Implementing JOTP-051-Compliant Safety Features in Lattice FPGAs

Technical Note

FPGA-TN-02150-1.1

March 2021

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 and with all faults, and all risk associated with such information is entirely with Buyer. Buyer shall not rely on any data and performance specifications or parameters provided herein. 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. No Lattice products should be used in conjunction with mission- or safety-critical or any other application in which the failure of Lattice's product could create a situation where personal injury, death, severe property or environmental damage may occur. 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.

Contents

Figures

Tables

Acronyms in This Document

A list of acronyms used in this document.

Acronym	Definition
ASIC	Application-Specific Integrated Circuit
CPLD	Complex Programmable Logic Device
CRC	Cyclic Redundancy Check
FPGA	Field Programmable Gate Array
GSR	Global Set-Reset
JOTP	Joint Ordinance Test Procedure
LMMI	Lattice Memory Mapped Interface
MCU	Microprocessor Unit
MPU	Microcontroller Unit
PLD	Programmable Logic Device
SED	Single Event Detection
SF	Safety Feature
SPI	Serial Peripheral Interface
SRAM	Static Random Access Memory

1. Introduction

Today's defense systems often involve mission-critical and safety-critical functions. Field programmable gate arrays (FPGAs) are ideal for use in many of these systems, but developers must ensure that these devices are properly configured before they start operating and that they remain properly configured during operation.

There are several methods and specifications that may be used to verify an FPGA's configuration bitstream, both upon power-up and during operation. One such specification is the Joint Ordinance Test Procedure (JOTP-051) defined by U.S. Department of Defense.

JOTP-051 is a technical manual for the use of logic devices in safety applications. It includes specific guidelines to minimize unintentional and/or unrecognized modes of operation, including failure modes.

This document describes the implementation of JOTP-051-compliant safety features in Lattice FPGAs.

2. JOTP-051 Requirements

JOTP-051 defines *logic devices* as including, but not limited to, *programmable logic devices (PLDs), complex programmable logic devices (CPLDs), field programmable gate arrays (FPGAs), application specific integrated circuits (ASICs), microcontrollers, discrete logic, and others*.

JOTP-051 notes that there are many safety issues and requirements involved with the use of logic devices. Some are addressed by MIL-STD-1316, MIL-STD-1911, MIL-STD-1901 and STANAG-4187, STANAG4497, STANAG-4368. JOTP-051 is intended to clarify these requirements as applied to Safety Features (SFs) implemented with logic devices.

With regard to SFs implemented in FPGAs -- specifically with regard to the FPGA's configuration memory -- Section 2 of Appendix A states: *…a method of validating the integrity of the memory shall be performed prior to executing the safety function. The memory must be validated with the rigor equivalent to, or better than, that of a 16-bit Cyclic Redundant Check (CRC16). This computed result shall be externally compared against a known value that is stored externally….*

3. Implementing JOTP-051-Complient Safety Features in Lattice FPGAs

Lattice offers a wide variety of flash-based (non-volatile) and SRAM-based (volatile) FPGAs that meet the requirements of JOTP-051.

Lattice FPGAs -- including non-volatile MachXO2, MachXO3, and MachXO3D devices -- allow the configuration bitstream to be read out through any external slave configuration interface. Alternatively, the configuration can be accessed through an internal Wishbone configuration interface and passed to the programmable logic fabric.

In the case of the non-volatile FPGAs, the configuration bitstream resides in on-chip flash. Upon power-up, the configuration bits are transferred to on-chip SRAM and checked for integrity prior to becoming active.

In order to address the JOTP-051 specification, when the FPGA is first powered-up, a CRC associated with the configuration memory must be generated and compared to an external CRC value to ensure the integrity of the configuration.

A common configuration is for the FPGA to be connected to an external microcontroller. In such a case, the CRC associated with the configuration memory can either be generated on-chip using a soft CRC calculator, or off-chip using the external microprocessor unit (MPU) or microcontroller unit (MCU).

Once the external MPU/MCU compares the CRCs, it can either release the FPGA to commence normal operation or it can halt the FPGA and apply some form or remediation, such as reloading the configuration, for example.

Table 3.1 provides a summary of available options to meet the configuration bitstream integrity.

