Soft Error Detection (SED)/Correction (SEC) User Guide for Nexus Platform

Technical Note

FPGA-TN-02076-2.1

July 2024

Disclaimers

Contents

Contents 3
Acronyms in This Document 5
1. Introduction 6
2. SED Overview 7
2.1. Block Diagram 9
3. Port List 10
4. Port Descriptions 11
4.1. cfg_clk_i 11
4.2. sedc_rst_i 11
4.3. sedc_mode_i 11
4.4. sedc_start_i 11
4.5. sedc_en_i 11
4.6. sed_en_i 11
4.7. sedc_cof_i 12
4.8. sedc_busy_o 12
4.9. sedc_errc_o 12
4.10. sedc_err_o 12
4.11. sedc_errm_o 12
4.12. sedc_errcrc_o 12
4.13. sedc_frm_errloc_o[15:0] 12
4.14. sedc_dsr_errloc_o[12:0] 12
5. SED Clock and Reset 13
6. SEDC Flow 14
6.1. SED Mode 15
6.1.1. Continuous Mode 15
6.1.2. One-shot Mode 15
6.2. SEDC Error Handling 16
7. SEDC Run Time 22
8. Sample Code 23
8.1. SEDC Verilog Example 23
8.1.1. Verilog Example of SEDC IP 23
8.1.2. Verilog SEDC IP Instantiation 23
8.2. SEDC VHDL Example 24
8.2.1. VHDL Component Instantiation 24
8.2.2. VHDL Instantiation 24
9. Soft Error Injection (SEI) 25
10. Important Note 27
References 29
Technical Support Assistance 30
Revision History 31

Figures

Figure 2.1. Bitstream Data Structure 7
Figure 2.2. SEDC System Block Diagram 8
Figure 2.3. SEDC Block Diagram 9
Figure 5.1. SEDC CLK/RST 13
Figure 6.1. SEDC Flow 14
Figure 6.2. SED Continuous Mode 15
Figure 6.3. SED One-Shot Mode 16
Figure 6.4. One-shot Mode with Correctable Error (sedc_cof_i = 0 / 1) 17
Figure 6.5. SEDC Continuous Mode with Correctable Error (sedc_cof_i = 0 / 1) 17
Figure 6.6. SEDC One-shot Mode or Continuous Mode with Multiple Error in One Frame (sedc_cof_i = 0) 18
Figure 6.7. SEDC One-shot Mode with Multiple Error in One Frame (sedc_cof_i = 1) 18
Figure 6.8. SEDC Continuous Mode with Multiple Error in One Frame (sedc_cof_i = 1) 19
Figure 6.9. SEDC One-shot Mode with CRC Error (sedc_cof_i = 0 / 1) 19
Figure 6.10. SEDC Continuous Mode with CRC Error (sedc_cof_i = 0 / 1) 20
Figure 6.11. SEDC One-shot Mode with SEC Disabled (sed_en_i = 0) 20
Figure 6.12. SEDC Continuous Mode with SEC Disabled (sed_en_i = 0) 21
Figure 9.1. SEI Editor 25
Figure 9.2. Device Properties 26

Tables

Table 3.1. SEDC Primitive Port Definitions 10
Table 5.1. SEDC Internal Oscillator Divider Settings 13
Table 7.1. Configuration-related Parameters that Affect SED Run Time 22

Acronyms in This Document

A list of acronyms used in this document.

Acronym	Definition
BCH	The Bose, Chaudhuri, and Hocquenghem (BCH) codes form a large class of powerful random error-correcting cyclic codes. This class of codes is a remarkable generalization of the Hamming code for error correction.
CRC	Cyclic Redundancy Check
CRE	Cryptographic Engine
DRAM	Dynamic Random Access Memory
ECC	Error Correcting Code
ESFB	Embedded Security and Function Block
FIT	Failures in Time
GPIO	General Purpose Input Output
IP	Intellectual Property
ISC	In-System Configuration
PLD	Programmable Logic Device
SEC	Soft Error Correction
SED	Soft Error Detection
SEDC	Soft Error Detection/Correction
SEI	Soft Error Injection
SRAM	Static Random Access Memory

1. Introduction

This document describes the hard-logic based Soft Error Detection (SED) approach taken by Lattice Semiconductor for the Lattice Nexus™ platform devices, including Crosslink™-NX, Certus™-NX, CertusPro™-NX, and MachXO5™-NX families. Once soft error is detected, Lattice provides an easy way to optionally perform the Soft Error Correction (SEC) without disturbing the functionality of the device.

Memory errors can 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 Dynamic Random Access Memory (DRAM), requiring error detection and correction for large memory systems in high-reliability applications. As device geometries continue to shrink, the probability of memory errors in Static Random Access Memory (SRAM) becomes significant for some systems. Designers now use a variety of approaches to minimize the effects of memory errors on system behavior. FPGA devices built on the Lattice Nexus platform, which include CrossLink-NX, Certus-NX, CertusPro-NX, and MachXO5-NX devices, are unique because the underlying technology used to build them is much more robust and less prone to soft errors.

SRAM-based programmable logic devices (PLDs) store logic configuration data in SRAM cells. As the number and density of SRAM cells in a PLD increase, the probability that a memory error alters the programmed logical behavior of the system increases. A number of traditional approaches are taken to address this issue, but most involve soft Intellectual Property (IP) cores that instantiate into the logic of user design, utilizing valuable resources and possibly affecting design performance.

The Nexus platform devices have an improved hardware implemented Soft Error Detection (SED) circuit which can be used to detect SRAM errors and allow them to be corrected. There are two layers of SED implemented in these devices that make them more robust and reliable.

2. SED Overview

The SEDC module in a Nexus platform device is an enhanced version as compared to the SED modules implemented in other Lattice devices. Enhancements include:

- Frame by Frame SED check
- Single-bit and multi-bit error detection
- ECC to correct single-bit error at the frame level
- Programmable SEDC clock with a wider clock frequency option

The device is based on the Lattice Nexus platform which is developed using FDSOI technology. FDSOI transistors have a less active region, which helps to reduce bit flipping when exposed to alpha and neutron particles.

Some of the key advantages of Nexus platform devices as compared to other devices are:

- Improved radiation tolerance due to reduction in critical area separated by a thin layer of buried oxide (BOX)
- 100x improvement in soft errors

Due to the above technology, Nexus platform devices have extremely low bit error rate and FIT rate. Details about the FIT rate calculation can be found in Single Event Upset (SEU) Report for CrossLink-NX (FPGA-TN-02174).

This SEDC module is part of the Configuration block in the Nexus platform devices. The configuration data is divided into frames so that the FPGA can be programmed as a whole or in precise parts. The SED hardware reads serial data from the FPGA's configuration memory frame-by-frame in the background while the device is in User mode and performs Error Correcting Code (ECC) calculation on every frame of configuration data (Figure 2.1). Once a single bit error is detected, a Soft Error Upset (SEU) notification is generated and SED resumes operation. When Soft Error Correction (SEC) is enabled, single bit errors are corrected; the corrected value is rewritten to the particular frame using ECC information. If more than one-bit error is detected within one frame of configuration data, an error message is generated. In parallel, cyclic redundancy check (CRC) is calculated for the entire bitstream along with ECC.

After the ECC is calculated on all frames of configuration data, cyclic redundancy check (CRC) is calculated for the entire configuration data, that is, the bitstream.

Due to the dynamic contents of memories, the CRC and ECC calculations do not include EBR and Large RAM memory. Dynamic RAM should not be used with SED. Otherwise, SED reports failures when normal RAM content changes occur.

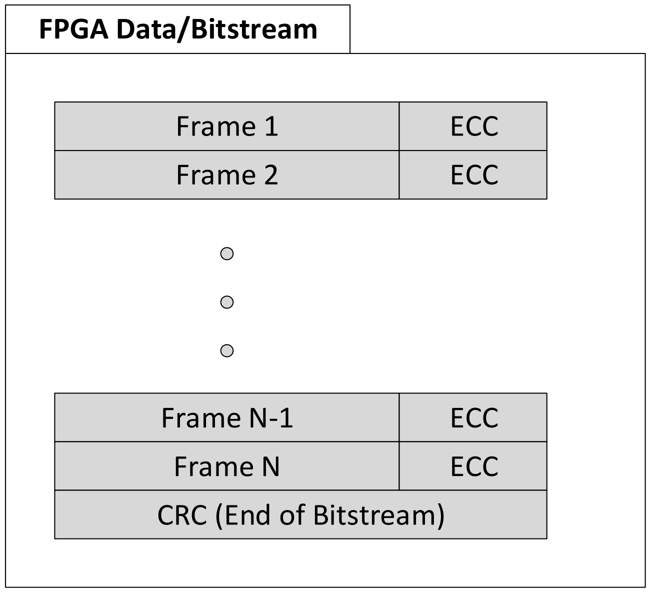


Figure 2.1. Bitstream Data Structure

The SEDC IP is part of the sysCONFIG block of devices built on the Nexus platform. Figure 2.2 shows the system-level view of the SEDC IP.

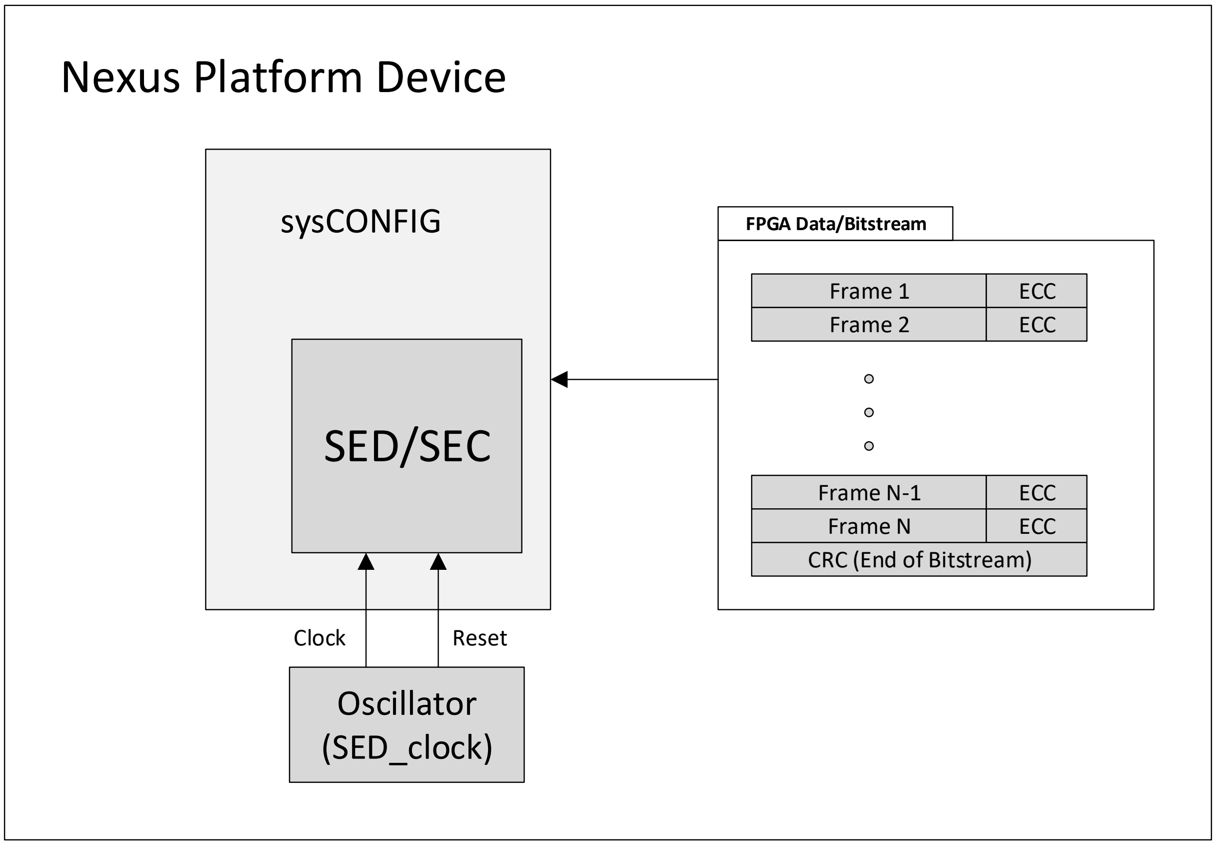


Figure 2.2. SEDC System Block Diagram

2.1. Block Diagram

The SED block in a Nexus platform device contains a number of inputs that control the actual SEDC block behavior. There are a number of modes that this SED block can operate in. Figure 2.3 shows a high-level block diagram of the user input and output ports.

