MachXO4 Soft Error Detection (SED) and Correction (SEC) User Guide

Technical Note

FPGA-TN-02406-1.0

December 2025

Disclaimers

Lattice makes no warranty, representation, or guarantee regarding the accuracy of information contained in this document or the suitability of its products for any particular purpose. All information herein is provided AS IS, with all faults, and all associated risk is the responsibility entirely of the Buyer. The information provided herein is for informational purposes only and may contain technical inaccuracies or omissions, and may be otherwise rendered inaccurate for many reasons, and Lattice assumes no obligation to update or otherwise correct or revise this information. Products sold by Lattice have been subject to limited testing and it is the Buyer's responsibility to independently determine the suitability of any products and to test and verify the same. LATTICE PRODUCTS AND SERVICES ARE NOT DESIGNED, MANUFACTURED, OR TESTED FOR USE IN LIFE OR SAFETY CRITICAL SYSTEMS, HAZARDOUS ENVIRONMENTS, OR ANY OTHER ENVIRONMENTS REQUIRING FAIL-SAFE PERFORMANCE, INCLUDING ANY APPLICATION IN WHICH THE FAILURE OF THE PRODUCT OR SERVICE COULD LEAD TO DEATH, PERSONAL INJURY, SEVERE PROPERTY DAMAGE OR ENVIRONMENTAL HARM (COLLECTIVELY, "HIGH-RISK USES"). FURTHER, BUYER MUST TAKE PRUDENT STEPS TO PROTECT AGAINST PRODUCT AND SERVICE FAILURES, INCLUDING PROVIDING APPROPRIATE REDUNDANCIES, FAIL-SAFE FEATURES, AND/OR SHUT-DOWN MECHANISMS. LATTICE EXPRESSLY DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY OF FITNESS OF THE PRODUCTS OR SERVICES FOR HIGH-RISK USES. The information provided in this document is proprietary to Lattice Semiconductor, and Lattice reserves the right to make any changes to the information in this document or to any products at any time without notice.

Inclusive Language

This document was created consistent with Lattice Semiconductor’s inclusive language policy. In some cases, the language in underlying tools and other items may not yet have been updated. Please refer to Lattice’s inclusive language FAQ 6878 for a cross reference of terms. Note in some cases such as register names and state names it has been necessary to continue to utilize older terminology for compatibility.

Contents

Contents 3
Abbreviations in This Document 5
1. Introduction 6
2. Soft Error Detection 7
2.1. Overview 7
2.2. SED Limitations 8
2.3. SED Operating Modes 8
2.3.1. Standard SED 9
2.3.2. One-Shot SED 9
2.4. Signal Descriptions 9
2.5. SED Clock Driver 10
2.6. SED Attributes 10
2.7. Port Descriptions 11
2.7.1. SEDENABLE 11
2.7.2. SEDSTART 11
2.7.3. SEDFRCERR 11
2.7.4. SEDSTDBY 11
2.7.5. SEDCLKOUT 11
2.7.6. SEDDONE 11
2.7.7. SEDINPROG 11
2.7.8. SEDERR 11
2.8. SED Flow 12
2.9. Timing Diagram for SED Operation 13
2.10. SED Run Time 14
2.11. Sample Code 15
2.11.1. SED VHDL Examples 15
2.11.2. SED Verilog Examples 16
3. Soft Error Correction 17
3.1. SPI Controller Code 18
References 19
Technical Support Assistance 20
Revision History 21

Figures

Figure 2.1. System Block Diagram 7
Figure 2.2. SEDFA Primitive Symbol 9
Figure 2.3. SEDFB Primitive Symbol 9
Figure 2.4. Example Schematic 12
Figure 2.5. Timing Diagram for SED Operation 13
Figure 3.1. MachXO4 Device Controller SPI Port with SPI Flash 18

Tables

Table 1.1. Soft Error Detection, Correction, and Injection Support 6
Table 2.1. SEDFA and SEDFB Primitive Port Definitions 9
Table 2.2. SED Internal Oscillator Supported Frequency Settings 10
Table 2.3. SED Attributes 10
Table 2.4. SED Operation Timing Parameters 13
Table 2.5. SED Run Time 14

Abbreviations in This Document

A list of abbreviations used in this document.

