platform Designer systems only)	Synthesis caching file that compares the .qsys and .ip files with the current parameterization of the Platform Designer system and IP core. This comparison determines if Platform Designer can skip regeneration of the HDL.
<*your_ip*>.qip	Contains all information to integrate and compile the IP component.
<*your_ip*>.csv	Contains information about the upgrade status of the IP component.
<*your_ip*>.bsf	A symbol representation of the IP variation for use in Block Diagram Files (.bdf).
<*your_ip*>.spd	Input file that ip-make-simscript requires to generate simulation scripts. The .spd file contains a list of files you generate for simulation, along with information about memories that you initialize.
<*your_ip*>.ppf	The Pin Planner File (.ppf) stores the port and node assignments for IP components you create for use with the Pin Planner.
<*your_ip*>_bb.v	Use the Verilog blackbox (_bb.v) file as an empty module declaration for use as a blackbox.
<*your_ip*>_inst.v or _inst.vhd	HDL example instantiation template. Copy and paste the contents of this file into your HDL file to instantiate the IP variation.
<*your_ip*>.regmap	If the IP contains register information, the Intel Quartus Prime software generates the .regmap file. The .regmap file describes the register map information of master and slave interfaces. This file complements the .sopcinfo file by providing more detailed register information about the system. This file enables register display views and user customizable statistics in System Console.
<*your_ip*>.svd	Allows HPS System Debug tools to view the register maps of peripherals that connect to HPS within a Platform Designer system. During synthesis, the Intel Quartus Prime software stores the .svd files for slave interface visible to the System Console masters in the .sof file in the debug session. System Console reads this section, which Platform Designer queries for register map information. For system slaves, Platform Designer accesses the registers by name.
<*your_ip*>.v <*your_ip*>.vhd	HDL files that instantiate each submodule or child IP core for synthesis or simulation.
mentor/	Contains a msim_setup.tcl script to set up and run a simulation.
aldec/	Contains a script rivierapro_setup.tcl to setup and run a simulation.
/synopsys/vcs /synopsys/vcsmx	Contains a shell script vcs_setup.sh to set up and run a simulation. Contains a shell script vcsmx_setup.sh and synopsys_sim.setup file to set up and run a simulation.
/cadence	Contains a shell script ncsim_setup.sh and other setup files to set up and run an simulation.
/xcelium	Contains an Parallel simulator shell script xcelium_setup.sh and other setup files to set up and run a simulation.
/submodules	Contains HDL files for the IP core submodule.
<*IP submodule*>/	Platform Designer generates /synth and /sim sub-directories for each IP submodule directory that Platform Designer generates.

1.6. Functional Description

With the Fault Injection IP core, designers can perform SEFI characterization in-house, scale FIT rates according to SEFI characterization, and optimize designs to reduce effect of SEUs.

1.6.1. Single Event Upset Mitigation

Integrated circuits and programmable logic devices such as FPGAs are susceptible to SEUs. SEUs are random, nondestructive events, caused by two major sources: alpha particles and neutrons from cosmic rays. Radiation can cause either the logic register, embedded memory bit, or a configuration RAM (CRAM) bit to flip its state, thus leading to unexpected device operation.

Intel Arria 10, Intel Cyclone 10 GX, Arria V, Cyclone V, Stratix V and newer devices have the following CRAM capabilities:

- Error Detection Cyclical Redundance Checking (EDCRC)
- Automatic correction of an upset CRAM (scrubbing)
- Ability to create an upset CRAM condition (fault injection)

For more information about SEU mitigation in Intel FPGA devices, refer to the *SEU Mitigation* chapter in the respective device handbook.

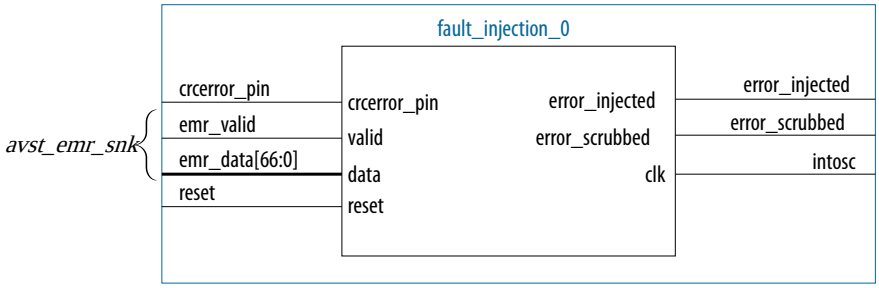
1.6.2. Fault Injection IP Pin Description

The Fault Injection IP core includes the following I/O pins.

Table 4. Fault Injection IP Core I/O Pins

Pin Name	Pin Direction	Pin Description
`crcerror_pin`	input	Input from Error Message Register Unloader Intel FPGA IP (EMR Unloader IP). This signal is asserted when a CRC error has been detected by the device's EDCRC.
`emr_data`	input	Error Message Register (EMR) contents. Refer to the appropriate device handbook for the EMR fields. This input complies with the Avalon Streaming data interface signal.
`emr_valid`	input	Indicates the `emr_data` inputs contain valid data. This is an Avalon Streaming valid interface signal.
`Reset`	input	Module reset input. The reset is fully controlled by the Fault Injection Debugger.
`error_injected`	output	Indicates an error was injected into CRAM as commanded via the JTAG interface. The length of time this signal asserts depends on your settings of the JTAG TCK and control block signals. Typically, the time is around 20 clock cycles of the TCK signal.
`error_scrubbed`	output	Indicates the device scrubbing is complete as commanded via the JTAG interface. The length of time this signal asserts depends on your settings of the JTAG TCK and control block signals. Typically, the time is around 20 clock cycles of the TCK signal.
`intosc`	output	Optional output. The Fault Injection IP uses this clock, for example, to clock the EMR_unloader block.