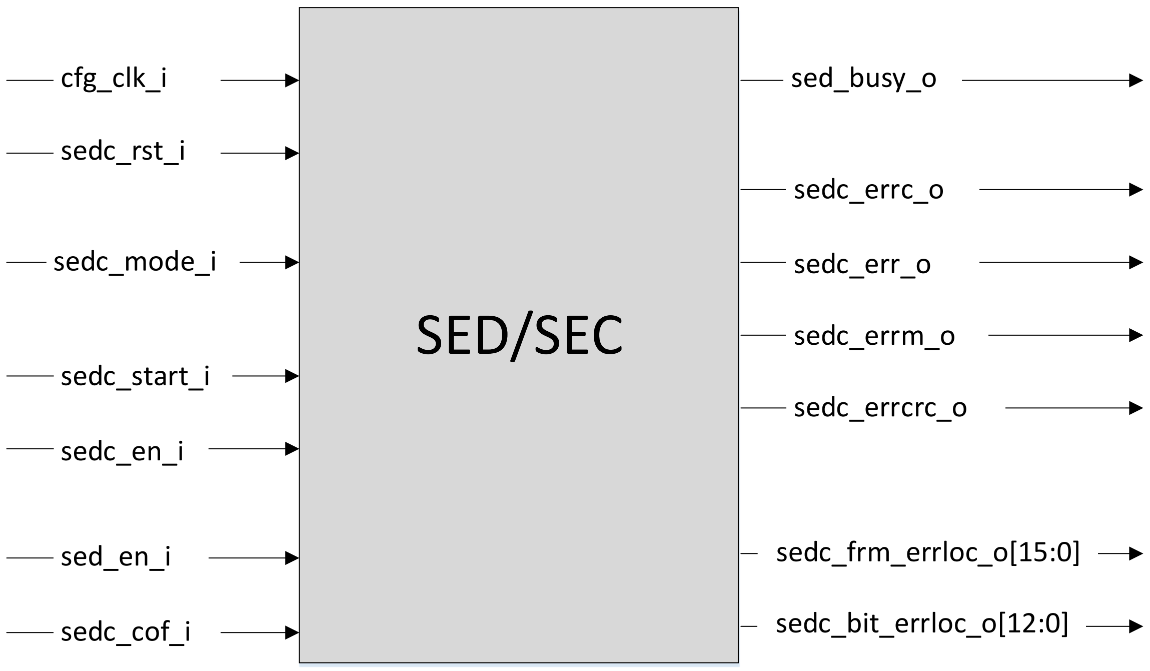


Figure 2.3. SEDC Block Diagram

3. Port List

To use the SEDC IP, instantiate the SEDC IP as well as the Oscillator primitive using the Lattice Radiant™ software IP Catalog. Refer to Figure 5.1 that shows how to connect the Oscillator to the SEDC IP. Below is a list of port signals used by the SEDC IP.

Table 3.1. SEDC Primitive Port Definitions

Port Name	Default Value	Active	Description
cfg_clk_i	0	Input	User input clock for SED. Connect this signal to the OSC module port *cfg_clk_o*.
sedc_rst_i	0	Input	Reset the SED IP. Connect this signal to the OSC module port *sedc_rst_o*.
sedc_mode_i	0	Input	SED mode signal Selects between two modes.
sedc_start_i	0	Input	SED signal used to start SED
sedc_en_i	0	Input	Signal to enable SED
sed_en_i	0	Input	Signal to enable Soft Error Correction
sedc_cof_i	0	Input	SED continue on failure signal
sedc_busy_o	0	Output	Signal to indicate SED in progress
sedc_errc_o	0	Output	Indicates Current SED error.
sedc_err_o	0	Output	Sticky bit, 1-bit correctable error detected.
sedc_errm_o	0	Output	Sticky bit, Multi-bit error detected within one frame.
sedc_errcrc_o	0	Output	CRC error for entire bitstream data
sedc_frm_errloc_o[15:0]	0	Output	Frame Error location
sedc_dsr_errloc_o[12:0]	0	Output	Bit error location within a frame

4. Port Descriptions

4.1. cfg_clk_i

The *cfg_clk_i* is a user-selectable clock signal used to run the SEDC IP. Table 5.1 defines the various clock divider settings that can be used to define the desired clock speed of the SED block.

4.2. sedc_rst_i

The *sedc_rst_i* signal is used to reset the SEDC IP. This is an asynchronous reset signal. Connect this signal to the OSC module port *sedc_rst_o.*

4.3. sedc_mode_i

The *sedc_mode_i* signal is used to choose the SED mode of operation. There are two SED modes that you can choose from, which are continuous mode and one-shot mode:

- For continuous mode, the *sedc_mode_i* signal is high and once *sedc_start_i* is HIGH, the SED operation keeps running continuously.
- For one-shot mode, the *sedc_mode_i* signal is low and once the SED module detects the low-to-high transition on the *sedc_start_i* signal, the SED operation runs one SED cycle.

4.4. sedc_start_i

This *sedc_start_i* signal is used to start the SED operation. Once the *sedc_start_i* signal goes high for at least one SEDCLK cycle, the SED cycle starts if *sedc_en_i* is high. There are two SED modes that you can choose from, which are continuous mode and one-shot mode:

- In continuous mode, the *sedc_start_i* signal must remain high to keep SED running. If *sedc_start_i* goes low during the SED cycle, the process is terminated and *sedc_busy_o* is deasserted at the end of the SED cycle.
- For one-shot mode, it is recommended that you keep *sedc_start_i* signal high until *sedc_busy_o* goes high to ensure the signal is captured and SED starts running.

4.5. sedc_en_i

The *sedc_en_i* signal is used to enable SED. The SED does not operate, and any in-progress operation aborted, if *sedc_en_i* is deasserted. Do not de-assert this signal while the SED is running, that is, *sedc_busy_o* = 1. Doing so interrupts the ECC and CRC calculation, and results in the assertion of the *sedc_errcrc_o* signal. It is recommended for you to assert this signal throughout the SED operation, the only time you need to deassert this signal is to perform Soft Error Injection (SEI).

4.6. sed_en_i

If *sed_en_i* is set, the soft error correction is performed immediately as soon as a single correctable error is detected. If this bit is disabled, the correction is not done.

4.7. sedc_cof_i

The *sedc_cof_i* stands for SED_Continue_On_Failure. This signal is used to tell the SED module to run or stop after a non-correctable multi-bit error is detected in a single configuration frame. If *sedc_cof_i* signal is set to HIGH, the SED operation continues even if non-correctable error is encountered. On the other hand, if the *sedc_cof_i* signal is LOW, the SED operation is terminated as soon as an error is detected. This bit is useful for debugging purposes but not recommended for normal use. Instead, it is recommended to reload the FPGA bitstream through REFRESH, PROGRAMN, assertion, power cycle, or through one of the slave sysCONFIG ports, in the event that a non-correctable error is detected.

4.8. sedc_busy_o

The *sedc_busy_o* signal indicates if SED/SEC operation is currently in progress. If the SED is running, *sedc_buys_o* is set to HIGH. Once SED operation is complete, this signal goes LOW.

4.9. sedc_errc_o

The SED error current flag indicates if a soft error is detected. As soon as an error is detected, this flag goes high indicating it is a current error. This flag is not sticky.

4.10. sedc_err_o

The *sedc_err_o* flag is used to indicate if there is a single-bit error in a frame. This flag is sticky. To clear this flag, the SED operation must be disabled.

This single-bit error detected is also correctable. The correction is performed only if the *sed_en_i* signal is set.

4.11. sedc_errm_o

The SED error multiple is used to indicate if non-correctable errors are encountered, such as two or more errors detected in a single frame. Multiple errors are not correctable. This flag is asserted high when non-correctable error occurs. The flag is asserted for two-bit error per frame, but it is not guaranteed to be asserted for an error that is more than 2 bits due to the nature of the Hamming coding algorithm. This flag is sticky. To clear this flag, disable the SEDC module or reload the bitstream.

4.12. sedc_errcrc_o

The SED error CRC indicates if there is a mismatch between calculated CRC of the bitstream as compared to the expected CRC. This error is generated once all the frames of bitstream are read. If there is a single-bit error detected, this flag is set. Once the single-bit error is corrected, this CRC flag is cleared when SED operation runs for the second time.

4.13. sedc_frm_errloc_o[15:0]

The SED frame Error location reports the last location of the frame that errored out. It only provides the frame location for the last one-bit error. This signal reports the 16-bit error location for the frame that is causing error. This signal is only used for information purposes which can be used for further analysis of the SED errors. This field contains invalid data if multiple errors per frame are detected.

Note: This signal is only valid when SEC is enabled, that is, *sed_en_i* is true.

4.14. sedc_dsr_errloc_o[12:0]

The SED bit error location reports the bit position in a particular frame that errored out. This information is useful so that you can perform detailed analysis of the bitstream on a bit-by-bit basis. This field contains invalid data if multiple errors per frame are detected.

Note: This signal is only valid when SEC is enabled, that is, *sed_en_i* is true.

5. SED Clock and Reset

The SEDC circuitry is driven by the FPGA device internal oscillator. You must instantiate the oscillator IP with SEDCLK option enabled along with the SEDC IP from the Lattice Radiant software IP catalog, and route the clock and reset signals between them, as shown in Figure 5.1. Note that the oscillator IP can be OSC IP or OSC for CRE IP.

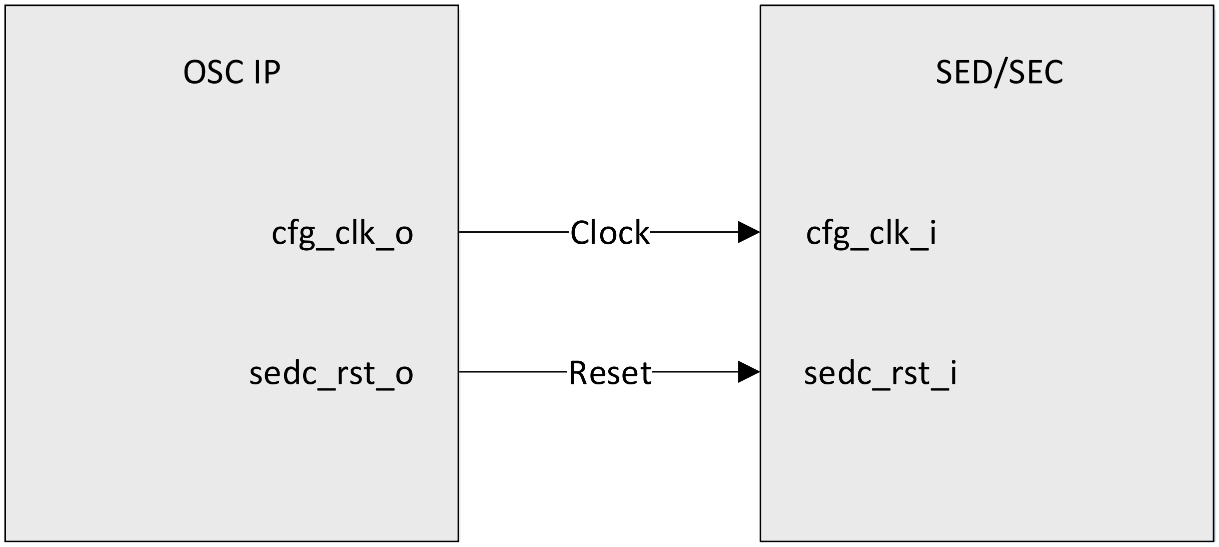


Figure 5.1. SEDC CLK/RST

The default oscillator frequency is 225 MHz. You can choose to lower the oscillator frequency by configuring the oscillator IP using the Lattice Radiant IP catalog. You can set the SEDCLK_divider setting anywhere between 2 to 256 in integer increments resulting frequency range from 225 MHz to 1.76 MHz (SED Oscillator Frequency = 450 MHz/SEDCLK_divider).

Table 5.1. SEDC Internal Oscillator Divider Settings

Divider Setting	SEDCLK_Divider	Oscillator Frequency (MHz)
Divide by 2	2	225
Divide by 3	3	150
Divide by 4	4	112.5
—	—	—
—	—	—
—	—	—
Divide by 256	256	1.76

6. SEDC Flow

This section describes the SEDC flow. The SEDC flow is executed once V_{CC} reaches the data sheet V_{CC} minimum recommended level and *sedc_en_i* and *sedc_start_i* are asserted.

Devices built on the Nexus platform have an advanced SEDC flow with two levels of SED checks. In the first level of SED check, the bitstream is read one frame at a time and the SED check is performed on a frame-by-frame basis. After all frames of the device bitstream are read, the SEDC module checks for CRC of the entire bitstream, second level SED, to check for the bitstream integrity giving the device improved SED performance. Figure 6.1 shows the SEDC flow in Nexus platform devices.

