sysCONFIG User Guide for Nexus Platform

Technical Note

FPGA-TN-02099-3.6

February 2026

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.

Contents

Contents 3
Abbreviations in This Document 11
1. Introduction 12
2. Features 13
3. Definition of Terms 14
4. Configuration Details 15
4.1. Bitstream/PROM Sizes 15
4.2. sysCONFIG Ports 16
4.3. Configuration Ports Arbitration 17
4.4. sysCONFIG Pins 19
4.4.1. PROGRAMN 20
4.4.2. INITN 20
4.4.3. DONE 21
4.5. Master SPI sysCONFIG Pins 22
4.5.1. MCLK 22
4.5.2. MCSN 28
4.5.3. MOSI/MD0 28
4.5.4. MISO/MD1 28
4.5.5. MD2 28
4.5.6. MD3 28
4.5.7. MCSNO/MSDO 28
4.6. Slave sysCONFIG Pins 29
4.6.1. TCK/SCLK 29
4.6.2. TMS/SCSN 29
4.6.3. TDI/SI/SD0 29
4.6.4. TDO/SO/SD1 29
4.6.5. SD2/SCL 30
4.6.6. SD3/SDA 30
4.7. PERSISTENT 30
5. Master Port Configuration Process and Flow 31
5.1. Power-Up Sequence 32
5.2. Initialization 32
5.3. Configuration 33
5.4. Wake-Up 33
5.5. Early I/O Release (EIO) 34
5.6. User Mode 34
5.7. Clearing the Configuration Memory and Re-initialization 34
5.8. Reconfiguration Priority 34
6. Configuration Modes 35
6.1. Master SPI Modes 35
6.1.1. Method to Enable the Master SPI Port 36
6.1.2. Dual-Boot and Multi-Boot Configuration Modes 37
6.1.3. Ping-Pong Boot 38
6.1.4. Dual and Quad Master SPI Read Mode 39
6.2. Slave SPI Mode 46
6.2.1. Method to Enable the Slave SPI Port 48
6.2.2. Specifications and Timing Diagrams – Slave SPI Port Waveforms 49
6.2.3. Slave SPI Port AC Timing Requirements 49
6.2.4. Dual and Quad Slave SPI Port 50
6.2.5. Slave SPI Configuration Flow Diagrams 50
6.2.6. Command Waveforms 52
6.2.7. Slave SPI to Master SPI Bridge 55
6.3. Slave I^2C/I3C Mode 56

6.3.1. Bus Sharing Between I2C and I3C 57
6.3.2. Slave I2C/I3C Configuration Mode 57
6.3.3. Slave Address for I2C and I3C Ports 58
6.3.4. Method to Enable the Slave I2C/I3C Port 58
6.3.5. Slave I2C/I3C Configuration Logic Access 60
6.3.6. Slave I2C/I3C Configuration Flow Diagrams 61
6.3.7. Slave I2C/I3C AC Timing Requirements 63
6.4. JTAG Mode 64
6.4.1. Method to Enable the JTAG Port 64
6.4.2. JTAG Port AC Timing Requirements 64
6.4.3. JTAG to Master SPI Bridge 64
6.4.4. JTAG ispTracy/Reveal Support 65
6.5. Device Status Register 67
6.6. Control Register 0 (CR0) 69
6.7. Control Register 1 (CR1) 71
6.8. TransFR Operation 72
6.9. Slave Command Support 73
6.9.1. Slave Command Format 73
6.9.2. Slave Command Set 73
6.9.3. Slave Command Details 77
6.10. JTAG Instruction Support 89
7. Software Selectable Options 96
7.1. SLAVE_SPI_PORT 97
7.2. MASTER_SPI_PORT 98
7.3. SLAVE_I2C_PORT/SLAVE_I3C_PORT 98
7.4. JTAG_PORT 98
7.5. DONE_PORT 98
7.6. INITN_PORT 98
7.7. PROGRAMN_PORT 99
7.8. BACKGROUND_RECONFIG 99
7.9. DONE_EX 99
7.10. DONE_OD 100
7.11. MCLK Frequency 100
7.12. TRANSFR 100
7.13. CONFIGIO_VOLTAGE_BANK0/CONFIGIO_VOLTAGE_BANK1 100
7.14. CONFIG_IOSLEW 100
7.15. CONFIG_SECURE 101
7.16. WAKE_UP 101
7.17. COMPRESS_CONFIG 101
7.18. EARLY_IO_RELEASE 101
7.19. BOOTMODE 101
7.20. USERCODE_FORMAT 101
7.21. USERCODE 102
7.22. UNIQUE_ID 102
8. Device Wake-Up Sequence 103
8.1. Wake-Up Signals 103
8.2. Wake-Up Clock 104
9. Daisy Chaining 105
9.1. Flow Through Option 105
9.2. Using Radiant Deployment Tool to Create a sysCONFIG Daisy Chain Hex File 106
9.3. MSPI Configuration Mode in Flow Through Mode 111
9.4. SSPI Configuration Mode in Flow Through Mode 112
Appendix A. Nexus Slave SPI Programming Guide 114
Appendix B. Nexus Bitstream File Format 115

B.1. Configuration Bitstream Format 115
B.2. Read Back Bitstream Format 126
B.3. Nexus Device Specific 127
Appendix C. Modifying the Nexus Feature Row 128
C.1. Breakdown of Feature Row Bits 128
C.2. Modifying the Feature Row Programming File 130
C.3. Modifying the Feature Row of a Programmed Device 131
C.3.1. Modifying the Feature Row Using the .fea Programming File 131
C.3.2. Modifying the Feature Row without Using the .fea Programming File 132
C.4. Programming No CDM Bit in CR0 134
C.5. Comparison of Feature Row Programming File (.fea) 134
Appendix D. Configuration Access from User Logic 135
D.1. CONFIG_LMMI and CONFIG_LMMI Primitive 135
D.2. CONFIG_LMMI Primitive and OSC/OSC for CRE IP Connection and Instantiation 136
D.3. CONFIG_LMMI Access Protocol 138
D.4. CONFIG_LMMI Supported Commands 140
Appendix E. Dual Boot Fail Safe Upgrade 142
E.1. Fail Safe Upgrade Process 142
E.2. Fail Safe Upgrade with Radiant Programmer 143
E.3. Bitstream Generation for Fail Safe Upgrade 144
E.3.1. Generate the Fail Safe Bitstream from Radiant 144
E.3.2. Converting the Original Bitstream using Deployment Tool 145
Appendix F. Enabling 32-bit (4-byte) MSPI Address and Command Mode (Dual-Boot/Multi-Boot/Ping-Pong Boot) 148
F.1. Setting 32-bit MSPI Address and Command in CR1 Through Bitstream 148
F.2. Setting No CDM in CR0 Through Bitstream 152
F.3. Setting 32-bit SPI Address and Command in Non-Volatile Configuration Memory 154
Appendix G. Modifying sysCONFIG Options in the Lattice Radiant Device Constraint Editor without Re-running Place and Route Design 155
G.1. Through Bitstream Strategy Settings Command Line 155
G.2. Through Tcl Command 156
Appendix H. Dry Run User Image 159
H.1. Dry Run the Bitstream Stored in External Flash 159
H.2. Dry Run the Bitstream Sent through sysCONFIG Ports 160
H.3. INITN Pin Behavior 160
H.4. Limitation 160
References 161
Technical Support Assistance 162
Revision History 163

Figures

Figure 4.1. Configuration Control Flow 18
Figure 4.2. Configuration from PROGRAMN Timing 20
Figure 4.3. Configuration Error Notification 21
Figure 4.4. Device Properties Window (Program Control Register 1) 24
Figure 4.5. Control Register 1 Window (Setting RX Edge) 25
Figure 4.6. Confirm Overwrite Register Dialog Box (Setting RX Edge) 25
Figure 4.7. Device Properties Window (Display Control Register 1) 26
Figure 4.8. Control Register 1 Window (Confirming RX Edge) 26
Figure 4.9. Control Register 0 Window (Setting No CDM) 27
Figure 4.10. Confirm Overwrite Register (Setting No CDM) 27
Figure 5.1. Master Port Configuration Flow 31
Figure 5.2. Configuration from Power-On-Reset Timing 32
Figure 6.1. Nexus Master SPI Port with SPI Flash 35
Figure 6.2. Jump Table 38
Figure 6.3. One Dual SPI Flash Interface (Dual) 39
Figure 6.4. One Quad SPI Flash Interface (Quad) 40
Figure 6.5. Programmer Icon from the Toolbar 40
Figure 6.6. Create New or Open Existing Project Window 40
Figure 6.7. Deployment Tool 41
Figure 6.8. Function Type and Output File Selections 41
Figure 6.9. Input File Selection 41
Figure 6.10. SPI Flash Settings 42
Figure 6.11. Output File Selection 42
Figure 6.12. Deployment Tool Summary and Programming File Generation 43
Figure 6.13. Programmer Icon from the Toolbar 43
Figure 6.14. Create New or Open Existing Project Window 44
Figure 6.15. Scan Device 44
Figure 6.16. Device Status 44
Figure 6.17. Device Properties Window 45
Figure 6.18. Program Device Icon on the Toolbar 45
Figure 6.19. Enable Quad Mode Operation 46
Figure 6.20. Nexus Slave SPI Port with CPU and Single or Multiple Devices 47
Figure 6.21. Nexus Slave SPI Port with SPI Flash 48
Figure 6.22. Slave SPI Read Waveforms 49
Figure 6.23. Slave SPI Write Waveforms 49
Figure 6.24. Slave SPI Configuration Flow 51
Figure 6.25. Class A Command Waveforms 52
Figure 6.26. Class B Command Waveforms 53
Figure 6.27. Class C Command Waveforms 53
Figure 6.28. Class D Command Waveforms 54
Figure 6.29. SSPI to MSPI Bridge Functional Diagram 55
Figure 6.30. SSPI to MSPI Bridge Block Diagram 55
Figure 6.31. Nexus Slave I^2C/I3C Port with CPU and Single or Multiple Devices 56
Figure 6.32. Bus Sharing between I^2C and I3C 57
Figure 6.33. Slave I^2C Configuration Activation Flow 59
Figure 6.34. Slave I3C Configuration Activation Flow 59
Figure 6.35. Typical I^2C Write 60
Figure 6.36. Typical I^2C Read 60
Figure 6.37. Typical I3C Read with 7'h7E 60
Figure 6.38. Typical I3C Read without 7'h7E 60
Figure 6.39. Typical I3C Write with 7'h7E 61
Figure 6.40. Typical I3C Write without 7'h7E 61

Figure 6.41. Slave I2C/I3C Configuration Flow 62
Figure 6.42. JTAG to MSPI Bridge Block Diagram 65
Figure 6.43. JTAG ispTracy Interface 66
Figure 6.44. Segmented Bitstream Burst Timing Waveforms 79
Figure 6.45. Continuous Bitstream Burst 80
Figure 6.46. Segmented Bitstream Burst with No Extra Clock Edge on SCL 80
Figure 6.47. Segmented Bitstream Burst when Extra SCL Edge Unavailable 80
Figure 6.48. 32-bit Hard IP Example; Frame size =4, Word size = 32 87
Figure 6.49. 10-bit EBR Example; Frame size = 5, Word size = 10 87
Figure 7.1. sysCONFIG Preferences in Global Tab, Lattice Radiant Device Constraint Editor 96
Figure 8.1. Wake-up Sequence Using Internal Clock 103
Figure 9.1. Creating New Deployment for sysCONFIG Daisy Chain Hex File 106
Figure 9.2. Select Input File(s) Window 107
Figure 9.3. sysCONFIG Daisy Chain Options Window 108
Figure 9.4. Select Output File(s) Window 109
Figure 9.5. Generate Deployment Window 110
Figure 9.6. Daisy Chain Configuration with Master SPI in Flow Through Mode 111
Figure 9.7. Daisy Chain Configuration with Slave SPI in Flow Through Mode 112
Figure 9.8. Slave SPI Daisy Chain Operation Sequence 113
Figure C.1. 32-bit NV CR1 Bits 128
Figure C.2. 96-bit FEATURE Bits 129
Figure C.3. 16-bit FEABITS 129
Figure C.4. Feature Row Programming Setup 131
Figure C.5. Feature Row Pseudo Programming Setup 131
Figure C.6. Update Feature Row User Interface 132
Figure C.7. Feature Row Bit Examples 133
Figure C.8. Control Register 0 Programming for No CDM Bit 134
Figure C.9. .fea File Sample 134
Figure D.1. CONFIG_LMMI Primitive Pin Diagram 135
Figure D.2. CONFIG_LMMI Primitive and OSC/OSC for CRE IP Connection (hf_clk_out_o as Clock Source) 136
Figure D.3. CONFIG_LMMI Primitive and OSC/OSC for CRE IP Connection (Alternative Clock Source) 137
Figure D.4. LMMI Class A Commands Execution 138
Figure D.5. LMMI Class B Commands Execution 139
Figure D.6. LMMI Class C Commands Execution 139
Figure E.1. SysCONFIG Chain Setup Example 142
Figure E.2. Fail Safe Upgrade with Radiant Programmer 144
Figure E.3. Strategy Setup for Bitstream Generation 145
Figure E.4. Fail Safe Bitstream Conversion: Launch Deployment Tool 145
Figure E.5. Fail Safe Bitstream Generation: Input Files Setup 145
Figure E.6. Fail Safe Upgrade Bitstream Generation: Option Setup 146
Figure E.7. Fail Safe Upgrade Bitstream Generation: Select Output File Name 146
Figure E.8. Fail Safe Upgrade Bitstream Generation: Generate the mcs file 147
Figure F.1. Control Register 1 Window 148
Figure F.2. Open Data File Window (Selecting .bit File) 149
Figure F.3. Control Register 1 Settings 149
Figure F.4. Changing 32-bit MSPI Commands and 32-bit MSPI Address Option Bit Settings 150
Figure F.5. Save As Window 150
Figure F.6. Write to File Successful Dialog Box 151
Figure F.7. Control Register 0 Window 152
Figure F.8. Open Data File Window (Selecting Modified .bit File) 152
Figure F.9. No CDM Option Bit Setting 153
Figure F.10. Changing No CDM Option Bit Setting 153
Figure F.11. Overwrite Dialog Box 154
Figure F.12. 32-bit SPIM Address and 32-bit SPIM Commands Option Bits in Nexus Feature Row 154

Figure G.1. Selecting Bitstream Settings Strategy 155
Figure G.2. Adding bitgen Option in Command Line Options Field 156
Figure G.3. Generating .bit File 156
Figure G.4. Implementation Folder 156
Figure G.5. Copying Bitgen Command from .bgn File 157
Figure G.6. Copying bitgen Command into Tcl Console 157
Figure G.7. Verifying sysCONFIG Option in .bgn File 158

Tables

Table 4.1. Maximum Configuration Bits 15
Table 4.2. Nexus Programming and Configuration Ports 16
Table 4.3. Slave Configuration Port Activation Key 17
Table 4.4. Default State of the sysCONFIG Pins 19
Table 4.5. Nexus MCLK Valid Frequencies 22
Table 4.6. sysCONFIG Pins Global Preferences[1] 30
Table 6.1. Master SPI Configuration Port Pins 35
Table 6.2. Bitstream Read Command Codes with MSPI Controller in 24-bit and 32-bit Addressing Modes 36
Table 6.3. Dual/Quad SPI Port Names 39
Table 6.4. Slave SPI Configuration Port Pins 47
Table 6.5. Slave SPI Configuration Port Activation Key 48
Table 6.6. Slave SPI Port AC Timing Requirements 49
Table 6.7. Special Commands for Dual and Quad Mode Enable/Disable 50
Table 6.8. Execution Time for ISC_ERASE Command 52
Table 6.9. Slave I2C/I3C Configuration Port Pins 56
Table 6.10. Slave I2C/I3C Configuration Port Activation Key 59
Table 6.11. Slave I2C/I3C Maximum Frequency Requirements 63
Table 6.12. JTAG AC Timing Requirements 64
Table 6.13. 64-bit Device Status Register 67
Table 6.14. Bit Definition for Control Register 0 69
Table 6.15. Bit Definition for Control Register 1 71
Table 6.16. Slave Command Support 73
Table 6.17. Slave Configuration Interface Command 73
Table 6.18. OPERAND1 Definition for LSC_DEVICE_CTRL Command 77
Table 6.19. OPERAND1 Definition for LSC_REFRESH Command 78
Table 6.20. OPERAND1 Definition for ISC_ENABLE Command 78
Table 6.21. Address Counter Field Definitions 81
Table 6.22. OPERAND Definition for LSC_PROG_INCR Command 82
Table 6.23. OPERAND Definition for LSC_READ_INCR Command 82
Table 6.24. Data Fields Definition for LSC_PROG_FEATURE Command 84
Table 6.25. Data Fields Definition for LSC_PROG_FEABIT Command 85
Table 6.26. Data Fields Definition for LSC_PROG_OTP Command 85
Table 6.27. Data Packet for LSC_INIT_BUS_ADDR 87
Table 6.28. Operand Definitions for LSC_INIT_BUS_WRITE 87
Table 6.29. Operand Definitions for LSC_INIT_BUS_READ 88
Table 6.30. LSC_PROG_SPI Operand 88
Table 6.31. JTAG Instruction Table 89
Table 7.1. sysCONFIG Options[1] 97
Table 7.2. BACKGROUND_RECONFIG Options 99
Table 8.1. Wake-Up Sequences 103
Table 9.1. Control Bit Settings for FPGA Device in a Daisy Chain 105
Table B.1. Nexus Compressed Bitstream Format 115
Table B.2. Nexus Uncompressed Bitstream Format – Without Early I/O Release 119
Table B.3. Nexus Uncompressed Bitstream Format – With Early I/O Release 123
Table B.4. Nexus Read Back File Format 126
Table B.5. Nexus Device Specifics 127
Table B.6. Nexus Device ID 127
Table D.1. CONFIG_LMMI Primitive Input/Output Ports1 135
Table D.2. CONFIG_LMMI Primitive Parameters[1] 136
Table D.3. CONFIG_LMMI Supported Command List 140
Table E.1. Fail Safe Upgrade Process Steps 142

Table E.2. Bitstream First Page Format 143

Abbreviations in This Document

A list of abbreviations used in this document.

Abbreviation	Definition
AES	Advanced Encryption Standard
BSE	Bitstream Engine
CDM	Control Information Download Mode
CIB	Common Interface Block
CRAM	Configuration Random Access Memory (also known as configuration SRAM or SRAM configuration memory in this document)
CRC	Cyclic Redundancy Check
EBR	Embedded Block RAM
ECDSA	Elliptic Curve Digital Signature Algorithm
FSM	Finite State Machine
HMAC	Hash-based Message Authentication Code
HSE	Hardware Security Engine/Module
I^2C	Inter-Integrated Circuit; A synchronous, multi-master, multi-slave, packet switched, single-ended, serial bus
JEDEC	Joint Electron Device Engineering Council
JTAG	Joint Test Action Group
LMMI	Lattice Memory Mapped Interface
LSB	Least Significant Bit
LUT	Look Up Table
MIB	Memory Interface Block
MSB	Most Significant Bit
MSI	Master Serial Input
MSPI	Master Serial Peripheral Interface
OTP	One Time Programmable
PCB	Printed Circuit Board
PLC	Programmable Logic Cell
POR	Power On Reset
QE	Quad Enable
RTL	Register Transfer Level
SCM	Serial Configuration Mode
SDM	Self-Download Mode
SEC	Soft Error Correction
SED	Soft Error Detection
SEI	Soft Error Injection
SER	Soft Error Rate
SFDP	Serial Flash Discoverable Parameters
SPI	Serial Peripheral Interface
SRAM	Static Random-Access Memory
SRME	Slow Response Mode Enable
SSPI	Slave Serial Peripheral Interface
TCK	Test Clock Pin
TDI	Test Data Input
TDO	Test Data Output
TMS	Test Mode Select

1. Introduction

The Lattice Nexus™ Platform is the next generation of Lattice FPGAs based on the FDSOI technology, which includes CrossLink™-NX, Certus™-NX, CertusPro™-NX, and MachXO5™-NX product families. These devices reap many benefits of the FDSOI technology like extremely low soft error rate (SER) and easy configurability of the same device for high-performance as well as low-power applications. The Nexus device has many new and unique features that make it one of the best-in-class FPGA in its device density range. The Nexus device has one of the fastest boot up times including ultrafast I/O booting capability, which moves this device a notch higher for booting compared to traditional FPGAs. This device has advanced options to enable multi-boot feature so you can easily switch between FPGA bitstreams. For more information on the MachXO5-NX device, refer to the MachXO5-NX Programming and Configuration User Guide (FPGA-TN-02271).

The configuration memory in the Nexus FPGA is built using volatile SRAM. Therefore, an external non-volatile configuration memory or external configuration engine is required to maintain the configuration data when the power is removed. This non-volatile memory or configuration engine supplies the configuration data to the Nexus device when it powers up or anytime the device needs to be updated. The Nexus device provides a rich set of features for configuring the FPGA or programming the external non-volatile memory. There are many options available for you to build the programming and configuration solution that fits your needs and each of the options available is described in detail. The waveforms presented in this document are for reference only. For detailed timing recommendations, refer to the CrossLink-NX Family Data Sheet (FPGA-DS-02049), Certus-NX Family Data Sheet (FPGA-DS-02078), CertusPro-NX Family Data Sheet (FPGA-DS-02086), and CrossLink-NX-33 and CrossLinkU-NX Data Sheet (FPGA-DS-02104).

2. Features

The following lists the key programming and configuration features of the Nexus device:

- Ultrafast early I/O (EIO) configuration for instant-on support.
- Fast full device configuration using quad SPI running at 150 MHz.
- I3C device configuration support.
- Bitstream dry-run support to ensure bitstream integrity.
- Extremely low SER due to inherent FDSOI technology.
- Easy switching between the high-performance and low-power modes using different bitstream configurations.
- Enhanced and flexible multi-boot support (32-bit addressing support with jump-table support).
- Configuration bridging for easy external SPI programming (SSPI to Master SPI bridge).
- User-selectable booting sequence.
- Bitstream encryption/decryption – 256 bits Advanced Encryption Standard (AES) (Available for the Lattice Radiant™ software V2.0SP1 or later). For detailed information about the cypher key handling and advanced security feature, refer to the Advanced Configuration Security User Guide for Nexus Platform (FPGA-TN-02176).
- Bitstream authentication – HMAC (Available for the Lattice Radiant software version 2.2 or later), for CrossLink-NX, Certus-NX, and CertusPro-NX product families. Refer to the Advanced Configuration Security User Guide for Nexus Platform (FPGA-TN-02176) for detailed information.
- Bitstream authentication – ECDSA (Available for the Lattice Radiant software version 2.2 or later), for CrossLink-NX, Certus-NX, and CertusPro-NX product families. Refer to the Advanced Configuration Security User Guide for Nexus Platform (FPGA-TN-02176) for detailed information.
- Multiple programming and configuration interfaces:
  - 1149.1 JTAG
  - Slave I2C
  - Slave I3C
  - Slave SPI, includes SERIAL, DUAL, and QUAD mode
  - Master SPI includes SERIAL, DUAL, and QUAD read mode
  - Dual boot and multi boot support
  - Ping-pong boot
  - Daisy chaining
- Transparent programming of external memory
- Readback security and encryption for design protection
- Compression of bitstreams

3. Definition of Terms

This document uses the following terms to describe common functions:

- Programming – Programming refers to the process used to alter the contents of the external configuration memory.
- Configuration – Configuration refers to a change in the state of the SRAM memory cells.
- Configuration Mode – The configuration mode defines the method the device uses to acquire the configuration data from the volatile memory.
- Configuration Data – This is the data read from the non-volatile memory and loaded into the FPGA's SRAM configuration memory. This is also referred to as a bitstream, or device bitstream.
- BIT – The BIT file is the configuration data for the device that is stored in an external SPI flash or other memory device. It is a binary file and is programmed unmodified into the SPI flash by the Lattice programming tool.
- HEX – The HEX file is also a configuration data file for the device. It is in HEX format and is normally requested by third-party programming vendors.
- Port – A port refers to the physical connection used to perform programming and some configuration operations. Ports on the Nexus device include JTAG, SPI, I^2C, and I3C physical connections.
- User Mode – The Nexus device is in user mode when configuration is complete, and the FPGA is performing the logic functions that you have programmed it to perform.
- Offline Mode – Offline mode is a term that is applied to SRAM configuration or external memory programming. When using offline mode, the FPGA no longer operates in user mode. The contents of the SRAM configuration memory or external memory device are updated. The FPGA does not perform the logic operations until offline mode programming/configuration is complete.
- Direct Mode (Foreground Mode) – The device is in a configuration mode and all the I/O pins are kept tri-stated.
- Dual Boot – Supports two configuration patterns that reside in a SPI flash device. Whenever loading failure occurs with the primary pattern, the FPGA searches for and loads a so-called *golden* or fail-safe pattern. Both patterns come from an off-chip non-volatile SPI memory.
- Ping-Pong Boot – Utilize the Jump Table to select an image for booting without changing the location of the image in SPI flash.
- Multi Boot – The FPGA determines and triggers the loading of the next pattern after a prior successful configuration. Multiple patterns (that is, more than two patterns) are available for the FPGA to choose to load on demand. All the patterns are stored in an external SPI Flash memory.
- Transparent Mode – Transparent mode, also referred to as the Background Mode is used to update the SRAM configuration while leaving the Nexus device in user mode and all I/O pins remain operational.
- Number Formats – The following nomenclature is used to denote the radix of numbers:
  - 0x: Numbers preceded by *0x* are hexadecimal
  - b (suffix): Numbers suffixed with *b* are binary
  - All other numbers are decimal
- SPI – Serial peripheral interface (SPI) bus. An industry standard, full duplex, synchronous serial data link that uses a four-wire interface. The interface supports a single master and single or multiple slaves.
- Master SPI – A configuration mode where the FPGA drives the master clock and issues commands to read the bitstream from an external SPI flash device.
- Slave SPI (SSPI) – A configuration mode where the CPU drives the clock and issues commands to the FPGA for writing the bitstream into the SRAM cells.
- Refresh – The process of re-triggering a bitstream write operation. It is activated by toggling the PROGRAMN pin or issuing a REFRESH command, which emulates the PROGRAMN pin toggling.
- Dry-Run – The process triggered by the LSC_DEVICE_CTRL command, which loads the bitstream and checks the cyclic redundancy check (CRC) of the non-volatile bits without writing the bits to the configuration SRAM (that is, it is done in the background during normal device operation), for the purpose of checking the bitstream file integrity.