Figure 4. Fault Injection IP Pin Diagram



1.7. Using the Fault Injection Debugger and Fault Injection IP Core

The Fault Injection Debugger works together with the Fault Injection IP core. First, you instantiate the IP core in your design, compile, and download the resulting configuration file into your device. Then, you run the Fault Injection Debugger from within the Intel Quartus Prime software or from the command line to simulate soft errors.

- The Fault Injection Debugger allows you to operate fault injection experiments interactively or by batch commands, and allows you to specify the logical areas in your design for fault injections.
- The command-line interface is useful for running the debugger via a script.

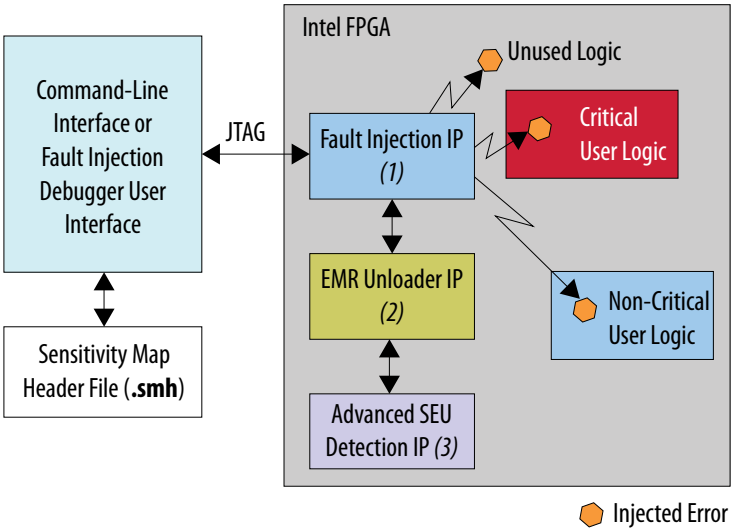
The Fault Injection Debugger communicates with the Fault Injection IP core via the JTAG interface. The Fault Injection IP accepts commands from the JTAG interface and reports status back through the JTAG interface.

Note: The Fault Injection IP core is implemented in soft logic in your device; therefore, you must account for this logic usage in your design. One methodology is to characterize your design's response to SEU in the lab and then omit the IP core from your final deployed design.

You use the Fault Injection IP core with the following IP cores:

- The Error Message Register Unloader IP core, which reads and stores data from the hardened error detection circuitry in Intel FPGA devices.
- (Optional) The Advanced SEU Detection Intel FPGA IP core, which compares single-bit error locations to a sensitivity map during device operation to determine whether a soft error affects it.

Figure 5. Fault Injection Debugger Overview Block Diagram



Notes:
1. The Fault Injection IP flips the bits of the targeted logic.
2. The Fault Injection Debugger and Advanced SEU Detection IP use the same EMR Unloader instance.
3. The Advanced SEU Detection IP core is optional.

Related Information

- About SMH Files on page 13
- About the EMR Unloader IP Core on page 10
- About the Advanced SEU Detection IP Core on page 11

1.7.1. Instantiating the Fault Injection IP Core

The Fault Injection IP core does not require you to set any parameters. To use the IP core, create a new IP instance, include it in your Platform Designer (Standard) system, and connect the signals as appropriate.

Note: You must use the Fault Injection IP core with the EMR Unloader IP core.

The Fault Injection and the EMR Unloader IP cores are available in Platform Designer and the IP Catalog. Optionally, you can instantiate them directly into your RTL design, using Verilog HDL, SystemVerilog, or VHDL.

1.7.1.1. About the EMR Unloader IP Core

The EMR Unloader IP core provides an interface to the EMR, which is updated continuously by the device's EDCRC that checks the device's CRAM bits CRC for soft errors.

Figure 6. Example Platform Designer System Including the Fault Injection IP Core and EMR Unloader IP Core