Abbreviation	Definition
CRC	Cyclic Redundancy Check
DRAM	Dynamic Random-Access Memory
DSP	Digital Signal Processing
EBR	Embedded Block RAM
EFB	Embedded Functional Block
FPGA	Field Programmable Gate Array
GPIO	General Purpose Input Output
I2C	Inter-Integrated Circuit
IEEE	Institute of Electrical and Electronics Engineers
IP	Intellectual Property
JTAG	Joint Test Action Group
PFU	Programmable Functional Unit
PLD	Programmable Logic Device
RAM	Random-Access Memory
SEC	Soft Error Correction
SED	Soft Error Detection
SEI	Soft Error Injection
SPI	Serial Peripheral Interface
SRAM	Static Random-Access Memory
VHDL	VHSIC Hardware Description Language

1. Introduction

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 have continued to shrink, the probability of memory errors in static random-access memory (SRAM) has become significant for some systems. Designers are using a variety of approaches to minimize the effects of memory errors on system behavior.

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 will alter the programmed logical behavior of a system increases. A number of approaches have been taken to address this issue, but most involve intellectual property (IP) cores that the user instantiates into the logic of their design, using valuable resources and possibly affecting design performance. MachXO4™ devices have a hardware implemented soft error detection (SED) circuit which can be used to detect SRAM errors and allow them to be corrected.

This document describes the hardware-based SED approach taken by Lattice Semiconductor for MachXO4 FPGAs. Once a soft error is detected, Lattice provides a way to perform soft error correction (SEC) without disturbing the functionality of the device. Lattice also provides a tool to help the user emulate soft error impact by injecting soft error into the device.

Table 1.1. Soft Error Detection, Correction, and Injection Support

Device	SED	SEC	SEI
MachXO4	Yes	Yes	No[1]

Note:

1. Soft error injection (SEI) will be supported in a future Lattice Radiant software release.

2. Soft Error Detection

2.1. Overview

The SED hardware in MachXO4 devices, which is part of the embedded functional block (EFB), consists of an access point to the PLD configuration logic, a control circuit, and a 32-bit register to store the cyclic redundancy check (CRC) for a given bitstream (see Figure 2.1). The SED hardware reads serial data from the PLD configuration memory and calculates a CRC. The data that is read and the CRC that is calculated do not include embedded block RAM (EBR) memory or programmable functional units (PFUs) used as random-access memory (RAM). The calculated CRC is then compared with the expected CRC stored in the 32-bit register. If the CRC values match, there is no configuration memory corruption. However, if the CRC values differ, an error signal is generated.

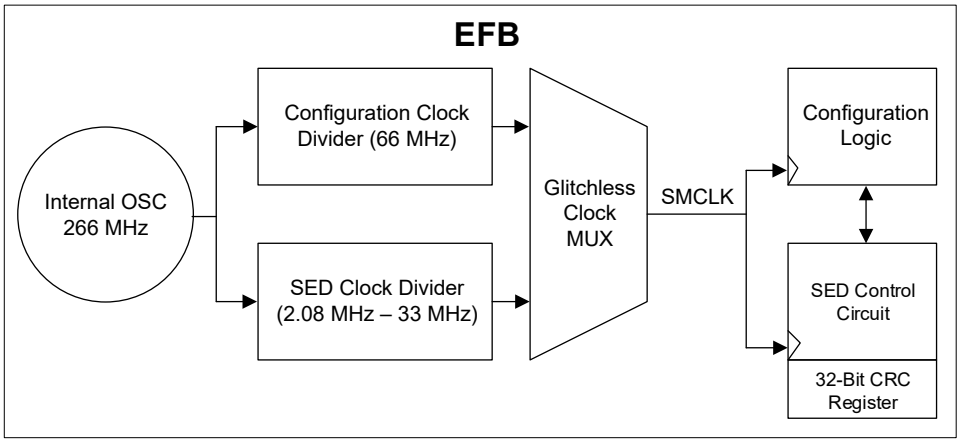

Figure 2.1. System Block Diagram

Note that the calculated CRC is based on the arrangement of configuration memory for a design. Consequently, the expected CRC cannot be specified until after the design is placed and routed. The Lattice Radiant™ bitstream generation software analyzes the configuration of a placed and routed design and updates the contents of the 32-bit CRC register during bitstream generation.

