Lattice Avant SEDC Controller Reference Design

Reference Design

FPGA-RD-02340-1.0

April 2026

Contents

Figures

Tables

Abbreviations in This Document

A list of abbreviations and abbreviations used in this document.

Abbreviations	Definition
CRAM	Configuration Random Access Memory
CRC	Cyclic Redundancy Check
FPGA	Field Programmable Gate Array
FSM	Finite State Machine
GUI	Graphical User Interface
HDL	Hardware Description Language
I/O	Input/Output
IP	Intellectual Property
LED	Light Emitting Diode
LMMI	Lattice Memory Mapped Interface
OEM	Original Equipment Manufacturer
OSC	Oscillator
RTL	Register Transfer Level
SEC	Soft Error Correction
SED	Soft Error Detection
SEDC	Soft Error Detection/Correction
SEI	Soft Error Injection
SRAM	Static Random Access Memory
TE	Trigger Expression
USB	Universal Serial Bus

1. Introduction

This reference design showcases the functionality of soft error detection (SED) and soft error correction (SEC) supported in the Lattice Avant™ device families. The reference design implements a state machine controller that interfaces with the SEDC Controller IP to enable the soft error detection/correction (SEDC) function, which can be used to detect and correct static random access memory (SRAM) errors.

For more information on the SEDC feature in Lattice Avant devices, refer to the Lattice Avant SED/SEC User Guide (FPGA-TN-02290) and SEDC Controller IP User Guide (FPGA-IPUG-02290).

1.1. Quick Facts

Download the reference design files from the Lattice reference design web page.

Table 1.1. Summary of the Reference Design

General	Target Devices	LAV-AT-X70
	Source Code Format	Verilog
Simulation	Functional Simulation	Performed
	Timing simulation	Not performed
	Test Bench	Available
	Test Bench Format	Verilog
Software Requirements	Software Tool and Version	Lattice Radiant™ Software 2025.2
	IP Version	OSC Module v2.1.0 SEDC Controller IP v1.1.0
Hardware Requirements	Board	Avant-X Versa board
	Cable	USB-A to Mini-B programming cable

1.2. Features

Key features of the Avant SEDC Controller reference design include:

- Ability to perform SEDC in continuous mode
- Ability to detect and flag single-bit and multi-bit errors when they occur
- Ability to correct 1-bit soft errors
- Ability to insert 1-bit or 2-bit soft errors through the soft error injection (SEI) tool
- Ability to measure the SED time
- Ability to measure the SEC time

1.3. Naming Conventions

1.3.1. Nomenclature

The nomenclature used in this document is based on Verilog HDL.

1.3.2. Signal Names

Signal names that end with:

- _n are active low signals (asserted when value is logic 0)
- _i are input signals
- _o are output signals

2. Directory Structure and Files

Figure 2.1 shows the directory structure.

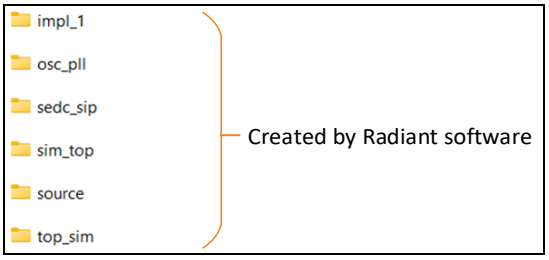

Figure 2.1. Directory Structure

The Avant SEDC Controller reference design includes the bitstream files, instantiated intellectual property (IP) files, register transfer level (RTL) source codes, and RTL simulation files.

Table 2.1 shows the list of directories included in the reference design package.

Table 2.1. Directory List

Directory	Description
impl_1	Contains the bitstream (.bit) files.
osc_pll	Contains the generated Oscillator Module package, which includes files as shown in Table 2.2.
sedc_sip	Contains the generated SEDC Controller IP package, which includes files as shown in Table 2.2.
source	Contains the RTL source codes.
top_sim	Contains the RTL simulation files.

Table 2.2 shows the list of files included in the reference design package.

Table 2.2. File List

Attribute	Description
<Component name>.ipx	Contains the information on the files associated with the generated IP.
<Component name>.cfg	Contains the parameter values used in the IP configuration.
component.xml	Contains the ipxact: component information of the IP.
design.xml	Documents the configuration parameters of the IP in IP-XACT 2014 format.
rtl/<Component name>.v	Provides an example RTL top file that instantiates the module.
rtl/<Component name>_bb.v	Provides the synthesis closed box.
misc/<Component name>_tmpl.v misc /<Component name>_tmpl.vhd	Provide instance templates for the module.

3. Functional Description

Figure 3.1 shows the top-level block diagram of the reference design.

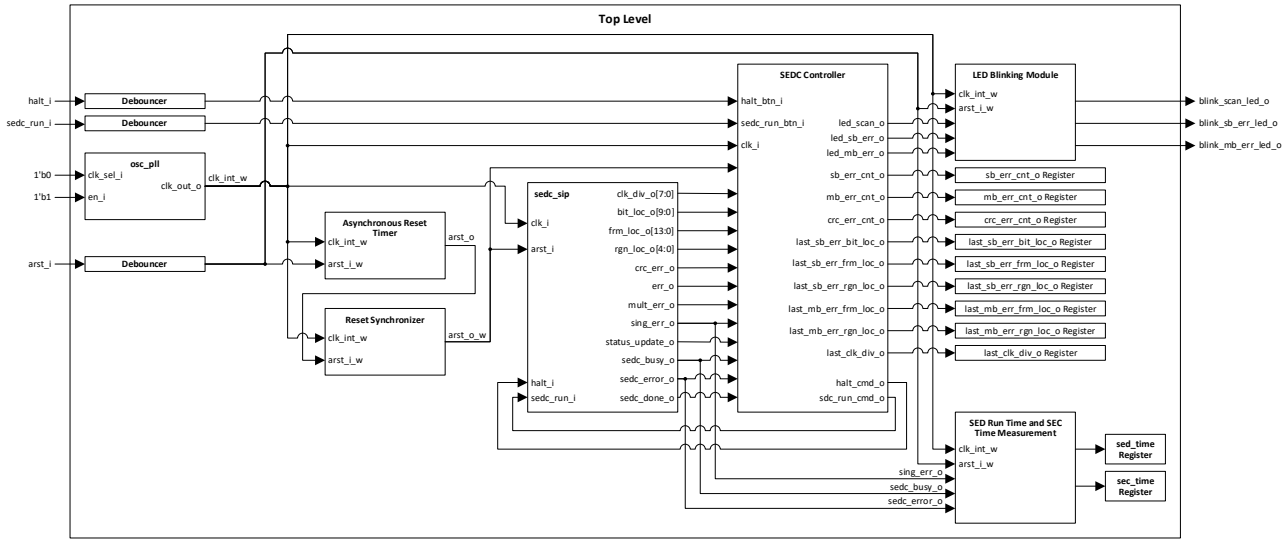

Figure 3.1. Reference Design Block Diagram

3.1. Design Components

The Avant SEDC Controller reference design includes the following blocks:

- osc_pll: An oscillator (OSC) module that generates the clock source (clk_int_w) for all other modules or blocks. The clk_int_w signal runs at 80 MHz. Refer to the Lattice Avant OSC Module User Guide (FPGA-IPUG-02184) for more information.
- Debouncer: A counter-based digital signal debouncer designed to eliminate noise and fluctuations (also known as bouncing) that occur in a digital signal when a mechanical switch or push button is pressed or released. It provides a clean and stable output signal by ensuring the input remains stable for a debounce period (required stable time) before the output is updated. The debounce period is configurable through the following parameter:
  - DEBOUNCE_COUNT (default: 800000): Counter threshold that determines the debounce period.
    To achieve a debounce period of 10 ms (recommended for most mechanical switches), set DEBOUNCE_COUNT as follows according to clock frequency:
    - 80 MHz clock: DEBOUNCE_COUNT = 800000
    - 40 MHz clock: DEBOUNCE_COUNT = 400000
    - 20 MHz clock: DEBOUNCE_COUNT = 200000
    - 10 MHz clock: DEBOUNCE_COUNT = 100000
- Asynchronous Reset Timer: The logic in this block enforces timing requirements for an asynchronous reset input (arst_i_w) and drives a controlled reset output (arst_o) using the system clock (clk_int_w). It guarantees an initial reset hold time after power-up and a minimum reset assertion duration on subsequent resets. Key parameters are as follows:
  - CLK_FREQ_HZ: Clock frequency (for example, 80 MHz).
  - INITIAL_RESET_DELAY_US: Duration to hold reset active, in microseconds, after startup.
  - MIN_RESET_ASSERTION_NS: The minimum time reset must stay asserted (active), in nanoseconds, on later events.
- Reset Synchronizer: The logic in this block converts an asynchronous, active-high reset (arst_o) into a clock-synchronous reset (arst_o_w) using a two-flip-flop synchronizer. arst_o_w asserts on asynchronous reset and deasserts synchronously after two clock cycles, thereby reducing metastability risk and ensuring a clean release of downstream logic.

- sedc_sip: The SEDC Controller IP for soft error detection and soft error correction function. Figure 3.2 shows the configuration of the SEDC Controller IP as follows:
  - SEDC Mode: Continuous Mode
  - Error Correction Mode: Auto-Correction Mode
  - Clock Divider: 2
  - Clock Frequency (MHz): 80

  For more information, refer to the SEDC Controller IP User Guide (FPGA-IPUG-02290).

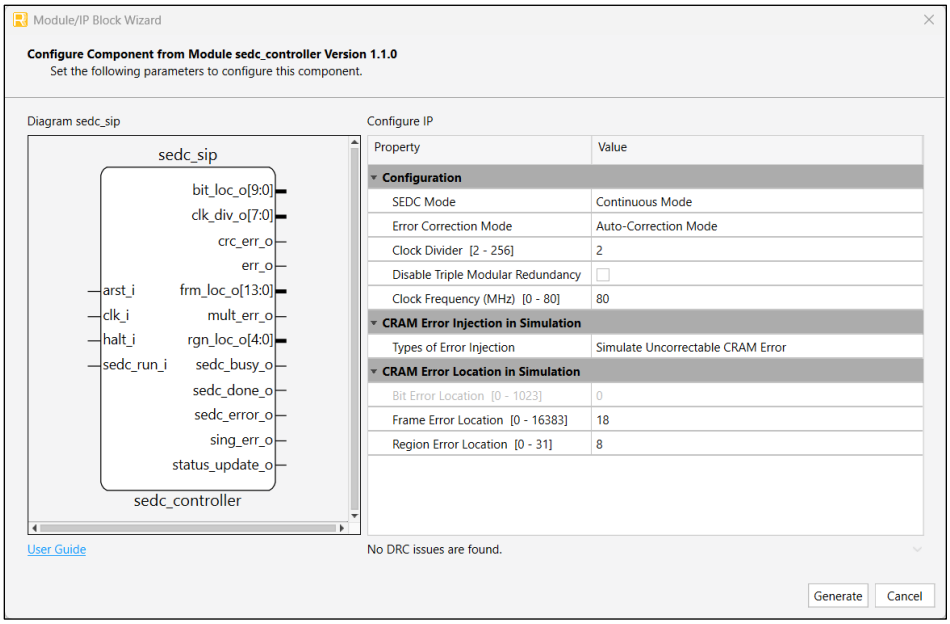

Figure 3.2. SEDC Controller IP Configuration

- LED Blinking Module: This module implements a counter that controls the blinking of the light emitting diodes (LEDs) on the board.
  - LED D45 blinks when an SEDC scan starts.
  - LED D46 blinks when a 1-bit error is detected.
  - LED D46 and D47 blink when a 2-bit or non-correctable error is detected.
- SED Run Time and SEC Time Measurement: This block measures the duration, in clock cycles, of two events driven by status signals:
  - SED runtime: The duration for which sedc_busy_o stays high. SEDC runtime measurement is obtained as follows:
    - While sedc_busy_o is high, sedc_count increments on each clock cycle.
    - When sedc_busy_o transitions from 1 to 0, the current value of sedc_count is latched into sed_time, and sedc_count is reset to 0.
  - SEC time: The duration for which the error condition, defined by sing_err_o and sedc_error_o being high, remains active. SEC time measurement is obtained as follows:
    - While the error condition, defined by sing_err_o and sedc_error_o being high, remains active, sec_count increments on each clock cycle.
    - When sedc_error_o transitions from 1 to 0, the current value of sec_count is latched into sec_time, and sec_count is reset to 0.
- Output Registers: All output registers are cleared only when asynchronous reset (arst_i) is asserted.
  - sed_time: 32-bit SED counter. This counter stores the number of clock cycles that sedc_busy_o was high in the most recent SED run.
  - sec_time: 32-bit SEC counter. This counter stores the number of clock cycles that the error condition, defined by sing_err_o and sedc_error_o being high, remained active in the most recent SEC event.