4. Configuration Details

The Nexus SRAM memory contains the active configuration, essentially the *fuses* that define the behavior of the FPGA. The active configuration is, in most cases, retrieved from an external memory. The non-volatile memory holds the configuration data that is loaded into the FPGAs SRAM.

4.1. Bitstream/PROM Sizes

The Nexus devices are SRAM-based FPGAs. The SRAM configuration memory must be loaded from an external non-volatile memory that can store all the configuration data. The size of the configuration data is variable. It is based on the amount of logic available in the FPGA, and the number of pre-initialized Embedded Block RAM (EBR) components and number of pre-initialized Large RAM block (LRAM) components. A Nexus design using the largest device, with every EBR and every LRAM pre-initialized with unique data values, and generated without compression turned on requires the largest amount of storage.

Table 4.1. Maximum Configuration Bits

Device[1]	Scenario	Uncompressed	SPI Mode	
		Unencrypted/Encrypted Bitstream Size (Mb)	Recommended SPI Flash Size (Mb)	Dual Boot Recommended SPI Flash Size (Mb)
LFD2NX-9	No LRAM, No EBR,	2.817	4	8
	No LRAM, MAX EBR	3.102	4	8
	MAX LRAM, No EBR	4.437	8	16
	MAX LRAM, MAX EBR	4.722	8	16
LFD2NX-15 LFD2NX-25	No LRAM, No EBR,	4.609	8	16
	No LRAM, MAX EBR	6.247	8	16
	MAX LRAM, No EBR	5.264	8	16
	MAX LRAM, MAX EBR	6.903	8	16
LIFCL-17 LFD2NX-17	No LRAM, No EBR,	2.817	4	8
	No LRAM, MAX EBR	3.273	4	8
	MAX LRAM, No EBR	5.517	8	16
	MAX LRAM, MAX EBR	5.873	8	16
LIFCL-33	No LRAM, No EBR	4.453	8	16
	No LRAM, MAX EBR	5.967	8	16
	MAX LRAM, No EBR	7.150	8	16
	MAX LRAM, MAX EBR	8.667	16	32
LIFCL-33U	No LRAM, No EBR	5.430	8	16
	No LRAM, MAX EBR	6.740	8	16
	MAX LRAM, No EBR	8.706	16	32
	MAX LRAM, MAX EBR	10.017	16	32
LFD2NX-28	No LRAM, No EBR,	6.232	8	16
	No LRAM, MAX EBR	7.286	8	16
	MAX LRAM, No EBR	7.281	8	16
	MAX LRAM, MAX EBR	8.335	16	32
LIFCL-40 LFD2NX-40	No LRAM, No EBR	6.232	8	16
	No LRAM, MAX EBR	7.758	8	16
	MAX LRAM, No EBR	7.281	8	16
	MAX LRAM, MAX EBR	8.807	16	32

Device[1]	Scenario	Uncompressed	SPI Mode	
		Unencrypted/Encrypted Bitstream Size (Mb)	Recommended SPI Flash Size (Mb)	Dual Boot Recommended SPI Flash Size (Mb)
LFD2NX-35	No LRAM, No EBR	10.611	16	32
	No LRAM, MAX EBR	13.233	16	32
	MAX LRAM, No EBR	11.922	16	32
	MAX LRAM, MAX EBR	14.543	16	32
LFD2NX-65	No LRAM, No EBR	10.611	16	32
	No LRAM, MAX EBR	13.233	16	32
	MAX LRAM, No EBR	11.922	16	32
	MAX LRAM, MAX EBR	14.543	16	32
LFCPNX-50	No LRAM, No EBR	15.484	16	32
	No LRAM, MAX EBR	17.466	32	64
	MAX LRAM, No EBR	16.139	32	64
	MAX LRAM, MAX EBR	18.122	32	64
LFCPNX-100	No LRAM, No EBR	17.293	32	64
	No LRAM, MAX EBR	18.749	32	64
	MAX LRAM, No EBR	18.589	32	64
	MAX LRAM, MAX EBR	22.333	32	64

Notes:

1. The Nexus device supports the usage of compression and encryption, provided that these are not simultaneously implemented in a single bitstream. You must not perform encryption on a compressed bitstream, as well as compressing an encrypted bitstream because this is not supported.
2. The SPI flash size can support up to 4 Gbit when the control register (CR1) bit 17 and 14 are set to 1. See Table 6.15 for more details regarding the CR1 register.

4.2. sysCONFIG Ports

Table 4.2. Nexus Programming and Configuration Ports

Interface	Port	Description
JTAG	JTAG (IEEE 1149.1 and IEEE 1532 compliant)	4-wire or 5-wire JTAG interface
sysCONFIG™	SSPI (SERIAL, DUAL, QUAD)	Slave serial peripheral interface (SPI)
	MSPI (SERIAL, DUAL, QUAD)	Master serial peripheral interface (SPI)
	Slave I^2C	Slave I^2C configuration interface
	Slave I3C	Slave I3C configuration interface

4.3. Configuration Ports Arbitration

At Power Up or PROGRAMN pin toggle LOW (for the period of $t_{PROGRAMN}$) or REFRESH command execution, the configuration logic puts the device into master auto booting mode to boot the device from external SPI boot PROM, if the PROGRAMN pin is *HIGH*, after a brief internal initialization time (Bulk erase the Configuration SRAM and CDM downloading). Holding the PROGRAMN LOW postpones the master auto-booting event, and allow the slave configuration ports (Slave SPI or Slave I^2C) to detect a *Slave Active* condition, which means: slave port is addressed, and the pre-defined Slave Configuration Port Activation Key, as shown in Table 4.3 is received. If any slave port declares active before PROGRAMN is released HIGH, the device is activated for slave configuration. If no slave port is declared active before the PROGRAMN pin is released HIGH, the device performs master auto booting sequence.

The PROGRAMN pin becomes a user I/O pin through a user feature setting, with default PROGRAMN pin state high to achieve a Master booting-only device. The PROGRAMN pin state could be set low to achieve a slave port configuration-only device.

The data format for slave configuration port activation is shown in Table 4.3.

Table 4.3. Slave Configuration Port Activation Key

Slave Port / Activation Key	Slave Configuration Port Activation Key	
Slave SPI port	Dummy bytes[1]	32'HA4C6F48A
Slave I^2C/I3C port	Slave I^2C/I3C port address[2]	32'HA4C6F48A

Notes:

1. Dummy bytes before the activation key are optional. Only the last 32 bits shifted in matter.
2. The slave I^2C address could be either 7 bits or 10 bits address.

Figure 4.1 shows the device configuration control flow of CFG_TOP.

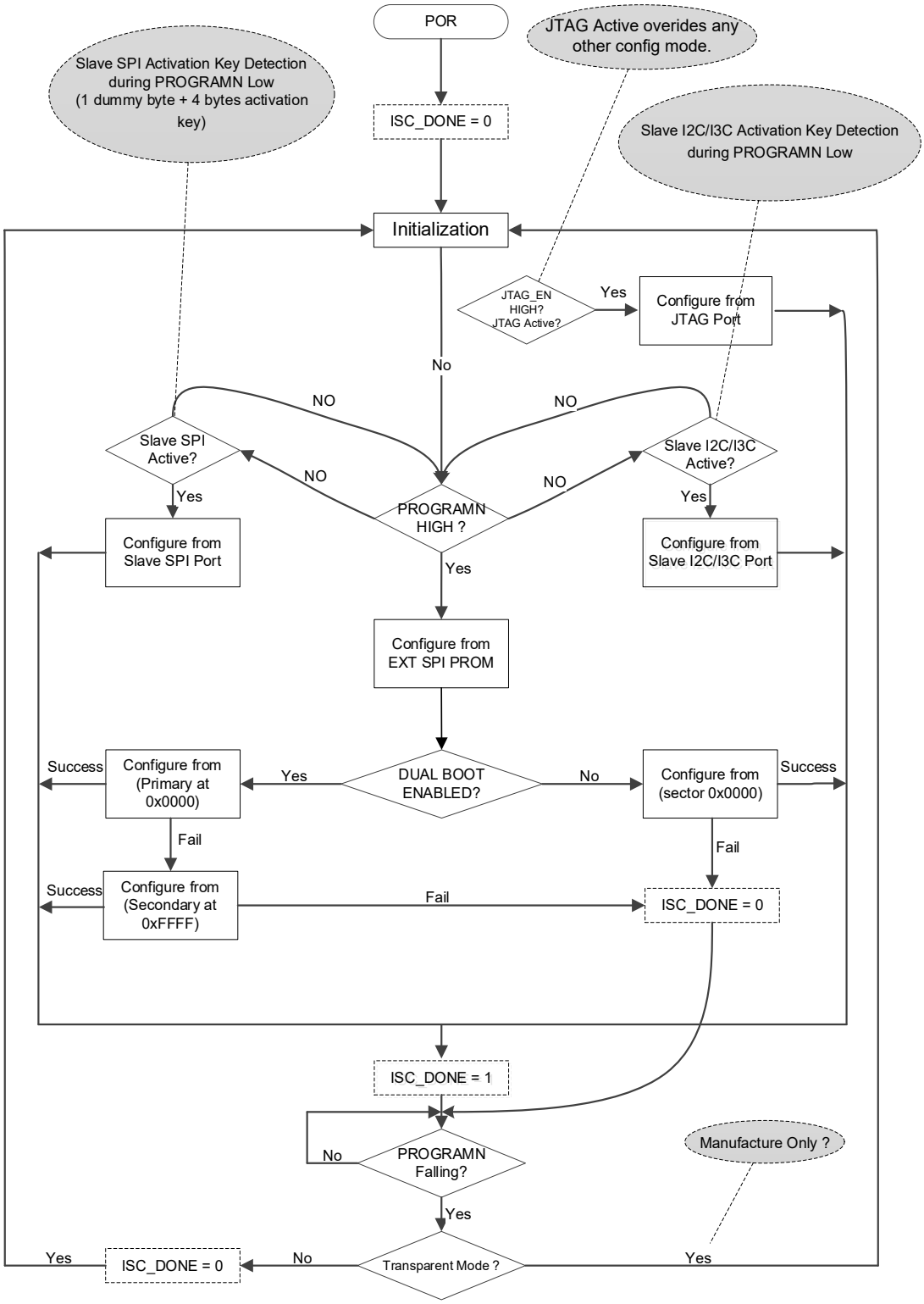


Figure 4.1. Configuration Control Flow

JTAG has higher priority than other modes. If JTAG becomes active by driving JTAG_EN HIGH during any other boot mode, that mode is canceled and JTAG takes over.

4.4. sysCONFIG Pins

The Nexus device provides a set of sysCONFIG I/O pins that you can use to program and configure the FPGA. The sysCONFIG pins are grouped together to create ports (such as JTAG, SSPI, MSPI) that are used to interact with the FPGA for programming, configuration, and access of resources inside the FPGA. The sysCONFIG pins in a configuration port group may be active, and used for programming the FPGA, or they can be reconfigured to act as general-purpose I/O.

Recovering the configuration port pins for use as general-purpose I/O requires you to adhere to the following guidelines:

- You must DISABLE the unused port. You can accomplish this by using the Lattice Radiant Device Constraint Editor under Global tab.
- You must prevent external logic from interfering with device programming.

Table 4.4 lists the shared sysCONFIG pins of the device, and the default state of these pins in user mode. After programming the Nexus device, the default state of the SSPI sysCONFIG pins become general-purpose I/O. This means that you do not have the ability to program the Nexus device using the SSPI or Slave I^2C/I3C when using the default sysCONFIG port settings. To retain the SSPI or Slave I^2C/I3C sysCONFIG pins in user mode, be sure to ENABLE them using the Lattice Radiant Device Constraint editor.

Unless specified otherwise, the sysCONFIG pins are powered by the VCCIO0 and VCCIO1 voltage. It is crucial for this to be taken into consideration when provisioning other logic attached to Bank 0 and Bank 1.

Table 4.4 lists the function of each sysCONFIG pin in detail.

Table 4.4. Default State of the sysCONFIG Pins

sysCONFIG Pins				Pull During Configuration	Configuration Modes			
Name	Type	Hardware Default	Software Default		JTAG	MSPI	SSPI	I^2C/I3C
JTAG_EN	Dedicated	JTAG_EN	JTAG_EN	DOWN	1'b1	1'bx	1'b0	1'b0
PROGRAMN	Shared[5]	PROGRAMN	PROGRAMN[7]	UP	1'b0	1'b1	1'b0	1'b0
INITN	Shared[5]	INITN	INITN[7]	UP	INITN			
DONE	Shared[5]	DONE	DONE[7]	UP	DONE			
MCLK[3]	Shared[6]	MCLK	GPIO	UP/DOWN[4]	—	MCLK	—	—
MCSN[2]	Shared[6]	MCSN	GPIO	UP	—	MCSN	—	—
MOSI/MD0	Shared[6]	MOSI/MDO	GPIO	UP	—	MOSI/D0	—	—
MISO/MD1	Shared[6]	MISO/MD1	GPIO	UP	—	MISO/D1	—	—
MD2	Shared[6]	MD2	GPIO	UP	—	D2	—	—
MD3	Shared[6]	MD3	GPIO	UP	—	D3	—	—
MCSNO/MSDO	Shared[6]	MCSNO/MSDO	GPIO	UP	—	MCSNO/MSDO	—	—
TCK/SCLK	Shared[6]	TCK/SCLK	TCK[7]	UP	TCK	—	SCLK	—
TMS/SCSN[1]	Shared[6]	TMS/SCSN	TMS[7]	UP	TMS	—	SCSN	—
TDI/SI/SD0	Shared[6]	TDI/SI/SD0	TDI[7]	UP	TDI	—	MOSI/D0	—
TDO/SO/SD1	Shared[6]	TDO/SO/SD1	TDO[7]	UP	TDO	—	MISO/D1	—
SD2/SCL	Shared[6]	SD2/SCL	GPIO	UP	—	—	D2	SCL
SD3/SDA	Shared[6]	SD3/SDA	GPIO	UP	—	—	D3	SDA
SD[15:4][8]	Shared	—	GPIO	UP	—	—	—	—

Notes:

1. SCSN should have 4.7 kΩ pull-up resistor on-board for SSPI.
2. MCSN should have 4.7 kΩ pull-up on-board resistor for MSPI.
3. Suggest 1 kΩ pull down resistor on-board for MCLK.
4. Inter weak pull-up or pull down is determined by the CR1 bit 22 (CPOL) setting. UP if CPOL = 1; DOWN if CPOL = 0.
5. Shared between configuration and user GPIO. User selectable through the Lattice Radiant Device Constraint Editor. Achieved through Nonvolatile EFUSE Feature setting.
6. Shared between configuration and user GPIO in User Function Mode. User-selectable through the Lattice Radiant Device Constraint Editor. Achieved through the SRAM fuse setting through bitstream.
7. These pins are set to DISABLE (become GPIO) in the Lattice Radiant Device Constraint Editor as Software Default in the Lattice Radiant software version 3.0 or earlier.

8. SD[15:4] are shared I/O pins reserved as dedicated configuration pins for internal testing only and shall be left unused when the device is in configuration mode. Once the device enters user mode, these pins can be used as general purpose I/O pins.

4.4.1. PROGRAMN

PROGRAMN is an input used to configure the FPGA. The PROGRAMN pin assertion enables you to control the device programming flow, allowing reconfiguration and exit from user mode for the device to re-initiate programming through either JTAG/SSPI/I2C.

The PROGRAMN pin is low level sensitive, and has an internal weak pull-up. When PROGRAMN is asserted low, the FPGA exits user mode and starts a device configuration sequence at the Initialization phase, as described earlier. Holding the PROGRAMN pin low prevents the Nexus device from leaving the initialization phase. The PROGRAMN has a minimum pulse width assertion period ($t_{PROGRAMN}$) for it to be recognized by the FPGA. You can find this minimum time in the sysCONFIG Port Timing Specifications of the CrossLink-NX Family Data Sheet (FPGA-DS-02049), Certus-NX Family Data Sheet (FPGA-DS-02078), CertusPro-NX Family Data Sheet (FPGA-DS-02086), and CrossLink-NX-33 and CrossLinkU-NX Data Sheet (FPGA-DS-02104).

Be aware of the following special cases when the PROGRAMN pin is active:

- If the device is currently being programmed through JTAG, PROGRAMN is ignored until the JTAG mode programming sequence is complete.
- Toggling the PROGRAMN pin during device configuration interrupts the process and restart the configuration cycle.
- PROGRAMN must not make a falling edge transition during the time the FPGA is in the Initialization state to avoid interrupting, restarting, or failing configuration. PROGRAMN must be asserted for a minimum low period of $t_{PROGRAMN}$ for it to be recognized by the FPGA. Failure to meet this requirement can cause the device to become non-operational, requiring power to be cycled.
- For slave port configuration or programming, if the PROGRAMN pin is persisted in user function mode, it must be driven high before the ISC_DISABLE command is issued.

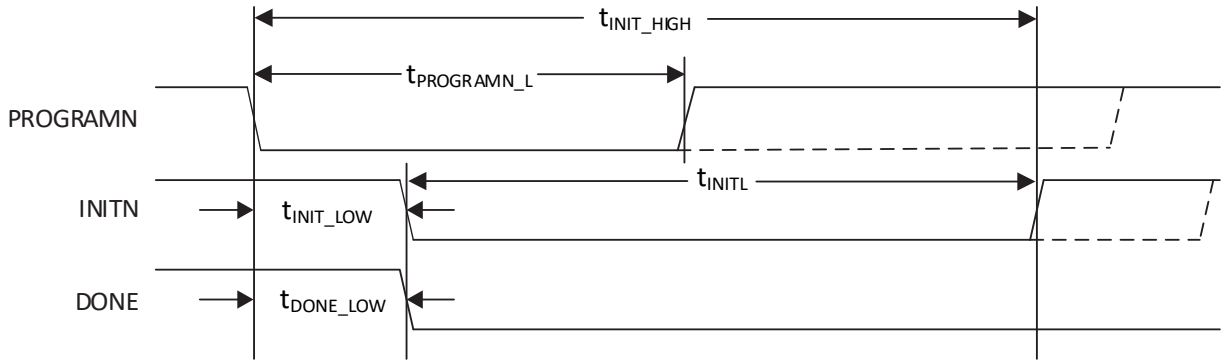


Figure 4.2. Configuration from PROGRAMN Timing

4.4.2. INITN

The INITN pin is a bidirectional open-drain control pin. It has the following functions:

- After power is applied, after a PROGRAMN assertion, or a REFRESH command it goes low to indicate the SRAM configuration memory is being erased. The low time assertion is specified with the t_{INIT_LOW} parameter.
- After the t_{INITL} time period has elapsed the INITN pin is deasserted (that is active high) to indicate the Nexus device is ready for its configuration bits. The Nexus device begins loading configuration data from an external SPI flash.
- INITN can be asserted low by an external agent before the t_{INITL} time period has elapsed to prevent the FPGA from reading configuration bits. This is useful when there are multiple programmable devices chained together. The programmable device with the longest t_{INITL} time can hold all other devices in the chain from starting to get data until it is ready itself.
- The last function provided by INITN is to signal an error during the time configuration data is being read. Once t_{INITL} has elapsed and the INITN pin has gone high, any subsequent INITN assertion signals the Nexus device has detected an error during configuration.

The following conditions cause INITN to become active, indicating the Initialization state is active:

- Power has just been applied.
- PROGRAMN falling edge occurred.
- The IEEE 1532 REFRESH command is sent using a slave configuration port (JTAG or SSPI). If the INITN pin is asserted due to an error condition, the error can be cleared by correcting the configuration bitstream and forcing the FPGA into the initialization state.

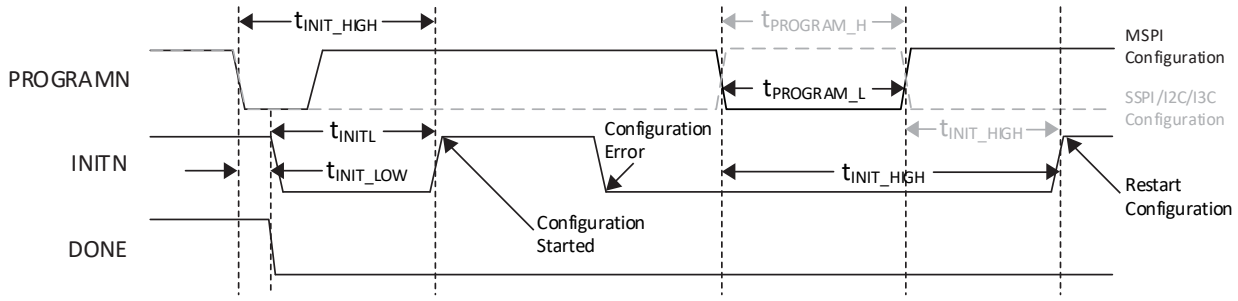


Figure 4.3. Configuration Error Notification

If an error is detected when reading the bitstream, INITN goes low, the internal DONE bit is not set, the DONE pin stays low, and the device does not wake up. The device configuration fails when the following happens:

- The bitstream CRC error is detected.
- The invalid command error is detected.
- A time out error is encountered when loading from the external flash memory. This can occur when the device is in MSPI configuration mode and the SPI flash device is not programmed.
- The program done command is not received when the end of on-chip SRAM configuration or external flash memory is reached.

Perform the following steps after a configuration error to enable the device to restart configuration:

1. After INITN goes low due to a configuration error, toggle the PROGRAMN pin. You can find the minimum pulse width for $t_{PROGRAMN\_H}$ and $t_{PROGRAMN\_L}$ in the sysCONFIG Port Timing Specifications of the CrossLink-NX Family Data Sheet (FPGA-DS-02049), Certus-NX Family Data Sheet (FPGA-DS-02078), CertusPro-NX Family Data Sheet (FPGA-DS-02086), and CrossLink-NX-33 and CrossLinkU-NX Data Sheet (FPGA-DS-02104).
2. Wait for t_{INIT_HIGH} time period to elapse. The INITN pin de-asserts high indicating that the Nexus device is ready to restart configuration.

4.4.3. DONE

The DONE pin is a bi-directional open drain with a weak pull-up that signals the FPGA is in user mode. DONE is first able to indicate entry into user mode only after an internal DONE bit is asserted. The internal DONE bit defines the beginning of the FPGA Wake-Up state. After the internal DONE bit is asserted, the GPIO will wake-up and the external DONE pin goes high after 10 µs. This means that the GPIO pin setting is applied before the DONE pin goes high.