The following sections describe the MachXO4 SED implementation and flow.

2.2. SED Limitations

SED should only be run when V_{CC} reaches the data sheet V_{CC} minimum recommend level. In addition, clock frequencies greater than 33.33 MHz are not supported for the SED.

The clock (SMCLK) of the SED circuit is shared with the configuration logic. As a result, the SED module interacts with several EFB functions in the following ways:

- If the EFB or configuration logic is accessed while the SED circuit is running:
  - The current SED cycle is terminated.
    - When the SED circuit is terminated, there is a delay of two SMCLK cycles before the EFB or configuration logic can be accessed. This is a result of the SMCLK transferring clock from the SED clock to the configuration clock domain. The two SMCLK cycles are defined by the slower SED clock.
    - When the SED circuit is terminated, SEDDONE and SEDERR remain low; SEDINPROG resets from high to low.
    - The EFB or configuration logic access which interacts with the SED circuit is defined as:
      - The following commands issued through the JTAG port or WISHBONE interface:
        - LSC_REFRESH
        - ISC_ENABLE
        - ISC_ENABLE_X
        - All IEEE 1532 instructions
        - ISC_DISABLE
    - Primary I2C configuration logic slave address match.
    - Serial peripheral interface (SPI) configuration logic chip select is asserted.
- The PROGRAMN pin detection logic requires the minimum low period to be longer than six SMCLK cycles. If the SED circuit is running, the six SMCLK cycles are defined by the SED clock.

2.3. SED Operating Modes

For MachXO4 devices, there are two operating modes available for SED:

- Standard mode allows the design to control when SED is run and to test the error detection operation.
- One-shot operation mode is used to run the SED once when the device is first configured to ensure that the configuration matches the desired configuration.

Both operations perform a single cycle which checks the CRC of all the bits in the SRAM except the EBR and RAM memory. Standard mode is activated using the SEDFA primitive while one-shot operation mode is activated using the SEDFB primitive. These primitives are described in the following sections.

If an error is detected during an SED operation, you can choose one of two actions: do nothing or initiate an on-demand user reconfiguration by pulsing the PROGRAMN pin. This can be done from another device or from an output of the MachXO4 device as shown in Figure 2.4.

The PROGRAMN pin detection logic requires the minimum low period to be longer than six SMCLK cycles. When error detection is enabled, the PROGRAMN pin minimum low period is six SEDCLK cycles because the active SMCLK switched to SEDCLK. This behavior is different from normal operation where SMCLK is operating at full speed. After booting, the SED function block behaves according to the new configuration programming.

2.3.1. Standard SED

Standard SED operation can be used by instantiating the SEDFA primitive as shown in Figure 2.2. The primitive port definitions are listed in Table 2.1. See the Port Descriptions section for more information about each of the ports.