- sb_err_cnt_o: 16-bit single-bit error counter. This counter increments by 1 on each status_update_o when both err_o and sing_err_o are high (indicating a single-bit error).
- mb_err_cnt_o: 16-bit multi-bit error counter. This counter increments by 1 on each status_update_o when both err_o and mult_err_o are high (indicating a multi-bit error).
- crc_err_cnt_o: 16-bit cyclic redundancy check (CRC) error counter. This counter increments by 1 on each status_update_o when err_o and crc_err_o are high (indicating a CRC mismatch).
- last_sb_err_bit_loc_o: This register captures the bit location (bit_loc_o) of the most recent single-bit error upon status_update_o when both err_o and sing_err_o are high.
- last_sb_err_frm_loc_o: This register captures the frame location (frm_loc_o) of the most recent single-bit error upon a qualifying status_update_o.
- last_sb_err_rgn_loc_o: This register captures the region location (rgn_loc_o) of the most recent single-bit error upon a qualifying status_update_o.
- last_mb_err_frm_loc_o: This register captures the frame location (frm_loc_o) of the most recent multi-bit error upon a qualifying status_update_o when both err_o and mult_err_o are high.
- last_mb_err_rgn_loc_o: This register captures the region location (rgn_loc_o) of the most recent multi-bit error upon a qualifying status_update_o when both err_o and mult_err_o are high.
- last_clk_div_o: This register latches the current clock divider value (clk_div_o) during scanning to provide visibility into the divider setting used for the latest scan window.

- SEDC Controller: This block implements a state machine controller to interact with the SEDC Controller IP and external switches (push buttons) to perform SED and SEC functions. The following describes the finite state machine (FSM) behavior and state transitions as shown in Figure 3.3:
  - Reset and Idle (S_IDLE): Upon arst_i assertion, the FSM initializes to the S_IDLE state. The output commands sedc_run_cmd_o and halt_cmd_o, which controls sedc_run_i (internal) and halt_i (internal) of the SEDC Controller IP, are deasserted. When the SEDC Controller IP is not busy (sedc_busy_o == 0), the falling edge of the signal generated by pressing the sedc_run_i button starts the SEDC Controller FSM sequence.
  - Start (S_START): The FSM asserts sedc_run_cmd_o to start or continue IP operation. It remains in the S_START state until the IP indicates it is busy (sedc_busy_o == 1), then transitions to the S_SCAN state.
  - Scan (S_SCAN): Scanning is performed continuously while sedc_run_cmd_o == 1. The current SEDC clock divider is latched (last_clk_div_o = clk_div_o). The following are exit conditions for the S_SCAN state:
    - If halt is requested (falling edge of the signal generated by pressing the halt_i button), the FSM transitions to the S_HALT state.
    - If an error is observed indicated by the sticky error flag err_seen_q == 1, the FSM transitions to the S_ERR state. The error flag err_seen_q is set by either of the following:
      - sedc_error_o == 1
      - Error condition where status_update_o == 1, err_o == 1, and at least one of sing_err_o, mult_err_o, or crc_err_o is 1
    - If the scan completes without errors (sedc_busy_o == 0 and sedc_done_o == 1), the FSM transitions to the S_DONE state.

    Note that led_scan_o tracks sedc_busy_o. When led_scan_o == 1, LED D45 on the board blinks.
  - Error (S_ERR): The FSM performs error handling while keeping the run command asserted (sedc_run_cmd_o == 1). On each status_update_o, the following counters and last-location registers are updated as applicable:
    - Single-bit error: sb_err_cnt_o and last_sb_err_*
    - Multi-bit error: mb_err_cnt_o and last_mb_err_*
    - CRC error: crc_err_cnt_o

    The following are exit conditions for the S_ERR state:
    - If halt is requested (falling edge of the signal generated by pressing the halt_i button), the FSM transitions to the S_HALT state.
    - If the error is cleared while scanning continues (sedc_busy_o == 1 and sedc_error_o == 0), the FSM transitions back to the S_SCAN state.
    - When the error is fully accounted for (sedc_busy_o == 0 and sedc_done_o == 1), the FSM transitions to the S_DONE state.

    The sticky error flag err_seen_q clears only when the FSM exits error handling (halt_i (internal) == 1 or sedc_error_o == 0 or sedc_done_o == 1). The LED D46 blinks for 1-bit errors when the value of the counter

sb_err_cnt_o or crc_err_cnt_o is greater than zero. The LEDs D46 and D47 blink for 2-bit (or non-correctable) errors when the value of the counter mb_err_cnt_o is greater than zero.

- Halt (S_HALT): The FSM asserts halt_cmd_o and deasserts sedc_run_cmd_o. It remains in the S_HALT state until the IP is not busy (sedc_busy_o == 0), then returns to the S_Idle state.
- Done (S_DONE): This is a transient state after a scan is completed. The FSM keeps sedc_run_cmd_o asserted to support continuous mode. The following are exit conditions for the S_DONE state:
  - If halt is requested (falling edge of the signal generated by pressing the halt_i button), the FSM transitions to the S_HALT state.
  - Otherwise, the FSM transitions to the S_WAIT_BUSY state.
- Wait Busy (S_WAIT_BUSY): This state facilitates staging of the continuous mode. The FSM keeps sedc_run_cmd_o == 1 and waits for sedc_busy_o to reassert before transitioning to the S_SCAN state. Halt can be requested at any time to transition the FSM to the S_HALT state.

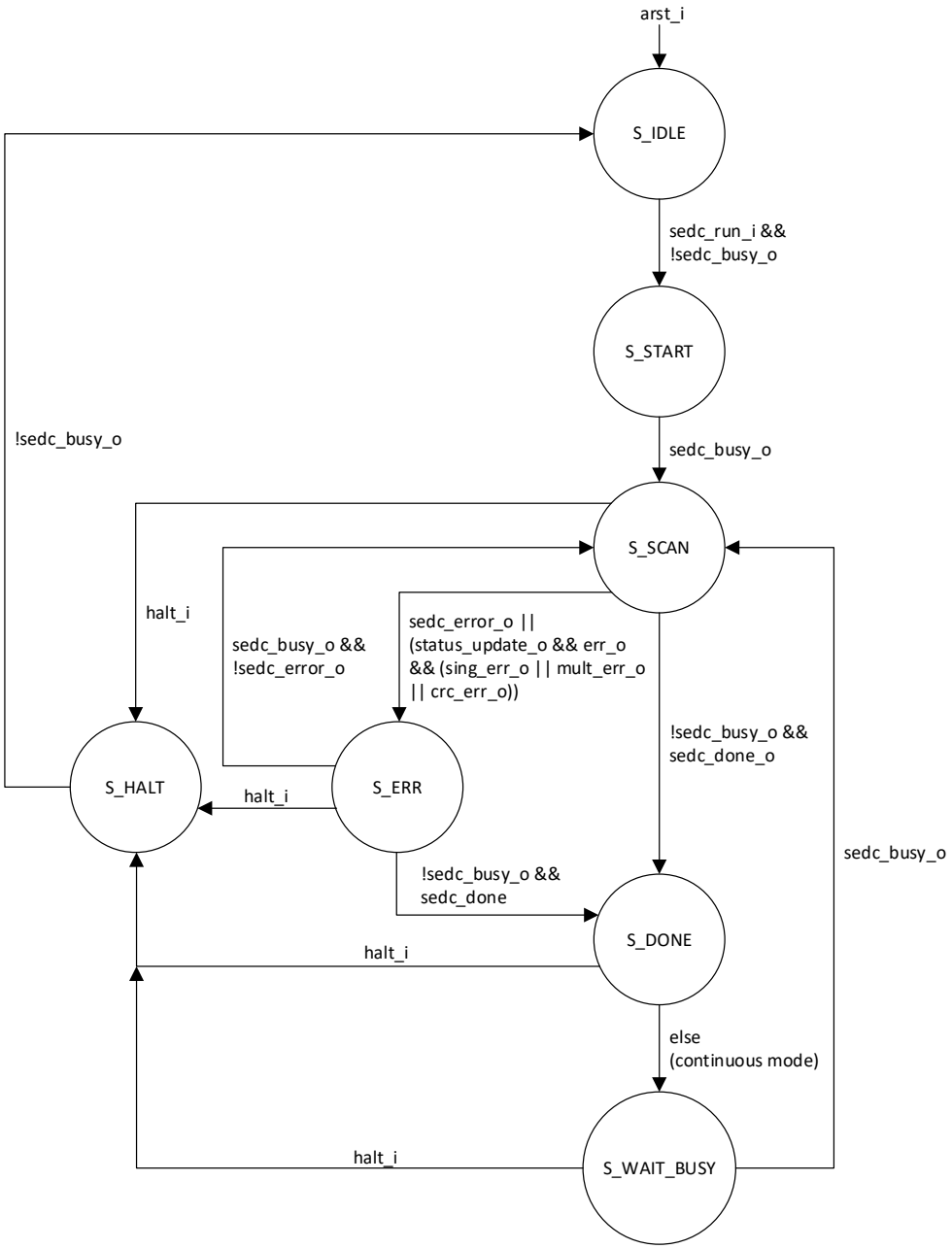

Figure 3.3. SEDC Controller FSM State Transition Diagram

4. Signal Description

The input/output interface signals for the top-level unit are shown in Table 4.1.

Table 4.1. Primary I/O

Port Name	I/O	Width	Description
arst_i	In	1	Clears the FSM state, counters, and status flags. arst_i is an active-low asynchronous reset input. Asserting arst_i low generates an internal active-high asynchronous reset for the SEDC Controller IP and SEDC Controller block. Driven by push button SW12. **Note:** The active-high asynchronous reset signal to the SEDC Controller IP and SEDC Controller block shall be referred to as *arst_i (internal)*, where appropriate, to differentiate it from this top-level input arst_i signal.
halt_i	In	1	Halts the current SEDC operation. Upon detection of the falling edge, the FSM transitions to the S_HALT state and asserts halt_cmd_o internally. Driven by push button SW13 with a normally high signal. **Note:** The active-high signal to the SEDC Controller IP shall be referred to as *halt_i (internal)*, where appropriate, to differentiate it from this top-level input halt_i signal.
sedc_run_i	In	1	Starts scanning. Upon detection of the falling edge, the FSM transitions from the S_IDLE state to the S_START state. The FSM subsequently transitions to the S_SCAN state when the SEDC Controller IP becomes busy. Driven by push button SW15 with a normally high signal. **Note:** The active-high signal to the SEDC Controller IP shall be referred to as the *sedc_run_i (internal)*, where appropriate, to differentiate it from this top-level input sedc_run_i signal.
blink_scan_led_o	Out	1	Blinks LED D45 when scanning is active (sedc_busy_o == 1).
blink_sb_err_led_o	Out	1	Blinks LED D46 when single-bit errors, CRC errors, or multi-bit errors are detected.
blink_mb_err_led_o	Out	1	Blinks LED D47 when multi-bit errors are detected.

5. Running the Reference Design

This section describes how to run the Avant SEDC Controller reference design using the Lattice Radiant software. For more details on the Lattice Radiant software, refer to the Lattice Radiant Software User Guide.

5.1. Compiling the Reference Design

The reference design file includes pre-compiled files and bitstreams (.bit files) to facilitate a quick start. However, you can recompile the reference design, after adding or modifying the user logic as desired, through the standard compilation flow.

5.2. Generating the Bitstream File

This section provides the procedure of creating your FPGA bitstream file using the Lattice Radiant software.
To create the FPGA bitstream file, follow the steps below.