The FPGA can be held from entering User mode indefinitely by having an external agent keep the DONE pin asserted low. A common reason for keeping DONE driven low is to allow multiple FPGAs to be completely configured. As each FPGA reaches the DONE state, it is ready to begin operation. The last FPGA to configure can cause all FPGAs to start in unison.

The DONE pin drives low in tandem with the INITN pin when the FPGA enters Initialization mode. As described earlier, this condition happens when power is applied, PROGRAMN is asserted, or an IEEE 1532 Refresh command is received through an active configuration port.

Sampling the DONE pin is a way for an external device to tell if the FPGA has finished configuration.

4.5. Master SPI sysCONFIG Pins

The following is a list of the dual-purpose Master SPI sysCONFIG pins. If any of these pins are used for configuration and for user I/O, you must adhere to the requirements listed at the start of the Configuration pin sections. These pins are powered by VCCIO0.

4.5.1. MCLK

The MCLK, when active, is a clock used to sequentially load the configuration data for the FPGA. When Master Configuration mode is used, MCLK becomes an output clock with the frequency that you set. The output is used to drive to external memory device. The maximum MCLK frequency and the data setup/hold parameters can be found in the sysCONFIG Port Timing Specifications of CrossLink-NX Family Data Sheet (FPGA-DS-02049), Certus-NX Family Data Sheet (FPGA-DS-02078), CertusPro-NX Family Data Sheet (FPGA-DS-02086), and CrossLink-NX-33 and CrossLinkU-NX Data Sheet (FPGA-DS-02104). MCLK actively drives until all of the configuration data is received. When the Nexus device enters user mode the MCLK output tri-states. The MCLK is reserved for use in MSPI mode, in most post-configuration applications, as the reference clock for performing memory transactions with the external SPI PROM. See Master SPI Modes section for details.

The Nexus device generates MCLK from an internal oscillator. The initial frequency of the MCLK is nominally 3.5 MHz. The MCLK frequency can be altered using the MCCLK_FREQ parameter. You can select the MCCLK_FREQ using the Lattice Radiant Device Constraint Editor. For a complete list of the supported MCLK frequencies, see Table 4.5.

Table 4.5. Nexus MCLK Valid Frequencies

MCLK Frequency (MHz)
3.5
7.0
14.1
28.1
56.2
75
90
112.5
150

At startup, the lowest frequency MCLK is used by the FPGA. During the initial stages of device configuration, the frequency value specified using MCCLK_FREQ contained in the bitstream is loaded into the FPGA. Once the Nexus device accepts the new MCCLK _FREQ value, the MCLK output begins driving the selected frequency. Make certain when selecting the MCCLK_FREQ that the user does not exceed the frequency specification of the configuration memory, or of the PCB.

4.5.1.1. Selecting MCCLK_FREQ

When selecting MCCLK_FREQ especially for 56.2 MHz and above, refer to the following guidelines:

- Review the configuration memory data sheet to ensure MCCLK_FREQ does not exceed the frequency specification of the configuration memory.
- Review the sysCONFIG Port Timing specifications in CrossLink-NX Family Data Sheet (FPGA-DS-02049), Certus-NX Family Data Sheet (FPGA-DS-02078), CertusPro-NX Family Data Sheet (FPGA-DS-02086), and CrossLink-NX-33 and CrossLinkU-NX Data Sheet (FPGA-DS-02104).
- Perform static timing analysis to evaluate and ensure the setup and hold timing slack meets the requirements stipulated in the respective Nexus device data sheet and to determine the MCCLK_FREQ. To evaluate the setup and hold timing slack in your system, refer to the following:
  - The data setup timing slack must be equal to or larger than the minimum data setup time t_{DSU}.
    - RX Edge = 0: Default setting for sampling clock edge for serial data input.
      $½ \times t_{MCLK} – (t_{BT\_MCLK} + t_{CLQV} + t_{BT\_DATA}) \geq t_{SU\_MSI}$
    - RX Edge = 1: Delays the sampling clock edge for half of the MCLK cycle to increase the setup time slack. This enables MSPI configuration to operate at a higher MCLK frequency compared to when RX Edge = 0.
      $t_{MCLK} – (t_{BT\_MCLK} + t_{CLQV} + t_{BT\_DATA}) \geq t_{SU\_MSI}$
  - The hold timing slack must be equal to or larger than the minimum data hold time t_{DH}.
  - $t_{BT\_MCLK} + t_{CLQX} + t_{BT\_DATA} \geq t_{HD\_MSI}$
  - In the mathematical expressions, the symbols represent the following:
    - t_{MCLK}: FPGA MCLK output clock period
    - t_{BT_MCLK}: Board trace propagation delay for MCLK
    - t_{CLQV}: Configuration memory clock low to output valid
    - t_{CLQX}: Configuration memory output hold time
    - t_{BT_DATA}: Board trace propagation delay for serial data input
    - t_{SU_MSI}: Minimum FPGA MSI to MCLK setup time
    - t_{HD_MSI}: Minimum FPGA MSI to MCLK hold time
    - f_{MCLK}: MCLK output frequency (= $1/t_{MCLK}$)

Note: To enable MSPI configuration to operate at a higher MCLK frequency (75 MHz or above), use a configuration memory with a smaller clock to output valid specification.

The following is an example of evaluating the setup timing slack and determining the MCCLK_FREQ setting:

- System specifications:
  - $t_{BT\_MCLK} = 0.172$ ns
  - $t_{CLQV} = 7$ ns
  - $t_{CLQX} = 1$ ns
  - $t_{BT\_DATA} = 0.227$ ns
  - $t_{SU\_MSI} = 3$ ns
  - $t_{HD\_MSI} = 0.5$ ns
- For RX EDGE = 0:
  $½ \times t_{MCLK} – (t_{BT\_MCLK} + t_{CLQV} + t_{BT\_DATA}) \geq t_{SU\_MSI}$
  $t_{MCLK} \geq 2 \times (t_{SU\_MSI} + (t_{BT\_MCLK} + t_{CLQV} + t_{BT\_DATA})$
  $t_{MCLK} \geq 2 \times (3 + 0.172 + 7 + 0.227)$
  $t_{MCLK} \geq 20.798$ ns
  $f_{MCLK} \leq 48.08$ MHz
- For RX EDGE = 1:
  $t_{MCLK} – (t_{BT\_MCLK} + t_{CLQV} + t_{BT\_DATA}) \geq t_{SU\_MSI}$
  $t_{MCLK} \geq (t_{SU\_MSI} + (t_{BT\_MCLK} + t_{CLQV} + t_{BT\_DATA}))$
  $t_{MCLK} \geq (3 + 0.172 + 7 + 0.227)$
  $t_{MCLK} \geq 10.399$ ns
  $f_{MCLK} \leq 96.72$ MHz
- Timing analysis shows that you can select MCCLK_FREQ = 28.1 MHz or below if RX Edge = 0; you can select MCCLK_FREQ = 90 MHz or below if RX EDGE = 1.

4.5.1.2. Setting RX Edge in Volatile SRAM Memory

If the FPGA fails to configure from configuration memory, you might need to change the RX Edge setting based on the desired MCCLK_FREQ setting. The following guides are for testing and ensuring that MSPI configuration can succeed with RX Edge set to 1 in volatile SRAM memory before committing the setting permanently to non-volatile configuration memory.

Setting RX Edge to 1 in Control Register 1

1. Power on the board and FPGA.
2. In the Lattice Radiant Programmer software, select **Run** > **Scan Device**.
3. Once device scanning is done, select **Edit** > **Device Properties**.
4. Select the following options under **Device Operation** in the Device Properties window:
   - Target Memory: Static Random Access Memory (SRAM)
   - Port Interface: JTAG (or any of the available interfaces such as Slave SPI, I2C, or I3C Bridge)
   - Access Mode: Direct Programming
   - Operation: Program Control Register1

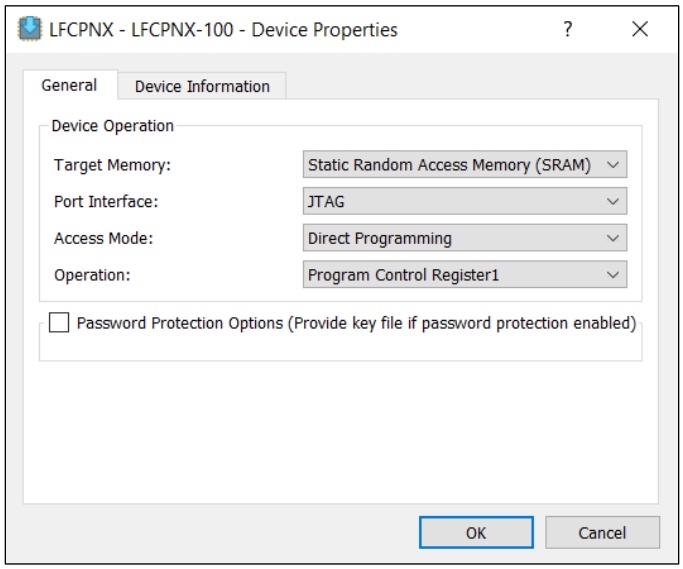


Figure 4.4. Device Properties Window (Program Control Register 1)

5. Click **OK**, then select **Run** > **Program Device**. The Control Register 1 window appears.

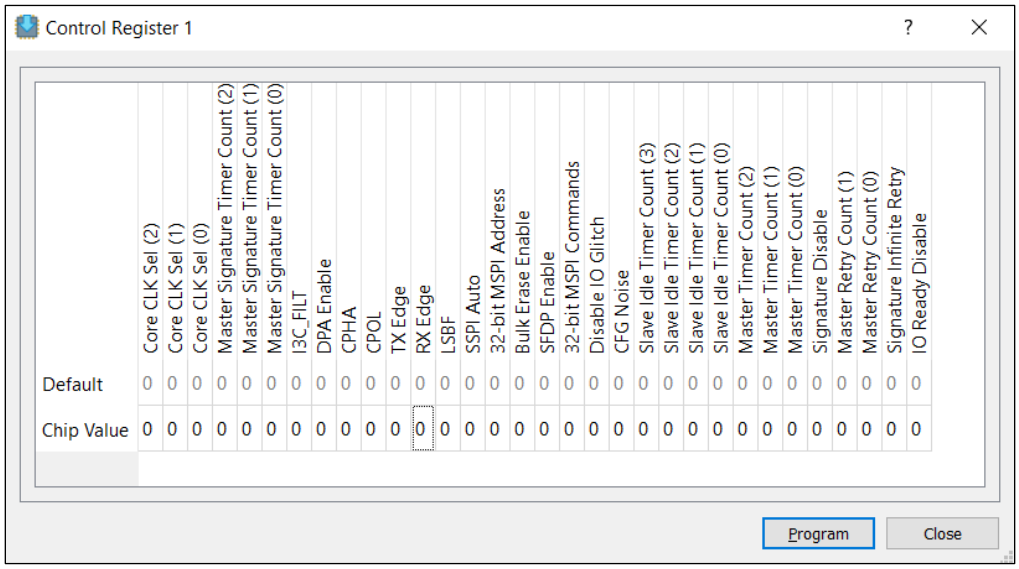


Figure 4.5. Control Register 1 Window (Setting RX Edge)

6. Edit the RX Edge bit from 0 to 1, then click **Program**. The Confirm Overwrite Register dialog box appears.

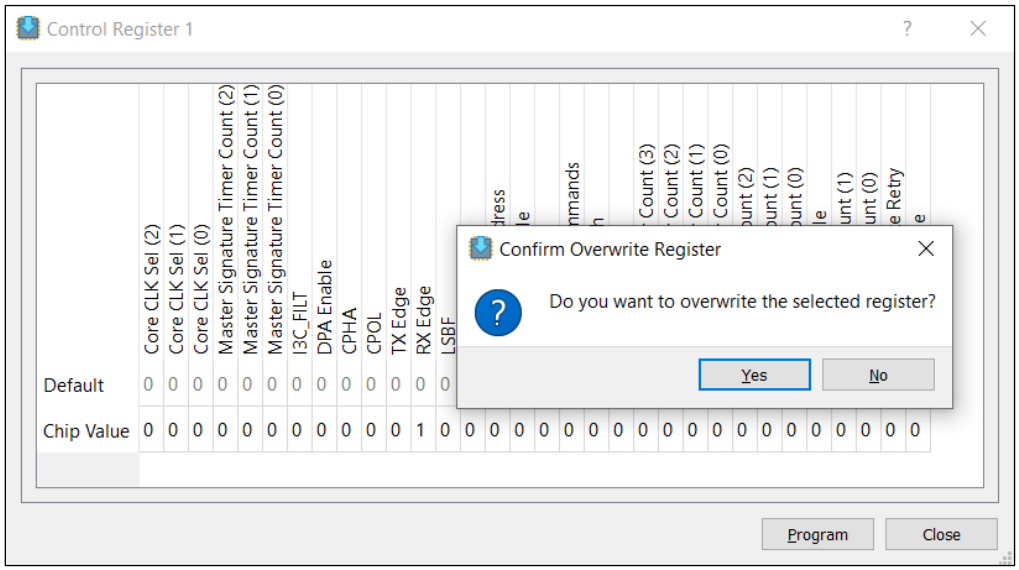


Figure 4.6. Confirm Overwrite Register Dialog Box (Setting RX Edge)

7. Click **Yes** to overwrite the RX Edge bit to 1 in Control Register 1.
8. Once programming is done, select **Edit** > **Device Properties**.
9. Change the Operation option under **Device Operation** in the Device Properties window to *Display Control Register1*.

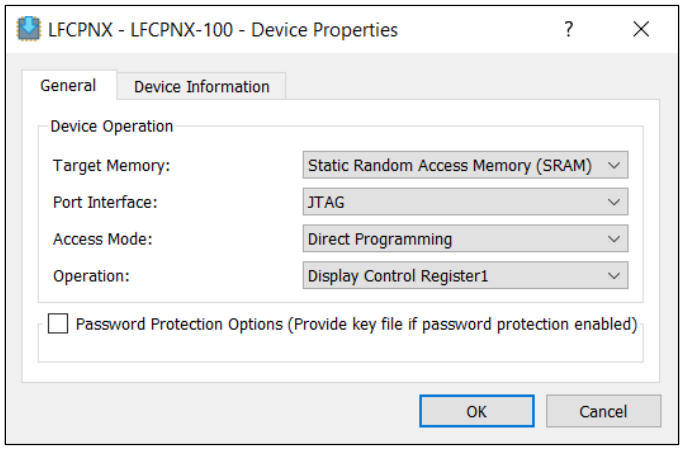


Figure 4.7. Device Properties Window (Display Control Register 1)

10. Click **OK,** then select **Run** > **Program Device**. The Control Register 1 window appears. Confirm that RX Edge is set to 1.

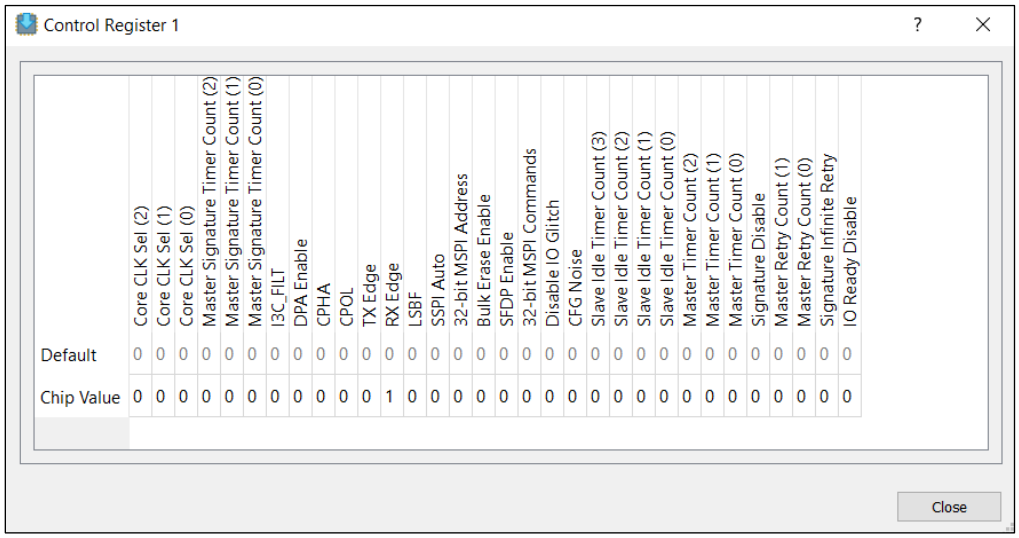


Figure 4.8. Control Register 1 Window (Confirming RX Edge)

Setting No CDM to 1 in Control Register 0

Setting No CDM to 1 in Control Register 0 is essential to retain the Control Register 1 settings in SRAM during reconfiguration.

1. In the Lattice Radiant Programmer software, select **Edit** > **Device Properties**.
2. Select the following options under **Device Operation** in the Device Properties window:
   - Target Memory: Static Random Access Memory (SRAM)
   - Port Interface: JTAG (or any of the available interfaces such as Slave SPI, I2C, or I3C Bridge)
   - Access Mode: Direct Programming
   - Operation: Program Control Register0

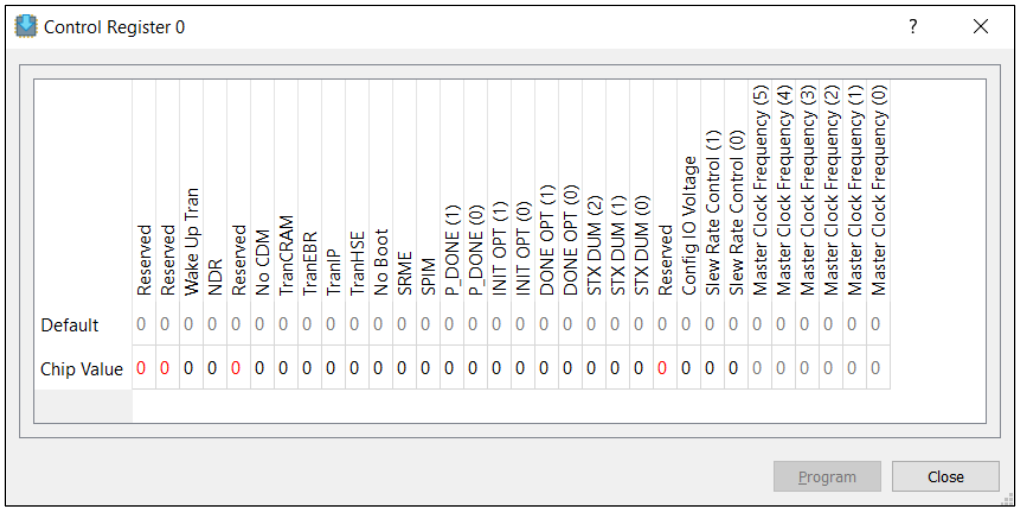


Figure 4.9. Control Register 0 Window (Setting No CDM)

3. Edit the No CDM bit from 0 to 1, then click **Program**. The Confirm Overwrite Register dialog box appears.

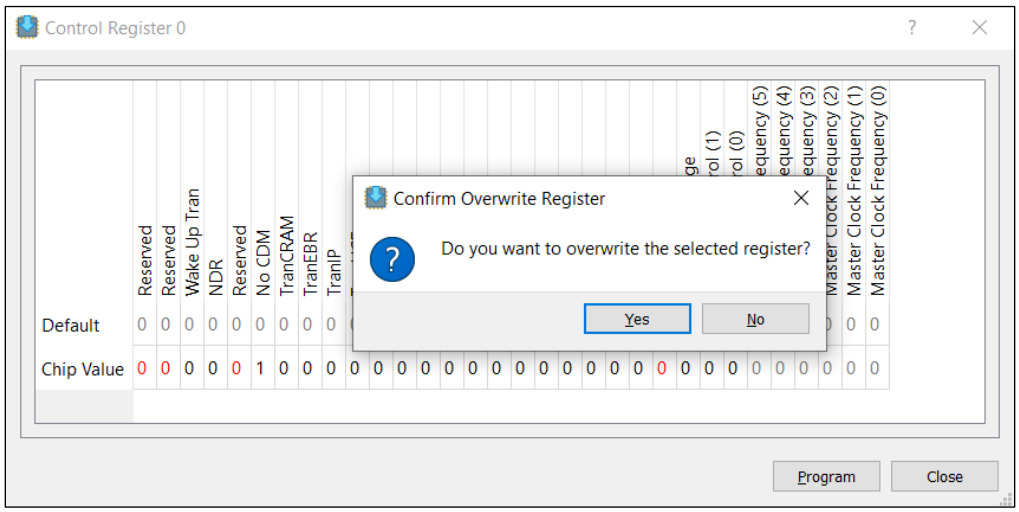


Figure 4.10. Confirm Overwrite Register (Setting No CDM)

4. Click **Yes** to overwrite the No CDM bit to 1 in Control Register 0.
5. Once programming is done, select **Edit** > **Device Properties**.
6. Change the Operation option under **Device Operation** in the Device Properties window to *Display Control Register0*.
7. Click **OK,** then select **Run** > **Program Device**. The Control Register 0 window appears. Confirm that No CDM is set to 1.

Toggling MSPI Configuration

1. Toggle the PROGRAMN pin to restart MSPI configuration. If the timing analysis indicates that RX Edge needs to be set to 1 to operate at high MCLK frequency, MSPI configuration should succeed.
2. Once MSPI configuration is successful, you may proceed to set RX Edge to 1 in non-volatile configuration memory.

4.5.1.3. Setting RX Edge in Non-Volatile Configuration Memory

To permanently set RX Edge, refer to Appendix C for guides on modifying the Nexus Feature Row using the Feature Row Editor.

Note: Feature row bits are one time programmable (OTP). These bits can only be modified once.

4.5.2. MCSN

MCSN – In MSPI mode, the MCSN becomes an active low Chip Select output that drives the SPI Serial Flash chip select. If SPI memory needs to be accessed using the SPI port while the part is in user mode (the DONE pin is high) then the MASTER_SPI_PORT= ENABLE, must be used to preserve this pin as MCSN. When the Nexus device is not in MSPI mode, the MCSN is a general purpose I/O with a weak pulldown. Adding a 4.7 kΩ to 10 kΩ pull-up resistor to MCSN pin on the Nexus device is recommended. MCSN must ramp in tandem with the SPI PROM VCC input. It remains a general purpose I/O when the FPGA enters user mode.

4.5.3. MOSI/MD0

MOSI/MD0 – In Master SPI configuration mode, the MOSI pin is the serial data output for SPI command and data. It becomes D0 of the data bus in Dual or Quad mode.

4.5.4. MISO/MD1

MISO/MD1 – In Master SPI configuration mode, the MISO pin is the serial data input. It becomes D1 of the data bus in Dual or Quad mode.

4.5.5. MD2

MD2 – In Master SPI configuration mode, this bit becomes the D2 of the data bus in Quad mode.

4.5.6. MD3

MD3 – In Master SPI configuration mode, this bit becomes the D3 of the data bus in Quad mode.

4.5.7. MCSNO/MSDO

This is an output pin for configuration daisy chain support, which has the following purposes:

MCSNO – For configuration daisy chaining implemented with the Flow-through attribute, this attribute allows the MCSNO pin to be driven when the done bit is set and configuration of the first device is complete. The MCSNO of the first device drives the CSN of the second part.

MSDO – MSDO pin is used in the Bypass mode. Since the Bypass mode is not supported for Nexus devices, the related function could be ignored.

4.6. Slave sysCONFIG Pins

4.6.1. TCK/SCLK

This pin is used by both the JTAG and Slave SPI configuration interface.

TCK – If the JTAG port is enabled, this pin serves as the clock pin for the JTAG interface. The test clock pin (TCK) provides the clock used to time the other JTAG port pins. Data is shifted into the instruction or data registers on the rising edge of TCK and shifted out on the falling edge of TCK. The TAP is a static design permitting TCK to be stopped in either the high or low state. The maximum input frequency for TCK is specified in the DC and Switching Characteristics section of the CrossLink-NX Family Data Sheet (FPGA-DS-02049), Certus-NX Family Data Sheet (FPGA-DS-02078), CertusPro-NX Family Data Sheet (FPGA-DS-02086), and CrossLink-NX-33 and CrossLinkU-NX Data Sheet (FPGA-DS-02104). The TCK pin does not have a pull-up. An external pull-down resistor of 2.2 kΩ is recommended to avoid inadvertently clocking the TAP controller as power is applied to the Nexus device.

SCLK – In slave SPI mode, this pin is the clock input for the slave SPI configuration interface. The maximum CCLK frequency and the data setup/hold parameters can be found in the sysCONFIG Port Timing Specifications of the CrossLink-NX Family Data Sheet (FPGA-DS-02049), Certus-NX Family Data Sheet (FPGA-DS-02078), CertusPro-NX Family Data Sheet (FPGA-DS-02086), and CrossLink-NX-33 and CrossLinkU-NX Data Sheet (FPGA-DS-02104).