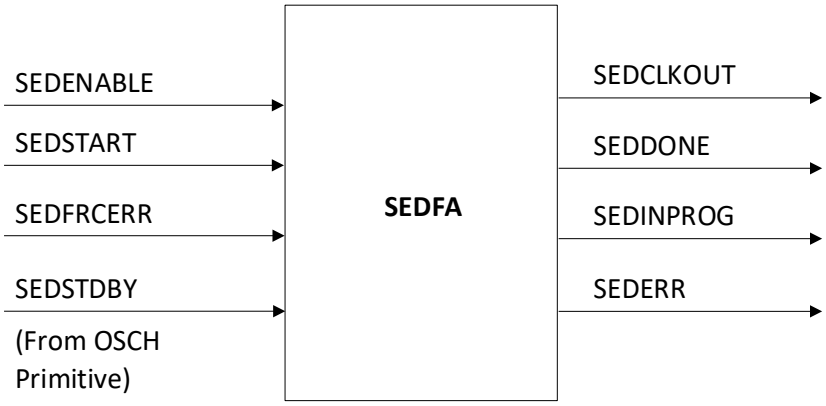

Figure 2.2. SEDFA Primitive Symbol

2.3.2. One-Shot SED

One-shot SED operation can be used by instantiating the SEDFB primitive as shown in Figure 2.3. The primitive port definitions are listed in Table 2.1. See the Port Descriptions section for more information about each of the ports.

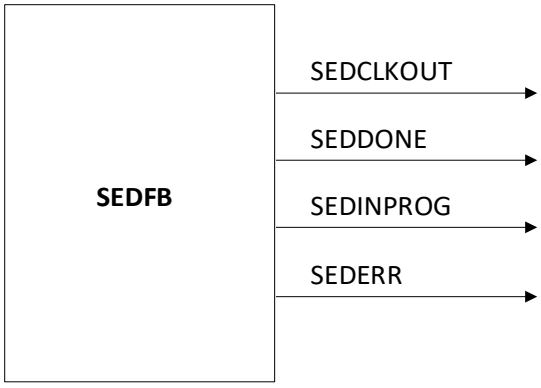

Figure 2.3. SEDFB Primitive Symbol

2.4. Signal Descriptions

Table 2.1. SEDFA and SEDFB Primitive Port Definitions

Signal Name	Direction	Active	Description
SEDENABLE	Input	High	Enables SRAM CRC
SEDSTART	Input	Rising edge	Starts SRAM CRC cycle
SEDFRCERR	Input	Rising edge	Forces an SRAM CRC error flag
SEDSTDBY	Input	High	Disables SRAM CRC while in standby mode
SEDCLKOUT	Output	—	Output clock
SEDDONE	Output	High	SRAM CRC cycle is complete
SEDINPROG	Output	High	SRAM CRC cycle is in progress
SEDERR	Output	High	SRAM CRC error flag

2.5. SED Clock Driver

The SED circuitry is driven by the MachXO4 internal oscillator when using either the SEDFA or SEDFB primitive. The maximum frequency supported is 33.25 MHz.

The MachXO4 internal oscillator can be used for several device functions including configuration and SED, and as an internal user clock. The frequency of the oscillator output can be set to different values depending on use. The settings available for the SED clock are shown in Table 2.2. When using the SED, the internal oscillator frequency is specified using the SED_CLK_FREQ attribute.

Table 2.2. SED Internal Oscillator Supported Frequency Settings

2.08	4.16	8.31	16.63
2.15	4.29	8.58	17.73
2.22	4.43	8.87	19.00
2.29	4.59	9.17	20.46
2.38	4.75	9.50	22.17
2.46	4.93	9.85	24.18
2.56	5.12	10.23	26.60
2.66	5.32	10.64	29.56
2.77	5.54	11.08	33.25
2.89	5.78	11.57	—
3.02	6.05	12.09	—
3.17	6.33	12.67	—
3.33	6.65	13.30	—
3.50	7.00	14.00	—
3.69	7.39	14.78	—
3.91	7.82	15.65	—

2.6. SED Attributes

There are three attributes that can be used with the SED primitives as shown in Table 2.3. Usage examples for these attributes can be found in the Sample Code section. The SEDFB primitive does not support the three attributes listed.

Table 2.3. SED Attributes

Attribute Name	Attribute Type	Description
SED_CLK_FREQ	String	Specifies the clock frequency when used with the SEDFA primitive. The SEDFA primitive uses the MachXO4 internal oscillator as the clock source. The available settings are shown in Table 2.2. If a value other than those listed is used, the software issues an error message and exits from the MAP process.
DEV_DENSITY	String	Specifies the device density for use by the MachXO4 device simulation model. The allowable values for the DEV_DENSITY attribute are: *010*, *015*, *015_256P*, *025*, *050*, *050_400P*, *080*, and *110*.
CHECKALWAYS	String	Reserved for future use. Not supported at this time.