1. Open the Lattice Radiant software, as shown in Figure 5.1.

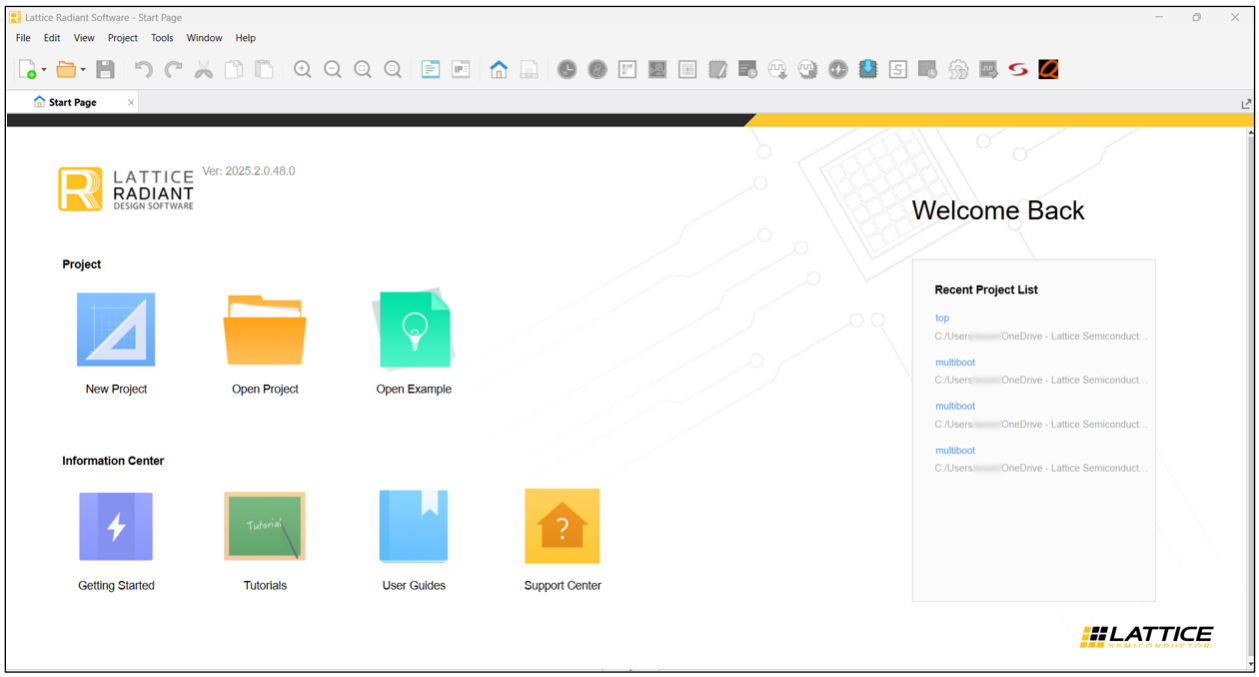

Figure 5.1. Lattice Radiant Software

2. Click **File > Open Project**. From the project database, navigate to the folder of the reference design project (for example, *sedc/25.2/top)* and open the Lattice Radiant project file *top.rdf* as shown in Figure 5.2.

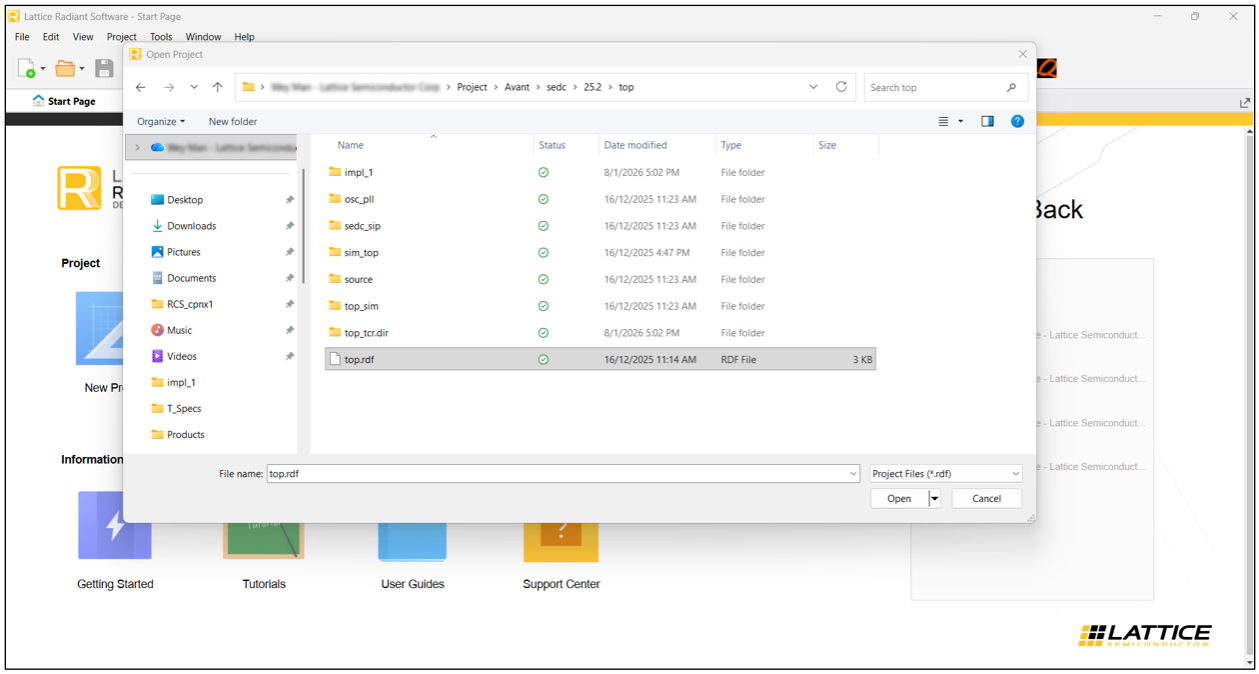

Figure 5.2. Open Project File

3. Click **Export Files** to generate the bit file. View the log message in the Export Reports folder for the generated bitstream, as shown in Figure 5.3. Once compilation is successful, the generated bit file can be found in the project implementation folder (for example, *top\impl_1\top_impl_1.bit*).

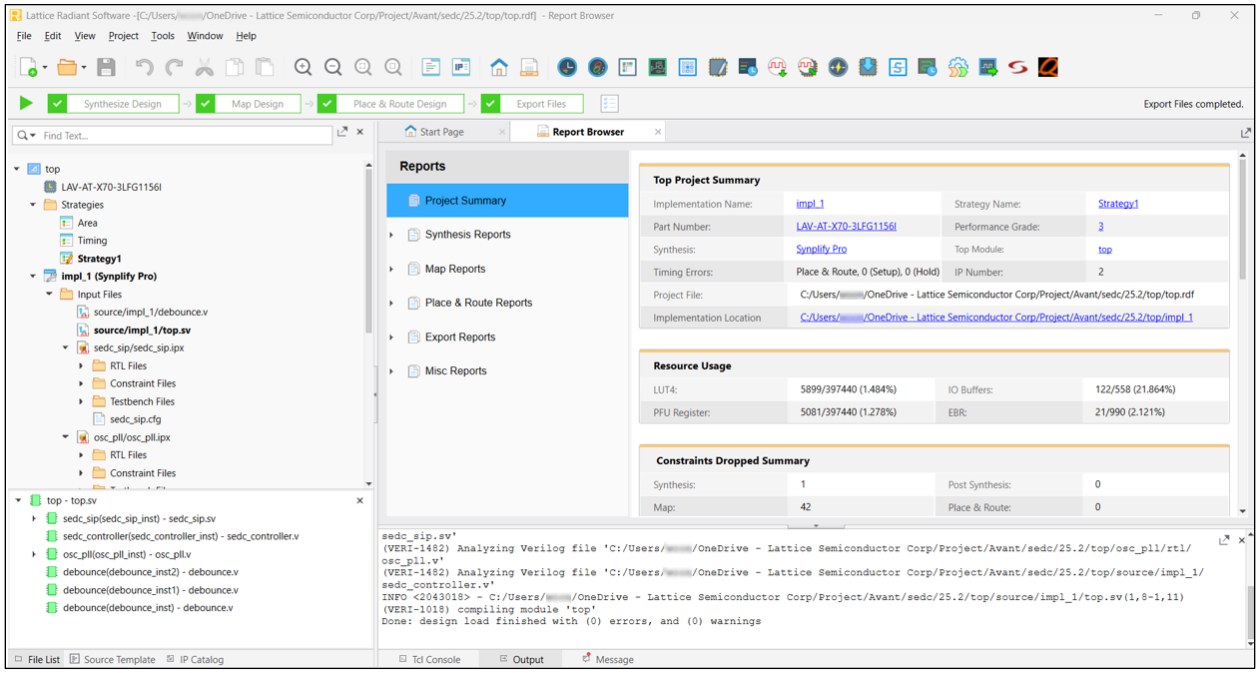

Figure 5.3. Generated Bitstream Log

6. Simulating the Reference Design

To simulate the design, perform the following steps:

1. Open the reference design project file (top.rdf) in the Lattice Radiant software, if not already open. The reference design file includes the pre-created simulation folder *sim_top* and the test bench *top_tf.v*. The pre-created simulation setup, selectable through the SEDC Controller IP graphical user interface (GUI), uses the **Simulate Uncorrectable CRAM Error** option, as shown in Figure 6.1. You can select other simulation options and regenerate the IP to enable the selected simulation option. You can also modify the initial test bench file top_tf.v if desired.

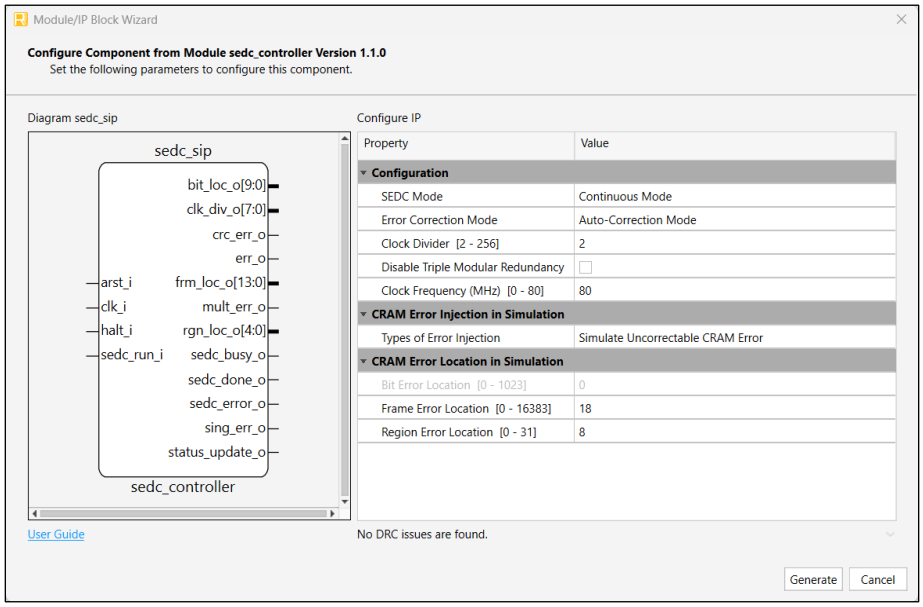

Figure 6.1. Simulation Setup in SEDC Controller IP GUI

2. Select **Tools > QuestaSim Lattice-Edition** to launch QuestaSim.
3. After the QuestaSim Lattice OEM Edition software loads, in the **Transcript** window:
   - Type *cd sim_top* to enter the *sim_top* directory.
   - Then, type *qrun -f sim_top.f* to launch the simulation.