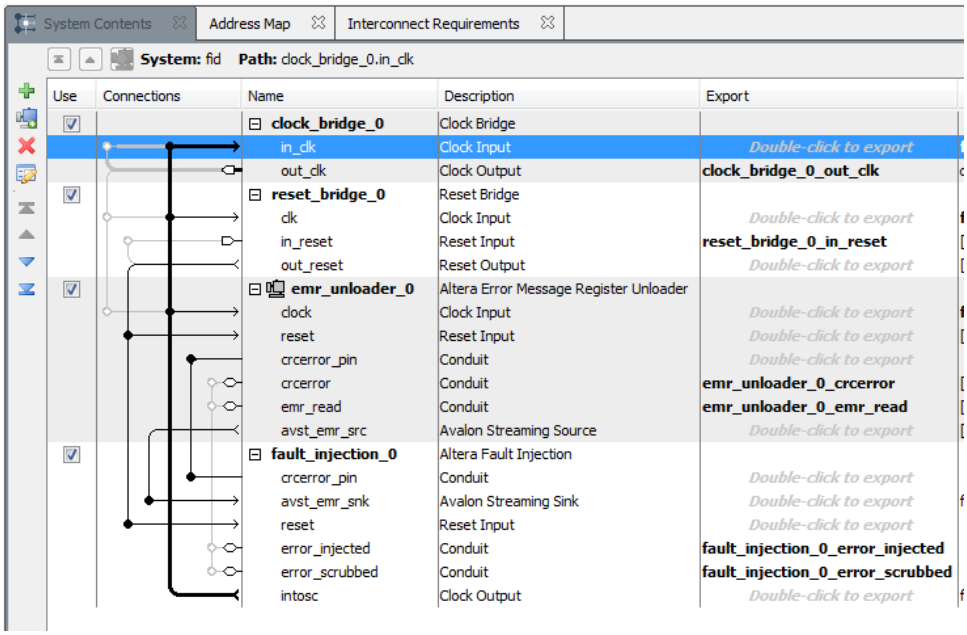
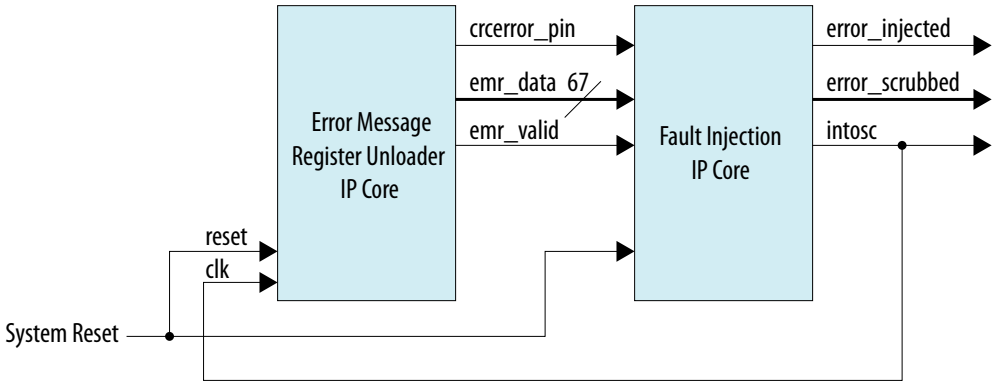


Figure 7. Example Fault Injection IP Core and EMR Unloader IP Core Block Diagram



Related Information

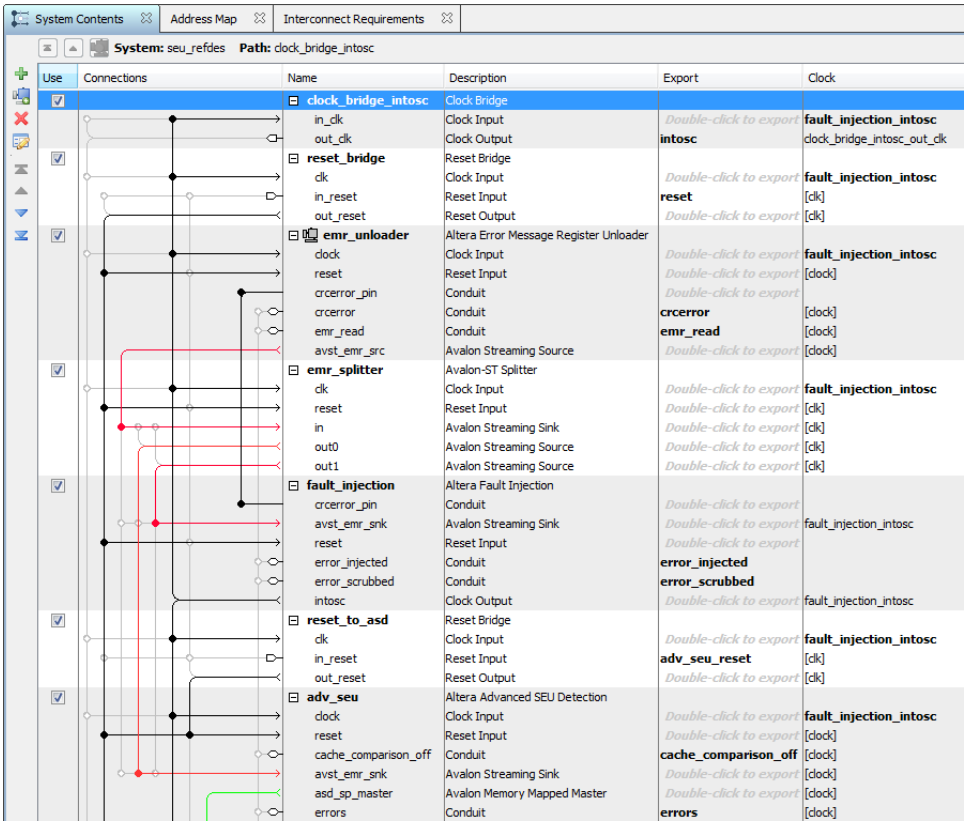
Error Message Register Unloader Intel FPGA IP Core User Guide

1.7.1.2. About the Advanced SEU Detection IP Core

Use the Advanced SEU Detection (ASD) IP core when SEU tolerance is a design concern.