2.7. Port Descriptions

2.7.1. SEDENABLE

SEDENABLE is a level-sensitive signal which enables SED checking when high. When this signal is low, the SED hardware is disabled. This can be tied high in a design if desired.

When performing background programming, SEDENABLE must be pulled low to disable SED and avoid contention with the configuration logic.

2.7.2. SEDSTART

SEDSTART is the signal which starts the SED process. The rising edge of the SEDSTART signal causes the SED cycle to start if SEDENABLE is high. The SEDSTART signal must remain high until the SED process has completed. If SEDSTART goes low during the SED cycle, the process is terminated without asserting SEDDONE or SEDERR.

2.7.3. SEDFRCERR

SEDFRCERR is used to force the SED process to return an error indication on the SEDERR signal. This is typically done to test the logic associated with the SEDERR output. The rising edge of the SEDFRCERR signal is detected by the SED hardware and latched by the rising edge of the SED clock driver signal. SEDFRCERR should be latched high while the SED is active for an error indication to be returned. The recommended use is for the user logic to drive the SEDFRCERR signal from low to high once the rising edge of the SEDINPROG signal is detected while following the setup/hold time requirements defined in Figure 2.5.

2.7.4. SEDSTDBY

The SEDSTDBY port is provided on the SEDFA primitive only and must be connected to the SEDSTDBY output port on the OSCH component. This signal is provided for simulation support of the STDBY function, which can be used to turn off the internal oscillator. When the STDBY function turns off the internal oscillator, the SEDFA block no longer operates because its clock source has been turned off. If you do not connect this signal on the SEDFA primitive, the SED still functions the same way in the hardware but may not match the simulation results when STDBY is used.

2.7.5. SEDCLKOUT

SEDCLKOUT is a gated version of the SED clock driver signal to the SED block. SEDCLKOUT is gated by SEDENABLE. This signal can be used to synchronize the inputs to the SED block or the outputs from the SED block.

2.7.6. SEDDONE

SEDDONE is an output which indicates that SED checking has completed a cycle. This signal is an active high output from the SED hardware, clocked out on the rising edge of SEDCLKOUT. SEDDONE is reset by a low SEDSTART signal.

2.7.7. SEDINPROG

SEDINPROG is an output which indicates that SED checking is in progress. This signal is an active high output from the SED hardware, clocked out on the rising edge of SEDCLKOUT. SEDINPROG follows SEDSTART to go high after a delay as shown in Figure 2.5.

2.7.8. SEDERR

SEDERR is an output which indicates that SED checking has completed a cycle with an error. This signal is an active high output from the SED hardware, clocked out on the rising edge of SEDCLKOUT. SEDERR is reset by a low SEDSTART signal after a delay as defined in Figure 2.5.

2.8. SED Flow

The general SED flow once V_{CC} reaches the data sheet V_{CC} minimum recommend level is as follows.

- User logic sets SEDENABLE high. This signal may be tied high if desired.
- User logic sets SEDSTART high and holds it high for the duration of the SED cycle. SEDINPROG goes high. If SEDDONE or SEDERR is already high, it is driven low.
- SED starts reading back data from the configuration SRAM.
- SED finishes checking. SEDERR is updated, SEDINPROG goes low, SEDDONE goes high, and another SED cycle is started by asserting SEDSTART. When SEDSTART is asserted, the SEDERR signal is reset.
- If SEDERR is driven high, it can be reset by reconfiguring the PLD.
- SEDENABLE goes low when or if the user specifies, and SED is no longer in use.

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 through a general purpose input output (GPIO) pin externally connected to PROGRAMN.