4.6.2. TMS/SCSN

This pin is used by both the JTAG and Slave SPI configuration interface.

TMS – If the JTAG port is enabled, this pin serves as the Test Mode Select pin for the JTAG interface. The Test Mode Select (TMS) pin is an input pin that controls the progression through the 1149.1 compliant state machine states. The TMS pin is sampled on the rising edge of TCK. The JTAG state machine remains in or transitions to a new TAP state depending on the current state of the TAP, and the present state of the TMS input. An internal pull-up resistor is present on TMS per the JTAG specification.

SCSN – In slave SPI mode, this pin is the active low chip select input for the slave SPI configuration interface. A 4.7 kΩ external pull-up resistor is recommended.

4.6.3. TDI/SI/SD0

This pin is used by both the JTAG and Slave SPI configuration interface.

TDI – If the JTAG port is enabled, this pin serves as the Test Data Input (TDI) pin, which is used to shift in serial test instructions and data. This pin should be wired to TDI of the JTAG connector, or to TDO of an upstream device in a JTAG chain. An internal pull-up resistor on the TDI pin is provided.

SI/SD0 – In Slave SPI configuration mode, the SI pin is the serial data input for SPI command and data. It becomes D0 of the data bus in Dual or Quad mode.

4.6.4. TDO/SO/SD1

This pin is used by both the JTAG and Slave SPI configuration interface.

TDO – If the JTAG port is enabled, this pin serves as the Test Data Output (TDO) pin, which is used to shift out serial test instructions and data. When TDO is not being driven by the internal circuitry, the pin is in a high impedance state. The only time TDO is not in a high impedance state is when the JTAG state machine is in the Shift IR or Shift DR state. This pin should be wired to TDO of the JTAG connector, or to TDI of a downstream device in a JTAG chain. An internal pull-up resistor on the TDO pin is provided.

SO/SD1 – In Slave SPI configuration mode, the SO pin is the serial data output for SPI data. It becomes D1 of the data bus in Dual or Quad mode.

4.6.5. SD2/SCL

This pin is used by both the Slave SPI and Slave I^{2}C/I3C configuration interface.

SD2 – In Slave SPI configuration mode, this bit becomes the D2 of the data bus in Quad mode.

SCL – In Slave I^{2}C/I3C configuration mode, this is the serial clock line of the I^{2}C or I3C bus.

4.6.6. SD3/SDA

This pin is used by both the Slave SPI and Slave I^{2}C/I3C configuration interface.

SD3 – In Slave SPI configuration mode, this bit becomes the D3 of the data bus in Quad mode.

SDA – In Slave I^{2}C/I3C configuration mode, this is the serial data line of the I^{2}C or I3C bus.

4.7. PERSISTENT

The internal PERSISTENT control bits are used to determine whether the dual-purpose Master and Slave sysCONFIG pins remain as sysCONFIG pins during normal operation. For example, to support transparent programming or configuration. The Nexus device has several PERSISTENT physical SRAM cells that determine the existence of the Master SPI port, Slave SPI port or I^{2}C/I3C port after entering user mode. These settings can be set in the Lattice Radiant Device Constraint Editor under Global tab as shown in Table 4.6.

Table 4.6. sysCONFIG Pins Global Preferences[1]

Port Setting	Value	Pins Affected	Details
SLAVE_SPI_PORT	SERIAL	MCLK, MCSN, MOSI/MD0, MISO/MD1	If enabled, persisted for configuration purpose.
	DUAL	MCLK, MCSN, MOSI/MD0, MISO/MD1	
	QUAD	MCLK, MCSN, MOSI/MD0, MISO/MD1, MD2, MD3	
MASTER_SPI_PORT	SERIAL	SCLK, SCSN, SI/SD0, SO/SD1	If enabled, persisted for configuration purpose.
	DUAL	SCLK, SCSN, SI/SD0, SO/SD1	
	QUAD	SCLK, SCSN, SI/SD0, SO/SD1, SD2, SD3	
SLAVE_I2C_PORT/ SLAVE_I3C_PORT	ENABLE	SCL, SDA	If enabled, persisted for configuration purpose.

Note:

1. JTAG Persist follows the JTAG enable pin.

5. Master Port Configuration Process and Flow

Prior to becoming operational, the FPGA goes through a sequence of states, including initialization, configuration, and wake-up.

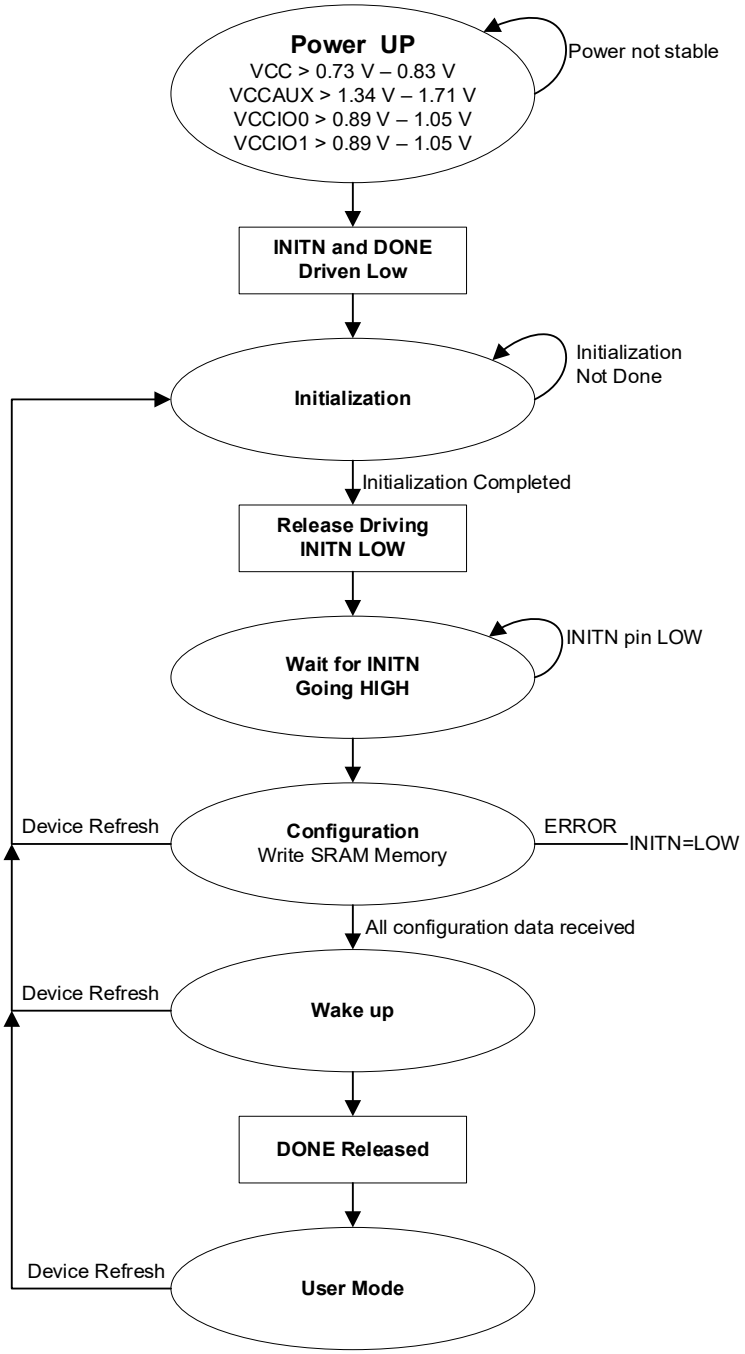


Figure 5.1. Master Port Configuration Flow

The Nexus device sysCONFIG ports provide industry standard communication protocols for programming and configuring the FPGA. Each of the protocols shown in Table 4.2 provides a way to access the configuration SRAM of the Nexus device. The Configuration Ports Arbitration section provides information about the availability of each sysCONFIG port.

5.1. Power-Up Sequence

For the Nexus device to operate, power must be applied to the device. During a short period of time, as the voltages applied to the system rise, the FPGA stays in an indeterminate state.

As power continues to ramp, a Power-On-Reset (POR) circuit inside the FPGA becomes active. During the power-up sequence phase, once the POR circuit is active, the POR circuit makes sure that the external I/O pins are in a high-impedance state. It also monitors the VCC, VCCAUX, VCCIO0, and VCCIO1 input rails. The POR circuit waits for the following conditions:

- VCC > 0.73 V – 0.83 V
- VCCAUX > 1.34 V – 1.71 V
- VCCIO0 > 0.89 V – 1.05 V
- VCCIO1 > 0.89 V – 1.05 V

When these conditions are met, the POR circuit releases an internal reset strobe, allowing the device to begin its initialization process. The Nexus device asserts INITN active low, and drives DONE low. When INITN and DONE are asserted low, the device moves to the initialization state, as shown in Figure 5.1.

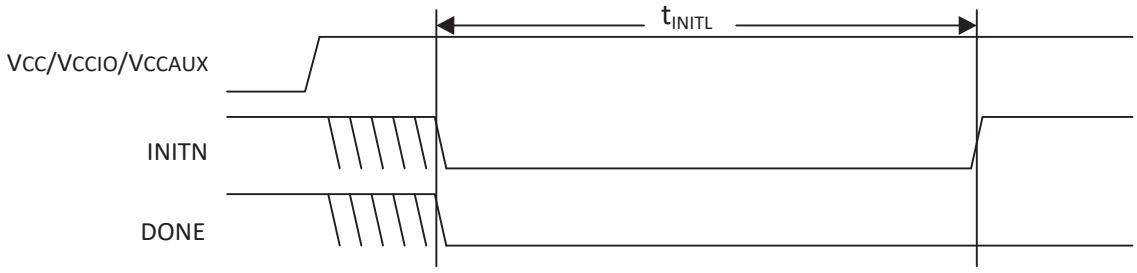


Figure 5.2. Configuration from Power-On-Reset Timing

5.2. Initialization

The Nexus device enters the memory initialization phase immediately after the Power-On-Reset circuit drives the INITN and DONE status pins low. The purpose of the initialization state is to clear all the SRAM memory inside the FPGA.

The FPGA remains in the initialization state until all the following conditions are met:

- The t_{INITL} time period has elapsed.
- The PROGRAMN pin is deasserted.
- The INITN pin is no longer asserted low by an external master.

The dedicated INITN pin provides two functions during the initialization phase. The first is to indicate the FPGA is currently clearing its configuration SRAM. The second is to act as an input preventing the transition from the initialization state to the configuration state.

During the t_{INITL} time period the FPGA is clearing the configuration SRAM. When the Nexus device is part of a chain of devices each device has different t_{INITL} initialization time. The FPGA with the slowest t_{INITL} parameter can prevent other devices in the chain from starting to configure. Premature release of the INITN in a multi-device chain may cause configuration of one or more chained devices to fail to configure intermittently.

The active-low, open-drain initialization signal INITN must be pulled high by an external resistor when initialization is complete. To synchronize the configuration of multiple FPGAs, one or more INITN pins should be wire-ANDed. If one or more FPGAs or an external device holds INITN low, the FPGA remains in the initialization state.

This initialization phase starts after the power-up sequence. In the power-up sequence phase, the GPIO of the device defaults to tri-stated outputs with active weak input pull-downs. After configuration, all GPIO included in the user design wake up in the user-defined condition. GPIO not defined in the user design remain output tri-stated and the input with a weak pull-down enabled. This default avoids inadvertent effects of the inputs rising while powering up. In some cases, this can cause a problem if other connected devices on the board reset or trigger from an active high signal.

Before/during configuration, the dedicated JTAG_EN pin has pull-down, and dual-purpose sysCONFIG I/O with pull-up, which are excluded from the GPIO default setting.

5.3. Configuration

The releasing of HIGH on the INITN pin causes the FPGA to enter the configuration state. The FPGA can accept the configuration bitstream created by the Lattice Radiant software.

Following power-up, the Nexus device begins the external SPI flash boot process with the Signature Verification phase. The Nexus device attempts a signature read-back in a finite loop until the expected result is received or the loop count is exceeded. A correct signature read allows the Nexus device to proceed to the Preamble Verification phase immediately. Opposed to waiting for a fixed power-ramp timer to expire, this process allows for the fastest possible boot times.

The Signature Verification process verifies either the Lattice Specified LSCC signature or JEDEC Standard SFDP, depending on the value of Control Register 1, Bit 15 (CR1 [15]). If CR1 [15] is *0* (default), the Nexus device reads the boot bitstream image from the base boot address and check for the LSCC signature (0x4C534343) using SPI flash command code 8'H03. If CR1 [15] is *1*, the Nexus device performs a SFDP read (SPI flash command code 8'H5A) and checking for SFDP code (0x50444653) contained in SFPD compliant SPI flash devices. The Nexus device retries the SFDP/LSCC signature read until three consecutive matches are found. When successful, the Nexus device sets the internal Signature Successful Flag and proceeds to the Preamble Verification step. If the loop timer expires (600,000 SPI clock cycles, or about 150 ms), the device proceeds to the Preamble Verification step without setting the internal Signature Successful Flag and sets the BSE Timeout Flag of the device status register to 1. When the internal Signature Successful Flag is set, the Signature Verification phase is bypassed for subsequent warm-boot events (for example, the PROGRAMN pin toggle or REFRESH command).

For proper bitstream data alignment, the bitstream Preamble must be detected once. If the preamble does not come before the preamble timer (counting for 600,000 SPI clock cycles) expires, a boot failure is declared, and the boot process aborts.

Once the preamble is detected, the Nexus device continues fetching data from non-volatile memory to configure the FPGA SRAM memory. The Nexus device does not leave the Configuration state if there is no valid configuration data.

INITN is used to indicate an error exists in the configuration data. When INITN is high, configuration is proceeding without issue. If INITN is asserted low, an error has occurred and the FPGA does not operate.

5.4. Wake-Up

Wake-up is the transition from configuration mode to user mode. The Nexus device's fixed four-phase wake-up sequence starts when the device has correctly received all of its configuration data. User can arrange the order of these four phases to meet specific implementation requirements. When all configuration data is received, the FPGA asserts an internal DONE status bit. The assertion of the internal DONE causes a Wake-Up state machine to run those four control sequences. The four control strobes are:

- Global Set/Reset (GSRN)
- Global Write Enable (GWE)
- Global Output Enable (GOE)
- External DONE

One phase of the Wake-Up process is for the FPGA to release the Global Output Enable. When it is asserted, permits the FPGA I/O to exit a high-impedance state and take on the programmed output function. The FPGA inputs are always active. The input signals are prevented from performing any action on the FPGA flip-flops by the assertion of the Global Set/Reset (GSRN).

Other phases of the Wake-Up process release the Global Set/Reset and the Global Write Enable controls.

The Global Set/Reset is an internal strobe that, when asserted, causes all I/O flip-flops, Look Up Table (LUT) flip-flops, distributed RAM output flip-flops, and Embedded Block RAM output flip-flops that have the *GSR enabled* attribute to be set/cleared per the hardware description language definition.

The Global Write Enable is a control that overrides the write enable strobe for all RAM logic inside the FPGA. As mentioned previously, the inputs on the FPGA are always active. Keeping GWE deasserted prevents accidental corruption of the instantiated RAM resources inside the FPGA.

Another phase of the Wake-Up process asserts the external DONE pin. The external DONE is a bi-directional, open-drain I/O only when it is enabled. An external agent that holds the external DONE pin low prevents the wake-up process of the FPGA from proceeding. Only after the external DONE, if enabled, is active high does the final wake-up phase complete. Wake-Up completes uninterrupted when the external DONE pin is not enabled.

Once the final wake-up phase is complete, the FPGA enters user mode. This process is detailed later in the document.

5.5. Early I/O Release (EIO)

The Nexus device supports an Early I/O Release feature, which allows the I/O that reside in the I/O Banks on the left and right of the device (I/O Bank 1, I/O Bank 2, I/O Bank 6, and I/O Bank 7), to assume user-defined drive states at the beginning of bitstream processing. The Early I/O Release feature releases the I/O after processing the I/O configuration for the left and right banks, which is located near the head of the bitstream data. Once data is programmed in the left/right Memory Interface Block (MIB) the I/O is released to a predefined state. This feature is enabled by setting the EARLY_IO_RELEASE preferences to ON in the Lattice Radiant Device Constraint Editor.

In addition, Early I/O Release requires you to instantiate an output buffer register with an asynchronous set or reset function, to indicate the desired drive *1* or drive *0* behavior, respectively, during the Early Release period. Unregistered outputs in Early-Release banks drive High-Z until full device configuration is complete. Be aware that some of the I/O in Bank 1, including the dual-purpose sysCONFIG I/O, cannot be utilized as Early Released I/O. Also, if the ECDSA bitstream authentication is enable for the Nexus device, the Early I/O Release feature is not supported.

5.6. User Mode

The Nexus device enters user mode immediately following the Wake-Up sequence has completed. User Mode is the point in time when the Nexus device begins performing the logic operations you designed. The device remains in this state until one of three events occurs:

- The PROGRAMN input pin is asserted
- A REFRESH command is received through one of the configuration ports
- Power is cycled or power supply levels drop below the specified trigger levels

5.7. Clearing the Configuration Memory and Re-initialization

The current user mode configuration of the Nexus device remains in operation until they are actively cleared, or power is lost. Several methods are available to clear the internal configuration memory of the Nexus device. The first is to remove power and reapply power. Another method is to toggle the PROGRAMN pin. Lastly, you can reinitialize the memory through a Refresh command. Any active configuration port can be used to send a Refresh command.

Invoking one of these methods causes the Nexus device to drive INITN and DONE low. The Nexus device enters the initialization state as described earlier.

5.8. Reconfiguration Priority

There are many sources that can initiate a reconfiguration while a configuration is already in process. When initiating a reconfiguration, the sources are prioritized depending on which of them initiated the original configuration. Note that if an interruption occurs, the reconfiguration occurs without informing the original configuration source that the configuration did not complete. JTAG always has the highest priority and any JTAG initiated configuration event causes a reconfiguration to occur. Toggling the PROGRAMN pin has the next highest priority. It interrupts any current configuration other than a JTAG configuration.

6. Configuration Modes

The Nexus device provides multiple options for loading the configuration SRAM from a non-volatile memory. The previous section describes the physical interface necessary to interact with the configuration logic of the Nexus device. This section focuses on describing the functionality of each of the different configuration modes. Descriptions of important settings required in the Lattice Radiant Device Constraint Editor View are also discussed.

6.1. Master SPI Modes

The Master SPI port is considered an intelligent port and it is the default mode before any other slave configuration port is activated. It can perform read and write actions based on the command shifted into the device. All commands are built inside the bitstreams. In Master SPI mode (MSPI), the Nexus device begins retrieving configuration data from the SPI flash when power is applied, a REFRESH command is received, or the PROGRAMN pin is asserted and released. The MCLK I/O takes on the Master Clock (MCLK) function, and begins driving a nominal 3.5 MHz clock to the SPI flash's SCLK input. MCSN is asserted low, commands are transmitted to the PROM over the MOSI/MD0 output, and data is read from the PROM on the MISO/MD1 input pin. When all the configuration data is retrieved from the PROM, the MCSN pin is deasserted, and the MSPI output pins are tri-stated.

Table 6.1. Master SPI Configuration Port Pins

Pin Name	Function	Direction	Description
MCLK	MCLK	Output with weak pull-up	Master clock used to time data transmission/reception from the Nexus device Configuration Logic to a slave SPI PROM.
MCSN[1]	MCSN	Output	Chip select used to enable an external SPI PROM containing configuration data.
MOSI/MD0	MOSI	Output	Carries output data from the Nexus device Configuration Logic to the slave SPI PROM.
	D0	Input	Input pin for bitstream data as DUAL and QUAD read mode
MISO/MD1	MISO	Input	Carries input data from the slave SPI PROM to the Nexus device Configuration Logic.
	D1	Input	Input pin for bitstream data as DUAL and QUAD read mode.
MD[3:2]	D[3:2]	Input	This is the input pins for bitstream data from SPI flash, used only on QUAD read mode.

Note:

1. Use 4.7 kΩ pull-up resistor.

The MCLK frequency always starts downloading the configuration data at the nominal 3.5 MHz frequency. The MCCLK_FREQ parameter, accessed using Device Constraint Editor, can be used to increase the configuration frequency. The configuration data in the PROM has some padding bits, and then the data altering the MCLK base frequency is read. The Nexus device reads the remaining configuration data bytes using the new MCLK frequency. For higher MCLK frequency with fast slew rate, a 33 Ω damping resistor is recommended.

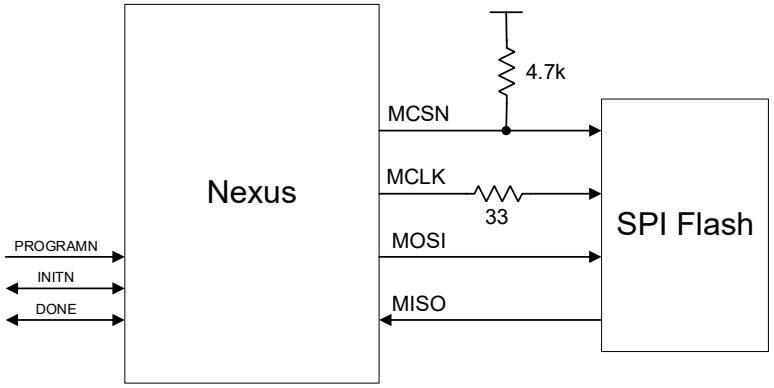


Figure 6.1. Nexus Master SPI Port with SPI Flash

Once the SPI flash contains the configuration data, you can test the configuration. Assert the PROGRAMN, transmit a REFRESH command, or cycle power to the board, and the Nexus device configures from the external SPI flash.

The following table shows the command codes for reading the bitstream from the configuration memory when the MSPI controller is operating in 24-bit and 32-bit addressing modes.

Table 6.2. Bitstream Read Command Codes with MSPI Controller in 24-bit and 32-bit Addressing Modes

Booting Command	24-bit Command Code	32-bit Command Code
SFDP Read	8'h5A	8'h5A
Slow Read	8'h03	8'h13
Fast Read	8'h0B	8'h0C
Dual Read	8'hBB	8'hBC
Quad Read	8'hEB	8'hEC

If you are using an SPI flash with a density of 256 Mbit or higher, you can use 32-bit addressing and commands so that the configuration engine can access data beyond the 128 Mbit boundary. Refer to Appendix F for details on setting the 32-bit MSPI address and 32-bit MSPI commands option bits in CR1 through the bitstream or in non-volatile configuration memory.

6.1.1. Method to Enable the Master SPI Port

The Master SPI port is enabled by default if there is no slave configuration port enabled.

- When the device is powered up, or when the PROGRAMN pin is toggled, or when the REFRESH command is executed, the Master SPI port is selected as the configuration port by default. A port is said to be a configuration port when it can execute both bitstream write and read commands. This is the only method in which the you can perform DUAL read and QUAD read from SPI flash.
- Enabling Master SPI port persistence.
  The MSPI port could be persisted in user mode by setting the desired Value (SERIAL, DUAL or QUAD) for MASTER_SPI_PORT in the Lattice Radiant Device Constraint Editor. Once set, the configuration bitstream contains optional Master SPI persistence bits. When the device completes configuration and wakes up, it checks the persistent bits to determine if the Master SPI port is to remain operational once in user mode. This selection is independent of the CFG port arbitration during configuration phase. A port enabled by persistence is a Transparent Mode port. It reserves those pins from being occupied by user logic.
  Note that both the DONE pin and the INITN pin must be high. If not, then the device is not in user mode. The persistent bits have no affect when the device is not in user mode.

6.1.2. Dual-Boot and Multi-Boot Configuration Modes

Both the primary and the golden (fail-safe) configuration data is stored in external SPI memory. The fail-safe pattern is available in case the primary pattern would fail to load. The primary image can fail in one of the following reasons:

- A time-out error is encountered while waiting for PREAMBLE code
- Device ID checking failed at the beginning of the bitstream
- An illegal command is encountered
- A bitstream CRC error is detected
- A time-out error is encountered when loading from the primary pattern stored in the external memory

A CRC error is caused by incorrect data being written into the SRAM configuration memory. Data is read from the external flash memory. As data enters the Configuration Engine the data is checked for CRC consistency. Before the data enters the Configuration SRAM the CRC must be correct. Any incorrect CRC causes the device to erase the Configuration SRAM and retrieve configuration data from the external golden pattern.