You must use the EMR Unloader IP core with the ASD IP core. Therefore, if you use the ASD IP and the Fault Injection IP in the same design, they must share the EMR Unloader output via an Avalon®-ST splitter component. The following figure shows a Platform Designer system in which an Avalon-ST splitter distributes the EMR contents to the ASD and Fault Injection IP cores.

Figure 8. Using the ASD and Fault Injection IP in the Same Platform Designer System



Related Information

Advanced SEU Detection Intel FPGA IP Core User Guide

1.7.2. Defining Fault Injection Areas

You can define specific regions of the FPGA for fault injection using a Sensitivity Map Header (**.smh**) file.

The SMH file stores the coordinates of the device CRAM bits, their assigned region (ASD Region), and criticality. During the design process you use hierarchy tagging to create the region. Then, during compilation, the Intel Quartus Prime Assembler generates the SMH file. The Fault Injection Debugger limits error injections to specific device regions you define in the SMH file.

1.7.2.1. 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. 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 a value from 0 to 16 for any partition to assign it to a specific ASD Region.
   — ASD region 0 is reserved to unused portions of the device. You can assign a partition to this region to specify it as non-critical..
   — ASD region 1 is the default region. All used portions of the device are assigned to this region unless you explicitly change the ASD Region assignment.

1.7.2.2. About SMH Files

The SMH file contains the following information:

- If you are not using hierarchy tagging (i.e., the design has no explicit ASD Region assignments in the design hierarchy), the SMH file lists every CRAM bit and indicates whether it is sensitive for the design.
- If you have performed hierarchy tagging and changed default ASD Region assignments, the SMH file lists every CRAM bit and it's assigned ASD region.

The Fault Injection Debugger can limit injections to one or more specified regions.

Note: To direct the Assembler to generate an SMH file:

- Choose **Assignments ➤ Device ➤ Device and Pin Options ➤ Error Detection CRC**.
- Turn on the **Generate SEU sensitivity map file (.smh)** option.

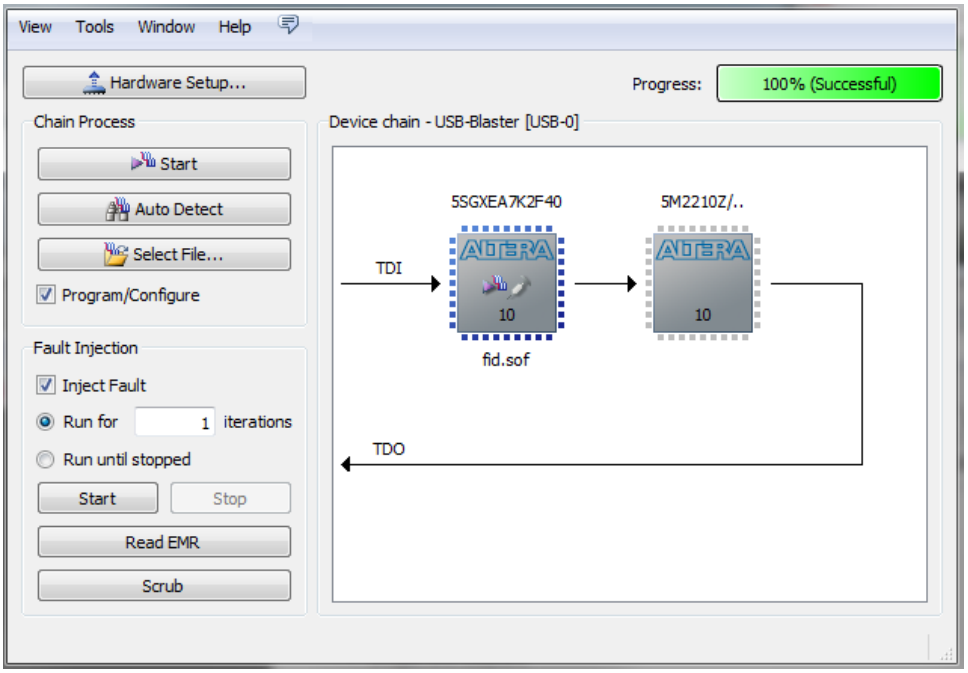
1.7.3. Using the Fault Injection Debugger

To use the Fault Injection Debugger, you connect to your device via the JTAG interface. Then, configure the device and perform fault injection.

To launch the Fault Injection Debugger, choose **Tools ➤ Fault Injection Debugger** in the Intel Quartus Prime software.

Note: Configuring or programming the device is similar to the procedure used for the Programmer or Signal Tap Logic Analyzer.

Figure 9. Fault Injection Debugger



To configure your JTAG chain:

1. Click **Hardware Setup**. The tool displays the programming hardware connected to your computer.
2. Select the programming hardware you wish 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

Targeted Fault Injection Feature on page 21

1.7.3.1. Hardware and Software Requirements

The following hardware and software is required to use the Fault Injection Debugger:

- FEATURE line in your Intel FPGA license that enables the Fault Injection IP core. For more information, contact your local Intel FPGA sales representative.
- Download cable (Intel FPGA Download Cable, Intel FPGA Download Cable II, , or II).
- Intel FPGA development kit or user designed board with a JTAG connection to the device under test.
- (Optional) FEATURE line in your Intel FPGA license that enables the Advanced SEU Detection IP core.