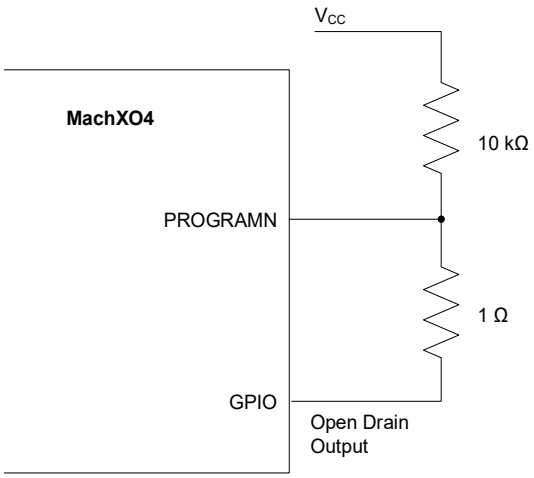

Figure 2.4. Example Schematic

Note: The 1-Ω resistor shown in Figure 2.4 allows you to recover from a configuration which always pulls the GPIO pin low in the MachXO4 device. If such a configuration is loaded into the MachXO4 device, the device is held in the reconfiguration state and is not able to communicate or cannot be erased to clear the error condition. To recover from this condition, remove the resistor and reprogram the device, then reinstall the resistor.

2.9. Timing Diagram for SED Operation

Figure 2.5 shows the timing diagram for the SED operation. Table 2.4 shows the SED operation timing parameters.

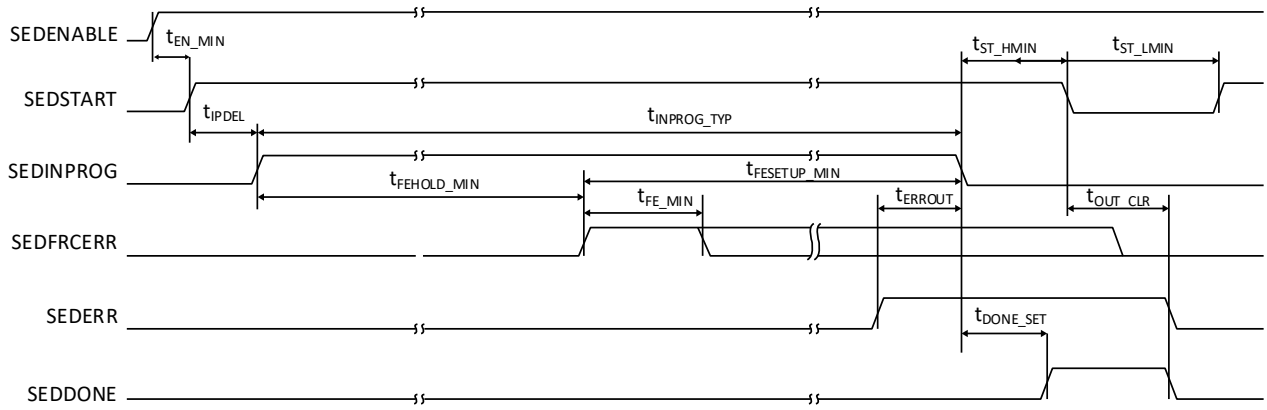

Figure 2.5. Timing Diagram for SED Operation

Table 2.4. SED Operation Timing Parameters

Parameter	Value	Unit
t_{EN_MIN}	0	SEDCLK
t_{IPDEL}	2	SEDCLK
t_{FEHOLD_MIN}	92	SEDCLK
$t_{FESETUP\_MIN}$	5	SEDCLK
t_{FE_MIN}	2	SEDCLK
t_{ERROUT}	1	SEDCLK
t_{ST_HMIN}	0	SEDCLK
t_{ST_LMIN}	1	SEDCLK
t_{OUT_CLR}	2	SEDCLK
t_{DONE_SET}	0	SEDCLK

2.10. SED Run Time

The amount of time needed to perform an SED check depends on the density of the device and the frequency of the SED clock driver signal. Some overhead time is also factored into the calculation but this is minimal. You can approximate the time required using the following formula:

(Maxbits / 8) / SED Clock Driver Frequency = Time (ms)

Maxbits is in kbits and depends on the density of the PLD (see Table 2.5). The SED clock driver signal frequency is shown in MHz. Time is in milliseconds.