Table 3.1. CRC Calculation Options

Mode	Interface	MachXO2™/ MachXO3™/ MachXO3D™	CrossLink™-NX	Certus™-NX	ECP5™	LatticeECP3™
External CRC Calculation	Requires direct read of SRAM bits through external JTAG SPI ports	JTAG/SPI	JTAG/SPI	JTAG/SPI	JTAG/SPI	JTAG
Internal CRC Calculation	Requires direct read of SRAM bits through external WISHBONE/LMMI ports	WISHBONE	LMMI	LMMI	JTAG/SPI[1]	JTAG/SPI[2]

Notes:

1. Requires external loopback read of JTAG/SPI
2. Requires external loopback read of JTAG

3.1. External CRC Calculation

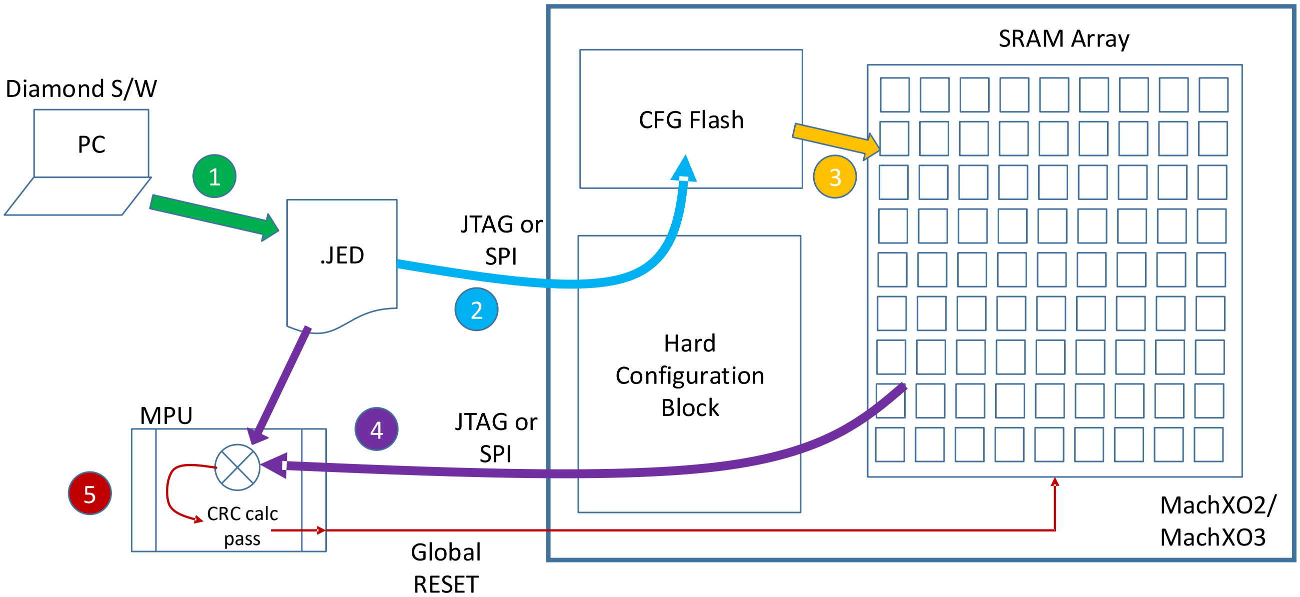

Figure 3.1. External CRC Calculation Flow

3.2. Internal Soft CRC Calculation

In this mode, the user generated FPGA image (1) is stored in the configuration flash (2). Upon power-up, the flash memory contents are transferred to the configuration SRAM while the entire FPGA array is in reset mode (3).

In the case of an internal CRC calculation, an on-chip soft IP CRC calculator function reads the entire configuration SRAM array via the internal Wishbone or similar (LMMI) interface. The internally calculated CRC is read out through an external interface to the external MPU/MCU (5) where the CRC values are compared. Upon successful comparison, the external controller releases the global reset and normal operation resumes (6).