1.7.3.2. Configuring Your Device and the Fault Injection Debugger

The Fault Injection Debugger uses a **.sof** and (optionally) a Sensitivity Map Header (**.smh**) file.

The Software Object File (**.sof**) configures the FPGA. The **.smh** file defines the sensitivity of the CRAM bits in the device. If you do not provide an **.smh** file, the Fault Injection Debugger injects faults randomly throughout the CRAM bits.

To specify a **.sof**:

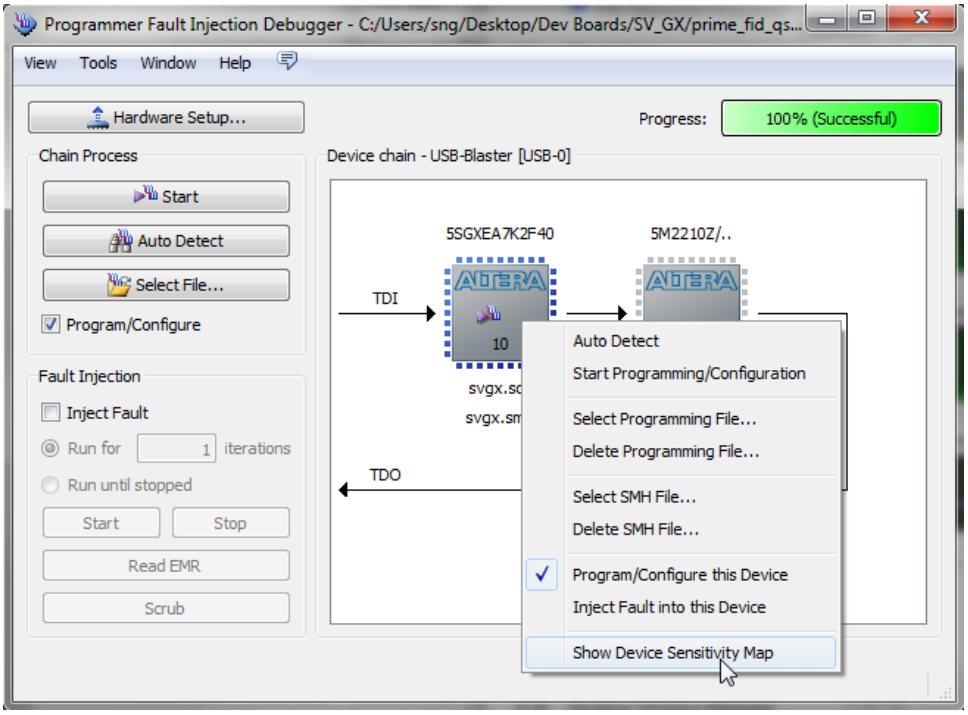
1. Select the FPGA you wish to configure in the **Device chain** box.
2. Click **Select File**.
3. Navigate to the **.sof** and click **OK**. The Fault Injection Debugger reads the **.sof**.
4. (Optional) Select the SMH file.

   If you do not specify an SMH file, the Fault Injection Debugger injects faults randomly across the entire device. If you specify an SMH file, you can restrict injections to the used areas of your device.

   a. Right-click the device in the **Device chain** box and then click **Select SMH File**.
   b. Select your SMH file.
   c. Click **OK**.
5. Turn on **Program/Configure**.
6. Click **Start**.

The Fault Injection Debugger configures the device using the **.sof**.

Figure 10. Context Menu for Selecting the SMH File



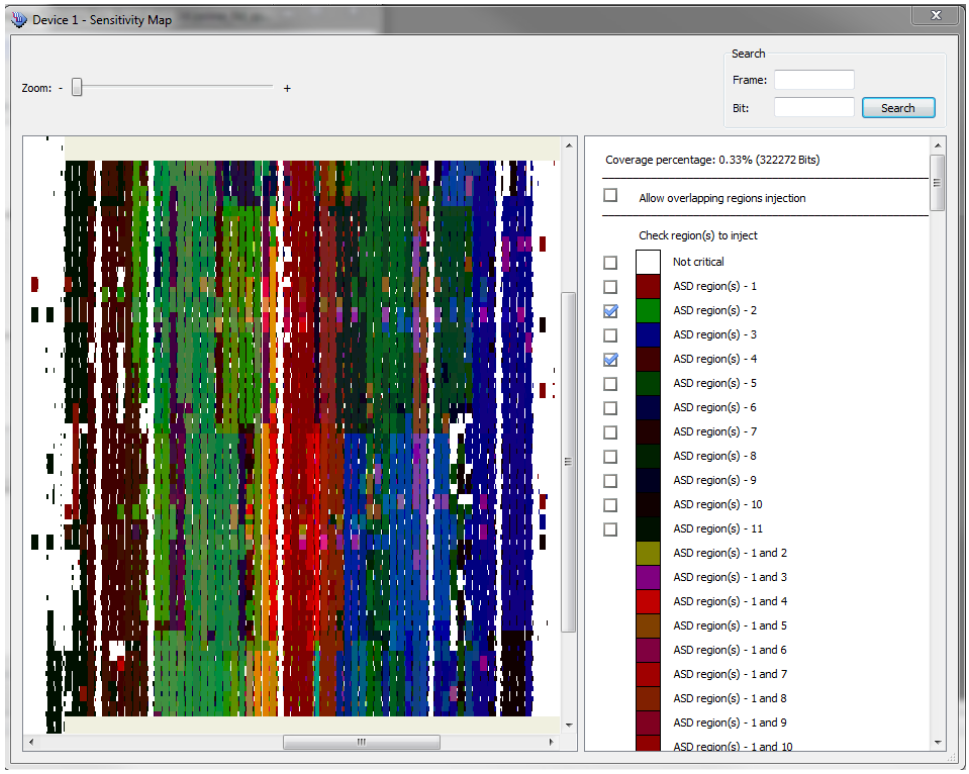
1.7.3.3. Constraining Regions for Fault Injection