SED checking in the MachXO4 device reads 8 bits (1 byte) on each SED clock cycle. For example, for a design using a MachXO4 device with 4300 look-up tables and an SED clock driver frequency of 7 MHz:

(972 kbits / 8) / 7.0 MHz = 17.4 ms

In this example, SED checking will take approximately 17.4 ms.

Note that the internal oscillator is used to generate the SED clock driver signal and its frequency can vary by ±5.5%.

Table 2.5. SED Run Time

Density[1]	LFMXO4-010	LFMXO4-015	LFMXO4-015 BGA256	LFMXO4-025	LFMXO4-050	LFMXO4-050 BGA400	LFMXO4-080	LFMXO4-110	Unit
	360K	360K	535K	535K	972K	1534K	1534K	2109K	
33.3 MHz	1.35	1.35	2.00	2.00	3.64	5.75	5.75	7.92	ms
3.05 MHz	14.8	14.8	21.9	21.9	39.9	62.9	62.9	86.4	ms

Note:

1. Density refers to the number of configuration bits in the device.

2.11. Sample Code

The following example code shows how to instantiate the SED primitive. In the example, the SED is always enabled and runs when the *sed_start* signal is driven high. The outputs of the SED hardware are available to route to the PLD output pins or for use in another module.

2.11.1. SED VHDL Examples

VHDL SEDFA

```
COMPONENT SEDFA
   GENERIC(
      SED_CLK_FREQ       : string    := "3.5";
      CHECKALWAYS        : string    := "DISABLED
      DEV_DENSITY        : string    := "010");
--"010", "015", "015_256P", "025", "050", "050_400P", "080", "110"

PORT(
     SEDENABLE    :     in    STD_LOGIC;
     SEDSTART     :     in    STD_LOGIC;
     SEDFRCERR    :     in    STD_LOGIC;
     SEDSTDBY     :     in    STD_LOGIC;
     SEDERR       :     out   STD_LOGIC;
     SEDDONE      :     out   STD_LOGIC;
     SEDINPROG    :     out   STD_LOGIC;
     SEDCLKOUT    :     out   STD_LOGIC);
END COMPONENT;

attribute SED_CLK_FREQ : string ;
attribute SED_CLK_FREQ of SEDinst0 : label is "13.30";
attribute CHECKALWAYS : string ;
attribute CHECKALWAYS of SEDinst0 : label is "DISABLED" ;
attribute DEV_DENSITY : string ;
attribute DEV_DENSITY of SEDinst0 : label is "1300L" ;


SEDinst0:   SEDFA
-- synthesis translate_off
      GENERIC MAP ( SED_CLK_FREQ => “13.30”;
           CHECKALWAYS => "DISABLED" ;
           DEV_DENSITY => "1300L" )

-- synthesis translate_on
PORT MAP   (SEDENABLE => '1',
         SEDSTART => sed_start,
         SEDFRCERR => '0',
         SEDSTDBY => sed_stdby,
         SEDERR => sed_err,
         SEDDONE => sed_done,
         SEDINPROG => sed_active,
         SEDCLKOUT => sed_clkout);
```

2.11.2. SED Verilog Examples

Verilog SEDFA

```
module SEDFA (SEDENABLE, SEDSTART, SEDFRCERR, SEDSTDBY, SEDERR, SEDDONE,
SEDINPROG, SEDCLKOUT);

input    SEDENABLE, SEDSTART, SEDFRCERR, SEDSTDBY;
output   SEDERR, SEDDONE, SEDINPROG, SEDCLKOUT;

parameter SED_CLK_FREQ = "3.5";
parameter CHECKALWAYS = "DISABLED";
parameter DEV_DENSITY = "010";
//"010", "015", "015_256P", "025", "050", "050_400P", "080", "110"

endmodule
```

