Soft Error Detection IP Core

User's Guide

March 2006
ipug49_01.0

Introduction

Soft errors occur when high-energy charged particles alter the stored charge in a memory cell in an electronic circuit. The phenomenon first became an issue in DRAM, requiring error detection and correction for large memory systems in high-reliability applications. As device geometries have continued to shrink, the probability of soft errors in SRAM is becoming significant for some systems. Designers are using a variety of approaches to minimize the effects of soft errors on system behavior.

SRAM-based FPGAs store logic configuration data in SRAM cells. As the number and density of SRAM cells in an FPGA increase, the probability that a soft error will alter the programmed logical behavior of the system increases. The module described in this document provides a mechanism for detecting errors in configuration memory for the FPGA.

Core Description

The core consists of an access point to FPGA configuration memory, a controller circuit, and a personality ROM. As serial data is read from configuration memory, CRC is calculated over the bitstream. After roughly 21 data frames, the current CRC calculation is compared with the expected CRC result stored in the personality ROM. If the CRC values match, it indicates that there has been no configuration memory corruption. If the values differ, an error signal is generated along with the row number where the error occurred. This process is repeated until every bit of configuration memory is analyzed. In addition, CRC is checked over the contents of the personality ROM.

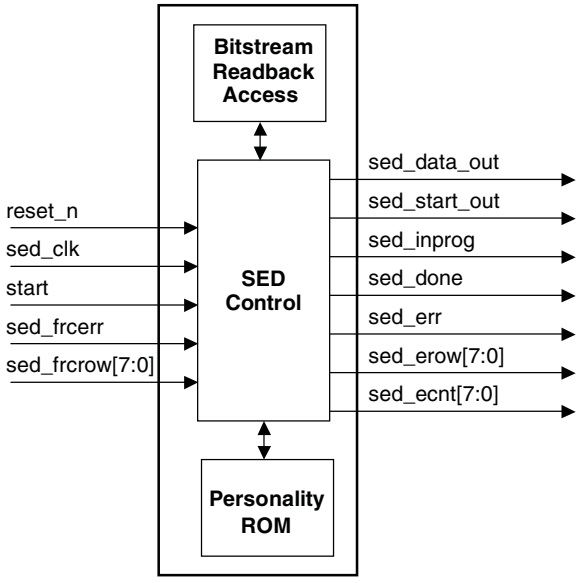
The Soft Error Detection Core utilizes a 17-bit CRC algorithm with the following attributes:

- 100% detection of single-bit errors
- 100% detection of two-bit errors within 10 adjacent frames
- 100% detection of multiple-bit errors within a 17-bit span
- $(1-2^{-17})*100\%$ detection of random multiple-bit errors

Note that the expected CRC calculations are based upon the particular arrangement of configuration memory for a particular design. Consequently, the expected results cannot be specified until after the design is placed and routed. ispLEVER® bit generation software analyzes the configuration of a placed and routed design, and updates the personality ROM contents during bitstream generation.