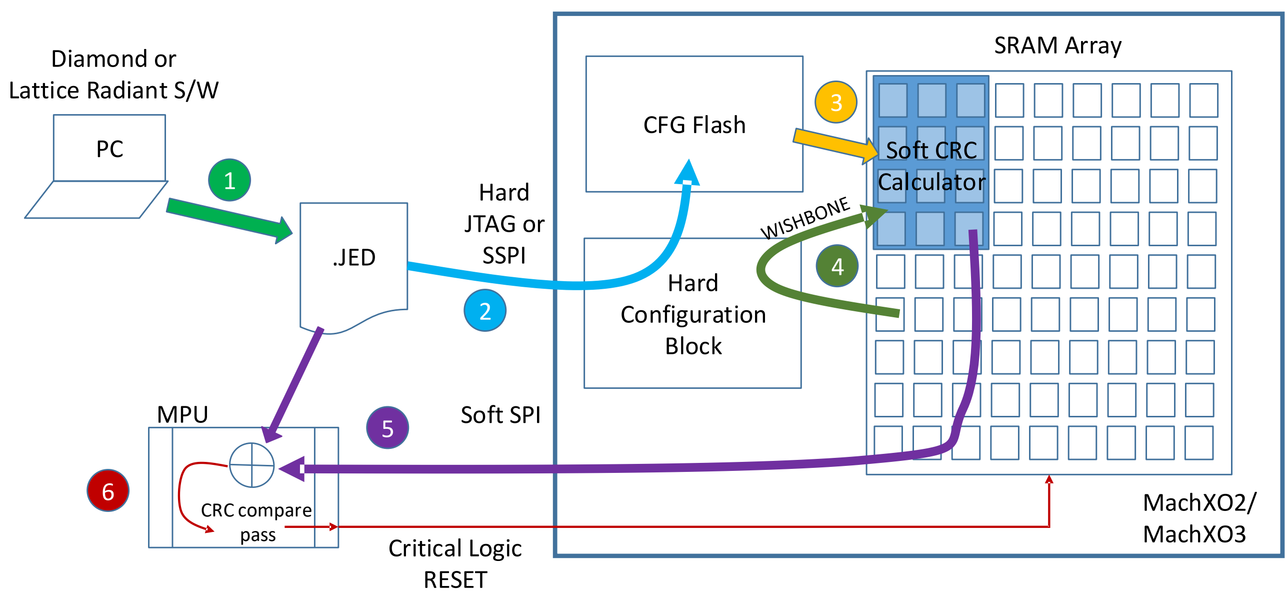

Figure 3.2. Internal Soft CRC Calculation Flow

3.2.1. FPGA Fabric Interface

To enable internal soft IP CRC calculation, Lattice provides a reference design to users to simplify implementation. This section describes the user logic interface to this reference design. Users are required to use the reference design as is to ensure proper operation. This CRC calculator logic block is optimized for Lattice MachXO2 and MachXO3 Non-volatile FPGA devices.

3.2.2. Interface Description

The Soft CRC calculator is a pre-compiled soft block implemented in the FPGA fabric array. When operated, it reads the SRAM configuration data, calculates the CRC value and makes the 32 bit value and status information available on a read-only SPI port. The design intent is for the balance of the user logic to be held in reset by an external controller until the CRC calculation is finished and the result CRC value retrieved and checked against an expected value.

Control signals are provided to operate the Soft CRC calculator automatically at Power-On, or anytime on demand. A read-only SPI port is provided for reading the result. The user logic is expected to implement a Function Enable input to allow an external controller to enable the primary user function after the CRC is successfully checked. Function Enable can be logically connected to the FPGA's Global Set-Reset (GSR) resource.