It is possible for the data to be correct from a CRC calculation perspective, but not be functionally correct. In this instance the internal DONE bit never becomes active. The Nexus device counts the number of master clock pulses it provided after the Power-On-Reset signal is released. When the count expires without DONE becoming active, the FPGA attempts to get its configuration data from the external golden pattern.

Dual boot configuration mode typically requires two configuration data files. One of the two configuration data files is a fail-safe image that is rarely, if ever, updated. The second configuration data file is a working image that is routinely updated. Both the working (primary) image and the fail-safe image are stored in the external SPI memory. One Lattice Radiant project can be used to create both the working and the fail-safe configuration data files. Configure the Lattice Radiant project with an implementation named *working*, and an implementation named *failsafe*. Read the Lattice Radiant Online Help for more information about using Lattice Radiant implementations.

The Nexus device supports dual-boot with the MSPI mode. If the primary pattern fails to load correctly, the Nexus device starts loading data from the golden sector in the SPI flash device. A blank external flash device causes a dual-boot event failure indicated by INITN going low. This is due to the absence of a primary or golden boot image.

The dual boot feature allows you to split a SPI flash device into two sections, the first containing a sacred *golden boot* file, and a second updatable *primary boot* file, which can be erased and reprogrammed. By default, the FPGA loads the primary boot file in block 0 (0x000000). If the FPGA fails in configuration, it automatically loads the jump table (containing addresses for instructions to load the golden boot file) stored in the last 256 bytes of the SPI memory device. This allows the system to boot to a known operable state, so that it continues to operate if for some reason (such as a power failure) the SPI flash fails to program correctly.

You can dynamically switch between up to five different design revisions stored in external flash. Multi-boot, or the ability to dynamically reconfigure from multiple design bitstreams, is similar to the dual-boot where there is one primary (working) bitstream and up to four alternate bitstreams.

For multi-boot operation, the next target address is set in memory that was loaded during the current configuration memory load. Initiating reprogramming by toggling the PROGRAMN pin or issuing a REFRESH through any sysCONFIG port causes the device to load from the defined SPI Flash address. Dual-boot can also be deployed with multi-boot, allowing a golden (fail-safe) design (or sixth design) to be available in the external Flash. For detail information regarding multi-boot operation, refer to Multi-Boot Usage Guide for Nexus Platform (FPGA-TN-02145).

The Lattice Radiant Deployment Tool allows you to assemble SPI flash images formatted to correctly match the hardware sector mapping.

Note: When using the dual-boot or multi-boot configuration mode, if the configuration memory is 256 Mb or larger, you must set both the 32-bit MSPI address and 32-bit MSPI commands option bits to 1. Otherwise, the fail-safe mechanism will not function properly. Refer to Appendix F for more information.

6.1.3. Ping-Pong Boot

The Ping-Pong boot mode utilize the JUMP table to select an image for booting without changing location of the image in flash. This is done with a jump table starting at address 0x000000 in flash containing two instructions LSC_PROG_SEC_BOOT and JUMP. For backup in case, the jump table were to become corrupted by a power failure. A backup JUMP instruction should be programmed at offset 0xFFFF00 pointing to the golden boot image. Refer to Figure 6.2 for the two images stored in flash Bitstream 0 and Bitstream 1. The secondary bitstream offset is programmed both with the command LSC_PROG_SEC_BOOT located in the jump table starting at offset 0x000000 and the JUMP instruction located in the backup jump table at offset 0xFFFF00, and the primary bitstream is selected with the JUMP command in the jump table. To swap the order of booting, these three instructions are all that needs to be re-programmed.

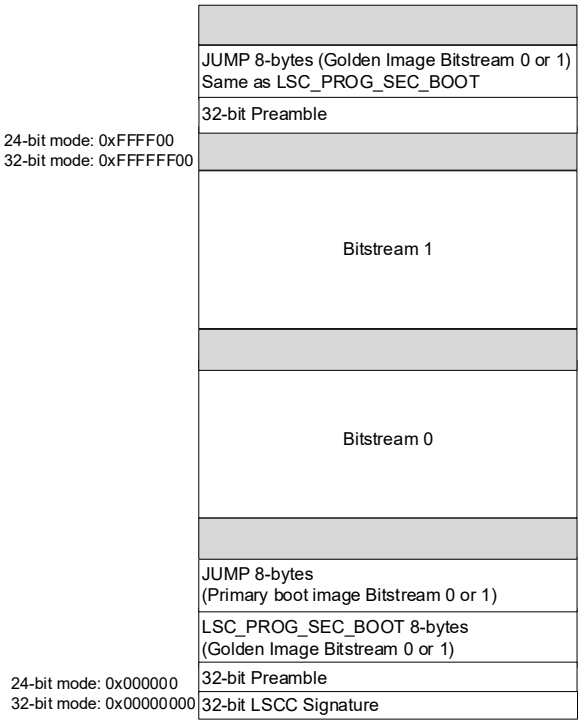


Figure 6.2. Jump Table

Note: When using the ping-pong boot configuration mode, if the configuration memory is 256 Mb or larger, you must set both the 32-bit MSPI address and 32-bit MSPI commands option bits to 1. Otherwise, the fail-safe mechanism will not function properly. Refer to Appendix F for more information.

6.1.4. Dual and Quad Master SPI Read Mode

The master SPI configuration mode in the Nexus device is expanded to support new industry standard Quad I/O SPI Flash memory. The support of (Serial Multi I/O) Flash memory enables fast parallel read.

A typical SPI flash interface uses either 4 or 6 interface signals to the FPGA. The Standard SPI flash uses CLK, CS, SI, and SO while Quad SPI flash uses CLK, CS, I/O0, I/O1, I/O2 and I/O3, maintaining function and pin-out compatibility with the standard SPI flash devices, while adding Dual-I/O and Quad-I/O SPI capability. All SPI modes are submodes of MASTER SPI, therefore there is no longer a separate CFGMDN pin decode for each SPI mode.

In Dual mode, the Fast-Read Dual I/O (BBh) instruction is issued and is similar to the standard Fast Read (0Bh) instruction except that data is output on two pins, SO and SIO0, instead of just SO. This allows data to be transferred from the dual output at twice the rate of standard SPI devices. In QUAD mode, the Fast-Read Quad I/O (EBh) instruction is issued and is similar to the standard Fast Read (0Bh) instruction except that data is output on four data pins, instead of just SO. This allows data to be transferred from the quad output at four times the rate of standard SPI devices.

To change the SPI read mode to fast read, dual read or quad read, the deployment tool must be used to generate the hex file used for programming the SPI flash device. The Lattice Radiant software flow only generates bitstreams with default SPI read mode, which is slow serial read (03h) mode. The sysCONFIG option in the Lattice Radiant Device Constraint Editor is for the software to preserve appropriated pins to help customer design flow. It does not serve the purpose of enabling SPI port nor setting appropriate bits in the PROM file.

Table 6.3. Dual/Quad SPI Port Names

Nexus Devices Pin Name	Flash Device Pin	MSPI Function DUAL(D), QUAD(Q)
MCLK	SCLK	Clock output from the Nexus Device Configuration Logic and Master SPI controller. Connect MCLK to the SCLK input of the Slave SPI device. (D)(Q)
MD [3:2]	SIO[3:2]	Data I/O between the Nexus device and SPI device. (Q)
MD[1:0]	SIO[1:0]	Data I/O between the Nexus device and SPI device. (D)(Q)
MCSN	CSN	Chip select output from the Nexus device configuration logic to the slave SPI flash holding configuration data for the Nexus device. (D)(Q)

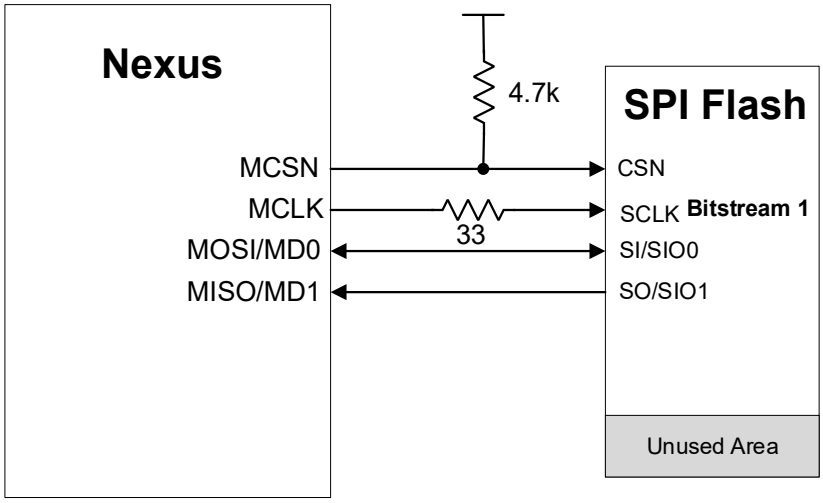


Figure 6.3. One Dual SPI Flash Interface (Dual)

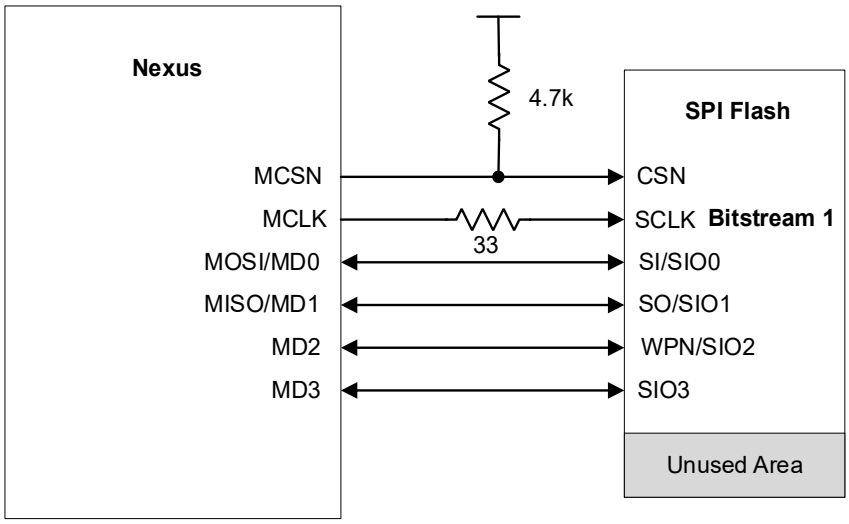


Figure 6.4. One Quad SPI Flash Interface (Quad)

6.1.4.1. Enabling Dual and Quad Master SPI Read Mode

By default, the Lattice Radiant software generates the bitstream in .bit file format that does not enable the Dual or Quad Master SPI read mode. To generate the bitstream that enables the Dual or Quad Master SPI read mode, you must generate the programming file using the Deployment Tool.

Follow these steps to generate the programming file that sets the Dual or Quad Master SPI read mode:

3. Launch the integrated Radiant Programmer in the Radiant software by clicking **Tools > Programmer** or click on the Programmer icon from the toolbar, as shown in Figure 6.5. Alternatively, you can run the stand-alone Radiant Programmer.

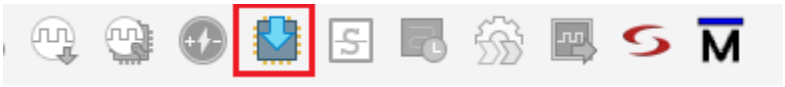


Figure 6.5. Programmer Icon from the Toolbar

4. Fill in the necessary information to create a new project or open an existing project. See Figure 6.6.

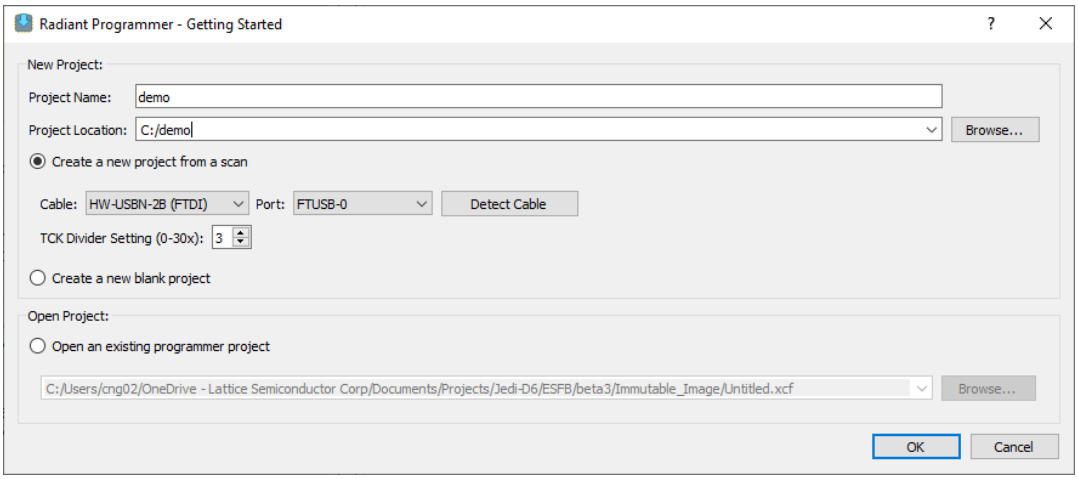


Figure 6.6. Create New or Open Existing Project Window

5. Launch the Deployment Tool by clicking **Tools > Deployment Tool**, as shown in Figure 6.7.

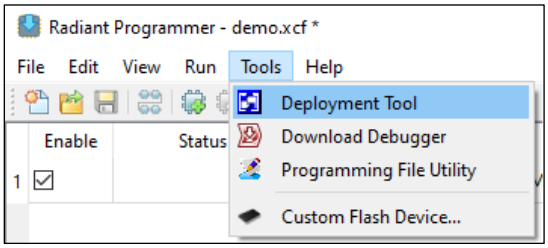


Figure 6.7. Deployment Tool

6. For Function Type, select **External Memory** and for Output File Type, select **Hex Conversion.** Click **OK.** See Figure 6.8. The following lists the available options and their use cases:
   - Hex Conversion: for single image conversion.
   - Dual Boot: for Dual Boot use case where the Primary image is stored at the flash offset 0x0.
   - Ping-Pong Boot: for Ping-Pong Boot use case where the Jump Table is generated and stored at the flash offset 0x0.
   - Advanced SPI Flash: for Multi-Boot use case. You may store up to six images in the flash.

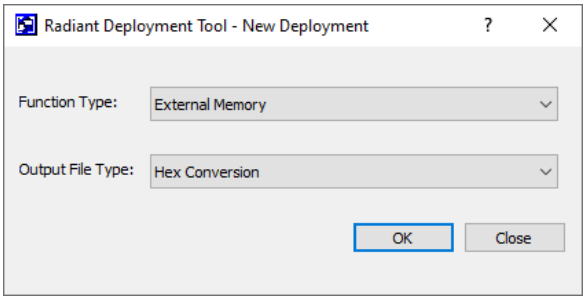


Figure 6.8. Function Type and Output File Selections

7. Click the **…** button to select the .bit file. Click **Next** to proceed. See Figure 6.9.

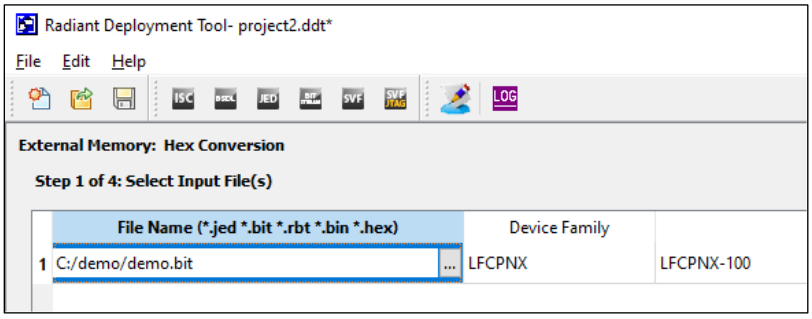


Figure 6.9. Input File Selection

8. Set the SPI Flash Size (Mb) to match with the QSPI flash on the board. Set the SPI Flash Read Mode to the intended mode, for example **Quad I/O SPI Flash Read**. Leave other settings as default or set them according to your use case. Click **Next** to proceed. See Figure 6.10.

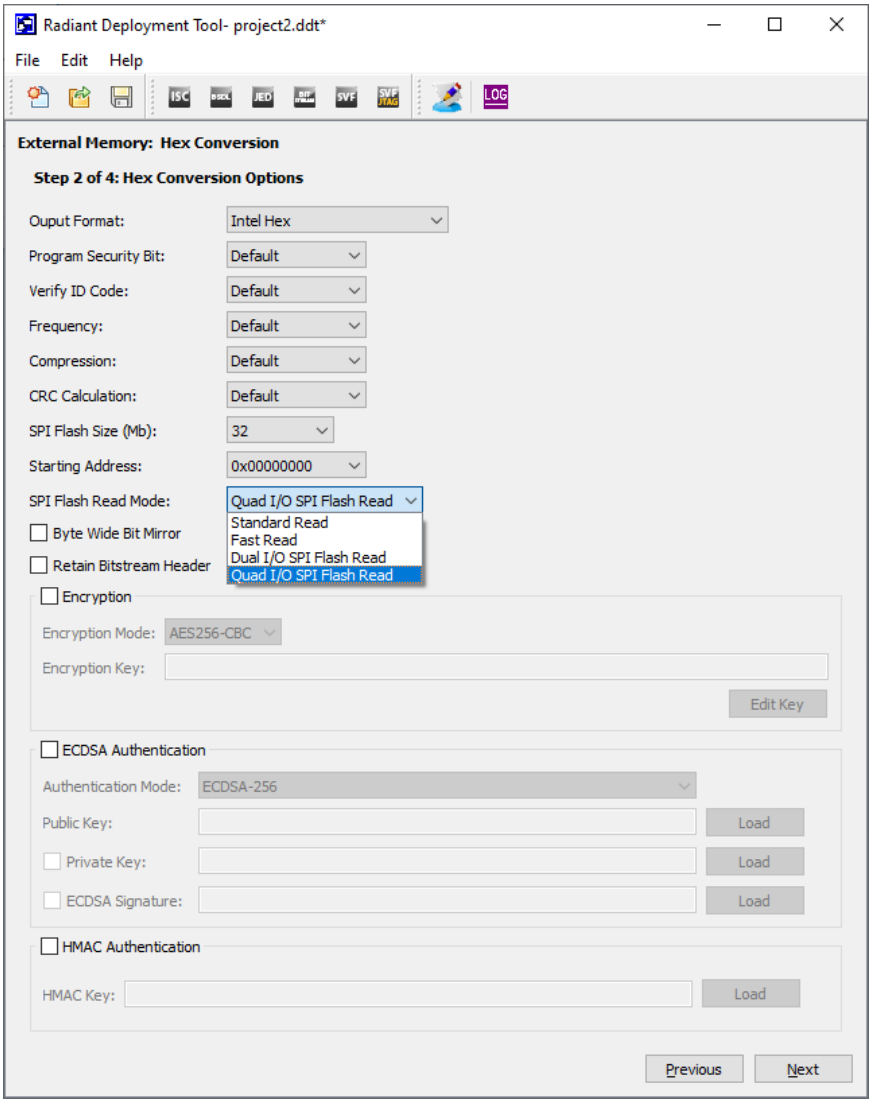


Figure 6.10. SPI Flash Settings

9. Specify the Output File name and click **Next** to proceed. See Figure 6.11.

Figure 6.11. Output File Selection

10. Verify the summary. To generate the programming file, click **Generate**. See Figure 6.12.

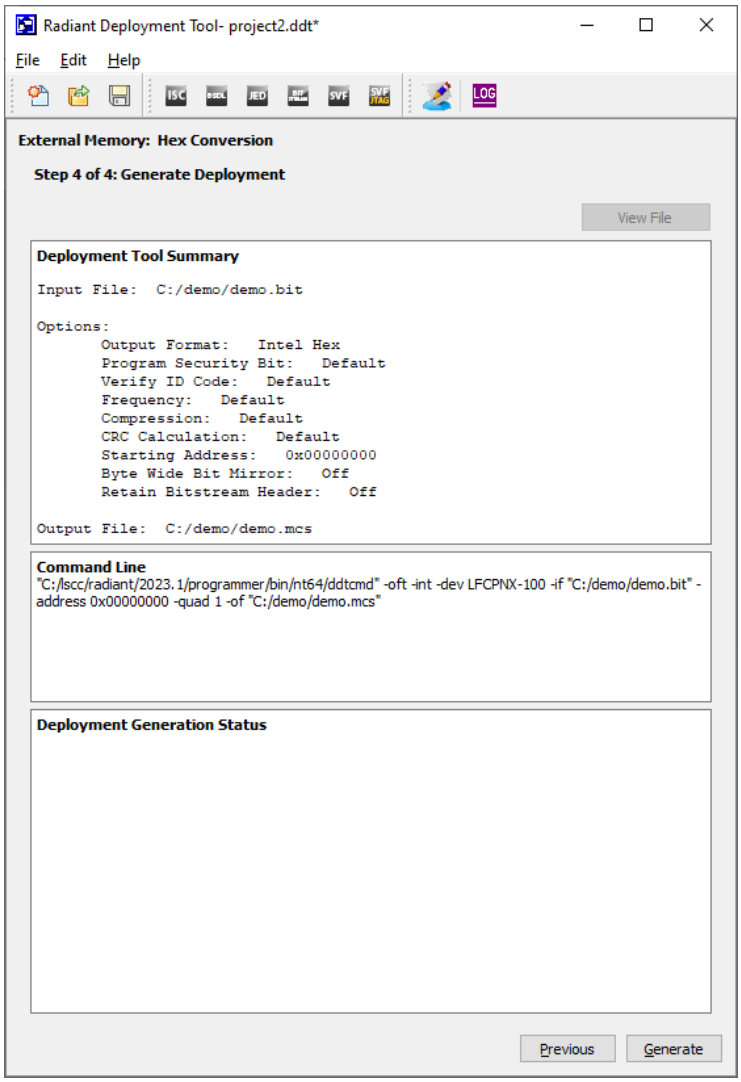


Figure 6.12. Deployment Tool Summary and Programming File Generation

6.1.4.2. Using the Radiant Programmer to Program the External SPI Flash via the JTAG Port

After you generate the programming file using the Deployment Tool, follow these steps to program the external SPI flash via the JTAG port:

1. Launch the integrated Radiant Programmer in the Radiant software by clicking **Tools > Programmer** or click on the Programmer icon from the toolbar, as shown in Figure 6.13Figure 6.5. Alternatively, you can run the stand-alone Radiant Programmer.

Figure 6.13. Programmer Icon from the Toolbar

2. Fill in the necessary information to create a new project or open an existing project. See Figure 6.14.

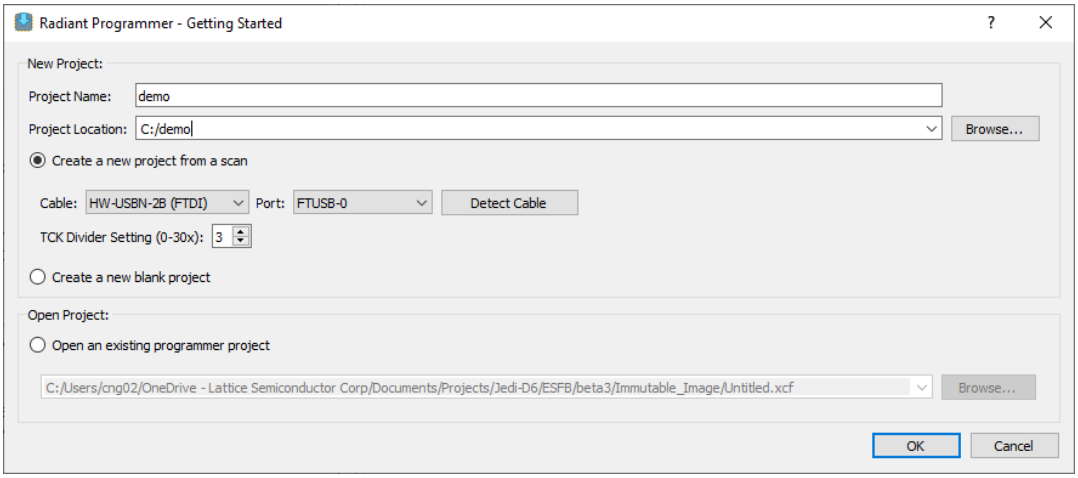


Figure 6.14. Create New or Open Existing Project Window

3. From the Run menu, select **Scan Device**. See Figure 6.15.

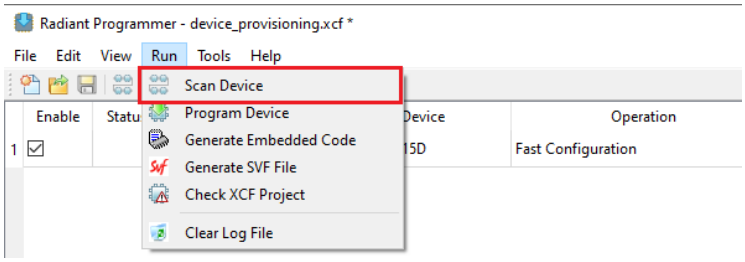


Figure 6.15. Scan Device

4. Make sure the Nexus device is detected in the Programmer, as shown in Figure 6.16.

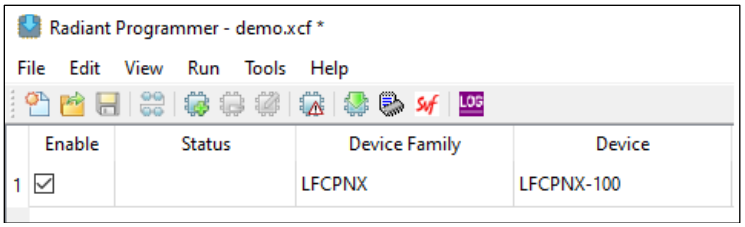


Figure 6.16. Device Status

5. Launch the Device Properties window by clicking **Edit > Device Properties...**
   Alternatively, you can double click on row 1 at the Operation column to launch the Device Properties window.
6. In the General tab of the Device Properties window, under Device Operation, select the following options:
   - Target Memory: External SPI Flash Memory (SPI flash)
   - Port Interface: JTAG2SPI
   - Access Mode: Direct Programming
   - Operation: Erase, Program, Verify
7. Under Programming Options, select the programming file to program the SPI flash.
8. Under SPI Flash Options, select the SPI flash that is used on the board, and click **OK**.