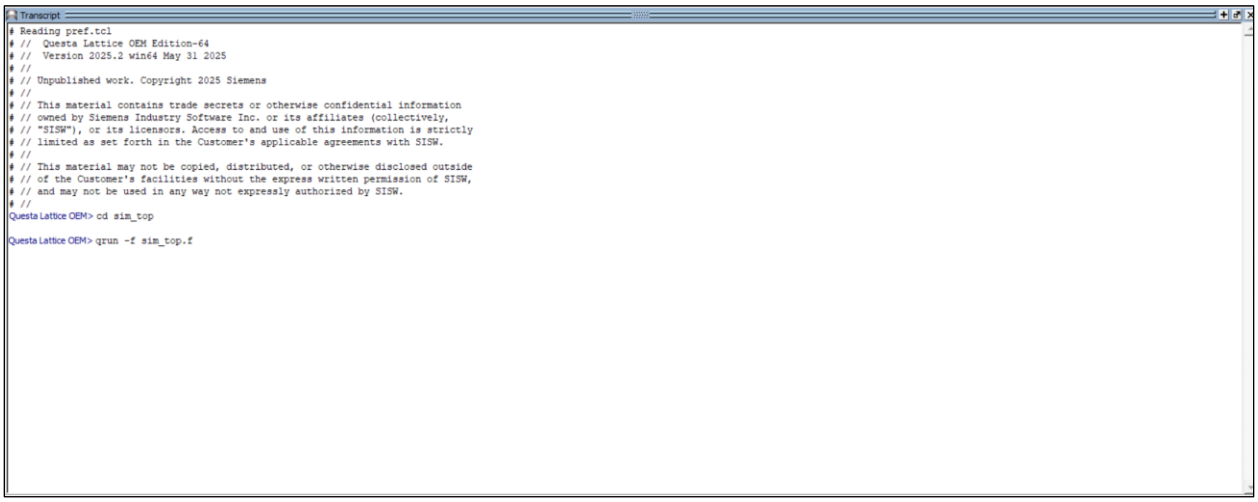

Figure 6.2. QuestaSim Interface – Launching the Simulation

4. Wait for the simulation to complete to see waveforms. Simulated waveforms will be visible in the **Wave** window.

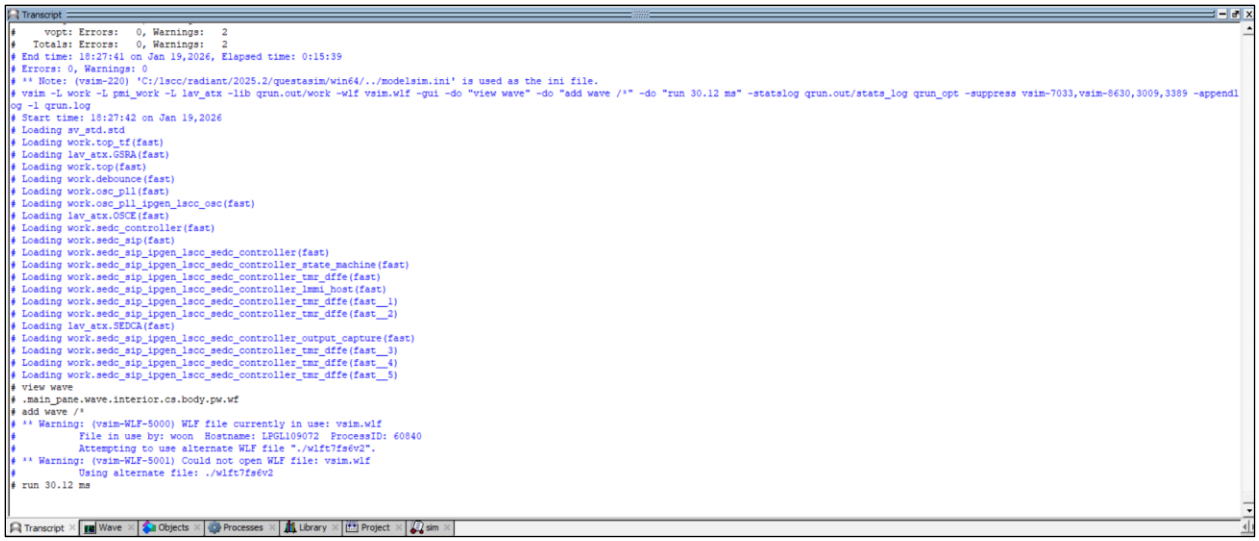

Figure 6.3. QuestaSim Interface – Simulation Completed

6.1. Simulation Results

The example simulation waveforms described in this section are based on the LAT-AT-X70-1LFG1156C device and the IP configuration as follows:

- SEDC Mode: Continuous Mode
- Error Correction Mode: Auto-Correction Mode
- Types of Error Injection: Simulate Uncorrectable CRAM Error
- Frame Error Location: 18
- Region Error Location: 8

Note: The following sections describe multi-bit error detection in continuous mode.

6.1.1. Start Scan

The following are characteristics of starting SEDC scan:

- Pressing the asynchronous reset push button (SW12) toggles the arst_i signal (asserted low and deasserted high) to generate an active-high pulse to reset the SEDC Controller IP and SEDC Controller block.
- After reset, pressing push button SW15 generates a pulse on sedc_run_i. The falling edge of sedc_run_i starts the SEDC scan.

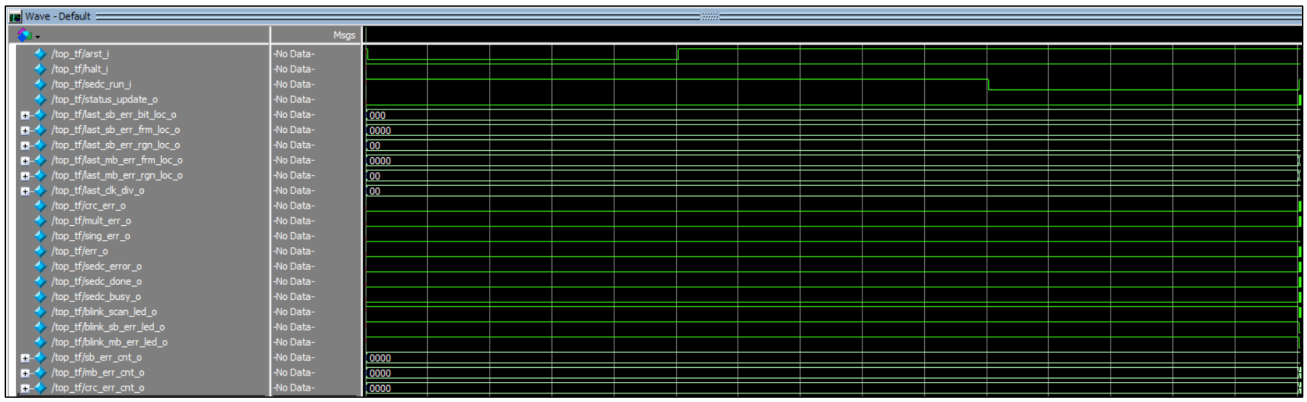

Figure 6.4. Simulation Waveforms – Asynchronous Reset and Triggering of SEDC Scan

- sedc_busy_o asserts high to indicate that scanning has started.
- blink_scan_led_o uses the value of blink_cnt[26] to blink LED D45 to indicate that scanning has started and is in progress.

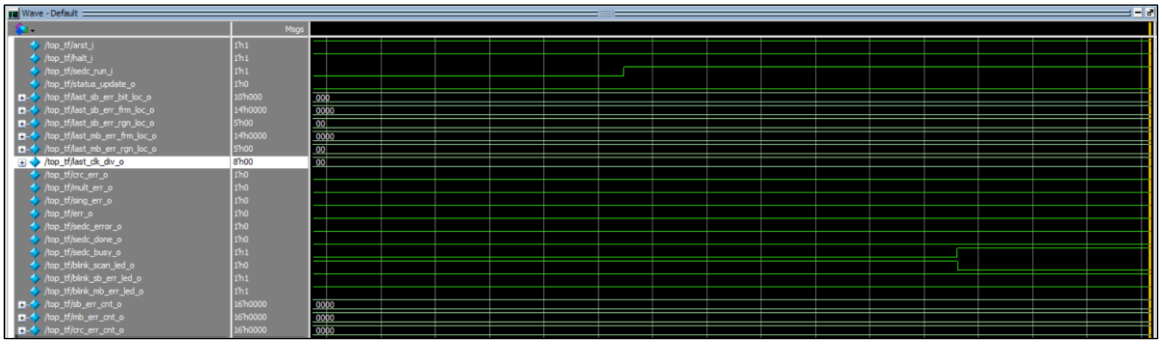

Figure 6.5. Simulation Waveforms – SEDC Scan in Progress

6.1.2. Error Detection and Resume Scan

The following are characteristics of multi-bit error detection:

- sedc_error_o asserts high when an error is detected.
- The internal soft logic of the SEDC Controller IP queries the SEDC status and configuration registers through the Lattice memory mapped interface (LMMI) and updates the following:
  - Error flags: err_o and mult_err_o
  - Error location: frm_loc_o[13:0] and rgn_loc_o[4:0]
  - Clock divider: clk_div_o[7:0]
- status_update_o pulses high for one clock cycle to indicate that updated values are ready to be read.
  - mb_err_cnt_o[15:0] increments by 1 after status_update_o asserts high and when the condition err_o && mult_err_o is true.
  - frm_loc_o[13:0] and rgn_loc_o[4:0] are captured in the last_mb_err_frm_loc_o[13:0] and last_mb_err_rgn_loc_o[4:0] registers.
  - The clock divider value is valid after sedc_busy_o asserts high. In this reference design, clk_div_o is captured in last_clk_div_o after sedc_busy_o asserts high.

    Note: The clock divider value is always incorrectly reported as 0 on the output port clk_div_o. This issue occurs regardless of the functional simulation type namely RTL, post-synthesis, post-route gate-level, or post-route gate-level+timing. This is a known issue in the Lattice Radiant software and will be fixed in a future Lattice Radiant software release.
- When a multi-bit error is detected and mb_err_cnt_o[15:0] counter is not zero, blink_sb_err_led_o and blink_mb_err_led_o use the values of blink_cnt[24] and blink_cnt[25], respectively, to blink LED D46 and D47 to indicate that a multi-bit error is detected.

The following is a characteristic of multi-bit error detection in continuous mode:

- Because multi-bit errors are non-correctable, sedc_error_o deasserts to enable resumption of scanning.

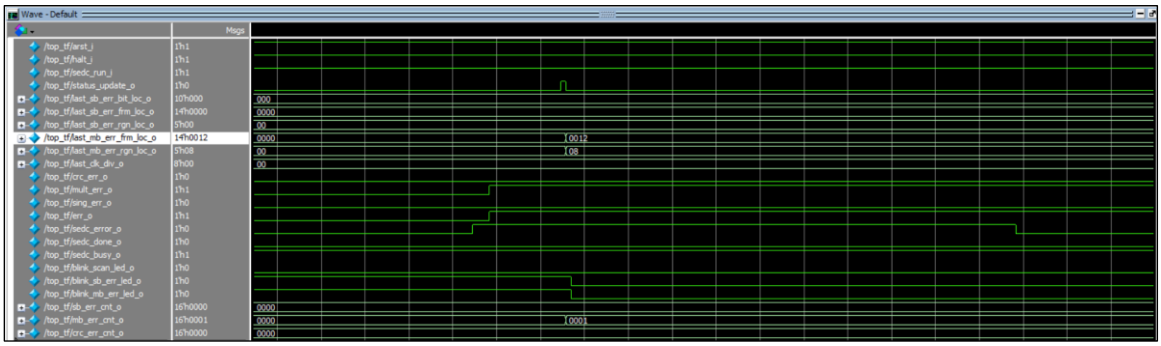

Figure 6.6. Simulation Waveforms – Multi-bit Error Detection and Resume Scan

6.1.3. CRC Error Near Scan Completion and End of Scan

The following are characteristics of CRC error near scan completion:

- Near the end of the scan, sedc_error_o asserts again to indicate that an error occurred during this scan.
- crc_error_o asserts, which is expected whenever a 1-bit or multi-bit error occurs.
- err_o remains high while mult_err_o deasserts, signaling the CRC error flag.
- crc_err_cnt_o[15:0] increments by 1 after status_update_o asserts high and when the condition err_o && crc_err_o is true.

The following are characteristics of end of scan:

- After sedc_done_o pulses high for one clock cycle, all flags (crc_err_o, err_o, sedc_busy_o, and sedc_error_o) deassert low, marking the end of the current scan.
- blink_scan_led_o asserts high once sedc_busy_o deasserts low (indicating scan is complete) to stop the blinking of LED D45.

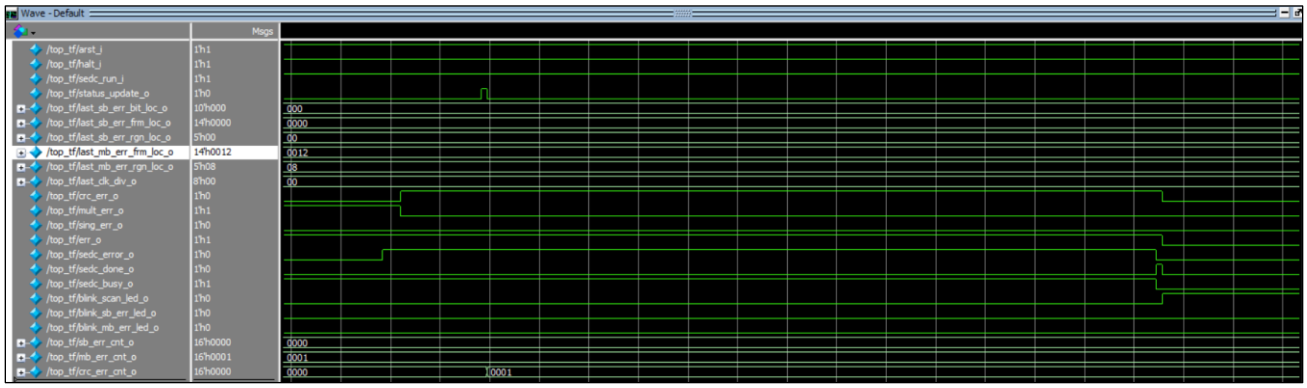

Figure 6.7. Simulation Waveforms – CRC Error and End of First SEDC Scan (Multi-bit Error)

6.1.4. Continuous Mode Behavior

The following is a characteristic of operation in continuous mode:

- sedc_busy_o reasserts while sedc_run_i (internal) remains high to indicate that the next scan has started.
  Note: Multi-bit errors reported in the first scan will continue to be reported in subsequent scans because they are uncorrectable.