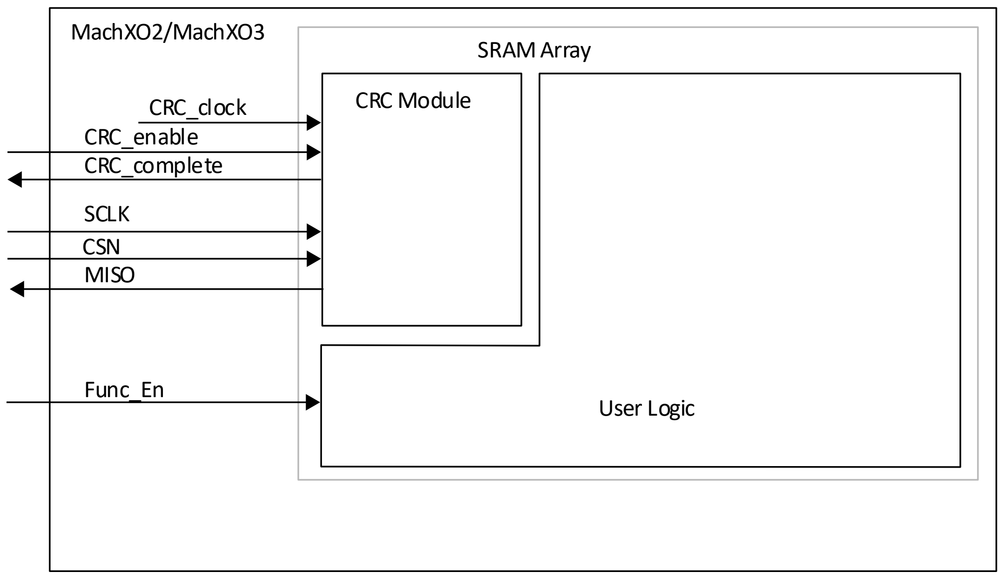

Figure 3.3. Soft CRC Interface Block Diagram

Refer to Table 3.1 for a detailed description of the signals.

Table 3.2. Soft CRC Interface Port Description

Signal	Description
CRC_clock	Input Clock can be driven from External or Internal (OSCH) clock source.
CRC_enable	Logic '1' (High) starts the CRC calculation. Can be tied to '1' externally or asserted after power-up. Logic '0' (Low) interrupts and resets the calculation.
CRC_complete	Asserts 'high' when calculation is complete. Calculation requires xxx CRC_clock periods after CRC_enable is asserted.
Func_En	User Logic Function enable. Hold low (0) until external controller extracts and validates calculated CRC. May be connected to user logic as active-low asynchronous reset or active high synchronous enable.
SCLK, CSN, MISO	SPI based read interface. Read 40 bits: 8 bits flag + 32 bits CRC. Data is valid when CRC_complete is 'high'.

3.2.3. Timing

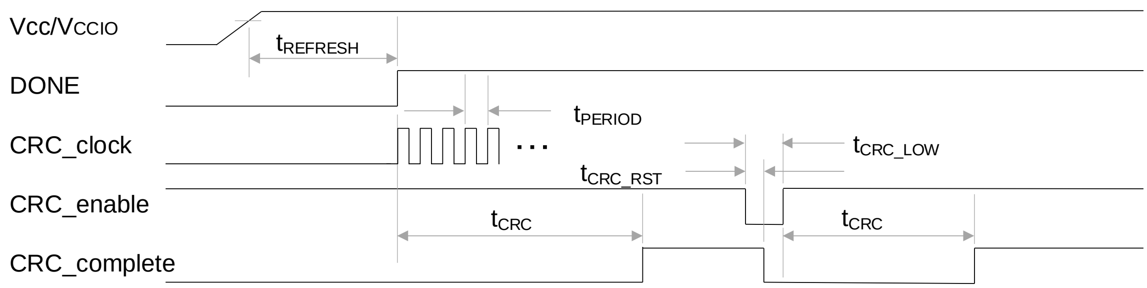

Figure 3.4. Interface Timing Diagram

Table 3.3. Timing Parameters

Parameter	Min (ns)	Max (ns)	Description
t_{PERIOD}	12.5	—	Equivalent to Fmax = 80 MHz
t_{CRC_RST}	—	5	Reset response time
t_{CRC_LOW}	3 X t_{PERIOD}	—	Minimum low pulse
t_{CRC}	See Table 3.4.		CRC Calculation time

Table 3.4. Soft CRC Calculation Times

MachXO2/MachXO3/MachXO3D		ASR Size (Frame)	DSR Size (Bit)	Number of Cycles	t_{CRC}*	Example t_{CRC} (ms)[1]
Device Size	256	186	504	35,460	35,460 * t_{PERIOD}	0.709
	640	215	888	72.045	72,045 * t_{PERIOD}	1.441
	1200	333	1080	135,387	135,387 * t_{PERIOD}	2.708
	2000	420	1272	200,934	200,934 * t_{PERIOD}	4.019
	4000	623	1560	365,157	365,157 * t_{PERIOD}	7.303
	7000	770	1992	576,054	576,054 * t_{PERIOD}	11.521
	10000	888	2376	792,216	792,216 * t_{PERIOD}	15.844