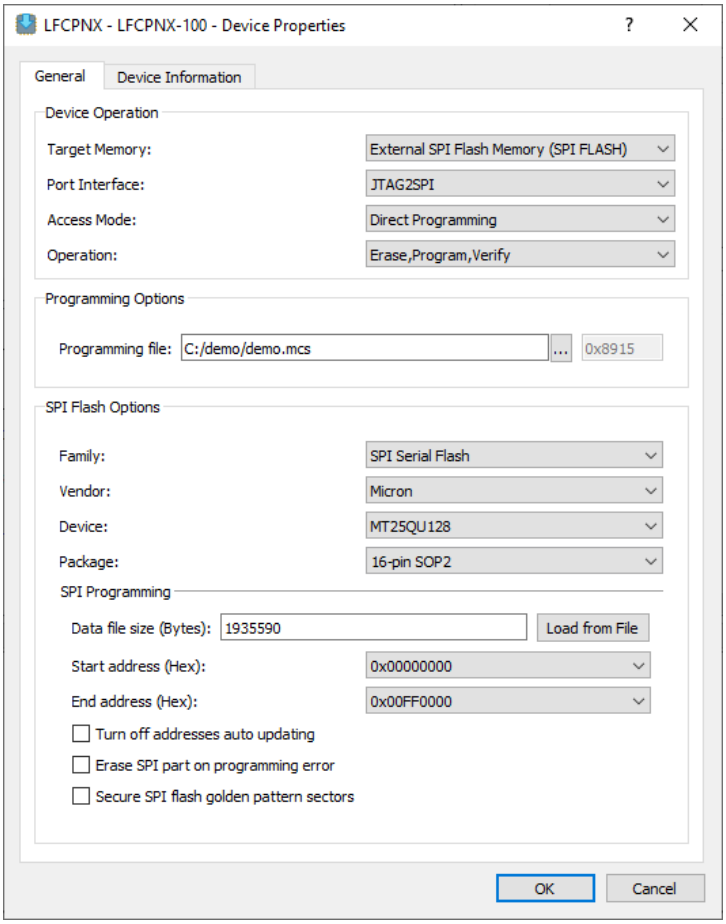


Figure 6.17. Device Properties Window

9. From the Run menu, select **Program Device** to program the external SPI flash. Alternatively, you can click on the Program Device icon from the toolbar, as shown in Figure 6.18.

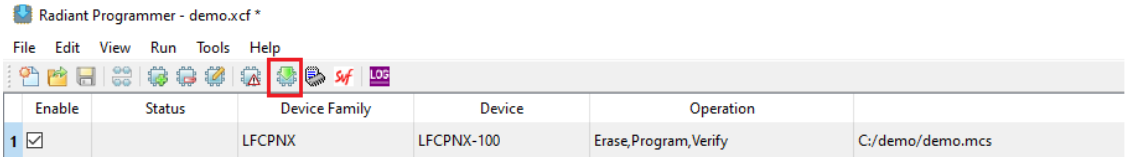


Figure 6.18. Program Device Icon on the Toolbar

6.1.4.3. Using the Radiant Programmer to Program Flash Quad Enable Bit via the JTAG Port

Certain SPI flash in the market requires you to program the quad enable (QE) bit for quad access, both read and write access. If your flash requires the QE bit to be set, select the **Enable Quad Mode** operation in the Device Properties window, as shown in Figure 6.19.

Note: Enabling the QE bit does not enable the Quad MSPI read mode, you must generate the programming file that enables the Quad MSPI read mode, as described in section 6.1.4.1 Enabling Dual and Quad Master SPI Read Mode.

You can also select the **Erase, Program, Verify Quad 1** operation mode in the Device Properties window to program the programming file and the QE bit at the same time.

Note: Not all SPI flash requires you to set the QE bit. To confirm the required settings, refer to your SPI flash data sheet.

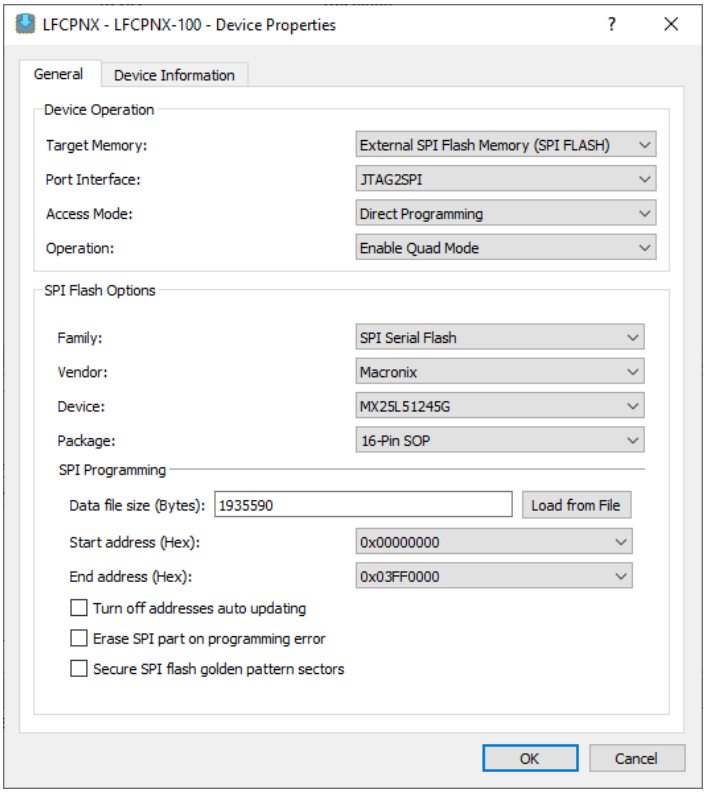


Figure 6.19. Enable Quad Mode Operation

6.2. Slave SPI Mode

The Nexus device provides a Slave SPI (SSPI) configuration port that allows you to access features provided by the Configuration Logic. You can reprogram the Configuration SRAM and access status/control registers within the Configuration Logic block. The external Flash memory updates are done using either offline or transparent operations. It is necessary to send a REFRESH command to load a new external Flash image into the configuration SRAM. When the Nexus device is in Transparent Mode, only read type commands for the Configuration SRAM are supported, allowing for device verification or debugging. The command set consists of 8-bit opcodes. Some of the commands have a 24-bit operand following the 8-bit opcode. The Slave SPI port is a byte bounded port; all input and output data must be byte bounded.

In the Slave SPI mode, the TCK/SCLK pin becomes SCLK (such as Slave SPI Clock). Input data is read into the Nexus device on the MOSI pin at the rising edge of SCLK. Output data is valid on the MISO pin at the falling edge of SCLK. The SCSN acts as the chip select signal. When SCSN is high, the SSPI interface is deselected and the MISO pin is tri-stated. Commands can be written into and data read from the Nexus device when SCSN is asserted.

The SSPI port can be activated through the CFG port arbitration procedure before the device enters user mode. The SSPI port can be persisted in user mode by setting the desired Value (SERIAL, DUAL or QUAD) for SLAVE_SPI_PORT in the Lattice Radiant Device Constraint Editor.

Using the SSPI port simplifies the Nexus device configuration process. Lattice provides *C* source code called sspiembedded to insulate you from the complexity of programming the Nexus device. Refer to the Lattice Radiant online help about sspiembedded.

Table 6.4. Slave SPI Configuration Port Pins

Pin Name	Function	Direction	Description
TCK/SCLK	SCLK	Input with weak pull-up	Clock used to time data transmission/reception from an external SPI master device to the Nexus device Configuration Logic.
SCSN[1]	CSN	Input with weak pull-up	Nexus device Configuration Logic slave SPI chip select input. SN is an active low input. High to Low transition: reset the device, prepare it to receive commands. Low to High transition: Completes or terminates the current command.
TDI/SI/SD0	MOSI	Input	Carries output data from the external SPI master to the Nexus device Configuration Logic.
	D0	Inout	D0 of the data bus for DUAL and QUAD mode
TDO/SO/SD1	MISO	Output	Carries output data from the Nexus Device Configuration Logic to the external SPI master. It is normally tri-stated with an internal pull-up. It becomes active only when the command is a read type command.
	D1	Inout	D1 of the data bus for DUAL and QUAD mode
SD2/SCL	D2	Inout	D2 of the data bus for QUAD mode
SD3/SDA	D3	Inout	D3 of the data bus for QUAD mode

Note:

1. Use external 4.7 kΩ pull-up resistor.

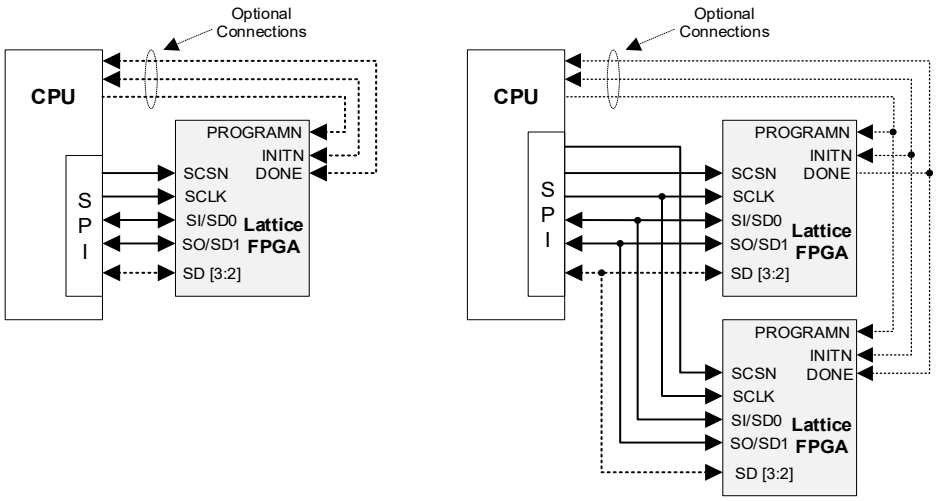


Figure 6.20. Nexus Slave SPI Port with CPU and Single or Multiple Devices

Notes:

- The dotted lines indicate optional connections.
- The wake-up time of the device does vary with the bitstream size and the speed of the SPI port. Lattice recommends connecting the DONE pin to the CPU to monitor when the configuration is complete.

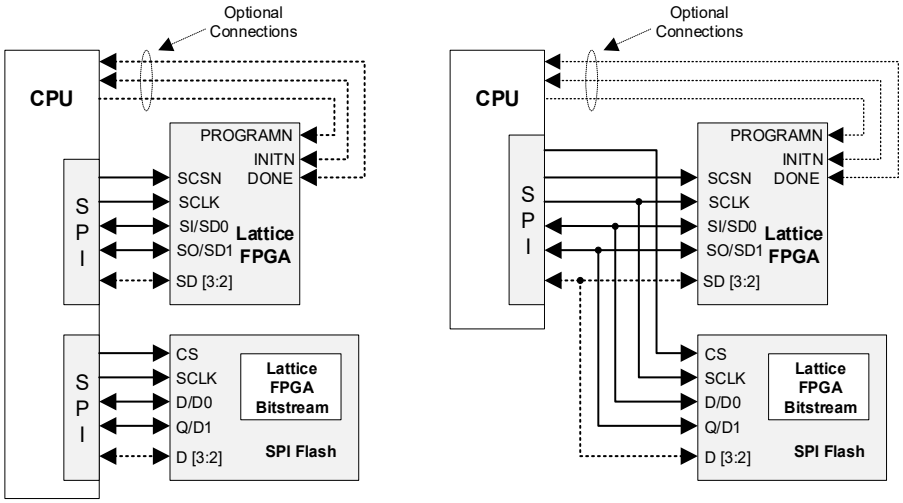


Figure 6.21. Nexus Slave SPI Port with SPI Flash

Notes:

- The dotted lines indicate the connection is optional.
- The Nexus device bitstream can reside in the SPI flash device instead of the system flash memory. The advantage of this is that the bitstream can be easily updated without changing the system software.

6.2.1. Method to Enable the Slave SPI Port

Similar to all configuration ports, the Slave SPI port is enabled by the two standard methods:

- Enable the Slave SPI Configuration Port in Configuration Mode.
  At Power Up or PROGRAMN pin toggle LOW (longer than $t_{PROGRAMN}$) or REFRESH command execution, holding the PROGRAMN LOW to postpone the master auto-booting event. Release PROGRAMN, then wait for INITN to go high. After INITN goes high, drive the SCSN of the Slave SPI Port and shift to the Slave Configuration Port Activation Key, as shown in Table 6.5. After the Slave SPI Configuration Port is activated, the state of the PROGRAMN pin could be either remained low or released high, but next PROGRAMN falling edge (longer than $t_{PROGRAMN}$) resets the Slave configuration port activation.

Table 6.5. Slave SPI Configuration Port Activation Key

Slave Port/Activation Key	Slave Configuration Port Activation Key	
Slave SPI port	Dummy bytes[1]	32'HA4C6F48A[1]

Note:

1. The number of dummy bytes should be at least 1, and only the last shifted in 32 bits matter.

- Enabling Slave SPI port persistence in user mode.
  The SSPI port could be persisted in user mode by setting the desired Value (SERIAL, DUAL or QUAD) for SLAVE_SPI_PORT in the Lattice Radiant Device Constraint Editor. Once set, the configuration bitstream contains optional Slave SPI persistence bits. When the device completes configuration and wakes up, it checks the persistent bits to determine if the SLAVE SPI port is to remain operational once in user mode. This selection is independent of the CFG port arbitration during configuration phase. A port enabled by persistence is a Transparent Mode port. It reserves those pins from being occupied by user logic.
  Both the DONE pin and the INITN pin must be high to qualify the Slave SPI port as a read back port. If not, then the device is not in user mode. The persistent bits have no affect when the device is not in user mode.

6.2.2. Specifications and Timing Diagrams – Slave SPI Port Waveforms

Data and commands shift into the MOSI pin on the rising edge of clock. Data is shifted out of the MISO pin on the falling edge of the clock. Only a read command causes the MISO pin to be enabled for data read out.

The Slave SPI read and write waveforms are shown in Figure 6.22 and Figure 6.23.

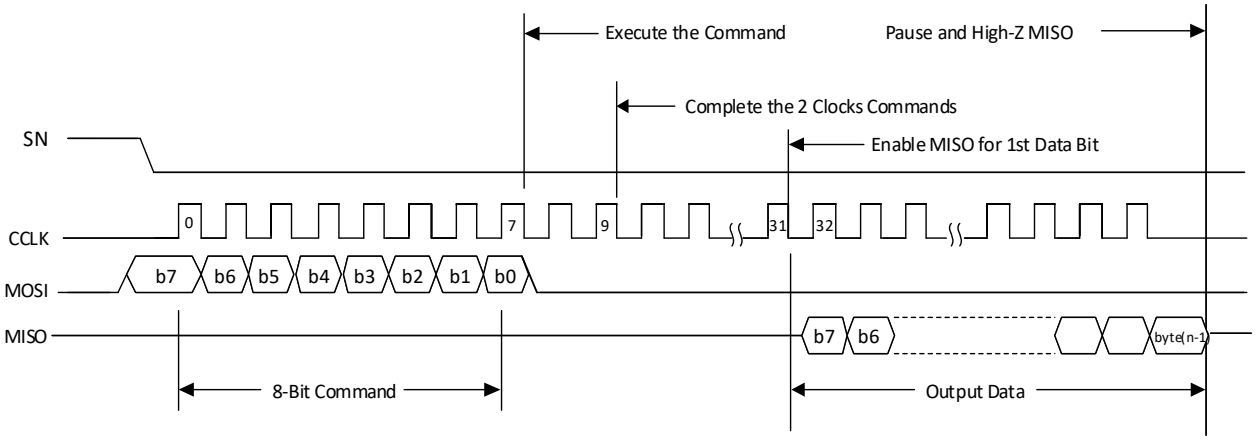
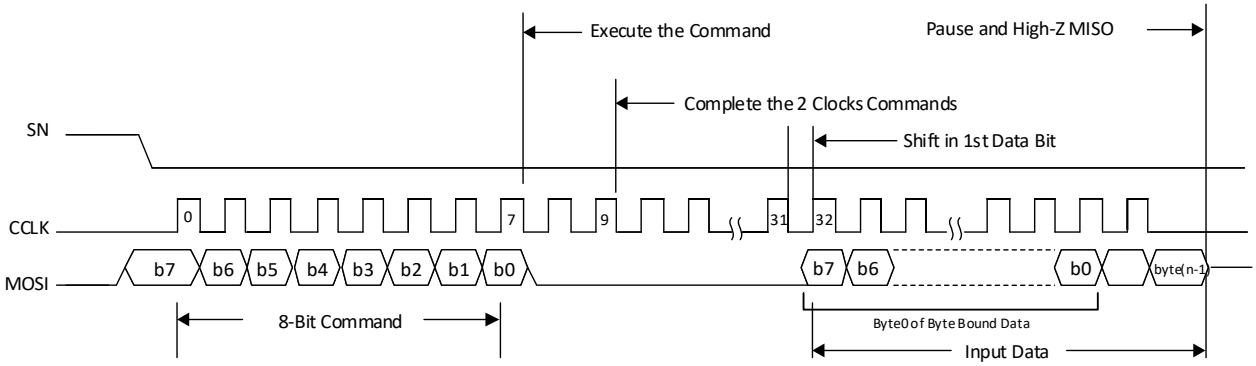


Figure 6.22. Slave SPI Read Waveforms



Note: The bitstream is transferred starting with the first byte of the data file, starting with the MSB of the byte.

Figure 6.23. Slave SPI Write Waveforms

6.2.3. Slave SPI Port AC Timing Requirements

The Slave SPI port maximum operation frequency requirement is shown in Table 6.6.

Table 6.6. Slave SPI Port AC Timing Requirements

Description	Parameter	Min	Max	Unit
SCLK frequency	f_{CCLK}	—	135	MHz

For detailed AC timing requirement for the Nexus Slave SPI configuration port, refer to the sysCONFIG Port Timing Specifications of CrossLink-NX Family Data Sheet (FPGA-DS-02049), Certus-NX Family Data Sheet (FPGA-DS-02078), CertusPro-NX Family Data Sheet (FPGA-DS-02086), and CrossLink-NX-33 and CrossLinkU-NX Data Sheet (FPGA-DS-02104).

6.2.4. Dual and Quad Slave SPI Port

By default, the SPI port data width is x1 (by 1). That is, there is only one bit of input and one bit of output. However, to allow faster loading of configuration information, some devices support wider versions of the SPI interface. For this reason, the Slave SPI configuration port of the Nexus device also support x2 (Dual) and x4 (Quad) modes of operation.

For Slave SPI, the Dual and Quad modes are enabled or disabled through 32-bit special commands. Those commands are processed at port level. The command codes are shown in Table 6.7.

Table 6.7. Special Commands for Dual and Quad Mode Enable/Disable

Commands	Command Code	Note
Dual Mode Enable	32'hD9B33EB3	Enable the DUAL Mode (X2)
Quad Mode Enable	32'hD9B33EBD	Enable the QUAD Mode (X4)
Parallel Mode Disable	32'hD9FFFFFF	Disable parallel mode, set back to X1 (serial) mode

The Dual and Quad mode can only be enabled from the default Serial mode. The Dual or Quad Mode Enable command must be shifted in serially from the slave SPI SI pin starting from the first clock, after the slave chip select (SCSN) pin is pulled low for 32 bits. The command execution happens upon the SCSN pulling high. Once the DUAL (X2) or QUAD (X4) mode is enabled, the Slave SPI data throughput, for either Non-JTAG command or data, is expanded from X1 to X2, X4 through the SD0 – SD3 pins as shown in Table 4.4.

At POR or in the event of PROGRAMN pin toggle (low) or REFRESH command execution, the Slave SPI port is reset to the default serial (X1) mode.

6.2.5. Slave SPI Configuration Flow Diagrams

The Slave SPI port supports the regular Nexus device bitstream and the encrypted bitstream (SSPI Mode).

The Slave SPI configuration flow diagram is shown in Figure 6.24 and the highlights are listed below.

- The bitstream file is a stand-alone file. It is not part of the driver or system code. This provides seamless system integration and flexible file management. For example, the bitstream can be switched on the fly without changing a single line of system code.
- The Nexus device is capable to wake up the I/O Banks on the left and right side of the device to user specified value at the beginning of the bitstream.
- The Nexus FPGA Fabric wakes up and enter user mode once it reads in the entire bitstream. If it is necessary to delay the wake-up, the simplest method is by using the DONE pin. Wake-up can be delayed by holding the open-drain DONE pin low until the wake-up is desired.

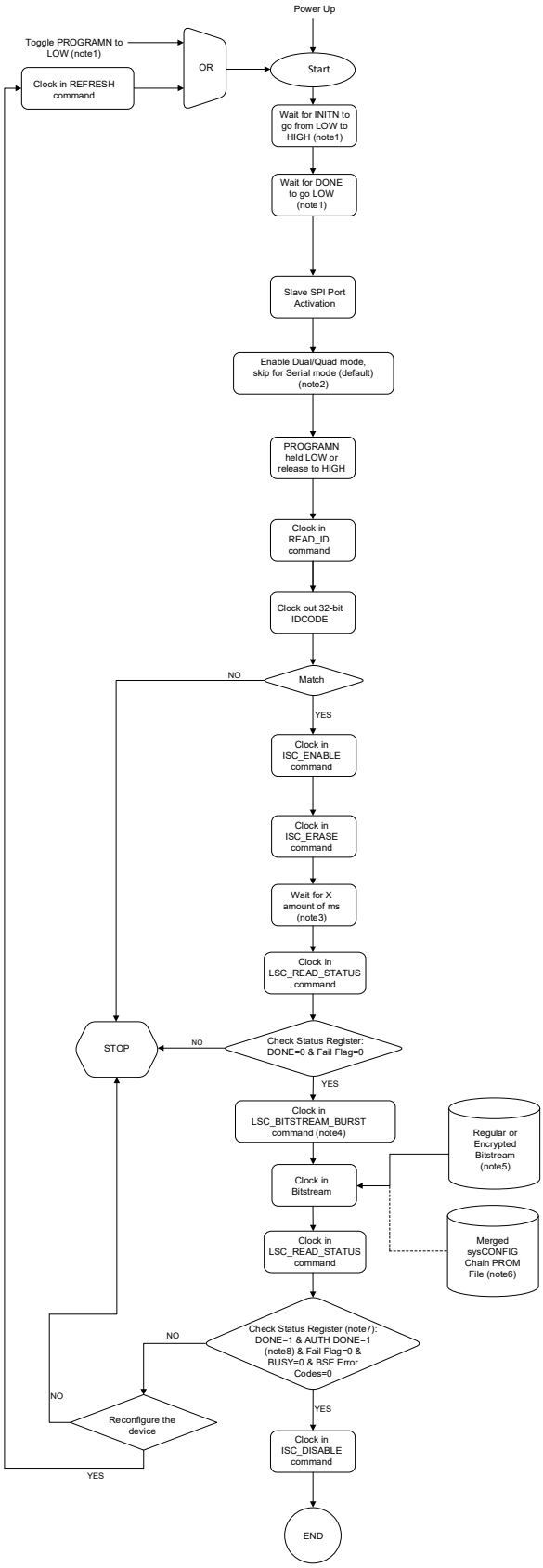


Figure 6.24. Slave SPI Configuration Flow

Notes:

1. For more information about the behavior of these pins, refer to the PROGRAMN, INITN, and DONE section. For more information about the timing specifications, refer to the CertusPro-NX Family Data Sheet.
2. For dual/quad mode, SCSN must be de-asserted (high) after the Enable command is sent.
3. Refer to Table 6.8 for X-amount of ms execution time per Nexus device.
4. For more information, refer to the LSC_BITSTREAM_BURST section and Figure 6.44.
5. The input file is a bitstream, which may be a standard or encrypted bitstream.
6. For a sysCONFIG chain of devices, the input file can be a merged PROM file.
7. The external DONE pin toggles high after the DONE bit and AUTH DONE bit (if applicable) in the status register are updated. If the external DONE pin is not available for monitoring, after clocking in the bitstream, observe the following wait times before checking the status register: 102.06 ms (encrypted bitstream) or 60 µs (regular bitstream). The status register can be checked at any time, but the DONE bit and AUTH DONE bit (if applicable) will only be updated after the specified wait time.
8. The AUTH DONE=1 check is only applicable if the encrypted bitstream is used.