Devices built on the Nexus platform support real time Soft Error Correction (SEC) feature in which a single bit error can be corrected using ECC at the frame level. Once the SEC is enabled, the SEDC module reports the error location, providing details about the error frame and the exact location of a single bit error in that frame as shown in Figure 6.1.

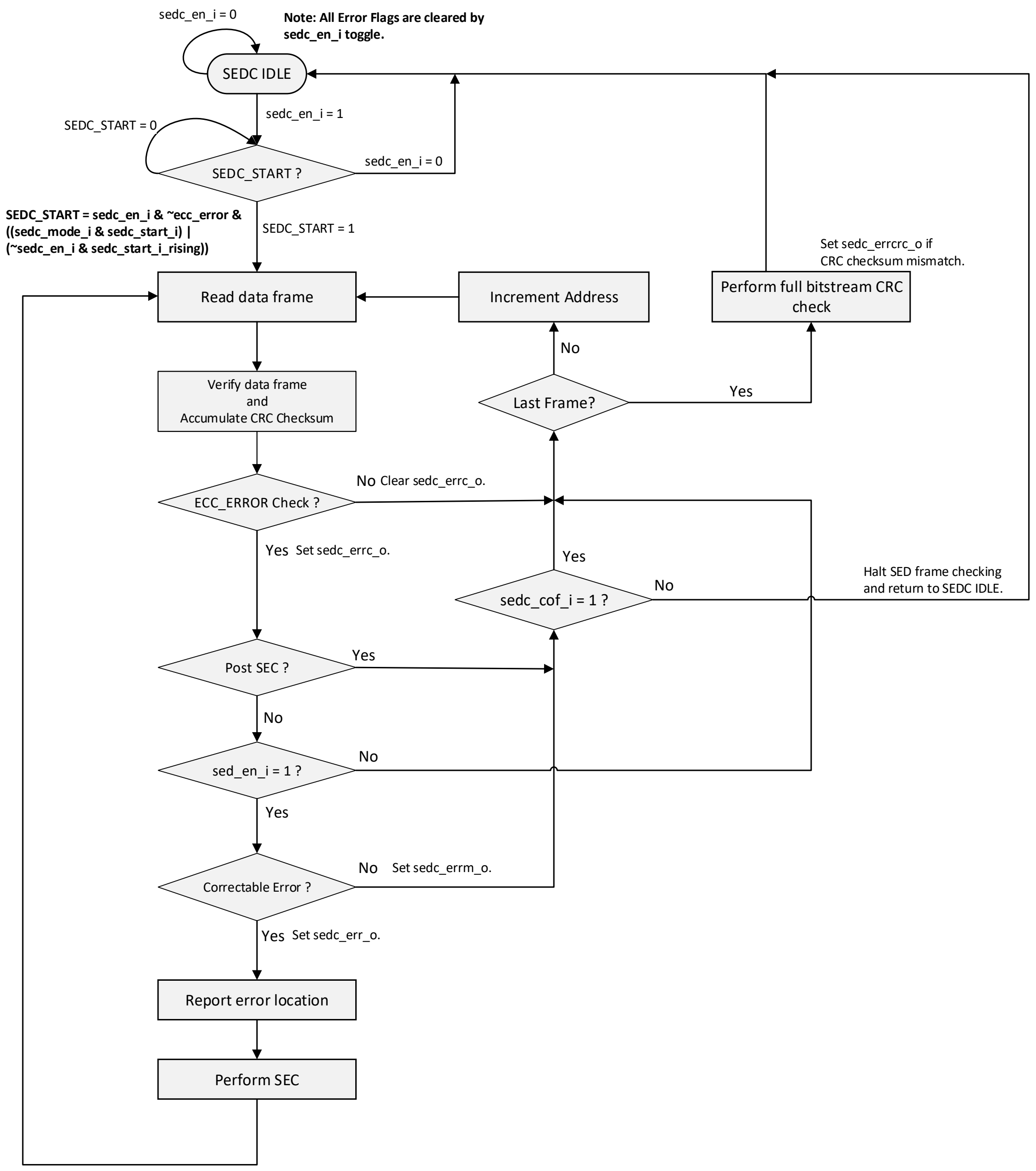


Figure 6.1. SEDC Flow

6.1. SED Mode

Devices built on the Nexus platform support two different SED modes. This provides the flexibility to run the SED. The first mode is the continuous mode in which the SED runs continuously. The other mode is one-shot mode in which SED runs once for each assertion of *sedc_start_i* signal.

6.1.1. Continuous Mode

As the name suggests, in continuous mode, the SED runs continuously as long as the *sedc_start_i* signal is high.

1. Once the SED is enabled, it starts reading bitstream data frame by frame and verifies if the data is read correctly from configuration SRAM. The *sedc_busy_o* signal is HIGH as long as SED is running.
2. Once SED finishes checking, the *sedc_busy_o* goes LOW when the SED cycles through for the first time.
3. The SED cycles through for the second time as long as *sedc_start_i* is HIGH since the operation is in Continuous Mode.

Note: Do not de-assert *sedc_en_i* while the detection is running, that is, *sedc_busy_o* = 1. This interrupts the CRC calculation and results in the assertion of the *sedc_errcrc_o* signal.

Once *sedc_mode_i* is set to 1 and *sedc_start_i* is always HIGH, the SED operation runs continuously, as shown in Figure 6.2.

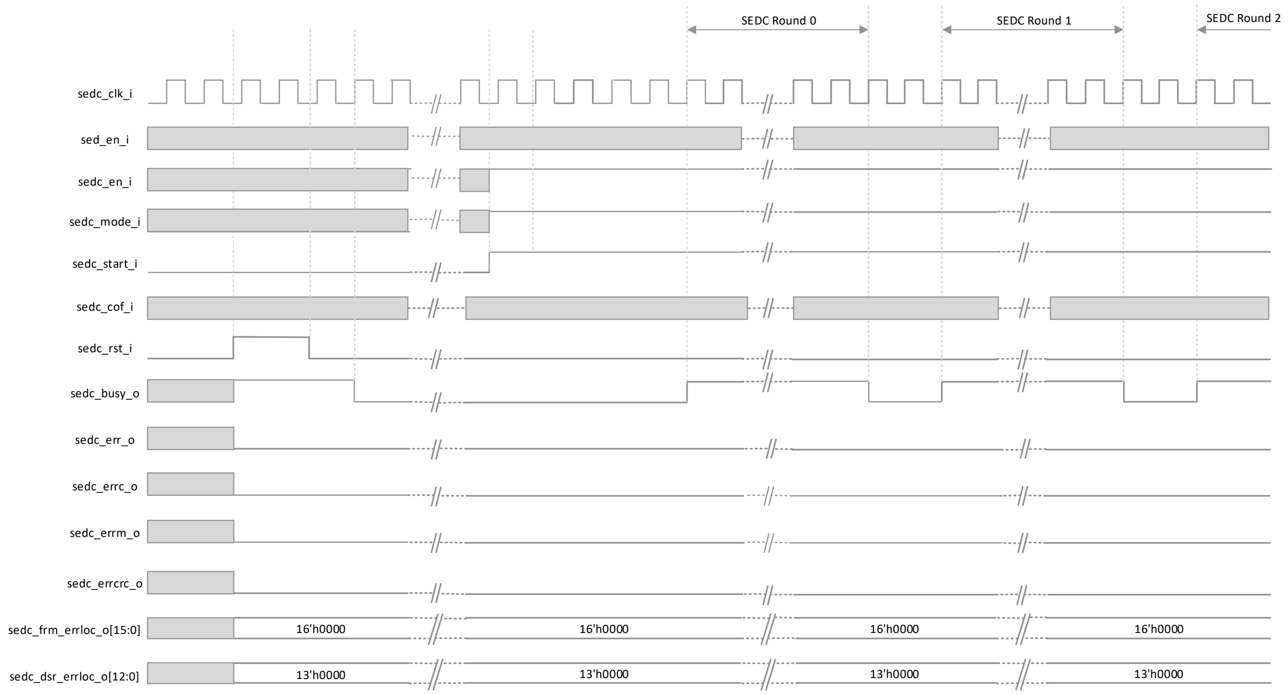


Figure 6.2. SED Continuous Mode

6.1.2. One-shot Mode

In this mode, the SED runs once for each assertion of the sedc_start_i signal.

1. For One-shot Mode, the *sedc_start_i* signal must have a LOW to HIGH transition to start the SED operation.
2. The SED starts reading bitstream data frame by frame and verifies if the data is read correctly from configuration SRAM. The *sedc_busy_o* signal is HIGH as long as SED is running.
3. The SED finishes checking. The SED error flags are updated and the *sedc_busy_o* flag goes LOW. Another SED cycle is started by making a LOW to HIGH transition on the *sedc_start_i* signal.

Notes:

- If there is any error, disable the *sedc_en_i* signal to reset all error flags.
- Do not de-assert *sedc_en_i* while the detection is running, that is *sedc_busy_o* = 1. This interrupts the CRC calculation and results in the assertion of the *sedc_errcrc_o* signal.

In this mode, the *sedc_mode_i* signal is set to zero. As soon as there is a low to high transition on the *sedc_start_i* signal, the SED operation starts. The SED operation is run once for each assertion of the *sedc_start_i* signal and when done, this *sedc_busy_o* goes LOW, as shown in Figure 6.3.

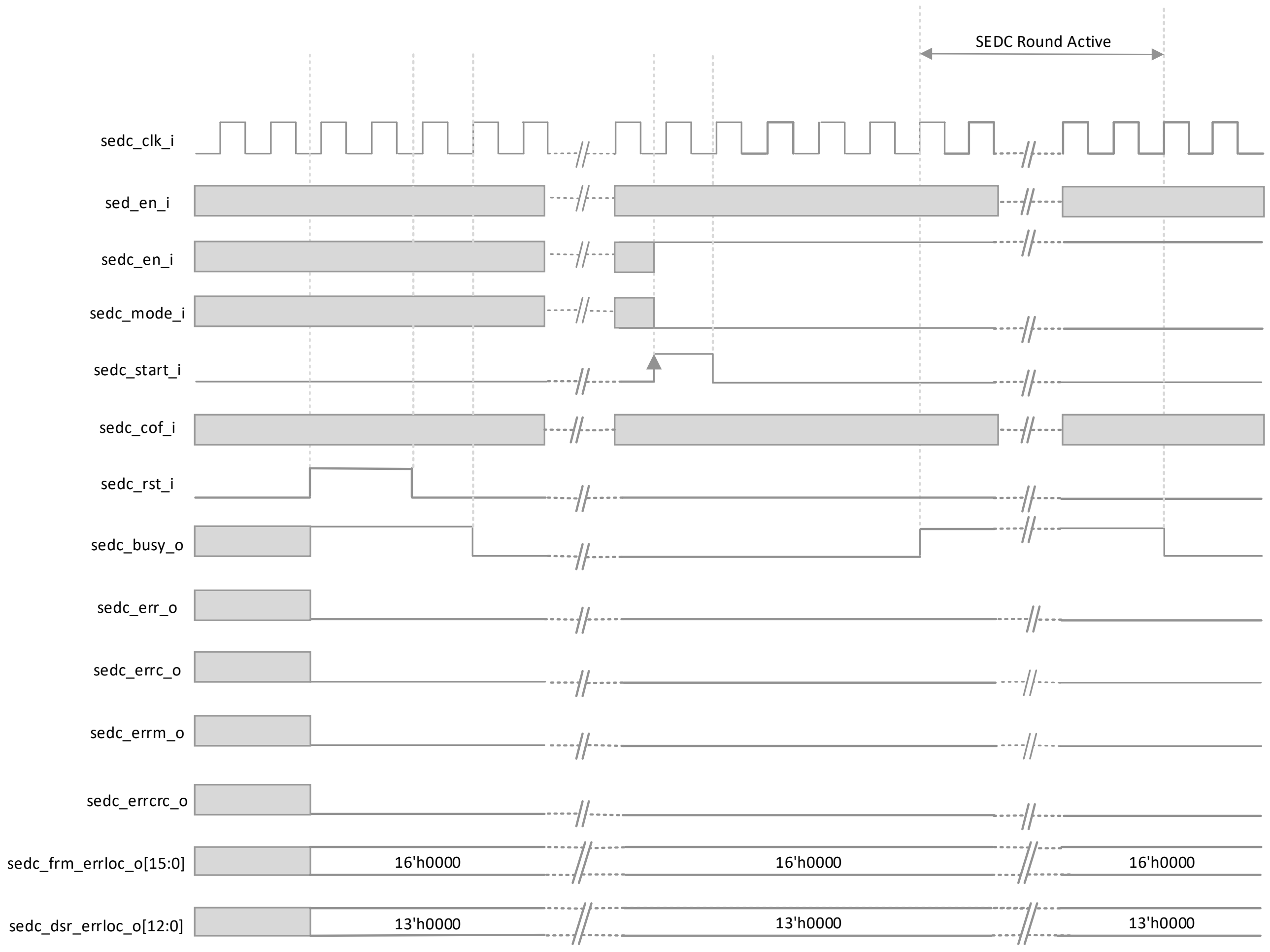


Figure 6.3. SED One-Shot Mode

The preferred action to take when an error is detected is to reconfigure the PLD. Reconfiguration can be accomplished by driving the PROGRAMN pin low. This can be done by externally connecting a GPIO pin to PROGRAMN.