***Note:** When t_{PERIOD} = 20 ns (50 MHz)

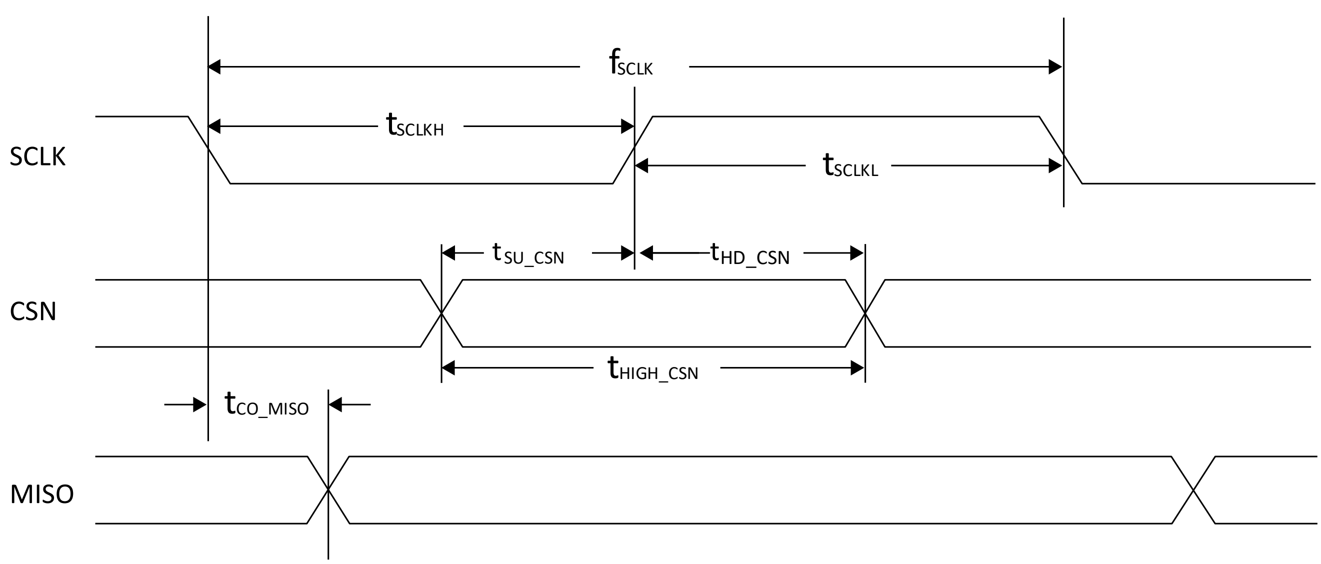

Figure 3.5. SPI Interface Timing

Table 3.5. SPI Interface Timing Parameters

Parameter	Min (ns)	Max (ns)		Description
f_{SCLK}	—	25	MHz	SPI clock frequency
t_{SCLKH}	19	—	ns	SPI clock pulse width high
t_{SCLKL}	19	—	ns	SPI clock pulse width low
t_{SU_CSN}	2	—	ns	SPI chip select setup time
t_{HD_CSN}	2	—	ns	SPI chip select hold time
t_{HIGH_CSN}	80	—	ns	SPI chip select high time
t_{CO_MISO}	—	16	ns	SPI clock falling edge to valid data output

SPI port read out data format:

Table 3.6. SPI Read Data Format

Bit[39]	Bit[38:33]	Bit[32]	Bit[31:0]
CRC_complete	Reserved	CRC_busy	CRC_checksum

Note: CRC Checksum (Bit[31:0]) is only valid when CRC_COMPLETE (Bit[39]) is 1.

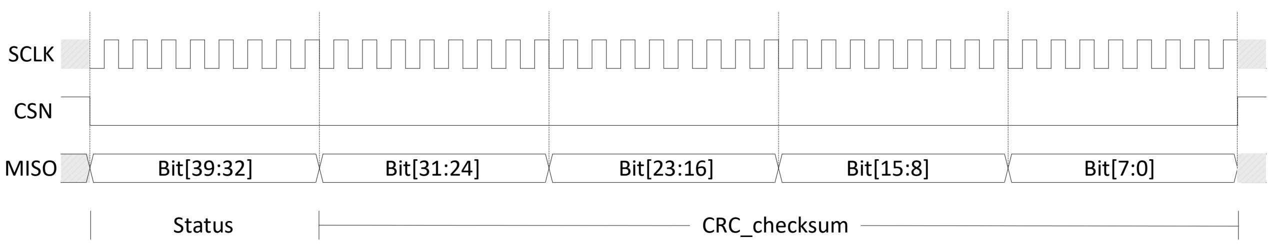

Figure 3.6. SPI Port Read Out Timing Diagram

3.2.4. Resource Utilization

Table 3.7. Resource Utilization, by Device

Device	LUTs	I/O
LCMXO3L/LF-640	268	7
LCMXO3L/LF-6900	268	7
LCMXO3L/LF-9400	268	7

3.3. Run-Time Safety

As described in the New FPGA Process from Lattice is Ideal for Military/Defense Applications whitepaper, select Lattice FPGAs are equipped with dedicated hard Single Event Detection (SED) circuits. During normal operation, Lattice SED circuitry, which is based on dedicated 32-bit CRC blocks, continuously verifies the bitstream CRC value.