Table 6.8. Execution Time for ISC_ERASE Command

Device Name	Fixed Execution Time Value (ms)
LIFCL-17, LFD2NX-9, LFD2NX-17	2.29
LIFCL-33	1.55
LIFCL-33U	1.55
LIFCL-40, LFD2NX-28, LFD2NX-40	2.65
LFCPNX-50	4.87
LFCPNX-100	4.87
LFD2NX-15	1.92
LFD2NX-25	1.92
LFD2NX-35	3.02
LFD2NX-65	3.02

6.2.6. Command Waveforms

6.2.6.1. Class A Command Waveforms

The Class A commands are ones that read data out from the Nexus device. MSB of the byte bound data or bitstream is read out first. The twenty-four (24) dummy clocks provide the device the necessary delay for the proper execution of the command.

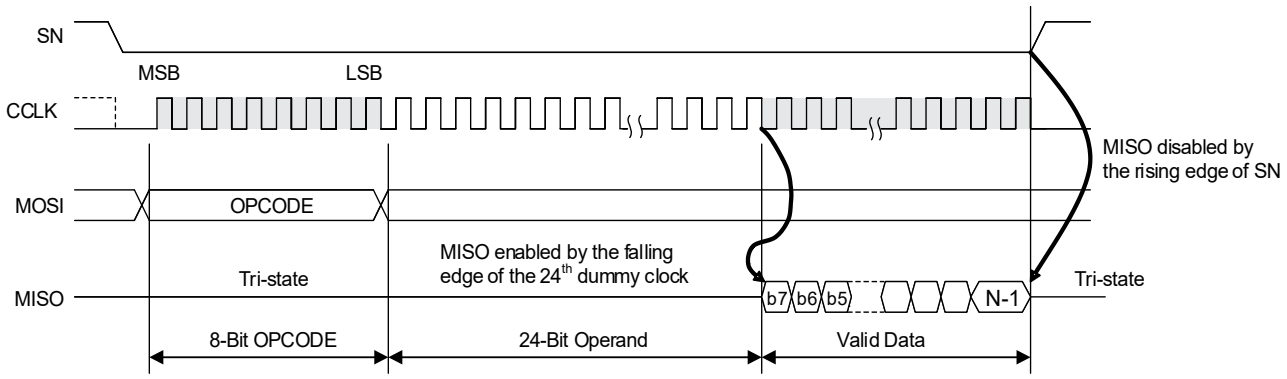


Figure 6.25. Class A Command Waveforms

6.2.6.2. Class B Command Waveforms

The Class B commands are used to shift data into the port. MSB of the byte bound data or bitstream is shifted in first. The twenty-four (24) dummy clocks provide the device the necessary execution time to execute the command properly.

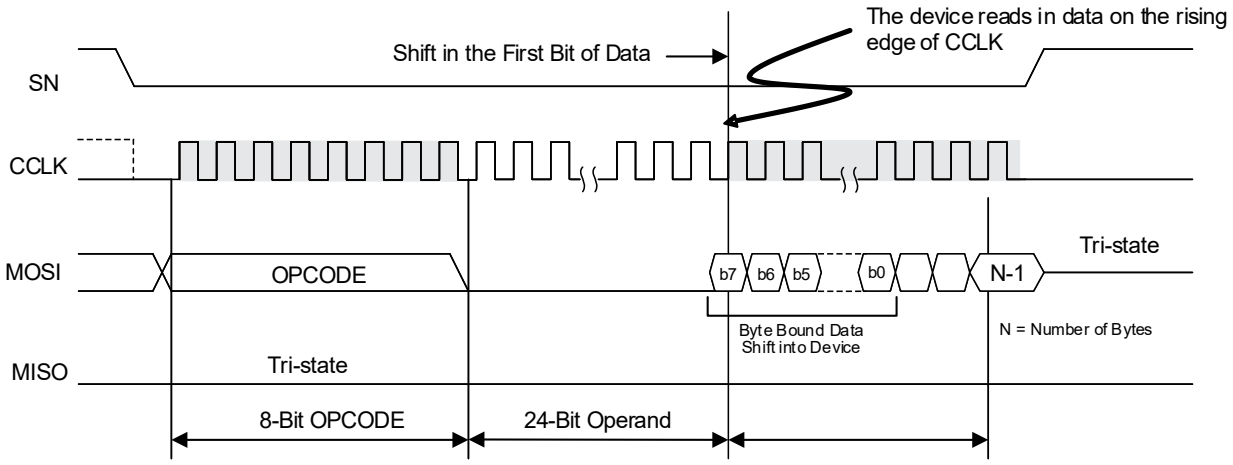


Figure 6.26. Class B Command Waveforms

6.2.6.3. Class C Command Waveforms

The Class C commands do not require any data to be shifted in or out. The twenty four (24) dummy clocks provide the device the necessary delay for the proper execution of the command. Even if extra dummy clocks are presented, the device ignores them.

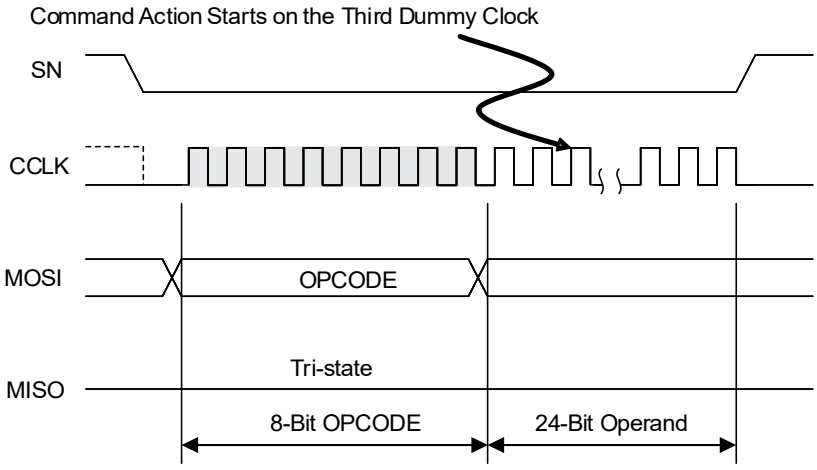


Figure 6.27. Class C Command Waveforms

6.2.6.4. Class D Command Waveforms

The Class D commands do not need to shift data in or out but still require a delay to execute the action associated with the command. This type of command cannot terminate the action of any commands including itself. After the 24th dummy clock, continuing to clock or suspending the clock or driving the SN pin high does not terminate the action. The action ends when it is complete. This class of commands is defined particularly for the benefit of the two unique commands: CLEAR and REFRESH.

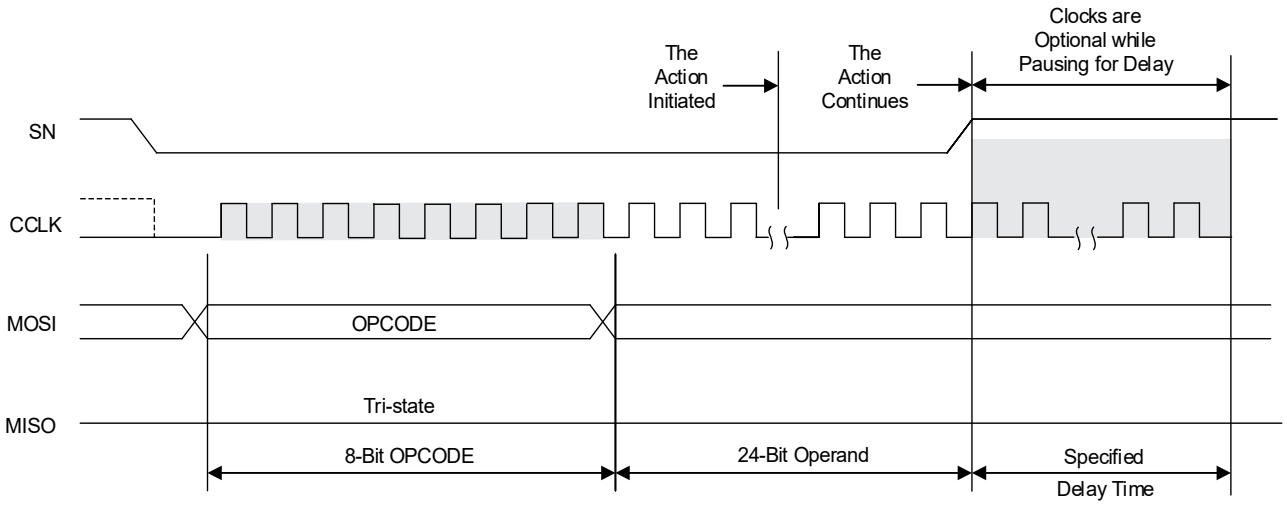


Figure 6.28. Class D Command Waveforms

6.2.7. Slave SPI to Master SPI Bridge

As the Master SPI port and Slave SPI port on Nexus device are allocated at different set of pins, the Nexus device provides the Slave SPI to Master SPI Bridge, which allows you to program the external SPI flash through the Nexus Slave SPI configuration port.

The LSC_PROG_SPI command is used to enable this bridging. Once this bridging is enabled, any data following this command on the SSPI interface is directed to the MSPI interface. Once this access is terminated by the external Host, by deasserting the Chip-Select (CSN) signal of SSPI, the bridging function is disabled and normal slave access for Configuration continues. The functional diagram of the SSPI to MSPI Bridge is shown in Figure 6.29.

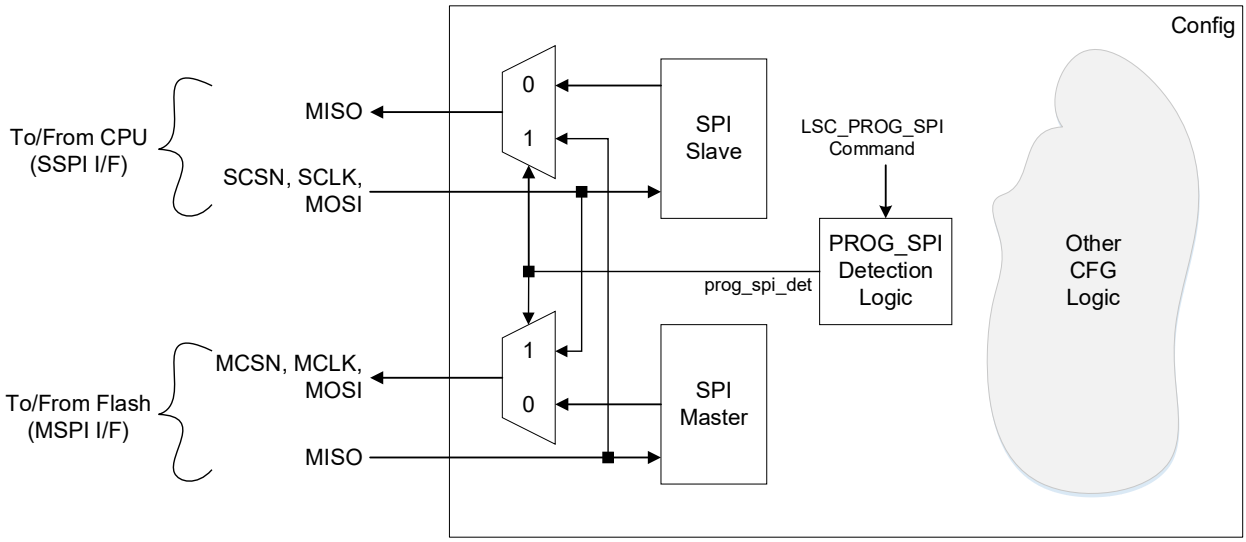


Figure 6.29. SSPI to MSPI Bridge Functional Diagram

Serial SSPI to MSPI bridging is supported. The diagram for SSPI to MSPI Bridge utilization is shown in Figure 6.30.

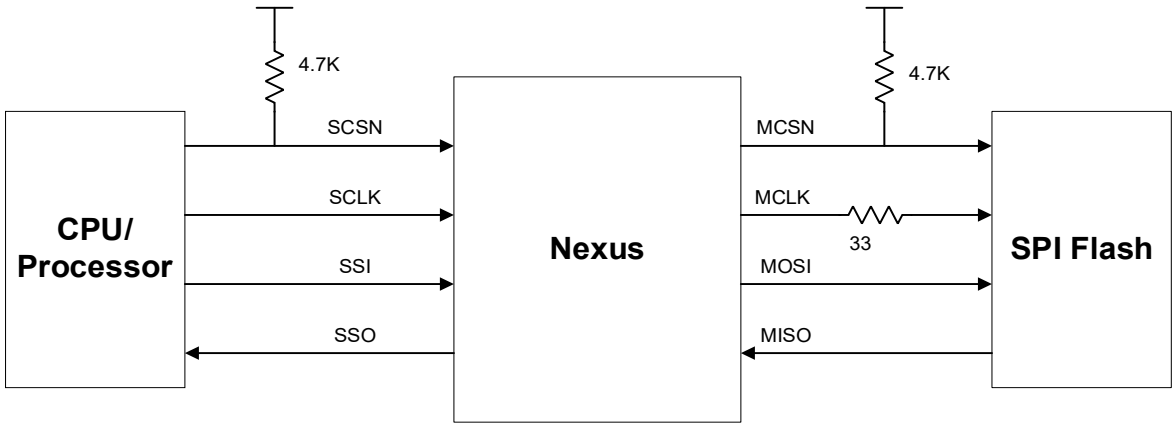


Figure 6.30. SSPI to MSPI Bridge Block Diagram

6.3. Slave I²C/I3C Mode

The Nexus device provides a Slave I²C/I3C configuration port that allows you to access features provided by the Configuration Logic. You can reprogram the Configuration SRAM and access status/control registers within the Configuration Logic block. When the Nexus device is in Transparent Mode, only read type commands for the Configuration SRAM are supported, allowing for device verification or debugging. The same non-JTAG command set, as shown in Table 6.17 is supported.

In the Slave I²C/I3C mode, as shown in Table 6.9, the SD2/SCL pin becomes SCL (that is, Serial Clock) of the I²C/I3C bus, and the SD3/SDA becomes the SDA (such as Serial Data) of the I²C/I3C bus.

The Slave I²C/I3C port can be activated through the CFG port arbitration procedure before the device enters user mode. The I²C/I3C port can be persisted in user mode by setting the desired Value for SLAVE_I2C_PORT/SLAVE_I3C_PORT in the Lattice Radiant Device Constraint Editor.

Using the I²C port simplifies the Nexus device configuration process. Lattice provides *C* source code called i2cembedded to insulate you from the complexity of programming the Nexus device. Refer to the Lattice Radiant online help about i2cembedded.

Table 6.9. Slave I²C/I3C Configuration Port Pins

Pin Name	Function	Direction	Description
SD2/SCL	SCL	Inout	SCL, Serial Clock, for I²C or I3C bus
SD3/SDA	SDA	Inout	SDA, Serial Data, for I²C or I3C bus

The Nexus device I²C/I3C configuration setup is shown in Figure 6.31.

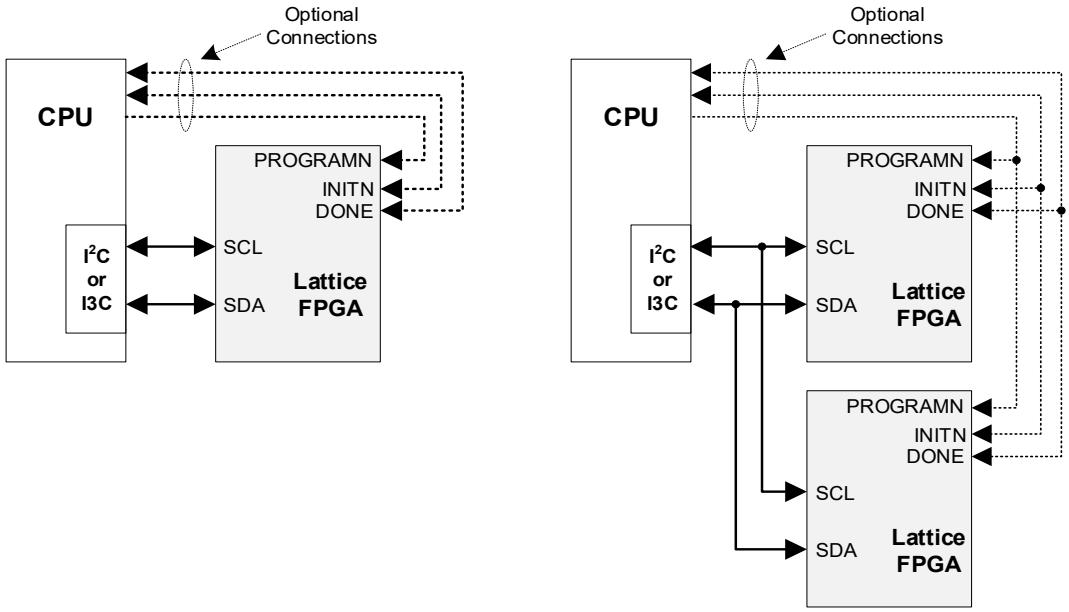


Figure 6.31. Nexus Slave I²C/I3C Port with CPU and Single or Multiple Devices

6.3.1. Bus Sharing Between I²C and I3C

The Nexus device configuration block support both slave I²C and slave I3C interface, the two share the same bus as shown in Figure 6.32.

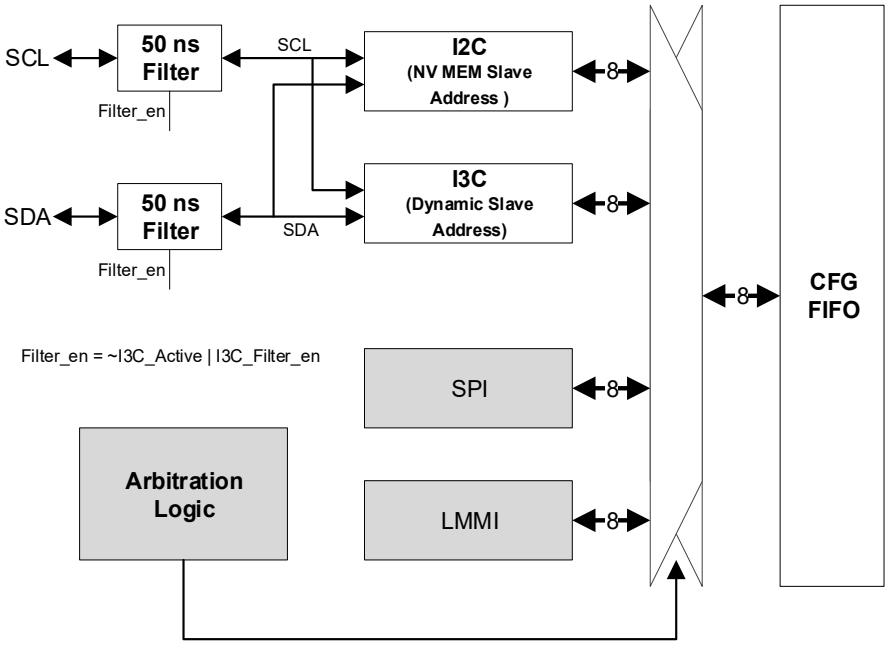


Figure 6.32. Bus Sharing between I²C and I3C

6.3.2. Slave I²C/I3C Configuration Mode

From configuration perspective, the I²C or I3C interface only support slave mode. They are for configuration purpose only, not usable by user logic. This hard slave I²C block and I3C block for configuration provide the industry standard two-pin interface for I²C/I3C masters to communicate with the Nexus device configuration engine.

6.3.2.1. Slave I²C Port Capabilities

- Standard I2C bus protocol to communicate with non-JTAG engine.
- User-programmable Slave address to override the hardware default one (EFUSE, One Time Programmable).
- Built in Port Activation Detection logic to detect the Lattice Slave Configuration Port Activation Code.
- Built in PREAMBLE detection logic for bitstream alignment when execute LSC_BITSTREAM_BURST command.
- Built in 50 ns filters for SCL and SDA make it capable of working within newer I3C bus.

6.3.2.2. I3C Slave Port Capabilities

This I3C slave configuration port is dedicated to device configuration only. It is not usable by user logic, The I3C slave configuration port is compliant with the MIPI I3C (Improved Inter Integrated Circuit) standard v1.0, and supports the following I3C features:

- Two wire serial interface up to 12.5 MHz using Push-Pull.
- Legacy I²C device co-existence on the same bus.
- Dynamic addressing while supporting static addressing for legacy I²C support.
- Legacy I²C messaging.
- I²C-like Single Data Rate messaging (SDR).
- Built in Port Activation Detection logic to detect the Lattice Slave Configuration Port Activation Code.
- Built in PREAMBLE detection logic for bitstream alignment when execute LSC_BITSTREAM_BURST command.

The Slave I3C configuration port provides the 48-bit Provisional ID (PID):

- Bit [47:33]: Hardwired MIPI Manufacturer ID (15 bits) setting at 0x019E
- Bit [32]: Provisional ID Type Selector: Selectable from EFUSE FEATURE setting (default 0).
- Bit [32] = 0: Provisional ID Bit [31:0] containing Vendor Fixed Value.
- Bit [32] = 1: Provisional ID Bit [31:0] containing Pseudo random value.

When Bit [32] is set at 0, the Provisional ID Bit [31:0] are defined as follows:

- Bit [31:16]: Part ID: 16-bit from IDCODE Bit [31:20], [15:12].
- Bit [15:12]: Instance ID: 4-bits from Pin EFUSE FEATURE setting.
- Bit [11:0]: 12-bit Device Characteristics code.
- Bit [11:8] from EFUSE FEATURE setting;
- Bit [7:0] from DCR register.

The Bus Characteristics Register (BCR) is set at 0x00.

- BCR[7:6] : I3C Slave
- BCR[5] : SDR only
- BCR[4] : Not a bridge device
- BCR[3] : Device always responds
- BCR[2] : No data byte follows the accepted IBI
- BCR[1] : Not IBI Capable
- BCR[0] : No speed limitation

The Device Characteristics Register (DCR) is set at 0xE1.

6.3.3. Slave Address for I²C and I3C Ports

6.3.3.1. Slave Address for I²C Configuration Port

An external I²C master accesses the Configuration Logic using 7'b100,0000 (0x40) for 7-bit addressing and 10'b11,1100,0000 (0x3C0) for 10-bit addressing as default unless the CFG I²C base address has been modified. The CFG I²C slave address could be altered by 8 bits user programmable EFUSE bits (YYYXXXXX) inside Feature Row. Using Program File Utility – Feature Row Editor, the user specified I²C slave address becomes XXXXX00 for 7-bit mode and YYYXXXXX00 for 10-bit address.

6.3.3.2. Slave Address for I3C Configuration Port

For Legacy I²C support, the Nexus device allows you to program 8 bits EFUSE bit (AAABBBBB) to setup the I3C configuration port using the default static address 7'b111,0000 (0x70) for 7-bit addressing and 10'b11,1111,0000 (0x3F0) for 10-bit addressing, with BBBBB00 for 7-bit mode and AAABBBBB00 for 10-bit mode. The Slave I3C Configuration Port receives its dynamic address during the Dynamic Address Assignment (DAA) process initiated by the main I3C master on the bus.