Verilog SEDFA Primitive Instantiation

```
//  instantiate SEDFA primitive module with parameter passing to SEDFA module
SEDFA # (.SED_CLK_FREQ("4.75"), .DEV_DENSITY("010"))

sedfa_tst (.SEDENABLE(1'b1), .SEDSTART(sed_start), .SEDFRCERR(1'b0), .SEDSTDBY(),
          .SEDERR(sed_err),   .SEDDONE(sed_done), .SEDINPROG(sed_active),
.SEDCLK-
OUT(sed_clkout) );
```

3. Soft Error Correction

Soft error correction (SEC) is performed with background reconfiguration. All bits are rewritten during the reconfiguration, and the erroneous bit is replaced with correct data. During background reconfiguration, writing the same value into configuration SRAM cells does not affect the SRAM cell output.

Soft error detection (SED) can be utilized in conjunction with the MachXO4 device dual-boot feature without restriction. However, SEC use is limited to the primary image only. Refer to MachXO4 Programming and Configuration User Guide (FPGA-TN-02393) for more information on the use of the dual-boot feature.

Upon completion of background reconfiguration, initialization of the device is bypassed, allowing user functions to be performed during and after reconfiguration.

Soft error correction is an optional feature of the MachXO4 device. It is used in conjunction with a sound system-wide error recovery process. When it is determined that the result of any SED within the MachXO4 user design can be contained, whether in the data or control planes, then SEC may be enabled. Refer to MachXO4 Programming and Configuration User Guide (FPGA-TN-02393) for more information about configuring a user design for SEC and background reconfiguration. Alternatively, if it is necessary or desired to re-initialize the user design to a known state in response to a SED event, then SEC should remain disabled to allow for warm-boot recovery.

To use SEC:

1. Select or enable the BACKGROUND_RECONFIG option in the Lattice Radiant Device Constraint Editor when you generate the bitstream.
2. Set the synthesis option to disable distributed RAM during the synthesize process.

Once soft error is detected, either inserted or by radiation, there are multiple ways to correct it as follows:

- Provide the original bitstream in background mode through an external sysCONFIG target port (for example, JTAG or I2C).
- Transfer the original bitstream from internal or external flash memory by issuing the sysCONFIG REFRESH command through an external sysCONFIG target port.
- Transfer the original bitstream from internal or external flash memory, initiated through external PROGRAMN pin assertion.

The following sections describe two examples of how to initiate SEC.

3.1. SPI Controller Code

If you are using the SPI controller mode to download bitstream from an external SPI flash to the MachXO4 device, you can issue a refresh instruction or pulse the PROGRAMN pin to reload the original bitstream from the SPI flash. The DONE pin goes low during configuration and goes back high after configuration.

Power cycle the device or issue offline mode instructions to the device to reinitialize the device or exit background mode.

Figure 3.1 shows the MachXO4 device controller SPI port to SPI flash.

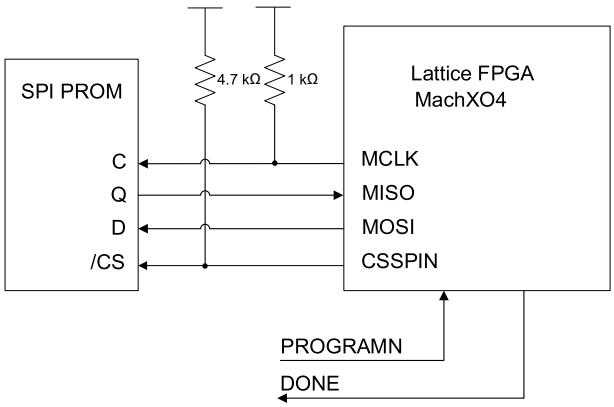

Figure 3.1. MachXO4 Device Controller SPI Port with SPI Flash

References

- MachXO4 Programming and Configuration User Guide (FPGA-TN-02393)
- MachXO4 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 1.0, December 2025

Section	Change Summary
All	Initial release.

www.latticesemi.com