You can launch the SED function internally using the Wishbone interface or externally through a slave configuration port such as JTAG or SPI.

Internal to the FPGA, the SED block can run periodically or continuously under user command and raise a warning flag when the expected results are not met. Based on this flag, an external MPU/MCU can address the problem, for example, by halting the FPGA and reloading the configuration.

For more detail about this functionality, refer to the CrossLink-NX Soft Error Detection (SED)/Correction (SEC) Usage Guide (FPGA-TN-02076).

3.4. Retrieving CRC Checksum from Bitstream File

In order to retrieve the software calculated SED CRC checksum from the bitstream file, the SEDC primitive has to be instantiated in the user design (even it is not used), this triggers the software to calculate and include the CRC checksum in the bitstream. There is a LSC_PROG_SED_CRC command with the SED CRC checksum data at the end of the bitstream file. To extract the expected SED CRC checksum from the bitstream file, launch the Diamond Programming File Utility and open the bitstream file (.bit file). It shows the bitstream file in readable ASCII format. At end of the file, you can find the *SED Address Frame*, as shown in Figure 3.7, where the A2 000000 is the command and operand for LSC_PROG_SED_CRC command and the actual CRC checksum is E00071DE.

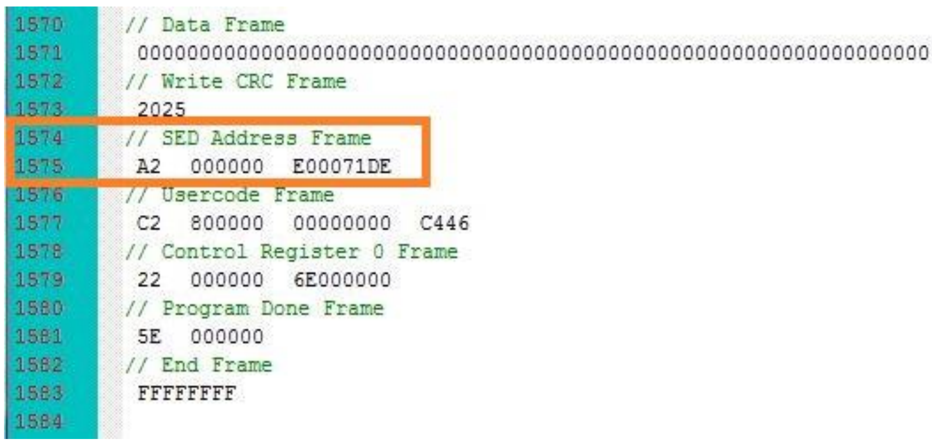

Figure 3.7. Bitstream File Sample

The SED CRC checksum only cover the FPGA data frame. To manually calculate it from the bitstream file, you have to parse through the bitstream file to extract the FPGA configuration data for each data frame, and feed them into the CRC engine with polynomial : $x^{32}+x^{26}+x^{23}+x^{22}+x^{16}+x^{12}+x^{11}+x^{10}+x^{8}+x^{7}+x^{5}+x^{4}+x^{2}+x+1$.

If instantiating the SEDC primitive solely for the purpose of generating a checksum in the bitstream file, do the following:

1. Tie the SEDENABLE input port to logic *1* and all other inputs to logic *0*. The output ports may be left unconnected.
2. Apply a constraint to keep the module from being removed or optimized by the synthesis engine. For example, */*synthesis syn_noprune=1*/* or similar.

References

- New FPGA Process from Lattice is Ideal for Military/Defense Applications
- CrosskLink-NX Soft Error Detection (SE,ccD)/Correction (SEC) Usage Guide (FPGA-TN-02076)

Technical Support Assistance

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

Revision History

Revision 1.1, March 2021

Section	Change Summary
Implementing JOTP-051-Compliant Safety Features in Lattice FPGAs	Added Retrieving CRC Checksum from Bitstream File section.

Revision 1.0, September 2020

Section	Change Summary
All	Initial release.