6.2. SEDC Error Handling

The figures in this section show the different types of errors reported by the SEDC module in different mode and error condition by Nexus platform devices. The *sedc_errc_o* signal flags as soon as there is an error. The *sedc_err_o*, and *sedc_errm_o* are sticky flags and the SEDC module has to be restarted to reset these error flags.

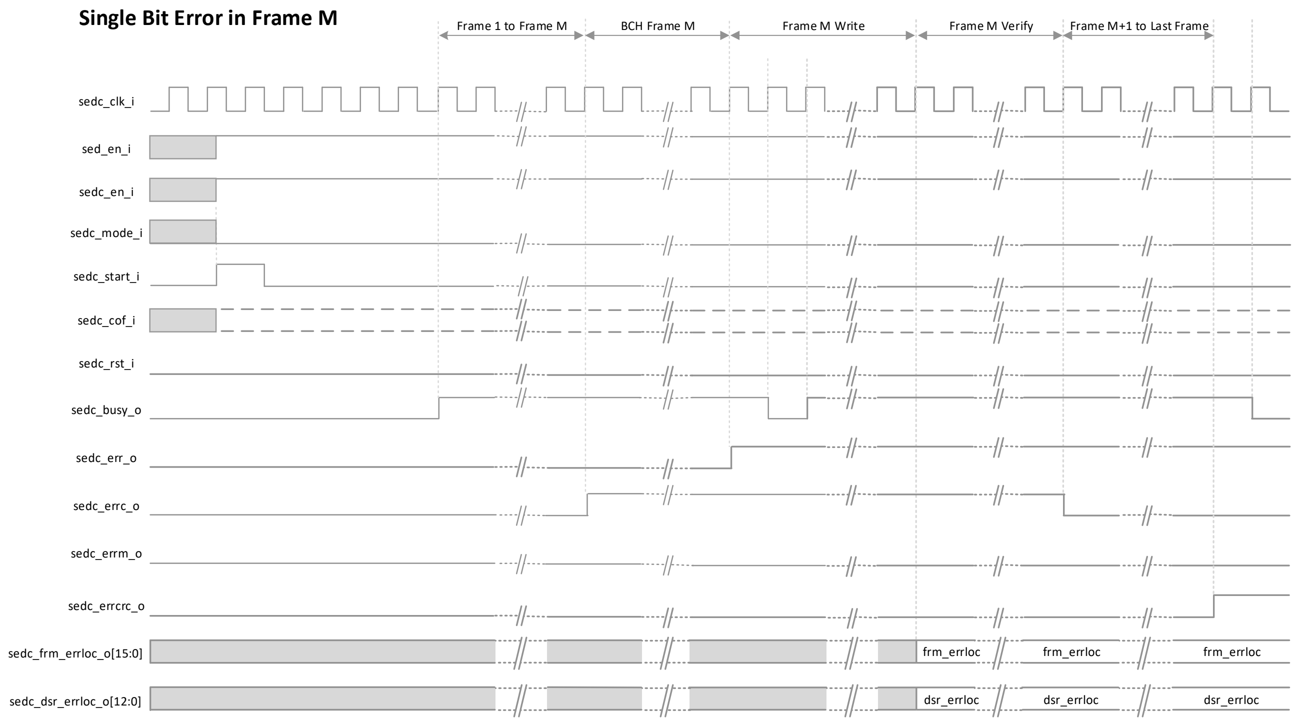


Figure 6.4. One-shot Mode with Correctable Error (sedc_cof_i = 0/1)

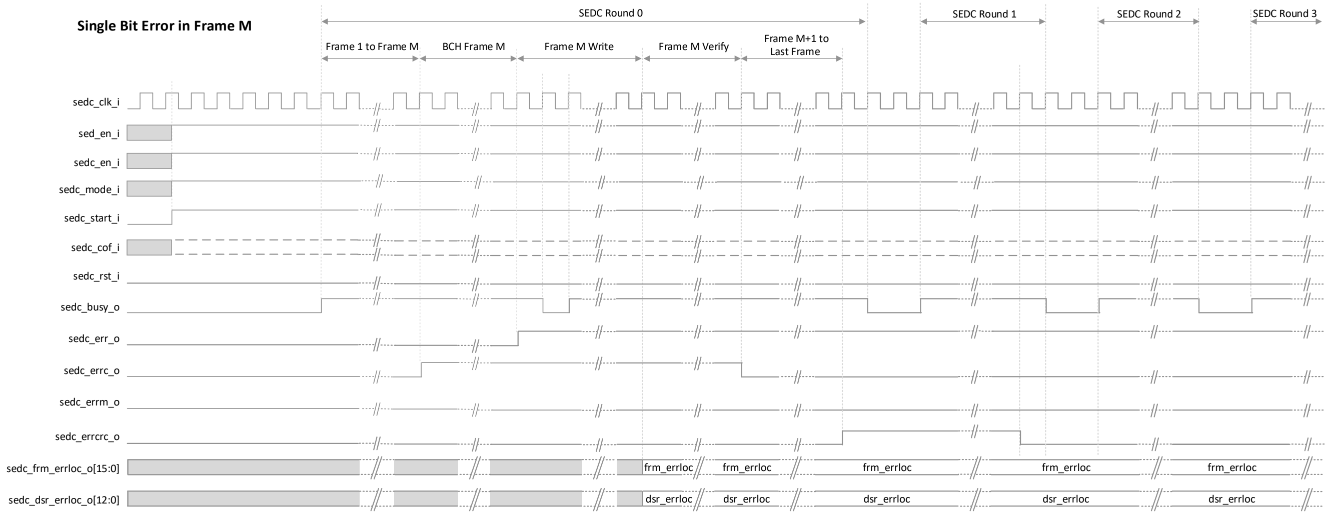


Figure 6.5. SEDC Continuous Mode with Correctable Error (sedc_cof_i = 0/1)

Note: Soft Error Correction only occurs if sed_en_i (soft error correction enable) is set. If sed_en_i is not set, sedc_busy_o remains asserted high until all frames are checked, regardless of error state as shown in Figure 6.4 and Figure 6.5.

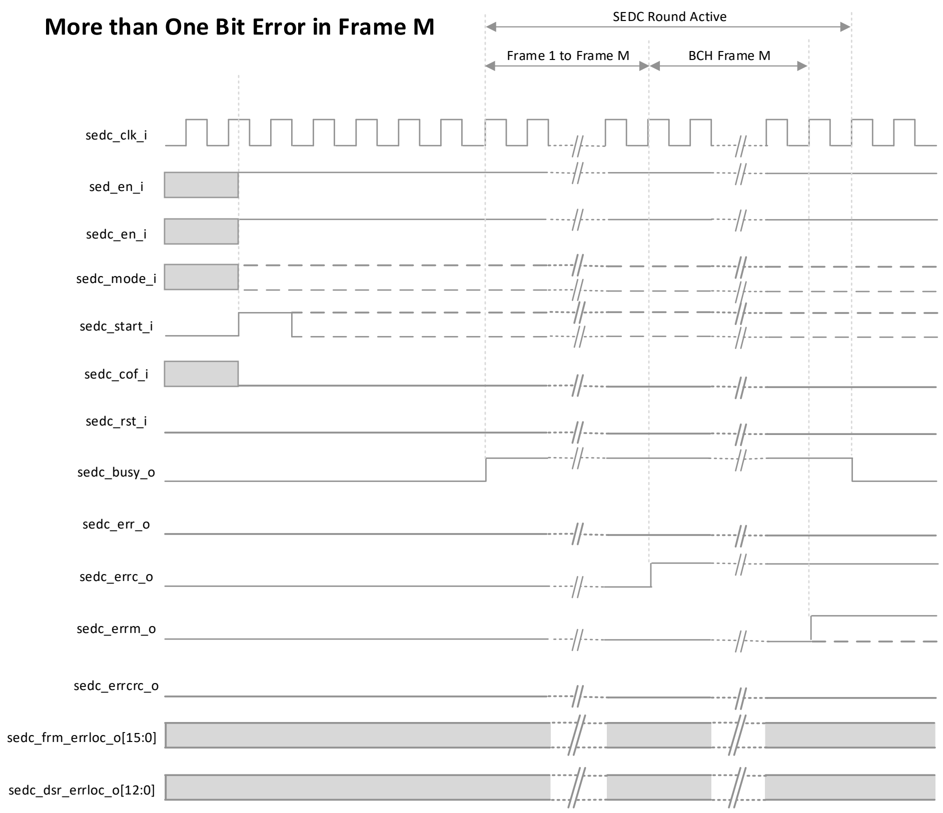


Figure 6.6. SEDC One-shot Mode or Continuous Mode with Multiple Error in One Frame (sedc_cof_i = 0)

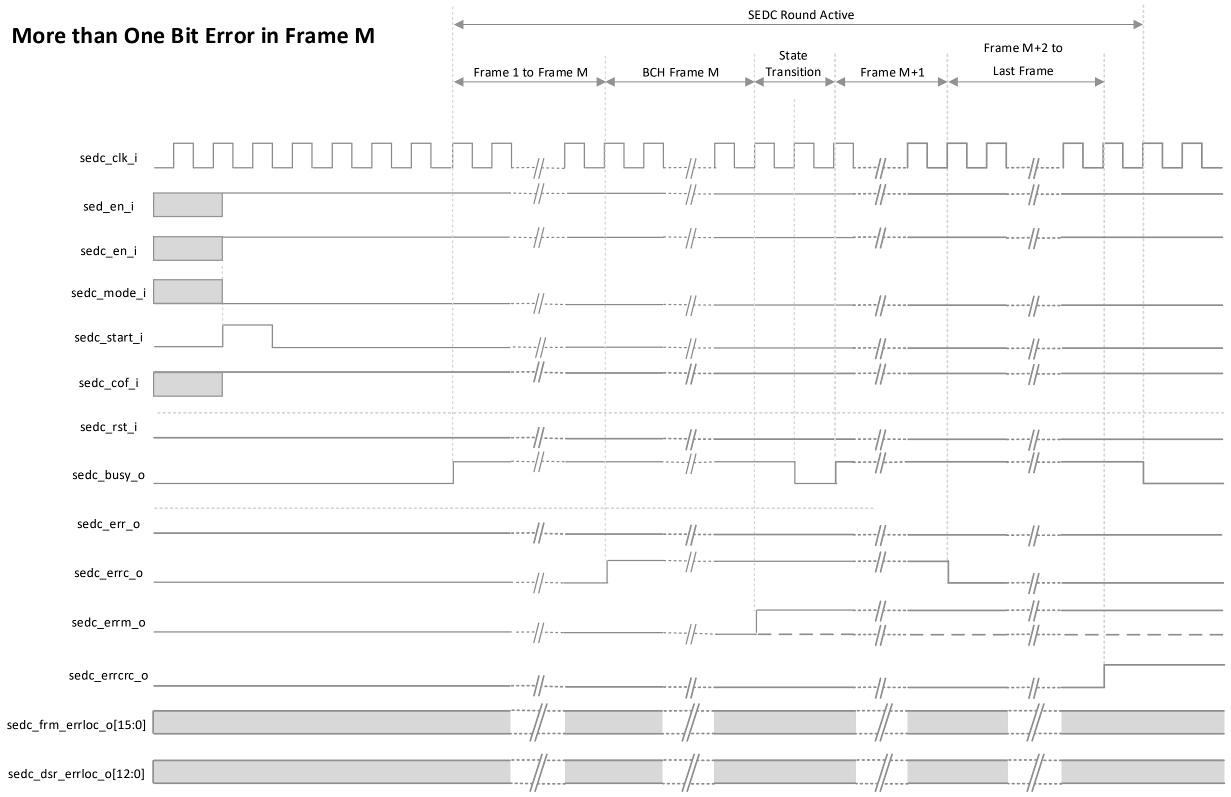


Figure 6.7. SEDC One-shot Mode with Multiple Error in One Frame (sedc_cof_i = 1)