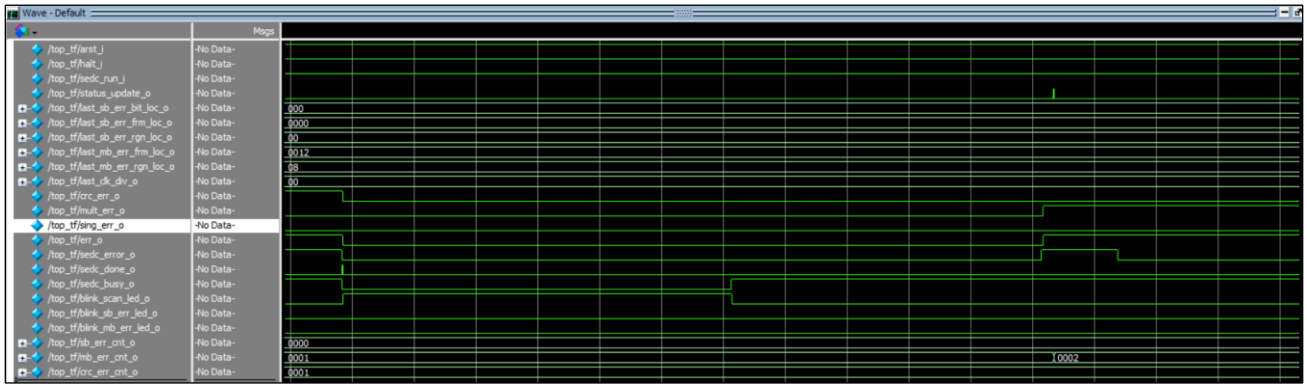

Figure 6.8. Simulation Waveforms – Second SEDC Scan (Mult-bit Error)

7. Implementing the Reference Design on Board

7.1. Hardware Requirements

The following are the hardware requirements:

- Avant-X Versa board
- USB-A to mini-B programming cable

7.2. Configuring the FPGA Using the Radiant Programmer Tool

To configure the FPGA device using the Radiant Programmer tool, follow the steps below:

1. Power on the board.
2. Launch the Radiant Programmer tool and select **Edit** > **Device Properties**.
3. Configure the device properties as shown in Figure 7.1, then click **OK**.

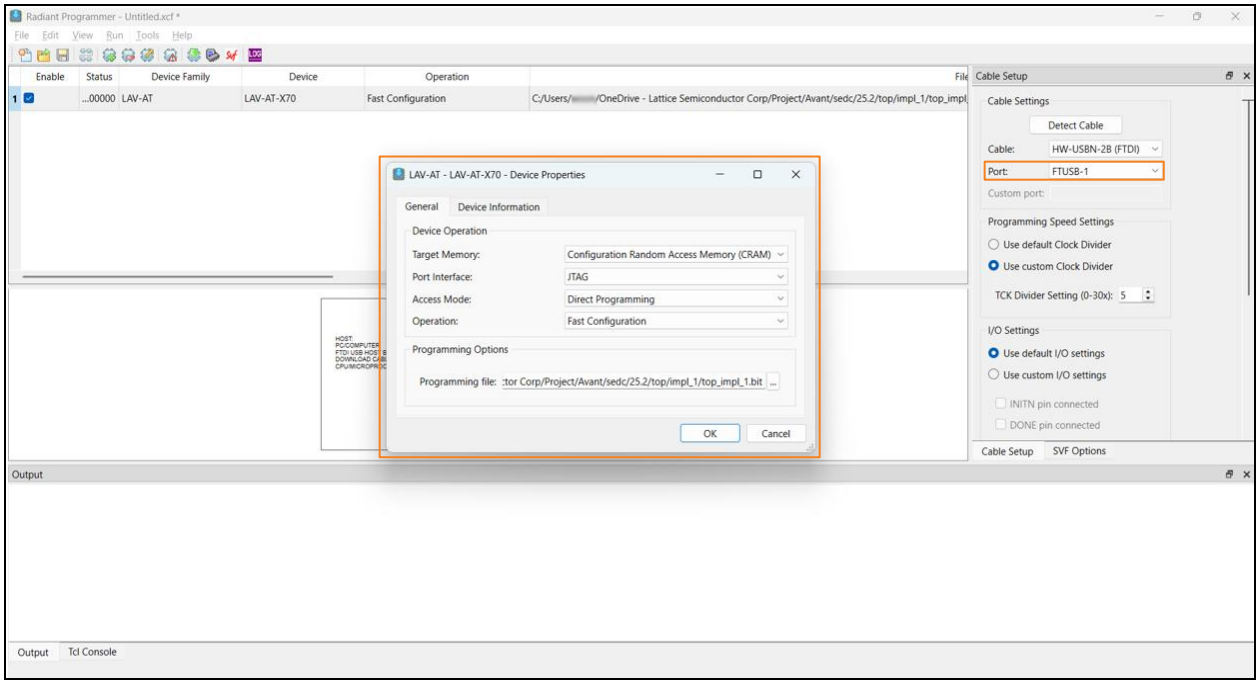

Figure 7.1. Setting Device Properties in the Radiant Programmer Tool

4. In the **Cable Settings** section of the **Cable Setup** panel, set **Port** to *FTUSB-1*.
5. Select **Run** > **Program Device** to start configuring the device.

7.3. Running the Design

To facilitate soft error injection into the configuration RAM (CRAM) and monitor the internal and external waveforms, follow the initial steps below:

1. Open the reference design project file (top.rdf) in the Lattice Radiant software, if not already open.
2. Launch the SEI Editor under **Tools** in the Lattice Radiant software and generate 1-bit and 2-bit SEI bitstreams if required. The reference design comes with pre-generated SEI bitstreams as shown in Figure 7.2. However, if you have recompiled the project after modifying the design, you must regenerate the SEI bitstreams by clicking the **Run** button in the SEI Editor.

Start Page | Report Browser | SEI Editor | Device Constraint Editor | Reveal Inserter

	Enable	ID	SEI File Name	Bit	Method	Block	Site	Frame	Bit In Frame
1	☑	0	impl_1_sei_0	1 Bit	Unused	PFU	R44C73B	data frame 10	bit 546
2	☑	1	impl_1_sei_1	1 Bit	Unused	DSP	DSP_CORE_R21C66	data frame 1029	bit 268
3	☑	2	impl_1_sei_2	1 Bit	Unused	ANY	R104C73E	data frame 4065	bit 1271
4	☑	3	impl_1_sei_3	1 Bit	Random	Routing	N/A	data frame 3559	bit 1153
5	☑	4	impl_1_sei_4	2 Bit	Unused	PFU,DSP	DSP_CORE_R21C252 R22C251A	data frame 1699 data frame 1699	bit 268 bit 269
6	☑	5	impl_1_sei_5	2 Bit	Unused	PFU,PFU	R97C76B R98C76A	data frame 3723 data frame 3723	bit 1186 bit 1188
7	☑	6	impl_1_sei_6	2 Bit	Unused	DSP,DSP	DSP_CORE_R85C84 DSP_CORE_R105C84	data frame 2747 data frame 2747	bit 1042 bit 1284
8	☑	7	impl_1_sei_7	2 Bit	Random	Routing	N/A	data frame 2908 data frame 2908	bit 1281 bit 866

Figure 7.2. SEI Editor Window with Pre-generated SEI Bitstreams

3. Launch the Reveal analyzer by double clicking *top.rva* in the Lattice Radiant software.
   Note: Due to memory limitations in the Reveal analyzer, you will be required manually select or deselect items in the Trigger Expression (TE) list and rerun the analyzer to view various SED or SEC operation waveforms.

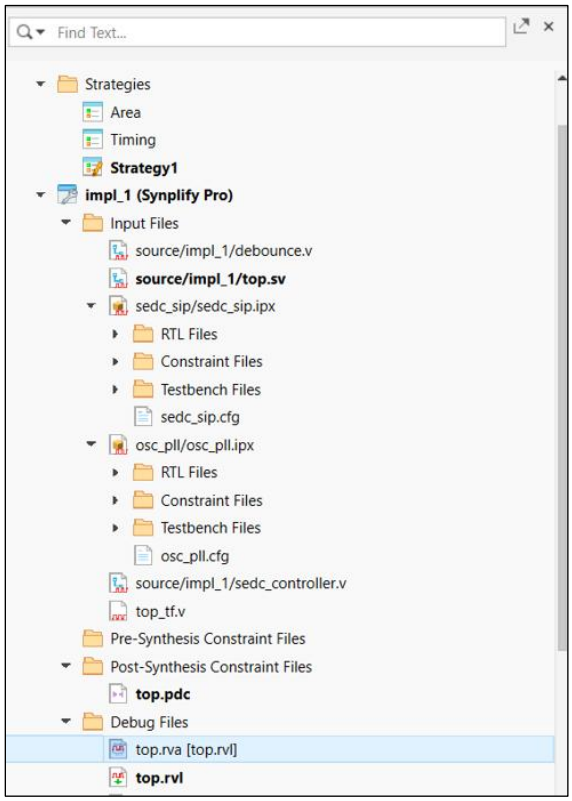

Figure 7.3. Launching the Reveal Analyzer

4. Continue with the steps in the No SEDC Error, 1-bit Error Detection and Correction (Continuous and Auto Correction Modes), and 2-bit Error Detection (Continuous and Auto Correction Modes) sections to observe SEDC waveforms under various conditions.

7.3.1. No SEDC Error

With the FPGA device configured to operate in continuous mode, to observe the SEDC waveforms when there is no SEDC error, follow the steps below:

1. Check **TE1** in the Reveal analyzer to set the trigger on sedc_run_i (internal), then click the **Run** button.

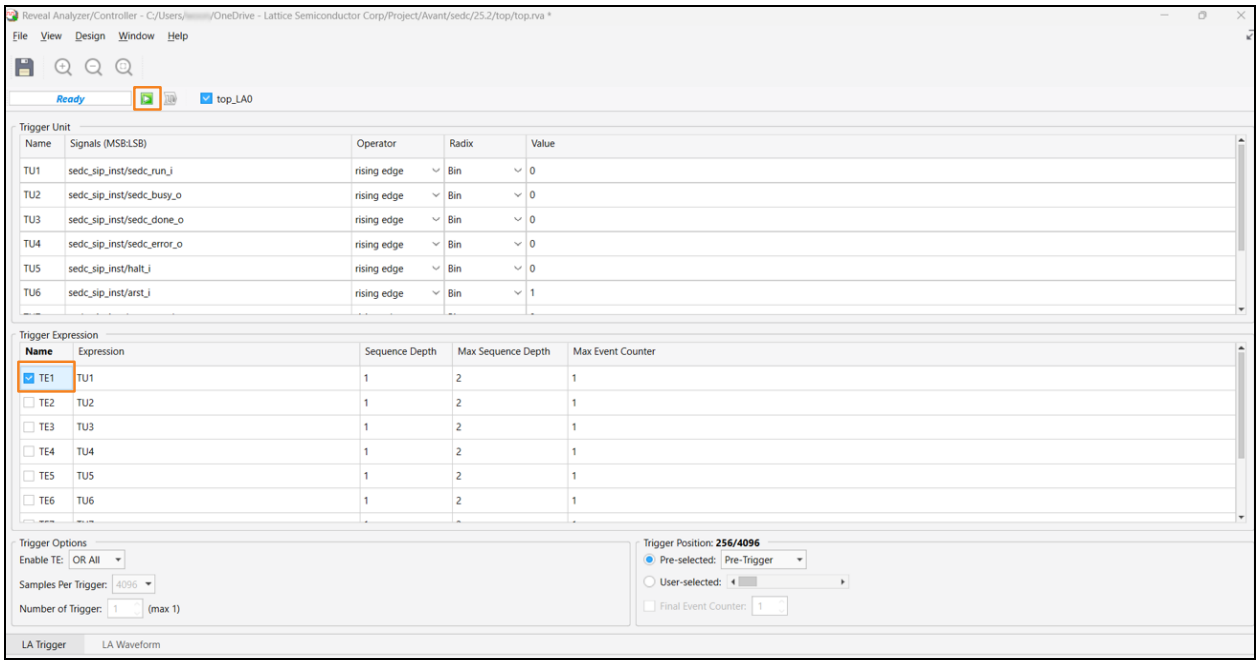

Figure 7.4. Reveal Analyzer Trigger Page – Selecting TE1 and Running Analyzer

2. With the Reveal analyzer running, briefly press the push button SW15 to set sedc_run_i (internal) to 1 to start the SEDC scan and capture the waveforms. Once the waveforms are captured, you can zoom in or out to view the SEDC signals.
   The following are observable characteristics:
   - sedc_busy_o asserts high to indicate that the SEDC scan has started and is in progress. LED D45 starts blinking.
   - The SEDC clock divider value (last_clk_div_o) is sampled after the SEDC scan starts or sedc_busy_o asserts high.