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 11. Device Sensitivity Map Viewer



1.7.3.4. Specifying Error Types

You can specify various types of errors for injection.

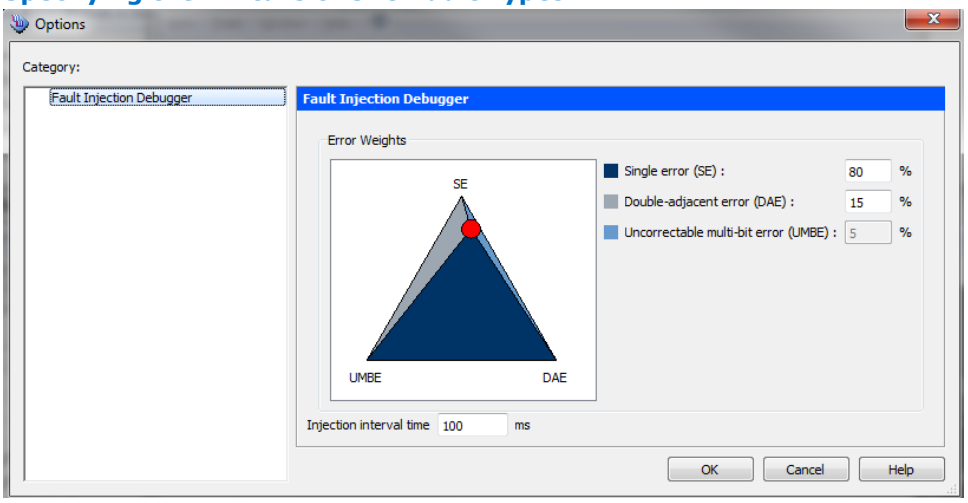
- Single errors (SE)
- Double-adjacent errors (DAE)
- Uncorrectable multi-bit errors (EMBE)

Intel FPGA devices can self-correct single and double-adjacent errors if the scrubbing feature is enabled. Intel FPGA devices cannot correct multi-bit errors. Refer to the chapter on mitigating SEUs for more information about debugging these errors.

You can specify the mixture of faults to inject and the injection time interval. To specify the injection time interval:

1. In the Fault Injection Debugger, choose **Tools ➤ Options**.
2. Drag the red controller to the mix of errors. Alternatively, you can specify the mix numerically.
3. Specify the **Injection interval time**.
4. Click **OK**.

Figure 12. Specifying the Mixture of SEU Fault Types



Related Information

Mitigating Single Event Upset

1.7.3.5. Injecting Errors

You can inject errors in several modes:

- Inject one error on command
- Inject multiple errors on command
- Inject errors until commanded to stop

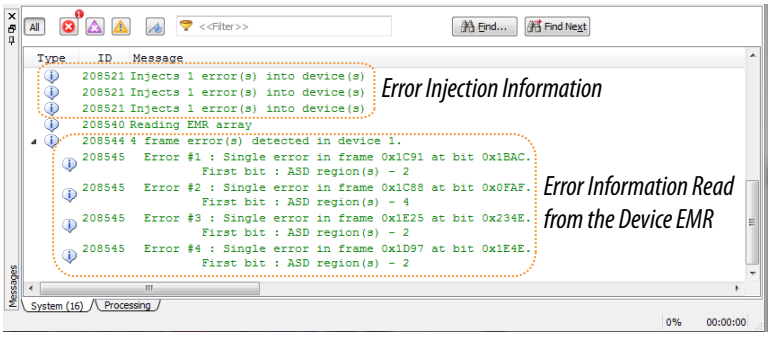
To inject these faults:

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**.

   Note: The Fault Injection Debugger runs for the specified number of iterations or until stopped.

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 device's EMR and displays the contents in the Messages window.

Figure 13. Intel Quartus Prime Error Injection and EMR Content Messages



1.7.3.6. Recording Errors

You can record the location of any injected fault by noting the parameters reported in the Intel Quartus Prime Messages window.

If, for example, an injected fault results in behavior you would like to replay, you can target that location for injection. You perform targeted injection using the Fault Injection Debugger command line interface.