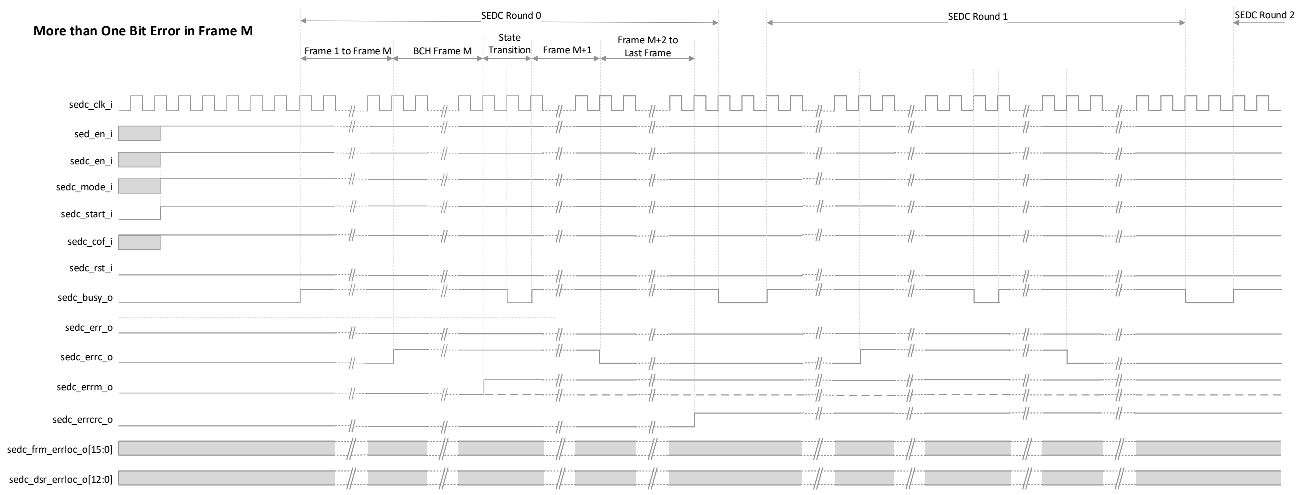


Figure 6.8. SEDC Continuous Mode with Multiple Error in One Frame (sedc_cof_i = 1)

In case of multiple errors in one frame and cannot be detected by the ECC logic, or the CRC checksum is not programmed in bitstream correctly, the sedc_errcrc_o is set in indicate the CRC checksum mismatch, as shown in Figure 6.9 and Figure 6.10.

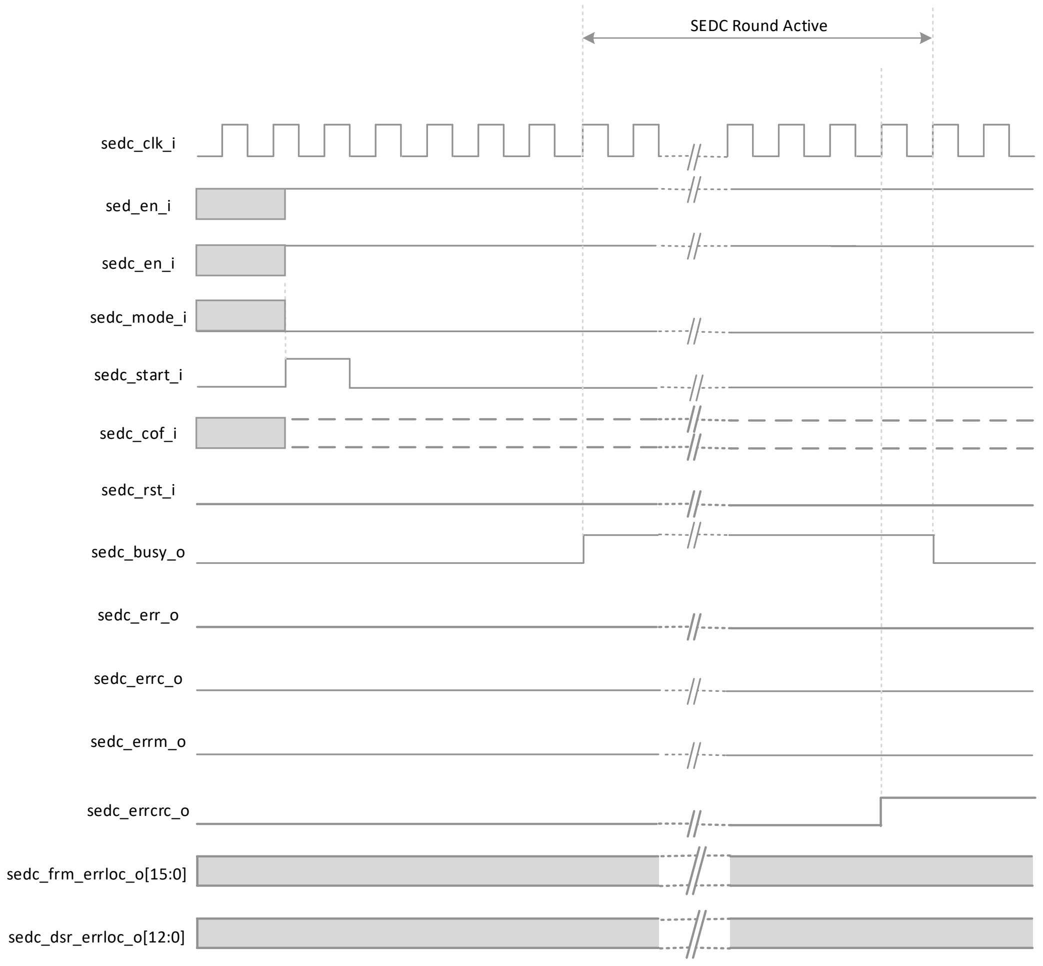


Figure 6.9. SEDC One-shot Mode with CRC Error (sedc_cof_i = 0/1)

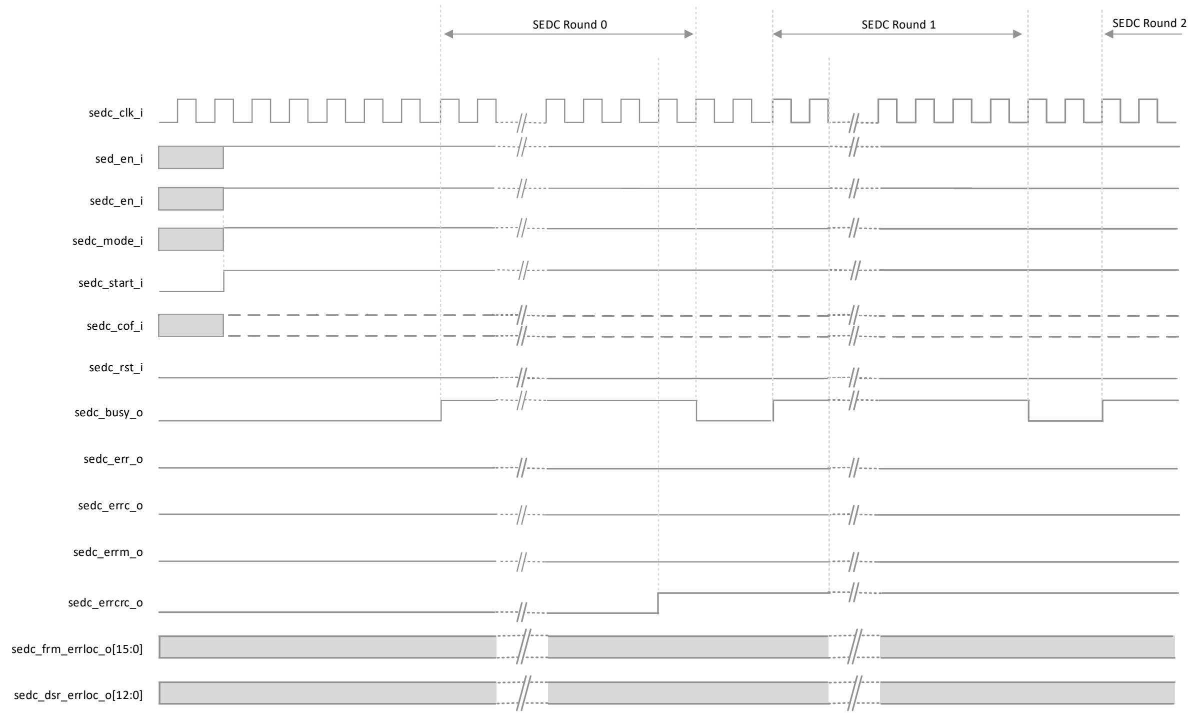


Figure 6.10. SEDC Continuous Mode with CRC Error (sedc_cof_i = 0/1)

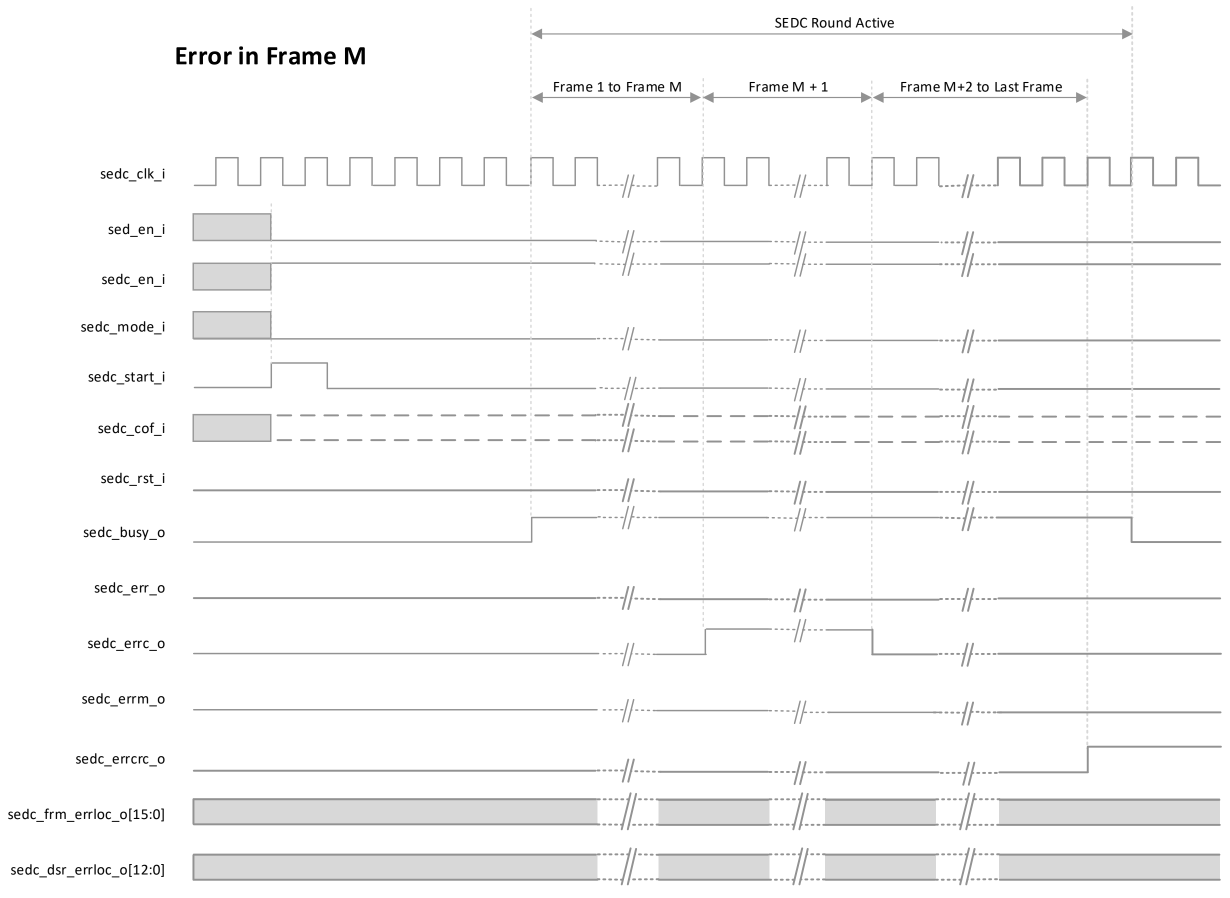


Figure 6.11. SEDC One-shot Mode with SEC Disabled (sed_en_i = 0)