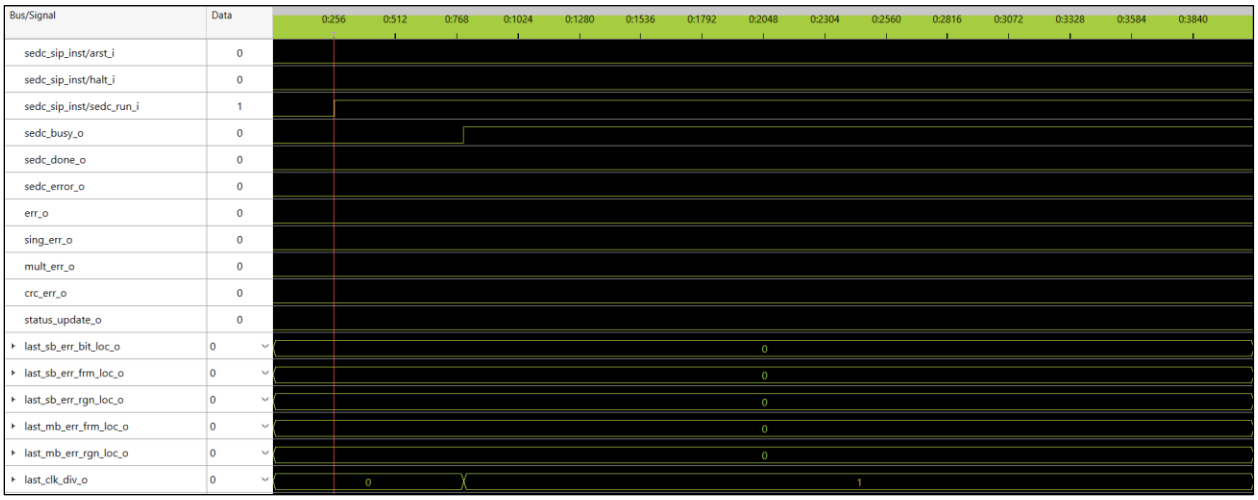

Figure 7.5. Reveal Analyzer Waveforms – Start of SEDC Scan

3. Uncheck **TE1** and check **TE3** in the Reveal analyzer to set the trigger on sedc_done_o, then click the **Run** button. The following are observable characteristics:
   - sedc_busy_o deasserts low and sedc_done_o asserts high for one clock cycle, indicating that one scan cycle is complete.

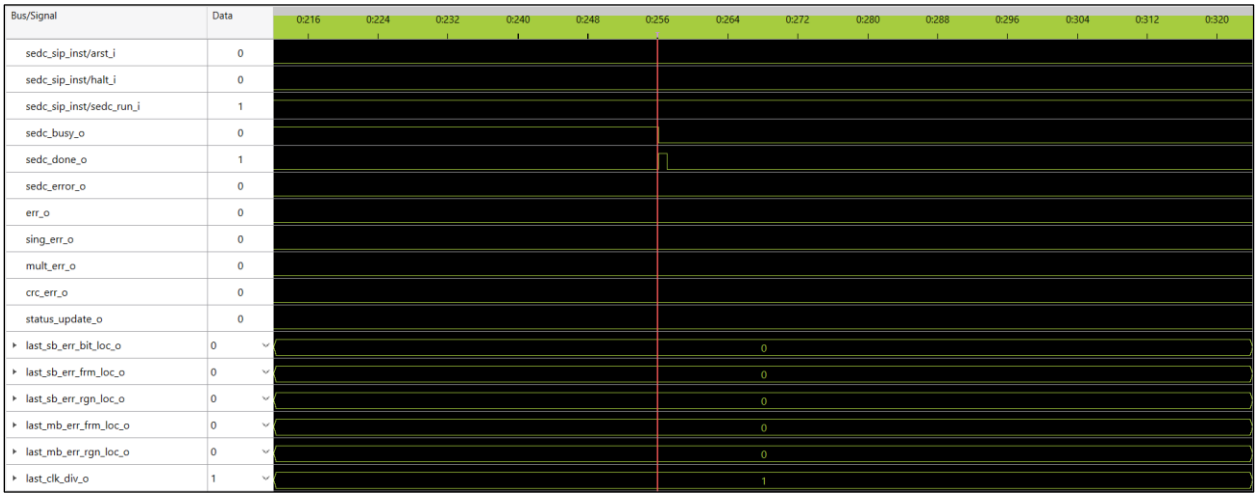

Figure 7.6. Reveal Analyzer Waveforms – End of SEDC Scan

   - In continuous mode, sedc_busy_o asserts high again to indicate that the next SEDC scan has started.

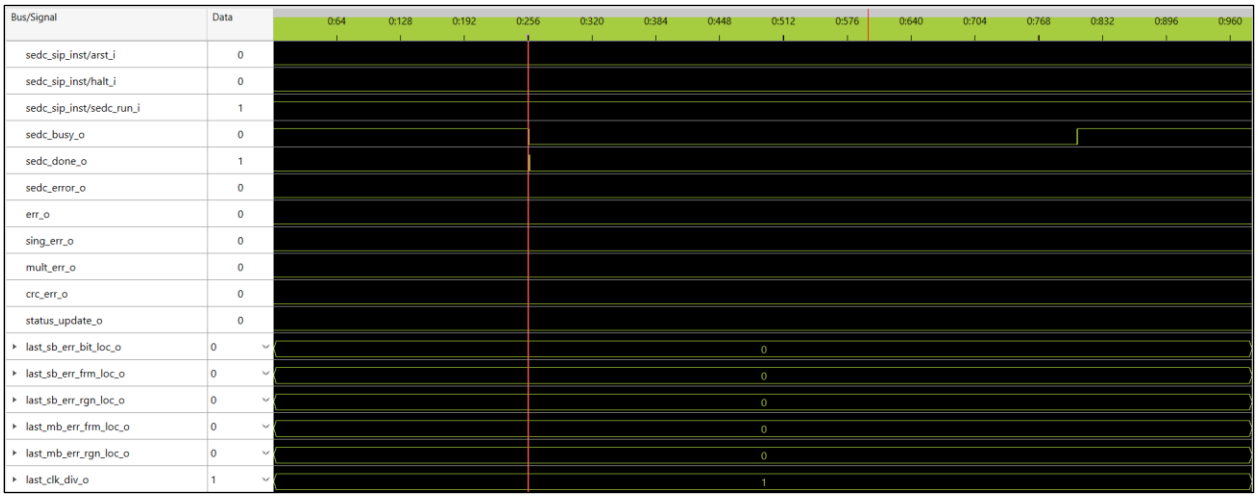

Figure 7.7. Reveal Analyzer Waveforms – Next SEDC Scan for Continuous Mode

4. Uncheck **TE3** and check **TE5** in the Reveal analyzer to set the trigger on halt_i (internal), then click the **Run** button.
5. Briefly press the push button SW13 to toggle the halt_i signal to halt the SEDC scan. The following are observable characteristics:
   - halt_i (internal) signal asserts high.
   - sedc_run_i (internal) signal deasserts low to stop the SEDC scan. Specifically, the SEDC Controller IP uses sedc_run_i (internal) to generate an internal signal used to enable or disable the SEDC hard block. This internal signal deasserts low, thereby stopping the SEDC scan.
   - sedc_busy_o deasserts low after the SEDC scan has stopped. When sedc_busy_o transitions from 1 to 0, sedc_done_o does not toggle high for one clock cycle, indicating that the SEDC scan has not completed a full scan cycle.

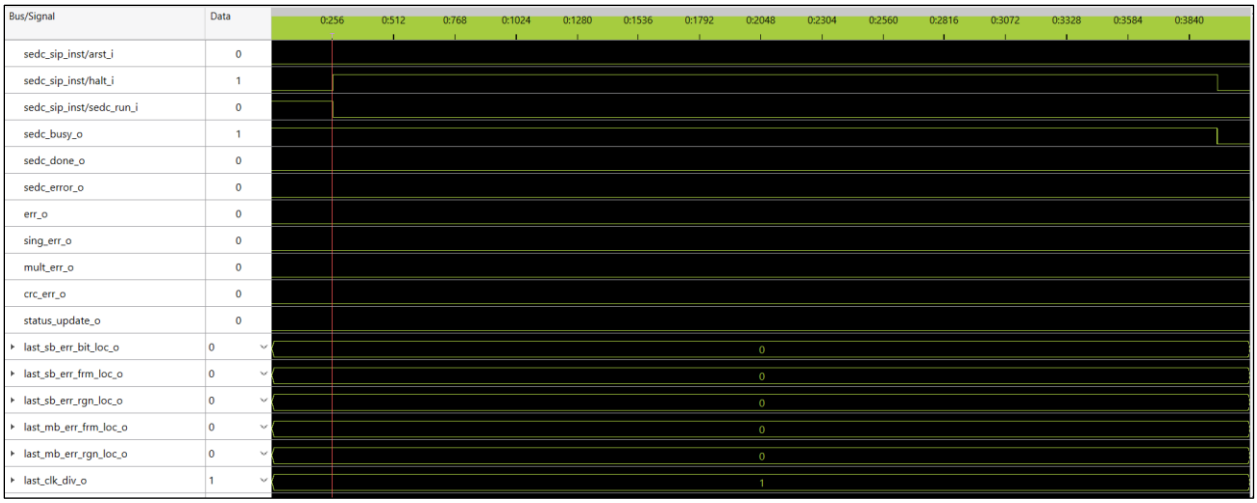

Figure 7.8. Reveal Analyzer **Waveforms** – Halt SEDC Scan

7.3.2. Calculating SED Runtime

Read the sed_time value in the Reveal analyzer to calculate the detection time.

For example:

SEDC clock divider = 2

- sed_time counter value = 1211935
- SEDC Controller IP input clock frequency = 80 MHz
- Detection time = 1/80 MHz × 1211935 = 15.149 ms

Figure 7.9. Counter for SED Time when SEDC Clock Divider = 2

7.3.3. 1-bit Error Detection and Correction (Continuous and Auto Correction Modes)

With the FPGA device configured to operate in continuous and auto-correction modes, to inject the 1-bit SEI bitstream and observe SEDC waveforms, follow the steps below:

1. Using the Radiant Programmer tool, configure the device properties as shown in Figure 7.10, then click **OK**.

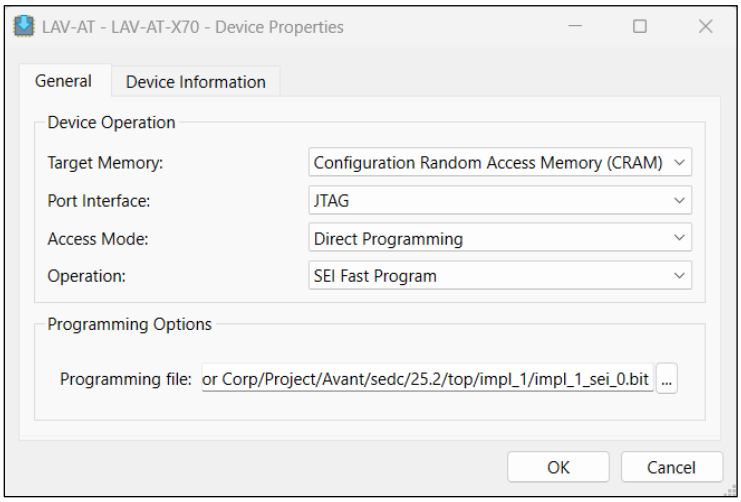

Figure 7.10. Setting Device Properties for 1-bit SEI Bitstream

Note: Use the **SEI Fast Program** operation with the 1-bit SEI bitstream as shown in Figure 7.11.

1	☑	0	impl_1_sei_0	1 Bit	Unused	PFU	R44C73B	data frame 10	bit 546

Figure 7.11. 1-bit SEI Bitstream Profile

2. Check **TE4** in the Reveal analyzer to set the trigger on sedc_error_o at the rising edge, then click the **Run** button.
   Note: To ensure the trigger on sedc_error_o is at the rising edge, set the **Operator** for **TU4** to **rising edge**.