Block Diagram

Figure 1. Soft Error Detection IP Core Block Diagram



Signal Descriptions

<table>
<tr><th>Name</th><th>I/O</th><th>Description</th></tr>
<tr><td>reset_n</td><td>In</td><td>Active low reset. When asserted, resets all internal registers and state machines</td></tr>
<tr><td>sed_clk</td><td>In</td><td>System clock. Maximum frequency is 200MHz. All inputs and outputs are referenced to the rising edge of the clock.</td></tr>
<tr><td>start</td><td>In</td><td>Active high control signal. When asserted, triggers the start of a soft error detect analysis cycle. Once the SED analysis starts, the state of the start signal is ignored. After the SED analysis completes and sed_done asserts, the core executes a 14 clock cycle wait state - and afterwards, the start signal state is monitored. Note that by tieing start permanently high, the SED core executes continuous back-to-back SED analysis cycles.</td></tr>
<tr><td>sed_frcerr</td><td>In</td><td>Active high control signal. It provides a diagnostic capability that forces an error to occur in the analyzed FPGA array bitstream. When sed_frcerr is asserted while the SED analysis is in progress, a datastream error is inserted in the row specified by the sed_frcrow signal. In response to the forced error, the SED core should detect the error. Errors are reported on the signals: sed_err, sed_erow, sed_ercnt.</td></tr>
<tr><td>sed_frcrow[7:0]</td><td>In</td><td>This signal specifies an 8-bit numeric value that indicates the FPGA row on which a forced error is inserted (MSB=bit7). This signal works in concert with the sed_frcerr signal. Note that the maximum row number is different for each of the FPGA array sizes in the LatticeSC™ family (see table below). Specifying a sed_frcrow number beyond the range of the FPGA array size causes the SED core to fail to insert the requested forced error.
<table>
<tr><th>LatticeSC Device</th><th>Valid Row Numbers</th></tr>
<tr><td>S15</td><td>0-45</td></tr>
<tr><td>S25</td><td>0-57</td></tr>
<tr><td>S40</td><td>0-71</td></tr>
<tr><td>S80</td><td>0-95</td></tr>
<tr><td>S115</td><td>0-117</td></tr>
</table></td></tr>
<tr><td>sed_data_out</td><td>Out</td><td>This signal displays the serial FPGA datastream analyzed by the SED core.</td></tr>
<tr><td>sed_start_out</td><td>Out</td><td>Active high signal that remains asserted throughout the entire SED analysis. The leading edge of the assertion begins 11 clock cycles after assertion of the start signal.</td></tr>
<tr><td>sed_inprog</td><td>Out</td><td>Active high signal that remains asserted throughout the entire SED analysis. The leading edge of the assertion begins when the core first detects the start sequence embedded in the serial FPGA datastream. The duration of the SED analysis is different for each of the FPGA array sizes in the LatticeSC family. See the table below for approximate duration intervals.
<table>
<tr><th>LatticeSC Device</th><th>In Progress Duration
(number of sed_clk cycles)</th></tr>
<tr><td>S15</td><td>4,423,344</td></tr>
<tr><td>S25</td><td>7,686,464</td></tr>
<tr><td>S40</td><td>11,763,264</td></tr>
<tr><td>S80</td><td>22,250,944</td></tr>
<tr><td>S115</td><td>30,613,424</td></tr>
</table></td></tr>
<tr><td>sed_done</td><td>Out</td><td>Active high signal that asserts when a SED analysis cycle ends. The signal deasserts when another SED analysis is started.</td></tr>
<tr><td>sed_err</td><td>Out</td><td>Active high signal that asserts when an error is detected during the SED analysis. sed_err is valid only when sed_done is asserted.</td></tr>
<tr><td>sed_erow[7:0]</td><td>Out</td><td>This signal specifies an 8-bit numeric value that indicates the FPGA row number associated with any detected SED errors. If errors are detected in multiple rows, this signal specifies the first row.</td></tr>
<tr><td>sed_ercnt[7:0]</td><td>Out</td><td>This signal specifies an 8-bit numeric value that indicates the number of FPGA rows that experienced errors during the SED analysis.</td></tr>
</table>

Core Operation

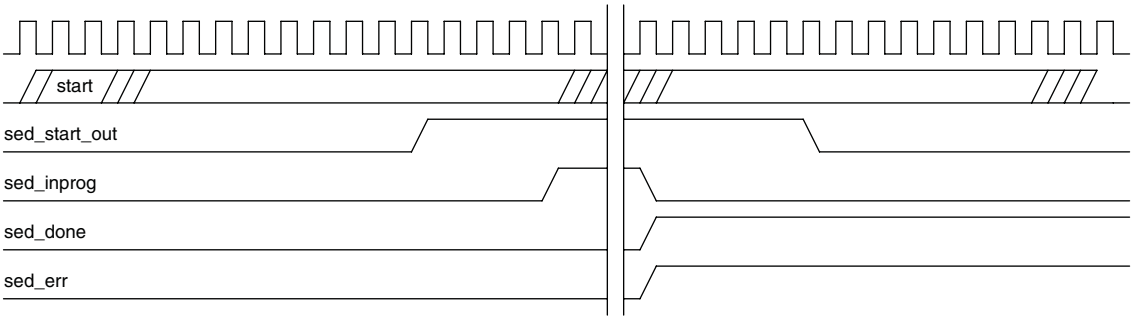
Overview

A typical SED analysis cycle is as follows:

- Trigger start of SED analysis
- Verify that SED analysis starts
- Wait for SED analysis to complete
- Check for errors at end of cycle

A typical SED timing diagram is shown below.

Figure 2. SED Timing Diagram



Starting a SED Analysis Cycle

A SED analysis is started by driving the `start` signal high. However `start` is not monitored while the analysis is in progress (`sed_inprog` asserted). It is also not monitored during the first 14 clock cycles after `sed_done` asserts. Therefore, an analysis cannot be started during these blind periods. A valid start signal can be as short as a single clock cycle, as long as it falls outside one of the blind periods.