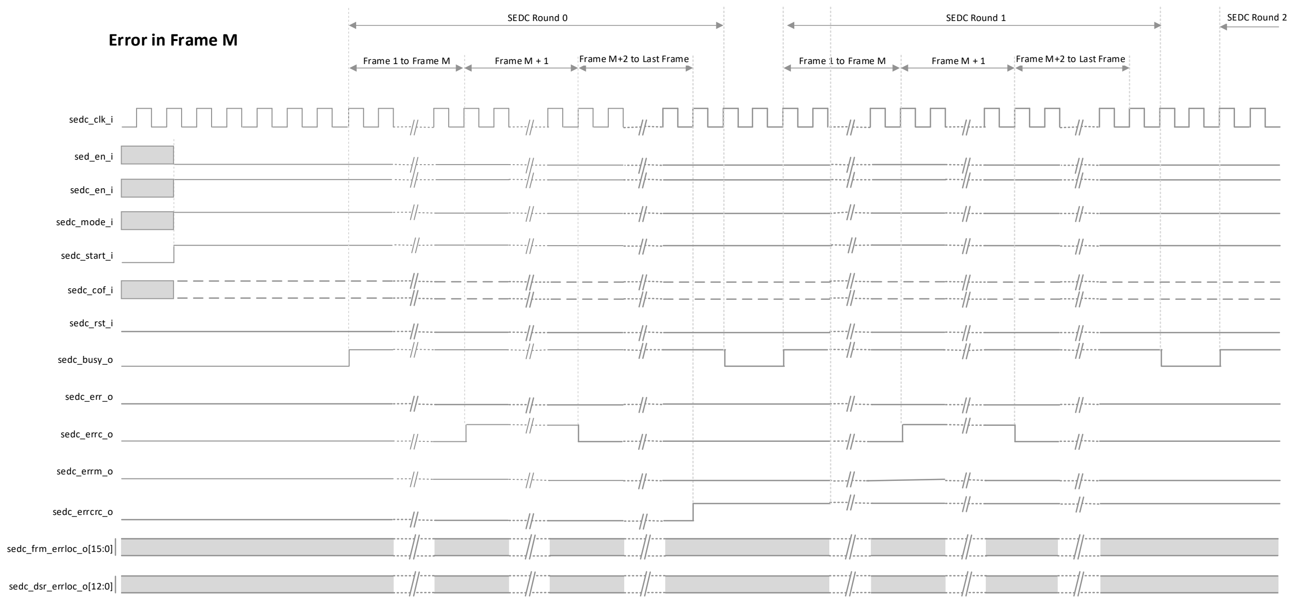


Figure 6.12. SEDC Continuous Mode with SEC Disabled (sed_en_i = 0)

7. SEDC Run Time

In the Nexus platform devices, the amount of time needed to perform a SED check depends on the density of the device, frequency of the SED clock driver signal and the number of shift lanes used to shift data into the device. There is also some overhead time for calculation, but it is short in comparison. An approximation of the SED run time can be found by using the following formula:

For LIFCL-17 and LFD2NX-17 devices:

Read Cycles per frame = 93

SED run time (ms) = (Total no of frames × Read Cycles per frame) / SED clock (MHz)

For LIFCL-40, LFD2NX-40, LFCPNX-50, LFCPNX-100, and LFMXO5-25 devices:

SED run time (ms) = (Total no of frames × (bytes per frame + overhead bytes)) / SED clock (MHz)

For example, for LIFCL-17 devices, the run time for SED clock running at 150 MHz is:

Total no. of frame = 7900

SED Run time = (7900 × 93) / 150 MHz = 4.9 ms

For example, for LIFCL-40 devices, the run time for SED clock running at 150 MHz is:

Total no. of frame = 9172

Bytes per frame = 85

Overhead bytes = 5

SED Run time = (9172 × (85 + 5)) / 150 MHz = (9172 × 90) / 150 MHz = 5.5 ms

Table 7.1. Configuration-related Parameters that Affect SED Run Time

Device	Total Number of Frames	Bytes per Frame	Overhead Bytes
LIFCL-17	7900	44	N/A
LIFCL-33	5344	127	5
LIFCL-40	9172	85	5
LFD2NX-9	7900	44	N/A
LFD2NX-17	7900	44	N/A
LFD2NX-28	9172	85	5
LFD2NX-40	9172	85	5
LFCPNX-50	16822	98	5
LFCPNX-100	16822	112	5
LFMXO5-25	6622	85	5

8. Sample Code

The following simple example code shows how to instantiate the SEDC primitive. Note that the SEDC IP can be created using the IP catalog in Lattice Radiant software version 2.0 or later. Refer to the Important Note section for additional design considerations once the SEDC primitive is instantiated.

8.1. SEDC Verilog Example

8.1.1. Verilog Example of SEDC IP

```
module sed_test (sed_en_i, sedc_cof_i, sedc_en_i, sedc_mode_i, sedc_start_i, cfg_clk_i, sedc_rst_i,
sedc_busy_o, sedc_err_o, sedc_errc_o, sedc_errcrc_o, sedc_errm_o, sedc_frm_errloc_o,
sedc_dsr_errloc_o)/* synthesis syn_black_box syn_declare_black_box=1 */;

    input  sed_en_i;
    input  sedc_cof_i;
    input  sedc_en_i;
    input  sedc_mode_i;
    input  sedc_start_i;
    input  cfg_clk_i;
    input  sedc_rst_i;
    output  sedc_busy_o;
    output  sedc_err_o;
    output  sedc_errc_o;
    output  sedc_errcrc_o;
    output  sedc_errm_o;
    output  [15:0]  sedc_frm_errloc_o;
    output  [12:0]  sedc_dsr_errloc_o;

endmodule
```

8.1.2. Verilog SEDC IP Instantiation

```
sed_test sed_module_name      (
        .sed_en_i(sed_enable),
        .sedc_cof_i(sedc_cof),
        .sedc_en_i(sedc_en),
        .sedc_mode_i(sedc_mode),
        .sedc_start_i(sedc_start),
        .cfg_clk_i(sedc_clk ),
        .sedc_rst_i(sedc_rst),
        .sedc_busy_o(sedc_busy),
        .sedc_err_o(sedc_err1),
        .sedc_errc_o(sedc_err_current),
        .sedc_errcrc_o(sedc_errcrc),
        .sedc_errm_o(sedc_errm),
        .sedc_frm_errloc_o(sedc_frm_loc),
        .sedc_dsr_errloc_o(sedc_bit_err_loc)
);
```

8.2. SEDC VHDL Example

8.2.1. VHDL Component Instantiation

```
component sed_test is
    port(
        sed_en_i: in std_logic;
        sedc_cof_i: in std_logic;
        sedc_en_i: in std_logic;
        sedc_mode_i: in std_logic;
        sedc_start_i: in std_logic;
        cfg_clk_i: in std_logic;
        sedc_rst_i: in std_logic;
        sedc_busy_o: out std_logic;
        sedc_err_o: out std_logic;
        sedc_errc_o: out std_logic;
        sedc_errcrc_o: out std_logic;
        sedc_errm_o: out std_logic;
        sedc_frm_errloc_o: out std_logic_vector(15 downto 0);
        sedc_dsr_errloc_o: out std_logic_vector(12 downto 0)
    );
end component;
```

8.2.2. VHDL Instantiation

```
SED_instance: sed_test port map(
    sed_en_i=> sed_enable,
    sedc_cof_i=> sed_cof,
    sedc_en_i=> sedc_en,
    sedc_mode_i=> sedc_mode ,
    sedc_start_i=> sedc_start,
    cfg_clk_i=> sedc_clk,
    sedc_rst_i=> sedc_rst,
    sedc_busy_o=> sedc_busy,
    sedc_err_o=> sedc_err1,
    sedc_errc_o=> sedc_err_current,
    sedc_errcrc_o=> sedc_errcrc,
    sedc_errm_o=> sedc_errm,
    sedc_frm_errloc_o=> sedc_frm_loc,
    sedc_dsr_errloc_o=> sedc_bit_err_loc );
```

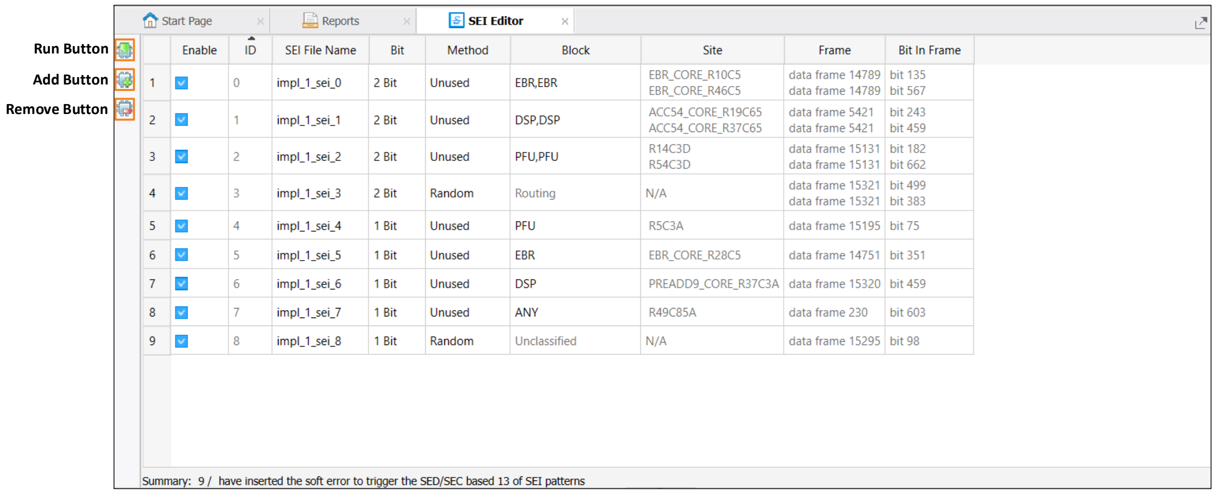
9. Soft Error Injection (SEI)

The Radiant SEI tool offers an easy and economical way to emulate soft error impact to the overall system. This tool allows you to randomly generate and program one or multiple soft errors into the device in background mode without disturbing the device function. Radiant SEI tool is supported in Radiant versions 2.1 and higher.

Note: The SEI feature does not support MachXO5-NX RoT device due to security reason. Any access to the MachXO5-NX RoT device SRAM is not allowed.

To use the Radiant SEI tool:

1. Select or enable the **BACKGROUND_RECONFIG** option in the Global setting under Device Constraint Editor when you generate the bitstream.
2. Run the SEI Editor (Figure 9.1) under Tools. This allows you to create one frame-special bitstream that has one bit or two bits different from the original bitstream.



	Enable	ID	SEI File Name	Bit	Method	Block	Site	Frame	Bit In Frame
1	✓	0	impl_1_sei_0	2 Bit	Unused	EBR,EBR	EBR_CORE_R10C5 EBR_CORE_R46C5	data frame 14789 data frame 14789	bit 135 bit 567
2	✓	1	impl_1_sei_1	2 Bit	Unused	DSP,DSP	ACC54_CORE_R19C65 ACC54_CORE_R37C65	data frame 5421 data frame 5421	bit 243 bit 459
3	✓	2	impl_1_sei_2	2 Bit	Unused	PFU,PFU	R14C3D R54C3D	data frame 15131 data frame 15131	bit 182 bit 662
4	✓	3	impl_1_sei_3	2 Bit	Random	Routing	N/A	data frame 15321 data frame 15321	bit 499 bit 383
5	✓	4	impl_1_sei_4	1 Bit	Unused	PFU	R5C3A	data frame 15195	bit 75
6	✓	5	impl_1_sei_5	1 Bit	Unused	EBR	EBR_CORE_R28C5	data frame 14751	bit 351
7	✓	6	impl_1_sei_6	1 Bit	Unused	DSP	PREADD9_CORE_R37C3A	data frame 15320	bit 459
8	✓	7	impl_1_sei_7	1 Bit	Unused	ANY	R49C85A	data frame 230	bit 603
9	✓	8	impl_1_sei_8	1 Bit	Random	Unclassified	N/A	data frame 15295	bit 98

Summary: 9 / have inserted the soft error to trigger the SED/SEC based 13 of SEI patterns

Figure 9.1. SEI Editor

Some of the main menus in the SEI Editor are described below:

- **Enable** – Select the checkbox of the specific bitstream to be generated when you click the **Run** button.
  Note: The generated bit stream is either Binary Bit File or ASCII Raw Bit File. It follows the Bit Stream Output Format setting in Lattice Radiant software. The default setting is in Binary Bit File.
- **ID** – Continuous error ID number assigned by the software. This value is read-only.
- **SEI File Name** – Default generated bitstream file name. This may be changed by the user.
- **Bit**
  - 1 Bit, single-bit error injection within one data frame.
  - 2 Bit, two-bit error injection within one data frame.
- **Method**
  - Unused – Error bit is introduced into an unused block which does not affect customer design logic.
  - Random – Error bit is introduced into a random block which may or may not affect customer design logic.
- **Block**
  - If **Bit** is set to 1 Bit and **Method** is set to Unused, **Block** can be selected among PFU, EBR, DSP, or ANY.
    Note: Selecting EBR inserts an error into its configuration bits which are bits in CRAM for EBR block settings. The EBR data field is not modified and not covered by the SED/SEC circuit.
  - If **Bit** is set to 2 Bit and **Method** is set to Unused, **Block** can be selected among the following options:
    PFU, PFU;
    EBR, EBR;
    DSP, DSP;
    EBR, PFU;

DSP, PFU.
Lower density devices with 30K logic cells or below have fewer options. The valid options are:
PFU, PFU;
EBR, PFU;
DSP, PFU.