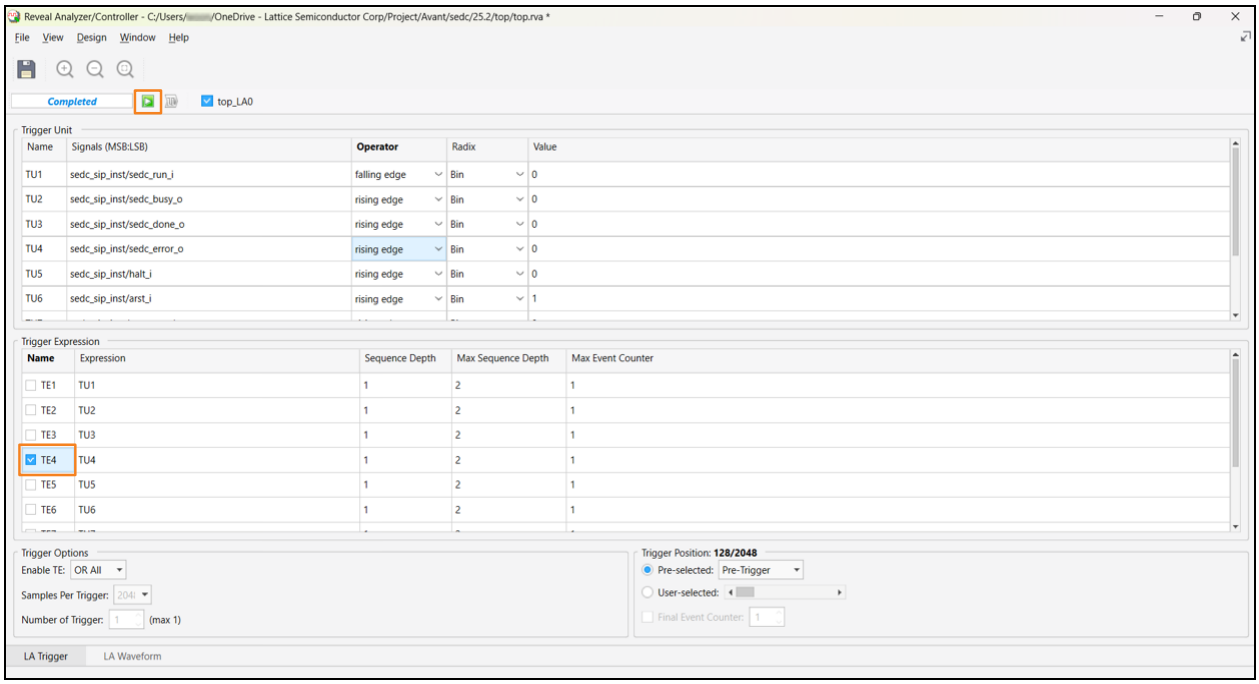

Figure 7.12. Reveal Analyzer Trigger Page – Selecting TE4 and Running Analyzer

3. With the Reveal analyzer running, briefly press the push button SW15 to start the SEDC scan and capture the waveforms. Once the waveforms are captured, you can zoom in or out to view the SEDC signals.
   The following are observable characteristics:
   - A 1-bit error is detected and reported.
     - sedc_error_o asserts high to indicate that an SEDC error is detected, halting the SEDC scan.
     - err_o and sing_err_o assert high to indicate that a 1-bit error is detected.
     - Error bit, error frame, and error region locations are reported:
       - bit_loc_o = 481
       - frm_loc_o = 10
       - rgn_loc_o = 7
     - status_update_o asserts high for one clock cycle to indicate that all error information can be read.
     - Error information is sampled and stored in last_sb_err_bit_loc_o, last_sb_err_frm_loc_o, and last_sb_err_rgn_loc_o.
     - sb_err_cnt_o counter increments by 1.
     - The reported error frame matches the SEI Editor error frame location.
       Note: There is a known issue whereby the error bit and region locations do not match the SEI Editor error bit and site locations. This issue will be fixed in a future Lattice Radiant software release.
   - sedc_error_o deasserts low to indicate that the 1-bit error is auto-corrected and the SEDC resumes scanning.

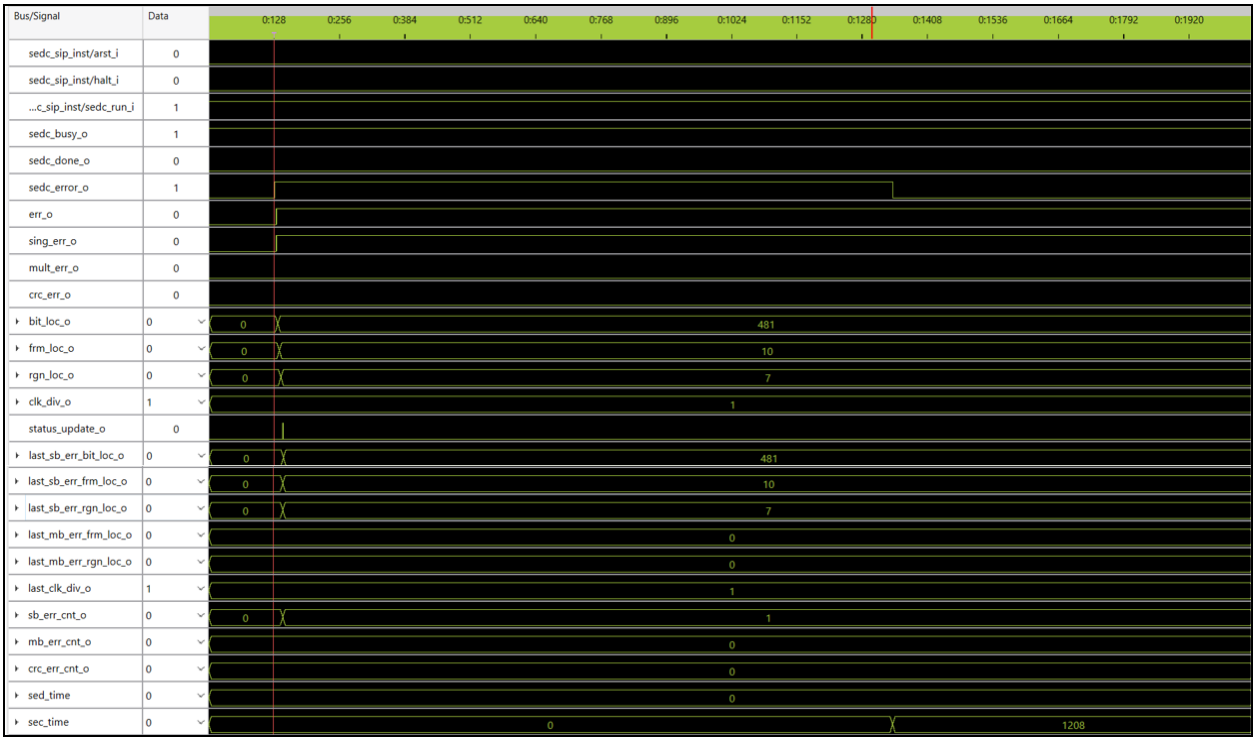

Figure 7.13. Reveal Analyzer Waveforms – First SEDC Scan 1-bit Error Detection and Correction

4. Briefly press the push button SW13 to toggle the halt_i signal to halt the SEDC scan.
5. Inject the same 1-bit SEI bitstream from Step 1 before proceeding to the next step. This addresses a limitation of the Reveal analyzer. The Reveal analyzer is unable to capture the whole SEDC scan in one shot thereby necessitating the need to rerun the SEDC scan to capture and view the desired waveforms in a scan cycle.
6. Uncheck **TE4** and check **TE8** in the Reveal analyzer to set the trigger on crc_error_o, then click the **Run** button.
7. With the Reveal analyzer running, briefly press the push button SW15 to start the SEDC scan and capture the waveforms for CRC errors occurring near the end of the scan cycle. Once the waveforms are captured, you can zoom in or out to view the SEDC signals.
   The following are observable characteristics:
   - sedc_error_o asserts high again to report the CRC error flag:
     - sedc_error_o asserts high.
     - crc_err_o and err_o assert high to indicate that a CRC error is detected.
     - sing_err_o deasserts low.
     - bit_loc_o, frm_loc_o, and rgn_loc_o are cleared to zero.
     - status_update_o asserts high for one clock cycle to indicate that all error information can be read. Note that last_sb_err_bit_loc_o, last_sb_err_frm_loc_o, and last_sb_err_rgn_loc_o are not updated because these registers are sticky error registers used to store the latest error information and will only be updated once a new single bit error is detected.
     - crc_err_cnt_o counter increments by 1.

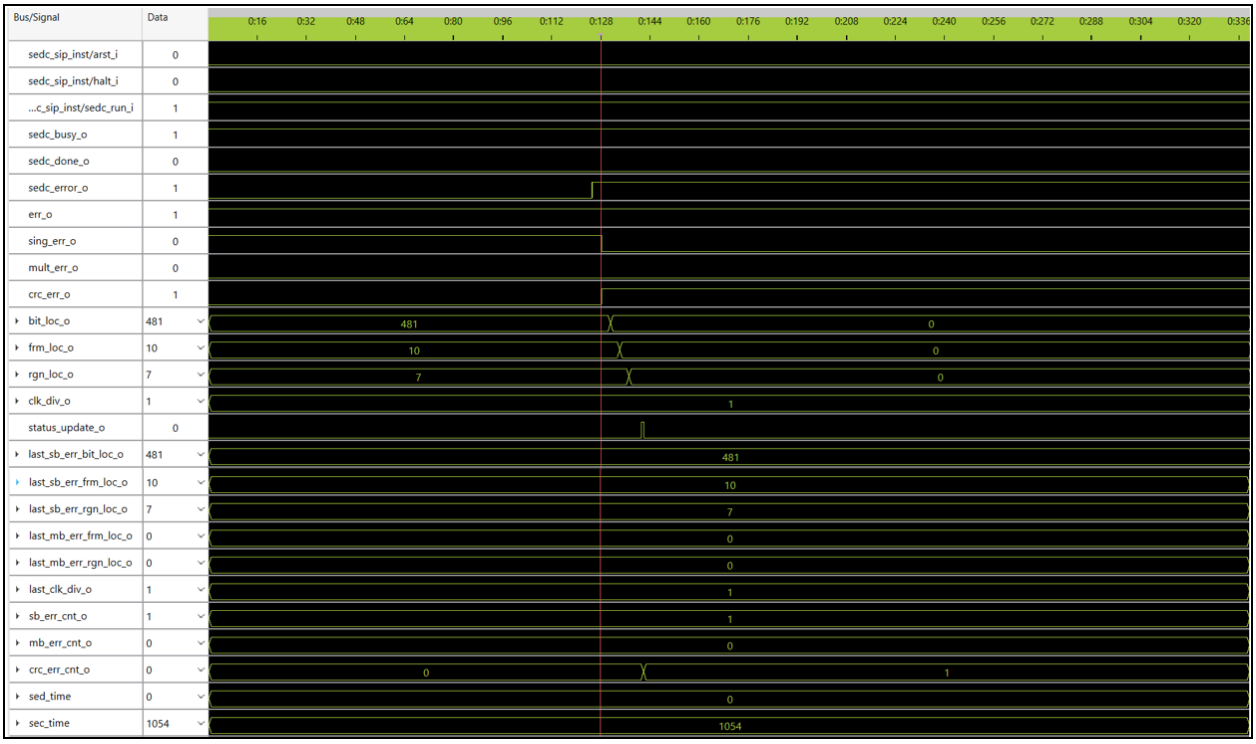

Figure 7.14. Reveal Analyzer Waveforms – CRC Error Detection Near End of Scan after 1-bit Error Detection and Correction

- sedc_busy_o and sedc_error_o deassert low at the same clock cycle when sedc_done_o asserts high.
- crc_err_o and err_o deassert low while clk_div_o clears to zero after the momentary one clock cycle assertion of sedc_done_o (more precisely when sedc_done_o deasserts low), indicating the end of a scan cycle.
- After a number of cycles, clk_div_o reads back as 1, indicating the SEDC Controller is preparing for the next SEDC scan.
- Because the SEDC Controller is set to run in continuous mode, sedc_busy_o asserts high again to indicate that a new scan cycle has started.

  Note: The 1-bit error will not be detected and reported in the next SEDC scan cycle because the 1-bit error has been corrected.

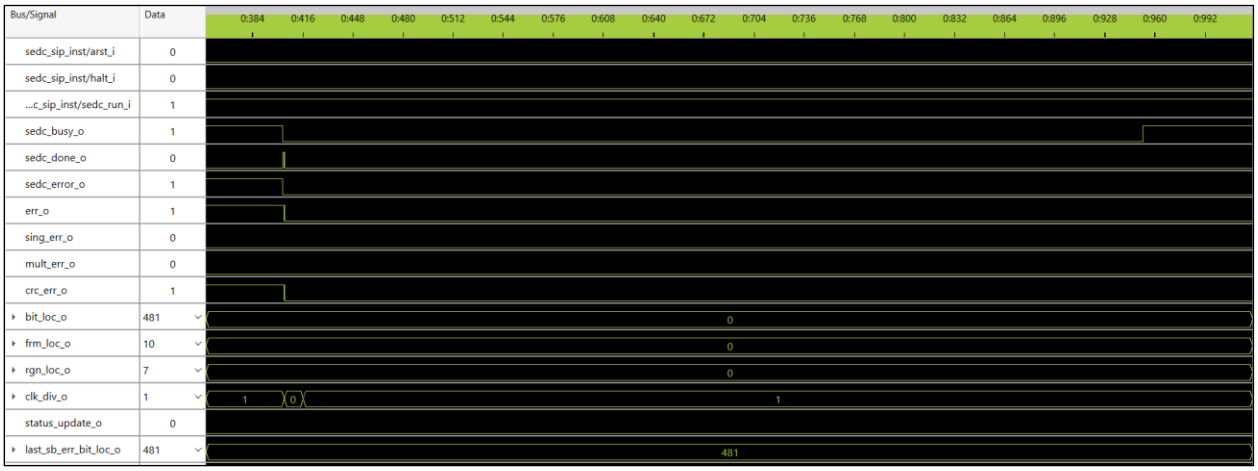

Figure 7.15. Reveal Analyzer Waveforms – End of First SEDC Scan and Start of Next Scan

7.3.4. Calculating 1-bit SEC Time

Read the sec_time value in the Reveal analyzer to calculate the error correction time.