Verifying that SED Analysis is Running

Eleven clock cycles after a valid `start` assertion, the `sed_start_out` signal asserts. Afterwards, the `sed_inprog` signal asserts when the readback datastream start sequence is detected. Both `sed_start_out` and `sed_inprog` remain asserted through the remainder of the SED analysis cycle.

Detecting the End of SED Analysis

The SED analysis is finished when the `sed_done` signal asserts. `sed_done` remains asserted until another analysis is started.

Detecting SED Errors

If a soft error is detected during the analysis, the `sed_err` signal is asserted. The `sed_err` signal in only valid while `sed_done` is also asserted. In addition, the `sed_erow` signal indicates the first array row that contains a soft error; the `sed_ecnt` signal indicates the total number of array rows that contained soft errors.

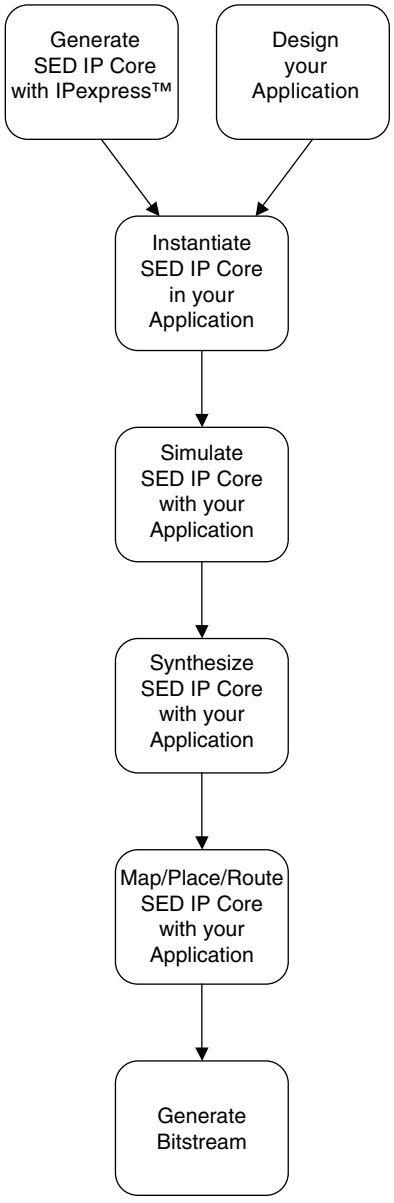
Forcing Errors

The `sed_frcerr` and `sed_frcrow` signals control forced soft error insertion. Driving `sed_err` high enables error insertion. The `sed_frcrow` signal specifies the row on which the forced soft error is inserted. Both signals must remain asserted throughout the entire period that the SED analysis is in progress. Note that forcing a soft error does not actually modify the contents of the FPGA array. Instead, the error is only inserted in the analyzer circuit. Therefore, forcing soft errors will not result in a malfunction in your FPGA design.

Design Flow Overview

A typical design flow using the SED IP core is illustrated below. The following few paragraphs explain the steps in detail.