- If method is set to Random, **Block** can be any functional block including a routing resource. In this situation, **Block** cannot be selected by the user.

- **Site** – The exact Site of the inserted error. Errors inserted into a routing block or unclassified block do not have a site. This field is read-only.
- **Frame** – The data frame of the inserted error. This field is read-only.
- **Bit in Frame** – This displays the address of the SEI frame.

Notes:

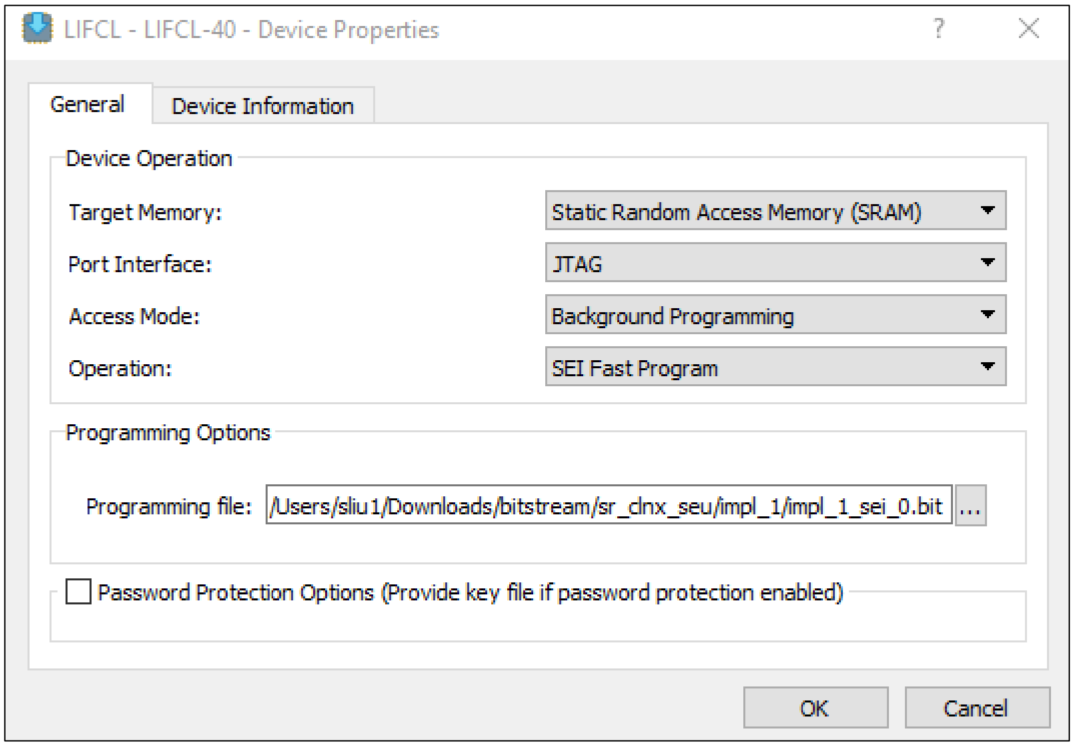
- The grey areas, including **Site**, **Frame**, and **Bit in Frame**, cannot be selected.
- Once SEI bit stream generation is completed, the grey areas report or display site name, data frame number, and bit number where the soft error is injected in.
- There is no site location when soft error injection is in a routing or unclassified block.

The major buttons on the SEI Editor are described below.

- **Add** Button – Click this button to add SEI files.
- **Remove** Button – Click this button to remove SEI files.
- **Run** Button – Click this button to generate SEI files.
  Note: This generates a data frame with flipped random bits, producing partial bitstream output in binary .bit or ASCII .rbt file formats. Encrypted partial bitstream is not supported. The encrypted bitstream implementing authentication feature will be supported in future Lattice Radiant versions.

3. To program the SEI bitstream, select the **SEI Fast Program** operation in the Radiant Programmer.
4. Once it is programmed into the device, you can use the general SED routine to detect the soft error.

   Note: While programming the device with soft error bitstream, SED checking must stop. You can deassert sedc_en_i to stop the SED checking.


LIFCL - LIFCL-40 - Device Properties

Device Operation

Target Memory: Static Random Access Memory (SRAM)
Port Interface: JTAG
Access Mode: Background Programming
Operation: SEI Fast Program

Programming Options

Programming file: /Users/sliu1/Downloads/bitstream/sr_clnx_seu/impl_1/impl_1_sei_0.bit

☐ Password Protection Options (Provide key file if password protection enabled)

Figure 9.2. Device Properties

10. Important Note

When the SEDC primitive is instantiated in a user design, you must follow below requirements to ensure the PROGRAMN[1] pin and REFRESH[1] command behave as expected for FPGA reconfiguration:

- Reset the boot address register, using one of the following[2]:
  - If using Radiant 3.1 or later, generate the bitstream as usual. No further action is required.
  - If using Radiant 3.0 or earlier, perform one of the following:
    - Manually set the Bulk Erase Enable One-Time Programmable (OTP) bit using the **Programmer Tool > Device Properties > Operation: Program Control NV Register1** in Radiant 3.0 or earlier. This must be done to every target device, but no special bitstream modification is required.
    - Manually set the Bulk Erase Enable register setting in Control Register 1 (CR1:16) in the bitstream file using the **Programmer Tool > Deployment Tool or Programmer Tool > Programming File Utility > Tools > Control Register1 Editor** in Radiant 3.0 or earlier. This must be done on every generated bitstream, but no extra programming step for the device is required.
- The SEDC primitive must be held in reset: the *sedc_rst_n_i* port of OSC/OSC for CRE IP must be asserted low a) before or while asserting PROGRAMN low, or b) before issuing REFRESH command, or c) before executing software reboot API in MachXO5-NX RoT device using one of the following methods[2].
  - External to the device, tie PROGRAMN to a GPIO pin which is routed to the *sedc_rst_n_i* port of the OSC/OSC for CRE IP.
  - Route the *sedc_rst_n_i* port of OSC/OSC for CRE IP to external pin which can be asserted independently of PROGRAMN, if desired.
  - Create user logic which can communicate with an external controller that asserts PROGRAMN, and asserts *sedc_rst_n_i* of OSC/OSC for CRE IP prior to PROGRAMN assertion.
  - If you implement Embedded Security and Function Block (ESFB) IP in MachXO5-NX RoT devices, take the following procedure. Instantiate the GPIO IP in Lattice Propel Builder to connect to *sedc_rst_n_i* of the OSC for CRE IP. Export the OSC CRE IP *cfg_clk_o* and *sedc_rst_o* pins in Lattice Propel Builder to connect them to SEDC IP *cfg_clk_i* and *sedc_rst_i* pins respectively in your RTL design. Execute the API of GPIO IP to assert *sedc_rst_n_i* before executing the ESFB software reboot API. Figure 10.1 shows the connection between GPIO IP, OSC for CRE IP, and SEDC IP.

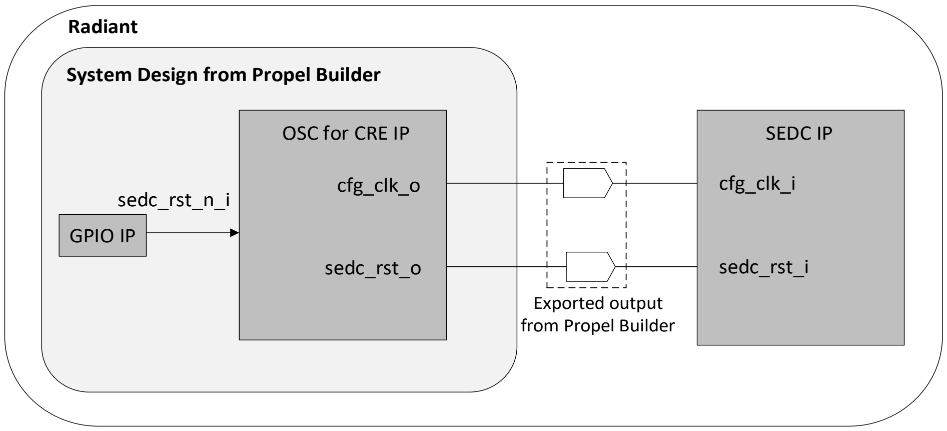


Figure 10.1. GPIO IP to Assert sedc_rst_n_i Before Executing Software Reboot API

- If you are using Dual Boot, Ping-Pong Boot or Multi-Boot, all the images need to consistently implement SEDC feature. Mixing the images with and without implementing SEDC feature can cause reconfiguration failure upon rebooting from one image to another. See Multi-Boot User Guide for Nexus Platform (FPGA-TN-02145) to learn more about Dual Boot, Ping-Pong Boot, and Multi-Boot.
- If Authentication is implemented together with SEDC IP, the SEDC IP must be held in disable, that is, sedc_en_i must be asserted to low, for user to check the Auth Done bit after device configuration is completed and the device enters user mode. Otherwise, once the SEDC engine runs, the Auth Done and std Preamble bits in the status register are cleared to 0. The clearing of these bits does not affect the device functionality negatively.

- The SEDC IP must be disabled to perform any ISC and programming commands via CONFIG_LMMI block, or JTAG, or Slave configuration interface, such as Slave SPI, I2C, or I3C interfaces, in direct/background programming operations. This can be done by the following sequence.
    a. Stop SEDC operation by de-asserting sedc_start_i from high to low.
    b. Wait sedc_busy_o signal to go low.
    c. De-asserts sedc_en_i from high to low.
    d. Perform any ISC and programming commands in background operations.
    e. Restart SEDC operation after ISC and programming command transaction is finished. The background operations may or may not work and might cause the device to exit user mode if you perform ISC and programming commands when SEDC engine is running. It might also cause the device to enter into unrecoverable error state and power cycling of the device is required.
- The SEDC IP operation must be stopped to perform any flash access operations using Flash Access IP. Otherwise, flash access operation does not work as expected.

Notes:

1. See sysCONFIG User Guide for Nexus Platform (FPGA-TN-02099) to learn more about PROGRAMN pin and REFRESH command.
2. If the first two conditions are met, then PROGRAMN and REFRESH operate as expected. If these conditions are not met, the PROGRAMN and REFRESH does not operate as expected which may cause the device to hang and not enter user mode. A power cycle is required to recover the device.

References

- Single Event Upset (SEU) Report for CrossLink-NX (FPGA-TN-02174)
- sysCONFIG User Guide for Nexus Platform (FPGA-TN-02099)
- Multi-Boot User Guide for Nexus Platform (FPGA-TN-02145)
- CrossLink-NX web page
- Certus-NX web page
- CertusPro-NX web page
- MachXO5-NX web page
- Lattice Radiant Software web page
- Lattice Insights web page for Lattice Semiconductor training courses and learning plans

Technical Support Assistance

Submit a technical support case through www.latticesemi.com/techsupport.

For frequently asked questions, refer to the Lattice Answer Database at www.latticesemi.com/Support/AnswerDatabase.

Revision History

Revision 2.1, July 2024

Section	Change Summary
SEDC Run Time	In Table 7.1. Configuration-related Parameters that Affect SED Run Time, added the data for *LFD2NX-9* and *LFD2NX-28* devices.
Soft Error Injection	In step 2 of how to use the Radiant SEI tool, updated the note for Block to *Selecting EBR inserts an error into its configuration bits which are bits in CRAM for EBR block settings. The EBR data field is not modified and not covered by the SED/SEC circuit*.