1.7.3.7. Clearing Injected Errors

To restore the normal function of the FPGA, click **Scrub**. When you scrub an error, the device's EDCRC functions are used to correct the errors. The scrub mechanism is similar to that used during device operation.

1.7.4. 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 5. Command line Arguments for Fault Injection

Short Argument	Long Argument	Description
`c`	`cable`	Specify programming hardware or cable. (Required)
`i`	`index`	Specify the active device to inject fault. (Required)
`n`	`number`	Specify the number of errors to inject. The default value is 1. (Optional)
`t`	`time`	Interval time between injections. (Optional)

Note: Use `quartus_fid --help` to view all available options.

The following code provides examples using the Fault Injection Debugger command-line interface.

```
##############################################
#
# Find out which USB cables are available for this instance
# The result shows that one cable is available, named "USB-Blaster"
#
$ quartus_fid --list
   . . .
    Info: Command: quartus_fid --list
```

```
    1) USB-Blaster on sj-sng-z4 [USB-0]
    Info: Intel Quartus Prime 64-Bit Fault Injection Debugger was successful.
     0 errors, 0 warning
##############################################
#
# Find which devices are available on USB-Blaster cable
# The result shows two devices: a Stratix V A7, and a MAX V CPLD.
#
$ quartus_fid --cable USB-Blaster -a
    Info: Command: quartus_fid --cable=USB-Blaster -a
    Info (208809): Using programming cable "USB-Blaster on sj-sng-z4 [USB-0]"
    1) USB-Blaster on sj-sng-z4 [USB-0]
      029030DD   5SGXEA7H(1|2|3)/5SGXEA7K1/..
      020A40DD   5M2210Z/EPM2210
    Info: Intel Quartus Prime 64-Bit Fault Injection Debugger was successful.
    0 errors, 0 warnings

##############################################
#
# Program the Stratix V device
# The --index option specifies operations performed on a connected device.
#   "=svgx.sof" associates a .sof file with the device
#   "#p" means program the device
#
$ quartus_fid --cable USB-Blaster --index "@1=svgx.sof#p"
 . . .
    Info (209016): Configuring device index 1
    Info (209017): Device 1 contains JTAG ID code 0x029030DD
    Info (209007): Configuration succeeded -- 1 device(s) configured
    Info (209011): Successfully performed operation(s)
    Info (208551): Program signature into device 1.
    Info: Intel Quartus Prime 64-Bit Fault Injection Debugger was successful.
    0 errors, 0 warnings

##############################################
#
# Inject a fault into the device.
# The #i operator indicates to inject faults
# -n 3 indicates to inject 3 faults
#
$ quartus_fid --cable USB-Blaster --index "@1=svgx.sof#i" -n 3
    Info: Command: quartus_fid --cable=USB-Blaster --index=@1=svgx.sof#i -n 3
    Info (208809): Using programming cable "USB-Blaster on sj-sng-z4 [USB-0]"
    Info (208521): Injects 3 error(s) into device(s)
    Info: Intel Quartus Prime 64-Bit Fault Injection Debugger was successful.
    0 errors, 0 warnings


##############################################
#
# Interactive Mode.
# Using the #i operation with -n 0 puts the debugger into interactive mode.
# Note that 3 faults were injected in the previous session;
# "E" reads the faults currently in the EMR Unloader IP core.
#
$ quartus_fid --cable USB-Blaster --index "@1=svgx.sof#i" -n 0
    Info: Command: quartus_fid --cable=USB-Blaster --index=@1=svgx.sof#i -n 0
    Info (208809): Using programming cable "USB-Blaster on sj-sng-z4 [USB-0]"
    Enter :
          'F' to inject fault
          'E' to read EMR
          'S' to scrub error(s)
          'Q' to quit
    E
    Info (208540): Reading EMR array
    Info (208544): 3 frame error(s) detected in device 1.
        Info (208545):   Error #1 : Single error in frame 0x1028 at bit 0x21EA.
        Info (10914):   Error #2 : Uncorrectable multi-bit error in frame
0x1116.
        Info (208545):   Error #3 : Single error in frame 0x1848 at bit 0x128C.
    Enter :
```

```
          'F' to inject fault
          'E' to read EMR
          'S' to scrub error(s)
          'Q' to quit
    Q
    Info: Intel Quartus Prime 64-Bit Fault Injection Debugger was successful.
        0 errors, 0 warnings
        Info: Peak virtual memory: 1522 megabytes
        Info: Processing ended: Mon Nov  3 18:50:00 2014
        Info: Elapsed time: 00:00:29
        Info: Total CPU time (on all processors): 00:00:13
```

1.7.4.1. Targeted Fault Injection Feature

The Fault Injection Debugger injects faults into the FPGA randomly. However, the Targeted Fault Injection feature allows you to inject faults into targeted locations in the CRAM. This operation may be useful, for example, if you noted an SEU event and want to test the FPGA or system response to the same event after modifying a recovery strategy.

Note: The Targeted Fault Injection feature is available only from the command line interface.

You can specify that errors are injected from the command line or in prompt mode.