For example:

SEDC clock divider = 2

- sec_time counter value = 1208
- Clock frequency = 80 MHz
- Correction time = 1/80 MHz × 1208 = 15.1 µs

Figure 7.16. Counter for SEC Time When SEDC Clock Divider = 2

7.3.5. 2-bit Error Detection (Continuous and Auto Correction Modes)

With the FPGA device configured to operate in continuous and auto-correction modes, to inject the 2-bit SEI bitstream and observe SEDC waveforms, follow the steps below:

1. Using the Radiant Programmer, configure the device properties as shown in Figure 7.10 (except replace the 1-bit SEI bitstream with the 2-bit SEI bitstream), then click **OK**.
   Note: Use the **SEI Fast Program** operation with the 2-bit SEI bitstream as shown in Figure 7.17.

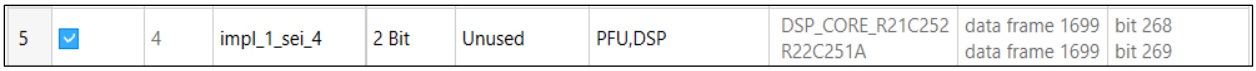

Figure 7.17. 2-bit SEI Bitstream Profile

2. Check **TE4** in the Reveal analyzer to set the trigger on sedc_error_o at the rising edge, then click the **Run** button.
   Note: To ensure the trigger on sedc_error_o is at the rising edge, set the **Operator** for **TU4** to **rising edge**.
3. With the Reveal analyzer running, briefly press the push button SW15 to start the SEDC scan and capture the waveforms. Once the waveforms are captured, you can zoom in or out to view the SEDC signals.
   The following are observable characteristics:
   - A 2-bit error is detected and reported.
     - sedc_error_o asserts high to indicate that an SEDC error is detected, halting the SEDC scan.
     - err_o and mult_err_o assert high to indicate that a multi-bit error is detected.
     - Error frame and error region locations are reported:
       - frm_loc_o = 1699
       - rgn_loc_o = 2
     - status_update_o asserts high for one clock cycle to indicate that all error information can be read.
     - Error information is sampled and stored in last_mb_err_frm_loc_o and last_mb_err_rgn_loc_o.
     - mb_err_cnt_o counter increments by 1.
     - The reported error frame matches the SEI Editor error frame location. The error bit location is reported as zero, which is expected by design when a 2-bit error is detected.
       Note: There is a known issue whereby the region location does not match the SEI Editor site location. This issue will be fixed in a future Lattice Radiant software release.
   - sedc_error_o deasserts low automatically and the SEDC resumes scanning because a multi-bit error is non-correctable.

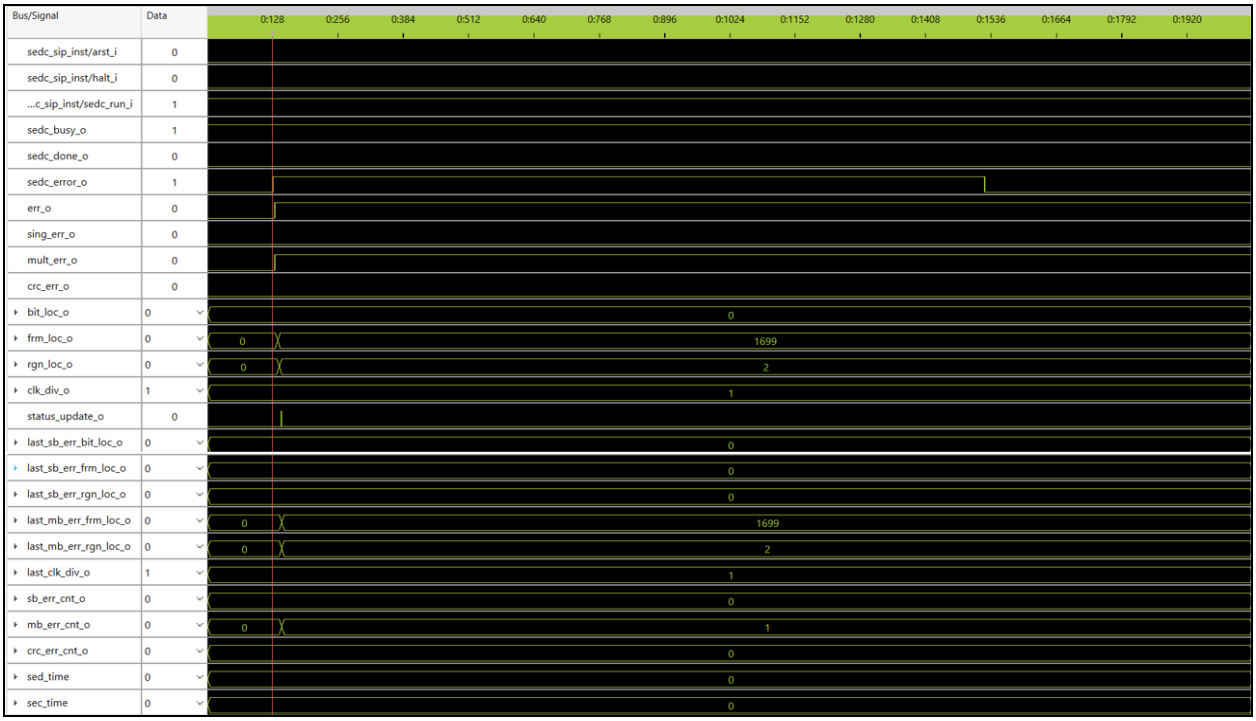

Figure 7.18. Reveal Analyzer Waveforms – First SEDC Scan 2-bit Error Detection

4. Uncheck **TE4** and check **TE8** in the Reveal analyzer to set the trigger on crc_error_o, then click the **Run** button to capture and verify the waveforms for CRC errors occurring near the end of the scan cycle.
   The following are observable characteristics:
   - sedc_error_o asserts high again to report the CRC error flag:
     - sedc_error_o asserts high.
     - crc_err_o and err_o assert high.
     - mult_err_o deasserts low.
     - bit_loc_o, frm_loc_o, and rgn_loc_o are cleared to zero.
     - status_update_o asserts high for one clock cycle to indicate that all error information can be read. Note that last_mb_err_frm_loc_o and last_mb_err_rgn_loc_o are not updated because these registers are sticky error registers used to store the latest error information and will only be updated once a new multi-bit error is detected.
     - crc_err_cnt_o counter increments by 1.

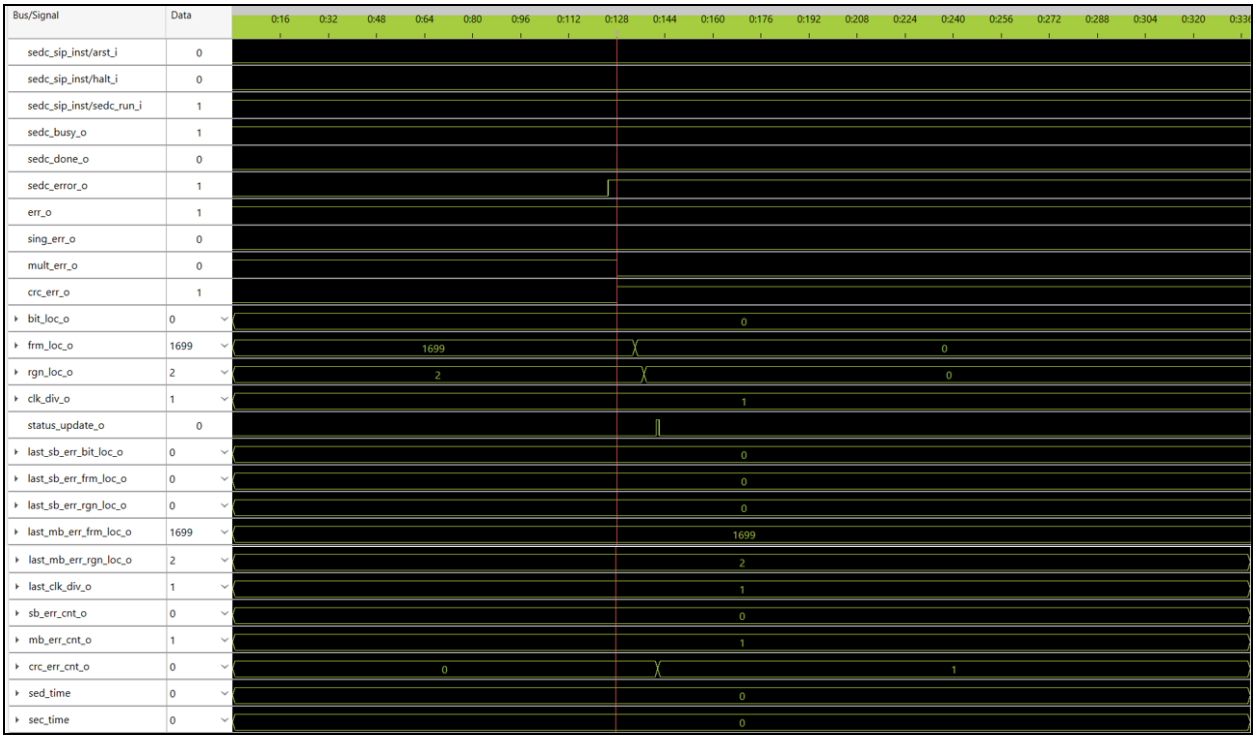

Figure 7.19. Reveal Analyzer Waveforms – CRC Error Detection Near End of Scan after 2-Bit Error Detection

- sedc_busy_o and sedc_error_o deassert low at the same clock cycle when sedc_done_o asserts high.
- crc_err_o and err_o deassert low while clk_div_o clears to zero after the momentary one clock cycle assertion of sedc_done_o (more precisely when sedc_done_o deasserts low), indicating the end of a scan cycle.
- After a number of cycles, clk_div_o reads back as 1, indicating the SEDC Controller is preparing for the next SEDC scan.
- Because the SEDC Controller is set to run in continuous mode, sedc_busy_o asserts high again to indicate a new scan cycle has started.

  Note: The multi-bit error will be detected and reported in the next SEDC scan cycle because a multi-bit error is non-correctable.

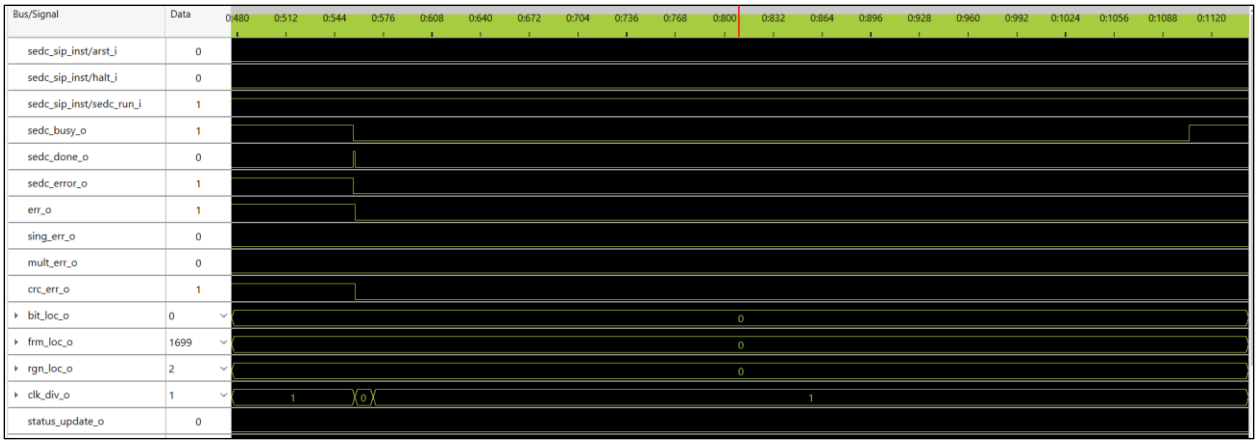

Figure 7.20. Reveal Analyzer Waveforms – End of First SEDC Scan with 2-bit Error Detection and Start of Next Scan

References

- Lattice Avant SED/SEC User Guide (FPGA-TN-02290)
- SEDC Controller IP User Guide (FPGA-IPUG-02290)
- Lattice Avant OSC Module User Guide (FPGA-IPUG-02184)
- Avant-X web page
- Avant-X Versa Board web page
- IP Cores and Reference Designs for Avant X Devices
- Kits, Boards, and Demonstrations for Avant X Devices
- Lattice Solutions Reference Designs web page
- Lattice Solutions IP Cores web page
- Lattice Radiant Software web page
- Lattice Insights for Lattice Semiconductor training courses and learning plan

Technical Support Assistance

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

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

Revision History

Revision 1.0, April 2026

Section	Change Summary
All	Initial release.

www.latticesemi.com