Figure 3. Design Flow



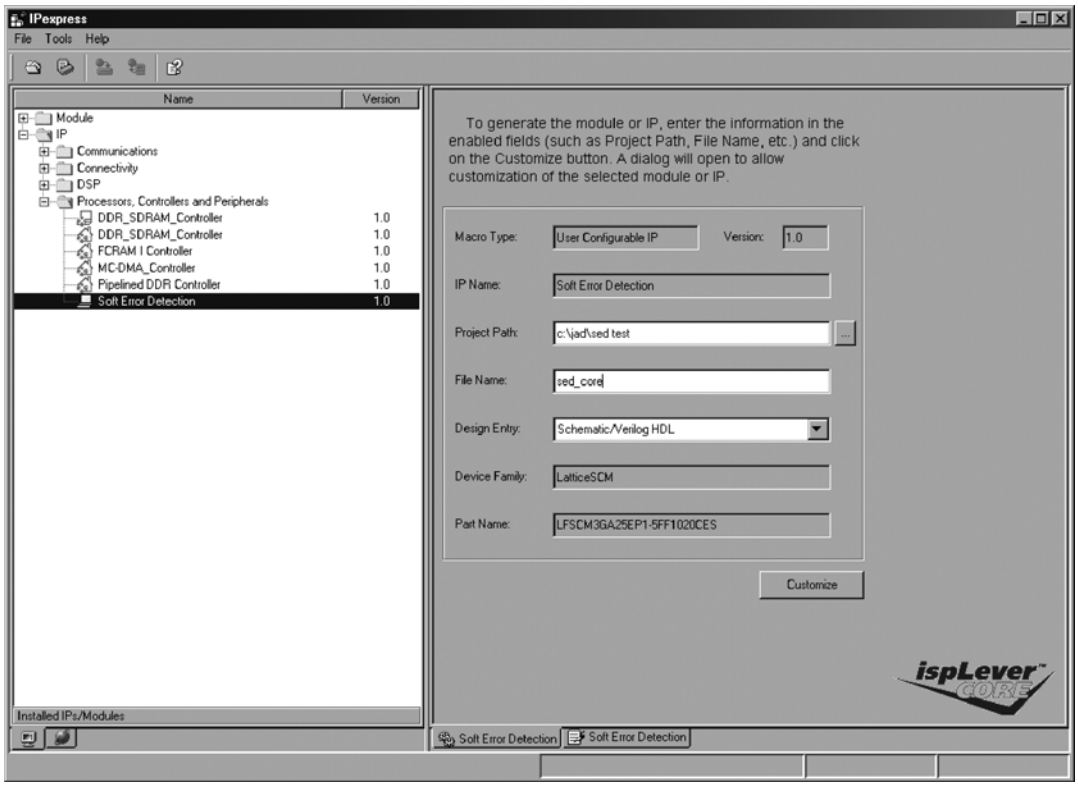
Core Generation

The SED IP core is available for download from the Lattice IP Server tab in the IPexpress main window.. The IP files are automatically installed using ispUPDATE technology in any customer-specified directory.

The ispLEVER IPexpress GUI window for the SED IP core is shown in Figure 4. The procedure for generating the SED core follows:

- Launch IPexpress by clicking **Start-> Programs-> LatticeSemiconductor-> Accessories-> IPExpress**. The window shown in Figure 4 will be displayed