Revision 2.0, May 2024

Section	Change Summary
Soft Error Injection (SEI)	• Added the following note regarding SEI feature support for MachXO5-NX RoT device: *The SEI feature does not support MachXO5-NX RoT device due to security reason. Any access to the MachXO5-NX RoT device SRAM is not allowed*. • Updated step 2 of using the Lattice Radiant Software SEI tool to*: Run the SEI Editor (Figure 9.1) under Tools. This allows you to create one frame-special bitstream that has one bit or two bits different from the original bitstream.* • Updated Figure 9.1. SEI Editor. • Added the following note to Enable: *The generated bit stream is either Binary Bit File or ASCII Raw Bit File. It follows the Bit Stream Output Format setting in Lattice Radiant software. The default setting is in Binary Bit File.* • Updated the description of ID to: *Continuous error ID number assigned by the software. This value is read-only.* • Updated the description of SEI File Name to: *Default generated bitstream file name. This may be changed by the user.* • Added the description of Bit: • *1 Bit, single-bit error injection within one data frame.* • *2 Bit, two-bit error injection within one data frame.* • Updated the description of Method to: • *Unused – Error bit is introduced into an unused block which does not affect customer design logic.* • *Random – Error bit is introduced into a random block which may or may not affect customer design logic.* • Updated the description of Block to: • *If Bit is set to 1 Bit and Method is set to Unused, Block can be selected among PFU, EBR, DSP, or ANY.* *Note: Selecting EBR inserts an error into its configuration bits. The EBR data field is not modified.* • *If Bit is set to 2 Bit and Method is set to Unused, Block can be selected among the following options:* *PFU, PFU;* *EBR, EBR;* *DSP, DSP;* *EBR, PFU;* *DSP, PFU.* *Lower density devices with 30K logic cells or below have fewer options. The valid options are:* *PFU, PFU;* *EBR, PFU;* *DSP, PFU.* • *If method is set to Random, Block can be any functional block including a routing resource. In this situation, Block cannot be selected by the user.*

Section	Change Summary
	• Added the description of Site: *The exact Site of the inserted error. Errors inserted into a routing block or unclassified block do not have a site. This field is read-only.* • Added the description of Frame: *The data frame of the inserted error. This field is read-only.* • Added the description of Bit in Frame: *This displays the address of the SEI frame.* • Added the following notes for Site, Frame, and Bit in Frame: • *The grey areas, including Site, Frame, and Bit in Frame, cannot be selected.* • *Once SEI bit stream generation is completed, the grey areas report or display site name, data frame number, and bit number where the soft error is injected in.* • *There is no site location when soft error injection is in a routing or unclassified block.* • Added the following note for the Run Button: *This generates a data frame with flipped random bits, producing partial bitstream output in binary .bit or ASCII .rbt file formats. Encrypted partial bitstream is not supported. The encrypted bitstream implementing authentication feature will be supported in future Lattice Radiant versions.* • In step 3, changed *one bit different bitstream* to *SEI bitstream*. • In step 4, removed *During the SEI Fast Program, you can see the DONE pin go low during configuration and back to high after the configuration*.
Important Note	• For the second condition, the SEDC primitive must be held in reset: • added *c) before executing software reboot API in MachXO5-NX RoT device to the second condition*; • replaced *OSC* with *OSC/OSC for CRE IP*; • added the following paragraph: *If you implement Embedded Security and Function Block (ESFB) IP in MachXO5-NX RoT device, take the following procedure. Instantiate the GPIO IP in Lattice Propel Builder to connect to sedc_rst_n_i of the OSC for CRE IP. Export the OSC CRE IP cfg_clk_o and sedc_rst_o pins in Lattice Propel Builder to connect them to SEDC IP cfg_clk_i and sedc_rst_i pins respectively in your RTL design. Execute the API of GPIO IP to assert sedc_rst_n_i before executing the ESFB software reboot API. Figure 10.1 shows the connection between GPIO IP, OSC for CRE IP, and SEDC IP.* • added Figure 10.1. GPIO IP to Assert sedc_rst_n_i Before Executing Software Reboot API. • Added the following paragraph: *If Authentication is implemented together with SEDC IP, the SEDC IP must be held in disable, that is, sedc_en_i must be asserted to low, for user to check the Auth Done bit after device configuration is completed and the device enters user mode. Otherwise, once the SEDC engine runs, the Auth Done and std Preamble bits in the status register are cleared to 0. The clearing of these bits does not affect the device functionality negatively.* • Added the following paragraph: *The SEDC IP must be disabled to perform any ISC and programming commands via CONFIG_LMMI block, or JTAG, or Slave configuration interface, such as Slave SPI, I2C, or I3C interfaces, in direct/background programming operations. This can be done by the following sequence.* *a. Stop SEDC operation by de-asserting sedc_start_i from high to low.* *b. Wait sedc_busy_o signal to go low.* *c. De-asserts sedc_en_i from high to low.* *d. Perform any ISC and programming commands in background operations.* *e. Restart SEDC operation after ISC and programming command transaction is finished. The background operations may or may not work and might cause the device to exit user mode if you perform ISC and programming commands when SEDC engine is running. It might also cause the device entered into unrecoverable error state and power cycling the device is required.*

Section	Change Summary
	• Added the following paragraph: *The SEDC IP operation must be stopped to perform any flash access operations using Flash Access IP. Otherwise, flash access operation does not work as expected.*

Revision 1.9, December 2023

Section	Change Summary
Disclaimers	Updated boilerplate.
Acronyms in This Document	Added acronyms for FIT.
SED Overview	• Updated the content to capitalize FIT. • Updated Figure 2.3.
Port List	• Updated the paragraph by replacing Figure 2.1 with Figure 5.1. • Moved the order of *sedc_rst_i* in Table 3.1.
Port Descriptions	• Moved the order of 4.2 sedc_rst_i section. • Updated the contents of 4.2 sedc_rst_i, 4.3 sedc_mode_i, 4.5 sedc_start_i, 4.6 sedc_en_i, and 4.14 sedc_dsr_errloc_o[12:0] sections.
SED Clock and Reset	Updated the paragraph by adding: *Note that the oscillator IP can be OSC IP or OSC for CRE IP*.
SEDC Flow	• Removed paragraph *Devices built on the Nexus platform support real time Soft Error Correction (SEC) feature in which a single bit error can be corrected using ECC at the frame level. Once the SEC is enabled, the SEDC module reports the error location, providing details about the error frame and the exact location of a single bit error in that frame as shown in Figure 6.1* due to repetition. • Added a new note in section 6.1.1 Continuous Mode and section 6.1.2 One-shot Mode. • Removed sedc_errcrc from the first paragraph of 6.2 SEDC Error Handling. • Updated Figure 6.2 – Figure 6.8, Figure 6.11, and Figure 6.12.
SEDC Run Time	Updated the total number of frames for LFCPNX-50 device from *10444* to *16822* in Table 7.1.
Sample Code	Updated the sample code for 8.1.2 Verilog SEDC IP Instantiation.
Soft Error Injection (SEI)	Updated the content for clarity.
Important Note	• Changed the *sedc_rst_i* to *OSP IP sedc_rst_n_i* • Updated the content for clarity.
References	Updated this section.

Revision 1.8, July 2023

Section	Change Summary
Acronyms in This Document	Added acronyms for BCH.
Port Descriptions	Added description for sedc_errm_o output *when non-correctable error occurs. The flag will be asserted for two bits error (per frame), but it's not guaranteed to be asserted for more than 2 bits error due to the nature of the Hamming coding algorithm* in sedc_errm_o section.
SEDC Flow	• Updated Figure 6.1. SEDC Flow in SEDC Flow Section. • Moved paragraph *Devices built on the Nexus platform support real time Soft Error Correction (SEC) feature in which a single bit error can be corrected using ECC at the frame level. Once the SEC is enabled, the SEDC module reports the error location, providing details about the error frame and the exact location of a single bit error in that frame as shown in Figure 6.1* from previous Section 7 to SEDC Flow Section. • Deleted previous Section 7 SEDC Flow, and Figure 7.1. • Updated Figure 6.3. SED One-Shot Mode to add SEDC Round Active label. • Updated Figure 6.4. One-shot Mode with Correctable Error (sedc_cof_i = 0 / 1) to update the label. • Updated Figure 6.5. SEDC Continuous Mode with Correctable Error (sedc_cof_i = 0 / 1) to update the label. • Updated Figure 6.6. SEDC One-shot Mode or Continuous Mode with Multiple Error in

Section	Change Summary
	One Frame (sedc_cof_i = 0) to change sedc_mode_i for both 1 & 0 and add label. • Updated Figure 6.7. SEDC One-shot Mode with Multiple Error in One Frame (sedc_cof_i = 1) to add label. • Updated Figure 6.8. SEDC Continuous Mode with Multiple Error in One Frame (sedc_cof_i = 1) to correct the sedc_mode_i, sedc_start_i state and update label. • Updated Figure 6.9. SEDC One-shot Mode with CRC Error (sedc_cof_i = 0 / 1) to add label. • Updated Figure 6.10. SEDC Continuous Mode with CRC Error (sedc_cof_i = 0 / 1) to correct the sedc_mode_i, sedc_start_i state.
SEDC Error Handling	• Added Figure 6.11. SEDC One-shot Mode with SEC Disabled (sed_en_i = 0). • Added Figure 6.12. SEDC Continuous Mode with SEC Disabled (sed_en_i = 0).
References	Added Reference section.

Revision 1.7, May 2023

Section	Change Summary
All	Change the module name from SED or SED/SEC to SEDC which is the actual primitive name in Radiant.
SEDC Flow	• Updated Figure 6.2. SED Continuous Mode in Continuous Mode section. • Updated Figure 6.3. SED One-Shot Mode in One-shot Mode section.
SEDC Error Handling	• Added below figures in SEDC Error Handling section: • Figure 6.4. One-shot Mode with Correctable Error (sedc_cof_i = 0 / 1) • Figure 6.5. SEDC Continuous Mode with Correctable Error (sedc_cof_i = 0 / 1) • Figure 6.6. SEDC One-shot Mode or Continuous Mode with Multiple Error in One Frame (sedc_cof_i = 0) • Figure 6.7. SEDC One-shot Mode with Multiple Error in One Frame (sedc_cof_i = 1) • Figure 6.8. SEDC Continuous Mode with Multiple Error in One Frame (sedc_cof_i = 1) • Figure 6.9. SEDC One-shot Mode with CRC Error (sedc_cof_i = 0 / 1) • Figure 6.10. SEDC Continuous Mode with CRC Error (sedc_cof_i = 0 / 1)
Technical Support Assistance	Added link for Lattice FAQ website.

Revision 1.6, October 2022

Section	Change Summary
SEDC Run Time	Updated Table 8.1. Configuration-related Parameters that Affect SED Run Time adding LIFCL 33 device support.

Revision 1.5, March 2022

Section	Change Summary
All	Minor adjustments in formatting across the document.
Introduction	Updated section content to add MachXO5-NX support.
SED Run Time	Updated section content to add LFMXO5-25, including in Table 8.1. Configuration-related Parameters that Affect SED Run Time.

Revision 1.4, February 2022

Section	Change Summary
All	Changed document name from Soft Error Detection (SED)/Correction (SEC) **Usage** Guide for Nexus Platform to Soft Error Detection (SED)/Correction (SEC) **User** Guide for Nexus Platform.
Soft Error Injection	• Updated step 1 to correct the location of the BACKGROUND_RECONFIG option. This option is in the Global setting under Device Constraint Editor when generating the bitstream. • Updated step 3 and corrected the operation (SEI Fast Program operation) to program the one-bit-different bitstream, in the Radiant Programmer.

Revision 1.3, July 2021

Section	Change Summary
SED Run Time	Added Table 7.1 to show configuration information and SED run times for different NX devices.
Important Note	New section to describe special handling of boot address register and 'sedc_lrst_n' port to ensure PROGRAMN and Refresh command works correctly.

Revision 1.2, June 2021

Section	Change Summary
All	Minor adjustments in formatting.
Introduction	Added CertusPro-NX support.

Revision 1.1, June 2020

Section	Change Summary
All	• Changed document name from CrossLink-NX Soft Error Detection/Correction Usage Guide to *Soft Error Detection/Correction Usage Guide for Nexus Platform*. • Added Nexus platform across the document.
Introduction	Updated section content.
SEC Overview	• Updated section content. • Updated Figure 2.2 and Figure 2.3.
Port List	Updated Table 3.1.
Port Descriptions	Updated section content.
SED Clock and Reset	Updated content to add Figure 5.1.
SED Flow	Updated Figure 6.4.
Sample Code	Updated codes in SED Verilog Example and SED VHDL Example.
Soft Error Injection (SEI)	Added this section.

Revision 1.0, February 2020

Section	Change Summary
All	Initial release