Related Information

AN 539: Test Methodology or Error Detection and Recovery using CRC in Intel FPGA Devices

1.7.4.1.1. Specifying an Error List From the Command Line

The Targeted Fault Injection feature allows you to specify an error list from the command line, as shown in the following example:

```
c:\Users\sng> quartus_fid -c 1 - i "@1= svgx.sof#i " -n 2 -user="@1=
0x2274 0x05EF 0x2264 0x0500"
```

Where:

`c 1` indicates that the fpga is controlled by the first cable on your computer.

`i "@1= svgx.sof#i "` indicates that the first device in the chain is loaded with the object file **svgx.sof** and will be injected with faults.

`n 2` indicates that two faults will be injected.

`user="@1= 0x2274 0x05EF 0x2264 0x0500"` is a user-specified list of faults to be injected. In this example, device 1 has two faults: at frame 0x2274, bit 0x05EF and at frame 0x2264, bit 0x0500.

1.7.4.1.2. Specifying an Error List From Prompt Mode

You can operate the Targeted Fault Injection feature interactively by specifying the number of faults to be 0 (-n 0). The Fault Injection Debugger presents prompt mode commands and their descriptions.

Prompt Mode Command	Description
F	Inject a fault
E	Read the EMR
S	Scrub errors
Q	Quit

In prompt mode, you can issue the F command alone to inject a single fault in a random location in the device. In the following examples using the F command in prompt mode, three errors are injected.

```
F #3 0x12 0x34 0x56 0x78 * 0x9A 0xBC +
```

- Error 1 – Single bit error at frame 0x12, bit 0x34
- Error 2 – Uncorrectable error at frame 0x56, bit 0x78 (an * indicates a multi-bit error)
- Error 3 – Double-adjacent error at frame 0x9A, bit 0xBC (a + indicates a double bit error)

```
F 0x12 0x34 0x56 0x78 *
```

One (default) error is injected:

Error 1 – Single bit error at frame 0x12, bit 0x34. Locations after the first frame/bit location are ignored.

```
F #3 0x12 0x34 0x56 0x78 * 0x9A 0xBC + 0xDE 0x00
```

Three errors are injected:

- Error 1 – Single bit error at frame 0x12, bit 0x34
- Error 2 – Uncorrectable error at frame 0x56, bit 0x78
- Error 3 – Double-adjacent error at frame 0x9A, bit 0xBC
- Locations after the first 3 frame/bit pairs are ignored

1.7.4.1.3. Determining CRAM Bit Locations

When the Fault Injection Debugger detects a CRAM EDCRC error, the Error Message Register (EMR) contains the syndrome, frame number, bit location, and error type (single, double, or multi-bit) of the detected CRAM error.

During system testing, save the EMR contents reported by the Fault Injection Debugger when you detect an EDCRC fault.

Note: With the recorded EMR contents, you can supply the frame and bit numbers to the Fault Injection Debugger to replay the errors noted during system testing, to further design, and characterize a system recovery response to that error.

Related Information

AN 539: Test Methodology or Error Detection and Recovery using CRC in Intel FPGA Devices

1.7.4.2. Advanced Command-Line Options: ASD Regions and Error Type Weighting

You can use the Fault Injection Debugger command-line interface to inject errors into ASD regions and weight the error types.

First, you specify the mix of error types (single bit, double adjacent, and multi-bit uncorrectable) using the `--weight <single errors>.<double adjacent errors>.<multi-bit errors>` option. For example, for a mix of 50% single errors, 30% double adjacent errors, and 20% multi-bit uncorrectable errors, use the option `--weight=50.30.20`. Then, to target an ASD region, use the `-smh` option to include the SMH file and indicate the ASD region to target. For example:

```
$ quartus_fid --cable=USB-BlasterII --index "@1=svgx.sof#pi" --
weight=100.0.0 --smh="@1=svgx.smh#2" --number=30
```

This example command:

- Programs the device and injects faults (`pi` string)
- Injects 100% single-bit faults (100.0.0)
- Injects only into `ASD_REGION` 2 (indicated by the `#2`)
- Injects 30 faults

1.8. Fault Injection IP Core User Guide Archives

If an IP core version is not listed, the user guide for the previous IP core version applies.

IP Core Version	User Guide
18.0	Fault Injection Intel FPGA IP Core User Guide
17.1	Intel FPGA Fault Injection IP Core User Guide
16.1	Altera Fault Injection IP Core User Guide
15.1	Altera Fault Injection IP Core User Guide

1.9. Document Revision History for Fault Injection IP Core User Guide

Document Version	Intel Quartus Prime Version	Changes
2019.07.09	18.1	Updated the *Fault Injection IP Pin Description* topic to clarify the `Reset`, `error_injected`, and `error_scrubbed` signals.
2018.05.16	18.0	• Added the following topics from Intel Quartus Prime Pro Edition Handbook: — *Defining Fault Injection Areas* and subtopics. — *Using the Fault Injection Debugger* and subtopics. — *Command-Line Interface* and subtopics. • Renamed Intel FPGA Fault Injection IP core to Fault Injection Intel FPGA IP.

Date	Version	Changes
2017.11.06	17.1	• Rebranded as Intel. • Added Intel Cyclone 10 GX device support.
2016.10.31	16.1	Updated device support.
2015.12.15	15.1	• Changed Quartus II to Quartus Prime software. • Fixed self-referencing related link.
2015.05.04	15.0	Initial release.