Figure 4. IP Selection Window

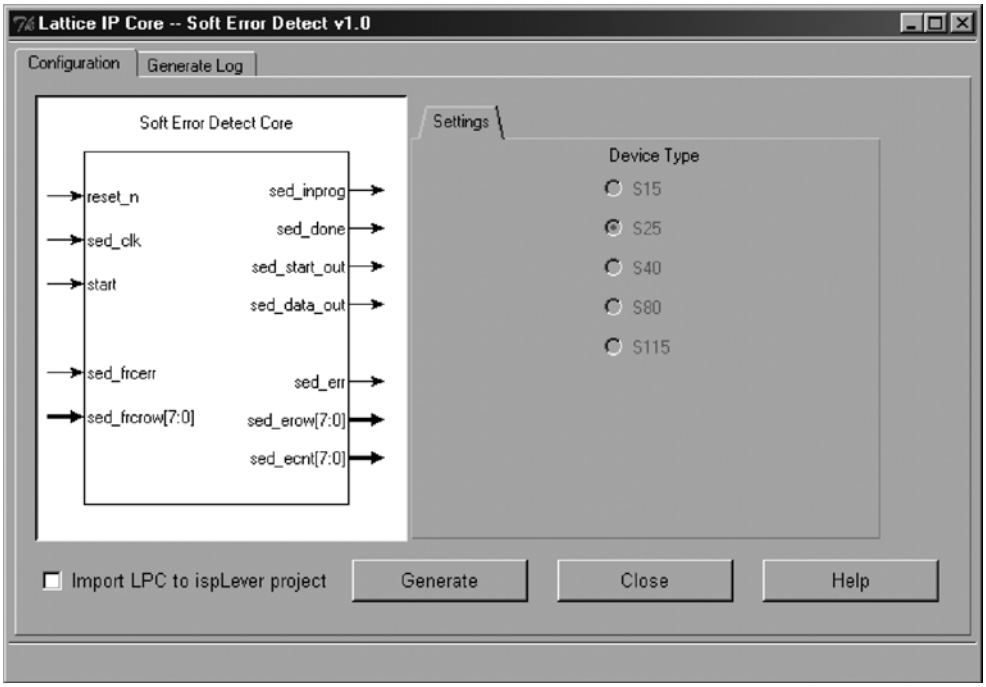


- Choose the SED IP Core by clicking **IP-> Soft Error Detect Core** in the left-hand pane of the IPexpress window.
- Choose the directory in which you want your IP core files to be generated by entering the appropriate path in the **Project Path** text box.
- Enter the name **sed_core** in the **File Name** text box. You must enter the name "sed_core". Otherwise, the IP core generation will fail.
- Select the desired HDL format (Verilog or VHDL) that you want for your generated files by choosing the desired option at the **Design Entry** test box.
- Select the **Device Family**.
- Select the **Part Name** to be used for your design. Note, it is important to choose the correct device type during core generation because operation of the SED core is tied to the array size. The SED core will not detect soft errors if it is downloaded into an array that is a different size than the array for which the core was originally generated.

  Note also that although array size is critical; for example S25 vs. S40, package type is not critical, i.e. if a core is generated using one package type and downloaded the design into a different package type, it will still operate correctly as long as the array size is correct. If IPexpress is called from within an existing project, Project Path, Design Entry, Device Family and Part Name default to the specified project parameters. Please refer to the IPexpress on-line help for further information.

- Hit the **Customize** button. The window shown in Figure 5 will be displayed.

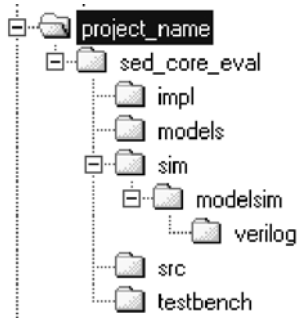
Figure 5. IP Generation Window



- Hit the **Generate** button in the new window. This will generate the IP core files. Close all windows when finished

After clicking the Generate button, the configuration-specific IP core and supporting files are generated in the user's project directory. The directory structure of the generated files is shown in Figure 6.

Figure 6. SED IP Core Generated Directory Structure



The table below shows the basic files generated in Verilog format. Similar files are generated for VHDL format. These are all of the files necessary to implement and verify the SED IP core in a top-level design.

Table 1. SED IP Core Generated Files

File Name	Description
Design Files	
sed_core_sc_1_001.ngo	Synthesized and mapped form of SED IP core. This file must be included in the same directory that your overall top-level application design is mapped with ispLEVER.
sed_core_bb.v	Black box description of SED IP core. This file should be used to describe the SED core when synthesizing your top level application.
sed_core_inst.v	Example of SED IP core instantiation.
sed_core_sim.v	Behavioral simulation model for SED IP core
Miscellaneous Files	
sed_core.lpc	Parameters associated with your configuration choices for SED IP core.
sed_core_generate.log	Lists IPexpress messages generated during SED core file generation.
sed_core_filelist.log	List of generated SED IP core files.

The \sed_core_eval and subtending directories provide files supporting SED core evaluation. The \sec_core_eval directory shown in Figure 6 contains files/folders with content that is constant for all configurations of the SED core. The \sec_core_eval directory is created by IPexpress the first time the core is generated and updated each time the core is regenerated.

Instantiating the Core

The generated SED IP core package includes black-box (sed_core_bb.v/sed_core_component.vhd) and instance (sed_core_inst.v/vhd) templates (Verilog or VHDL) that can be used to instantiate the core in a top-level design. An example RTL top-level reference source file that can be used as an instantiation template for the IP core is provided in \sed_core_eval\src. Users may also use this top-level reference as the starting template for the top-level for their complete design.

Running Functional Simulation

The generated IP core package includes a configuration-specific behavior model (sed_core_sim.v/vhd) for functional simulation. A top-level file supporting ModelSim® eval simulation is provided in \sed_core_eval\sim\modelsim.

ModelSim simulation is supported via testbench files provided in \sed_core_eval\testbench. Models required for simulation are provided in the corresponding \models folder.

Users may run the eval simulation by doing the following:

1. Open ModelSim

2. Under the File tab, select **New**, then **Project**

3. Set project location to **\sed_core_eval\sim\modelsim\verilog (or \vhdl)**

4. Specify a **Project Name** (may be any name)

5. Select **OK** to create the project

6. Close **Add items to the Project** box without selecting anything

7. Execute simulation **do** script **sed_core_eval.do** located in **\sed_core_eval\sim\modelsim\verilog (or \vhdl)**

Synthesizing and Implementing the Core in a Top-Level Design

The SED core itself is provided in NGO format when the core is generated. Users may synthesize the core in their own top-level design by instantiating the core in their top-level as described previously and then synthesizing the entire design with either Synplicity® or Precision® RTL Synthesis.

An example RTL top-level reference source file supporting SED core top-level synthesis and implementation is provided with the SED IP core in `\sed_core_eval\src`. This reference design supports the ability to synthesize and map just the SED core itself. This design is intended to provide an example of how to instantiate the core in a top-level design and an accurate indication of the device utilization associated with the SED core itself.

Push-button implementation of the reference design is supported via the ispLEVER project file `sed_core_eval.syn` located in `\sed_core_eval\impl`. To use this project file:

1. Select **Open Project** under the **File** tab in ispLEVER
2. Browse to **\sed_core_eval\impl** in the **Open Project** dialog box
3. Select and open **sed_core_eval.syn**. At this point, all of the files needed to support top-level synthesis and implementation will be imported to the project.
4. Implement the complete design via the standard ispLEVER GUI flow

Hardware Evaluation

Lattice's IP hardware evaluation capability makes it possible to create versions of IP cores that operate in hardware for a limited period of time (approximately one hour) without requiring the purchase on an IP license. The hardware evaluation capability is enabled by including a specific attribute string in the IP top-level design.

For Verilog flows, the attribute string is included in the top-level module declaration as follows:

```
module <top_design> (
signal_1,
signal_2,
.
.
signal_n
)
/* <attribute string> */
;
```

The specific attribute string to be inserted is a function of the Lattice FPGA device family being targeted and the synthesis tool being used. When synthesizing the top-level with Synplify, the following attribute string should be included for LatticeSC devices:

```
/* synthesis LSC_IP_SC_HT_SED="LSC_IP_sed_core_sc_ipe" */
```

When synthesizing with Precision RTL Synthesis, the following attribute string should be included:

```
/* pragma attribute <top_design> LSC_IP_SC_HT_SED LSC_IP_sed_core_sc_ipe */
```

For VHDL flows, the attribute string is included in the architecture declaration as follows:

```
architecture <arch> of <top_design> is

attribute LSC_IP_SC_HT_SED : string;

attribute LSC_IP_SC_HT_SED of <arch> : architecture is "LSC_IP_sed_core_sc_ipe";
```

For VHDL flows the same attribute string is used for both Synplify and Precision RTL Synthesis.

The same string may be included in customer top-level designs to enable hardware evaluation of the SED IP core in customer-defined applications. When the attribute string is included in the top level of the design, it is possible to generate a programming file that may be downloaded into the device. After initialization, the IP core will be operational for approximately one hour. After one hour, the IP core will stop working and it will be necessary to reprogram the device to re-enable operation. This hardware evaluation capability is only enabled if the core has not been licensed. During implementation, a license check is performed. If a license is not detected, a pop-up window will be displayed indicating a license failure. Click **OK** in the window and generation will proceed to completion with hardware evaluation enabled. If a license is detected, no pop-up window is displayed and core generation is completed with no restrictions.

References

The following documents provide more information on implementing this core.

- ispLEVER Software User Manual
- ispLeverCORE™ IP Module Evaluation Tutorial available on the Lattice website at www.latticesemi.com

Technical Support Assistance

Hotline: 1-800-LATTICE (North America)
+1-503-268-8001 (Outside North America)
e-mail: techsupport@latticesemi.com
Internet: www.latticesemi.com

Appendix for LatticeSC FPGAs

Table 2. Performance and Resource Utilization1

Device	SLICEs	LUTs	Registers	External Pins	sysMEM™ EBRs	f_{MAX} (MHz)
SC25	283	478	313	0	1	200MHZ

1. Performance and utilization characteristics are in Lattice's ispLEVER 5.1 SP2 software with Synplify synthesis and targeting a LatticeSC LFSC3GA25EP1-5FF1020CES device. When using this IP core in a different density, speed, or grade within the LatticeSC family or in a different software version, performance may vary.

Supplied Netlist Configurations

The Ordering Part Number (OPN) for the Soft Error Detection IP core targeting LatticeSC devices is SED-CORE-SC-U1. You can use the IPexpress software tool to help generate new configurations of this IP core. IPexpress is the Lattice IP configuration utility, and is included as a standard feature of the ispLEVER design tools. Details regarding the usage of IPexpress can be found in the IPexpress and ispLEVER help system. For more information on the ispLEVER design tools, visit the Lattice web site at: www.latticesemi.com/software.