6.3.4. Method to Enable the Slave I²C/I3C Port

Similar to all configuration ports, the Slave I²C or I3C port is enabled by the two standard methods:

- Enable the Slave I²C Configuration Port in Configuration Mode.
  At Power Up or PROGRAMN pin toggle LOW (for certain period) or REFRESH command execution, holding the PROGRAMN LOW to postpone the master auto booting event. Then the Master performs the write activity to the slave address of the Nexus Slave I²C or I3C configuration port along with the Slave Configuration Port Activation Key, as shown in Table 6.10. After the Slave I²C or I3C Configuration Port has been activated, the state of the PROGRAMN pin could be either remained low or released high, but next PROGRAMN falling edge (longer than $t_{PROGRAMN}$) can reset the Slave configuration port activation.

Table 6.10. Slave I²C/I3C Configuration Port Activation Key

Slave Port/ Activation Key	Slave Configuration Port Activation Key	
Slave I²C/I3C Port	Slave I²C/I3C Port Address[1]	32'HA4C6F48A

Note:

1. The slave I²C/I3C address could be either 7 bits or 10 bits address.

- Enabling Slave I²C/I3C port persistence in user mode.
  The I²C/I3C port could be persisted in user mode by setting the desired Value (ENABLE) for SLAVE_I2C_PORT/SLAVE_I3C_PORT in Lattice Radiant Device Constraint Editor. Once set, the configuration bitstream contains optional I²C/I3C persistence bits. When the device completes configuration and wakes up, it checks the persistent bits to determine if the SLAVE I²C/I3C port is to remain operational once in user mode. This selection is independent of the CFG port arbitration during configuration phase. A port enabled by persistence is a Transparent Mode port. It reserves those pins from being occupied by user logic.
  Both the DONE pin and the INITN pin must be high to qualify the Slave I²C or I3C port as a read back port. If not, then the device is not in user mode. The persistent bits have no affect when the device is not in user mode.

6.3.4.1. Slave I²C Port Activation Flow

The standard Slave I²C Configuration Port activation flow is shown in Figure 6.33.

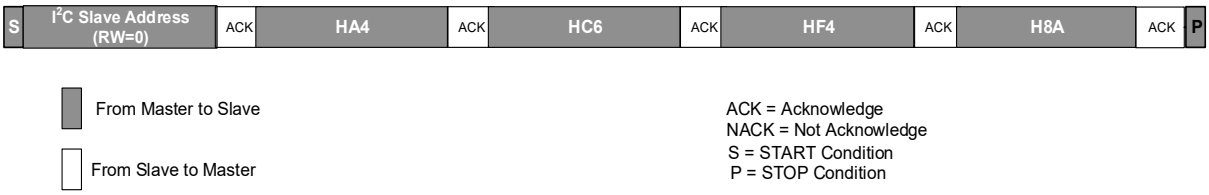


Figure 6.33. Slave I²C Configuration Activation Flow

6.3.4.2. Slave I3C Port Activation Flow

The typical Slave I3C Configuration Port activation flow is shown in Figure 6.34.

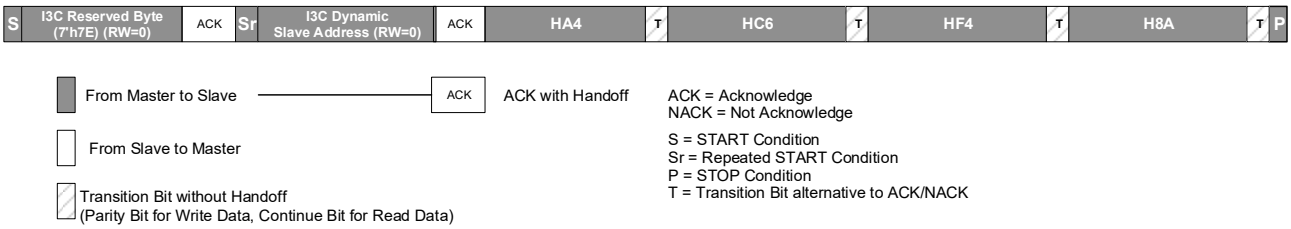


Figure 6.34. Slave I3C Configuration Activation Flow

6.3.5. Slave I²C/I3C Configuration Logic Access

The Slave I²C Configuration Port or Slave I3C Configuration Port follows corresponding standard to access the Nexus device by non-JTAG configuration engine, using the same non-JTAG command set, as shown in Table 6.17.

6.3.5.1. Slave I²C Configuration Access Flow

The typical I²C configuration write and read flows are shown in Figure 6.35 and Figure 6.36.

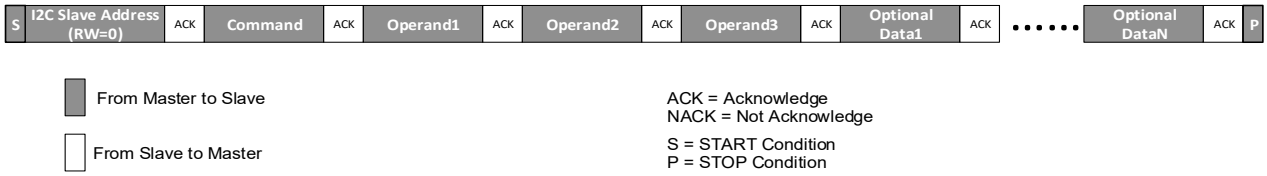


Figure 6.35. Typical I²C Write

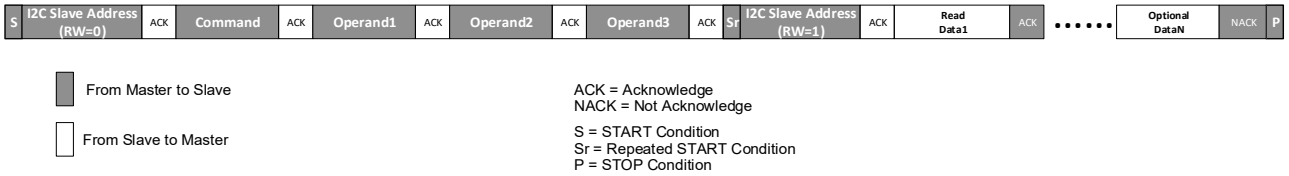


Figure 6.36. Typical I²C Read

6.3.5.2. Slave I3C Configuration Access Flow

The typical I3C read flows for class A commands are shown in Figure 6.37 and Figure 6.38. The typical I3C write flows for class B, class C, and class D commands are shown in Figure 6.39 and Figure 6.40. For more information on I3C dynamic address, refer to the *Enter Dynamic Address Assignment* section in the I3C Controller IP Core User Guide (FPGA-IPUG-02228).

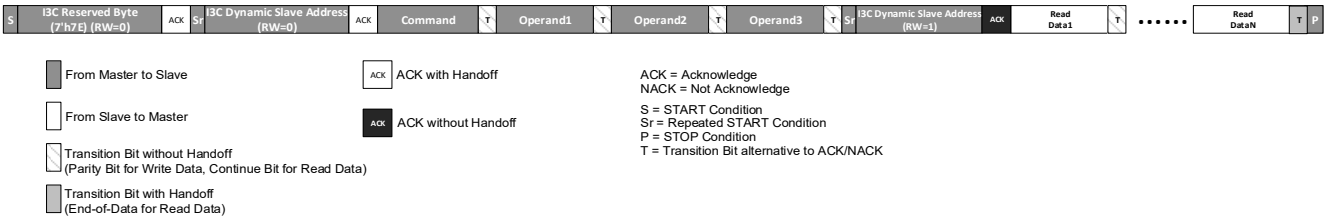


Figure 6.37. Typical I3C Read with 7'h7E

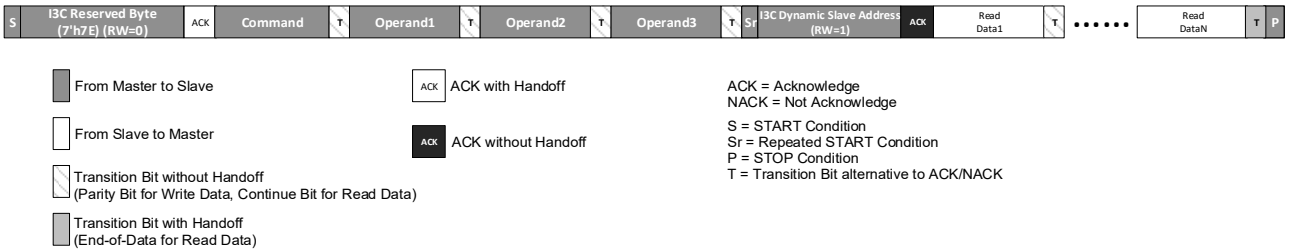


Figure 6.38. Typical I3C Read without 7'h7E

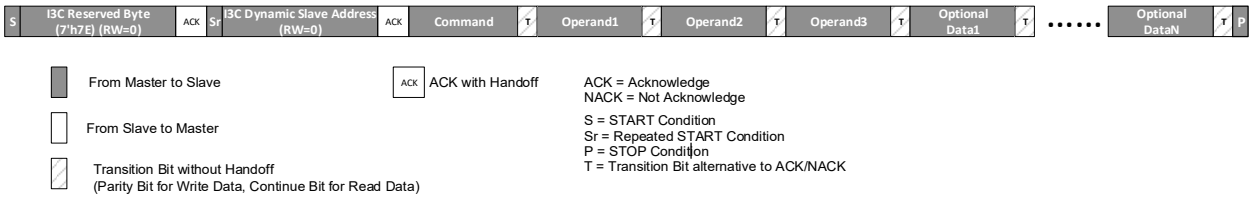


Figure 6.39. Typical I3C Write with 7'h7E

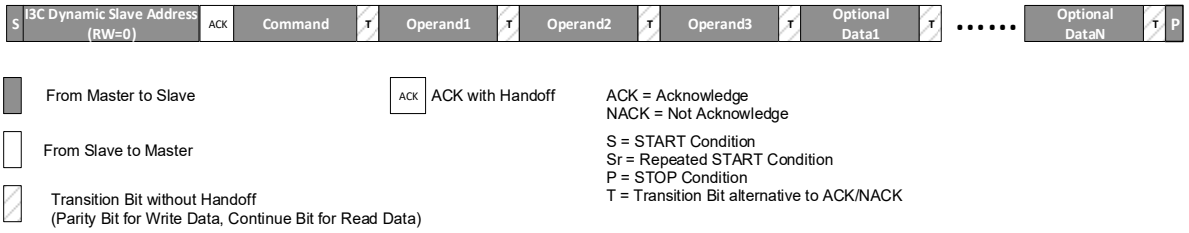


Figure 6.40. Typical I3C Write without 7'h7E

Notes:

1. The optional data frames are for class B commands.
2. There should be a specific delay after the class D commands to allow the execution to finish before the next command. To make the delay adaptive, the LSC_CHECK_BUSY command loop is recommended after the class D commands except for the ISC_ERASE command.

The Nexus device is fully compliant to all required I3C Common Command Code (CCC) under MIPI specification. Refer to the MIPI I3C Basic v1.0 for the set of I3C command codes and definitions.

6.3.6. Slave I²C/I3C Configuration Flow Diagrams

The Slave I²C/I3C port supports the regular Nexus device bitstream and the encrypted bitstream.

The Slave I²C/I3C configuration flow diagram is shown in Figure 6.41 and the highlights are listed below.

- The bitstream file is a stand-alone file. It is not part of the driver or system code. This provides seamless system integration and flexible file management. For example, the bitstream can be switched on the fly without changing a single line of system code.
- The Nexus device is capable to wake up the I/O banks on the left and right side of the device to user-specified value at the beginning of the bitstream.
- The Nexus FPGA Fabric wakes up and enter user mode once it reads in the entire bitstream. If it is necessary to delay the wake-up, the simplest method is by using the DONE pin. Wake-up can be delayed by holding the open-drain DONE pin low until the wake-up is desired.

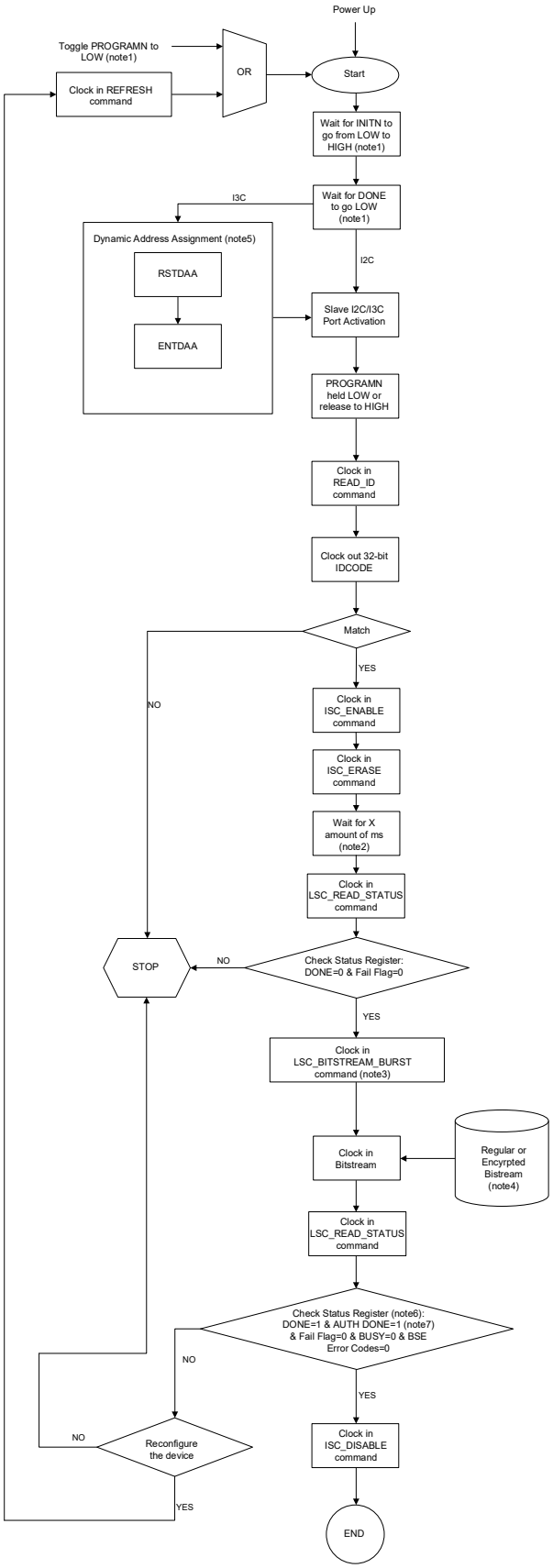


Figure 6.41. Slave I²C/I3C Configuration Flow

Notes:

1. For more information about the behavior of these pins, refer to the PROGRAMN, INITN, and DONE sections. For more information about the timing specifications, refer to the respective Nexus family data sheet.
2. Refer to Table 6.8 for X-amount of ms execution time per Nexus device.
3. The following should be considered when sending the bitstream using the LSC_BITSTREAM_BURST command:
   a. The bitstream burst can be transmitted in several segments or as a single transaction.
   b. When sending the bitstream in segments, the first transaction should begin with a START condition. Subsequent transactions must begin with a RESTART condition.
   c. The bus controller must occupy the I^2C/I3C bus until the complete bitstream has been transmitted.
   d. The bus controller must ensure the I^2C/I3C bus does not enter the STOP condition before the complete bitstream has been transmitted. The STOP condition causes the Bitstream Engine (BSE) to terminate the programming sequence – if the bitstream condition has not been fully transmitted when the STOP condition occurs, the BSE reports an *Abort Error* through the BSE error status register.
   e. The bus controller should ensure there are no additional edges on SCL, which can cause bit alignment issues. Under these conditions, the BSE reports an *Illegal Command Error* or *CRC Error*.
   f. For more information on the usage of the LSC_BITSTREAM_BURST command, refer to the LSC_BITSTREAM_BURST section.

   For debug purposes, through the Radiant Programmer, you have the option to send the bitstream using the **Erase, Program, Verify** operation. Using this option, the bitstream is sent one frame at a time until the end of the bitstream. Sending each frame begins with a START condition and ends with a STOP condition.
4. The input file is a bitstream, which may be a standard or encrypted bitstream.
5. The master performs RSTDAA (Reset Dynamic Address Assignment) before assigning a Dynamic Address to the Nexus device. The master can perform ENTDAA (Enter Dynamic Address Assignment) to assign Dynamic Address to the Nexus device. The master can also assign a Dynamic Address to the Nexus device with a known Static Address, using the Command Code SETDASA (Set Dynamic Address from Static Address). The I3C Bridge programming solution supported in the Radiant Programmer using the Command Code ENTDAA to assign a Dynamic Address to the Nexus device. For more information on I3C dynamic address assignment, refer to the *Enter Dynamic Address Assignment* section in the I3C Controller IP Core User Guide (FPGA-IPUG-02228). The Nexus device is fully compliant to all the required I3C Common Command Code (CCC) under the MIPI specification. Refer to the MIPI I3C Basic v1.0 for the set of I3C command codes and definitions.
6. The external DONE pin toggles high after the DONE bit and AUTH DONE bit (if applicable) in the status register are updated. If the external DONE pin is not available for monitoring, after clocking in the bitstream, observe the following wait times before checking the status register: 102.06 ms (encrypted bitstream) or 60 µs (regular bitstream). The status register can be checked at any time, but the DONE bit and AUTH DONE bit (if applicable) will only be updated after the specified wait time.
7. The AUTH DONE=1 check is only applicable if the encrypted bitstream is used.

6.3.7. Slave I^2C/I3C AC Timing Requirements

The Slave I^2C and I3C maximum operation frequency requirements are listed in Table 6.11.

Table 6.11. Slave I^2C/I3C Maximum Frequency Requirements

Description	Parameter	Min	Max	Unit
I^2C Maximum SCL Clock Frequency	F_{SCL_I2C}	—	1	MHz
I3C Maximum SCL Clock Frequency	F_{SCL_I3C}	—	12	MHz

For the detailed AC timing requirements for the Nexus device Slave I^2C/I3C configuration port, refer to the sysCONFIG Port Timing Specifications of the CrossLink-NX Family Data Sheet (FPGA-DS-02049), Certus-NX Family Data Sheet (FPGA-DS-02078), CertusPro-NX Family Data Sheet (FPGA-DS-02086), and CrossLink-NX-33 and CrossLinkU-NX Data Sheet (FPGA-DS-02104).

6.4. JTAG Mode

The Nexus device provides a four-pin JTAG engine, which is fully compliance with IEEE 1149 (Standard Test Access Port and Boundary-Scan Architecture) and IEEE1532 (Standard for In-System Configuration of Programmable Devices) standards. A separate dedicated JTAG_EN pin can enable the JTAG port at any time. The JTAG port on Nexus devices provides:

- Offline external flash memory programming
- Background external Flash memory programming
- Direct SRAM configuration
- Full access to the Nexus Device Configuration Logic
- Device chaining
- IEEE 1149.1 testability
- IEEE 1532 Compliant programming

The advantages of keeping the JTAG port available include:

- Multi-chain Architectures – The JTAG port is the only configuration and programming port that permits the Nexus device to be combined in a chain of other programmable logic.
- Reveal Debug – The Lattice Reveal debug tool is an embeddable logic analyzer tool. It allows you to analyze the logic inside the Nexus device in the same fashion as an external logic analyzer permits analysis of board level logic. Reveal access is only available through the JTAG port. Refer to the Appendix A – Reveal User Guide of the Lattice Radiant Software User Guide for more information on the Lattice Reveal debug tool.
- SRAM Readback – The JTAG port can directly access the configuration SRAM. It is occasionally necessary to perform failure analysis for SRAM based FPGAs. A key component for failure analysis is reading the configuration SRAM.
- Boundary Scan Testability – Board level connectivity testing performed using IEEE 1149.1 JTAG is a key capability for assuring the quality of assembled printed-circuit-boards. Lattice provides Boundary Scan Description Language files for the Nexus device on the Lattice website.

6.4.1. Method to Enable the JTAG Port

The four pins for the JTAG port on Nexus devices are dual-purpose I/O, which are shared with user functionalities. Once the dedicated JTAG_EN pin is driven HIGH, it can enable the JTAG port at any time, no matter whether the device is in configuration mode or user functional mode.

6.4.2. JTAG Port AC Timing Requirements

The JTAG port AC timing requirements are listed in Table 6.12.

Table 6.12. JTAG AC Timing Requirements

Description	Parameter	Min	Max	Unit
TCK Frequency	f_{MAX}	—	25	MHz

For detailed AC timing requirements for the Nexus JTAG port, refer to the JTAG Port Timing Specifications of CrossLink-NX Family Data Sheet (FPGA-DS-02049), Certus-NX Family Data Sheet (FPGA-DS-02078), CertusPro-NX Family Data Sheet (FPGA-DS-02086), and CrossLink-NX-33 and CrossLinkU-NX Data Sheet (FPGA-DS-02104).

6.4.3. JTAG to Master SPI Bridge

The Nexus device provides the JTAG to Master SPI Bridge, which allows you to program the external SPI flash through the Nexus JTAG port.

The LSC_PROG_SPI instruction is used to enable this bridging. There is a 16-bit TDR between TDI and TDO for this instruction. Upon the TAP controller going through the Update-DR state first time, the internal TDI signal, the gated TCK signal, the Shift-DR signal, and the internal TDO signal are unconditionally connected respectively to the MOSI/MD0 pin, the MCLK pin, the MCSN pin, and the MISO/MD1 pin.

As an example, the diagram for JTAG to MSPI Bridge utilization is shown in Figure 6.42.

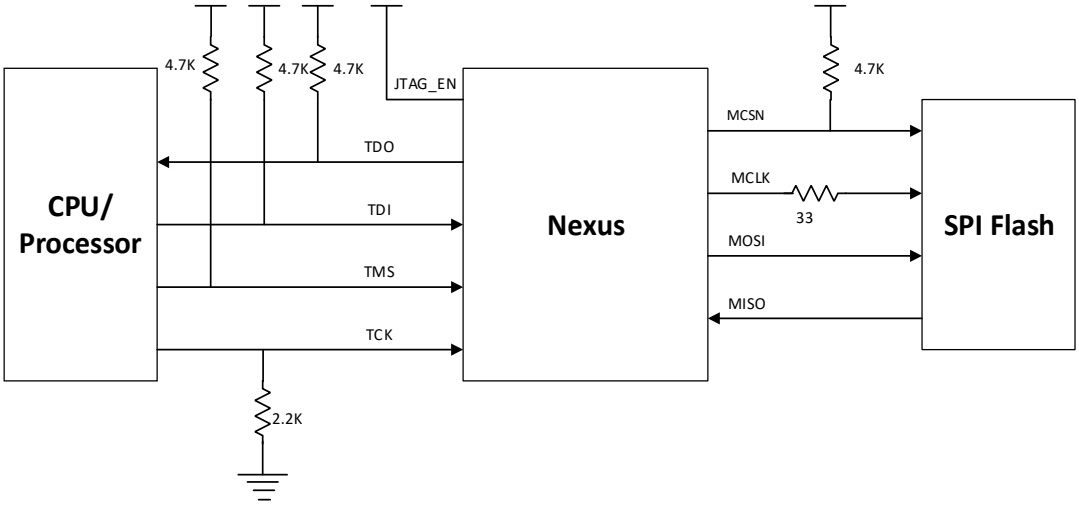


Figure 6.42. JTAG to MSPI Bridge Block Diagram

6.4.4. JTAG ispTracy/Reveal Support

The JTAG ispTracy feature supported in FPGAs is the optional addition of internal logic analyzers (ILA). These ILAs have features similar to external logic analyzers such as programmable event and trigger conditions and deep trace memory. Logic analyzer IP (two current versions are currently supported: ispTRACY and REVEAL) consists of one ispJTAG core plus numerous ispLA logic analyzer cores (device limited). The IP block ispJTAG implements two embedded registers to support logic analyzers. To implement the ispJTAG core, the configuration block has an 11-port I/O interface to the core logic as seen in Figure 6.43.

To be IEEE 1149.1 compliant, there is a default 16-bit register implemented for registers ER1 and ER2 to allow the size of this register to be a fixed length whenever these registers are not implemented in FPGA logic. Whenever these registers are implemented in FPGA logic, these 16-bit registers are replaced with a new fixed length register that is based on the customer application – to allow for IEEE 1149.1 compliance, the BSDL file needs to be modified by the customer to define the lengths of these new ER1 and/or ER2 registers. This new implementation is different from that of all previous device architectures.

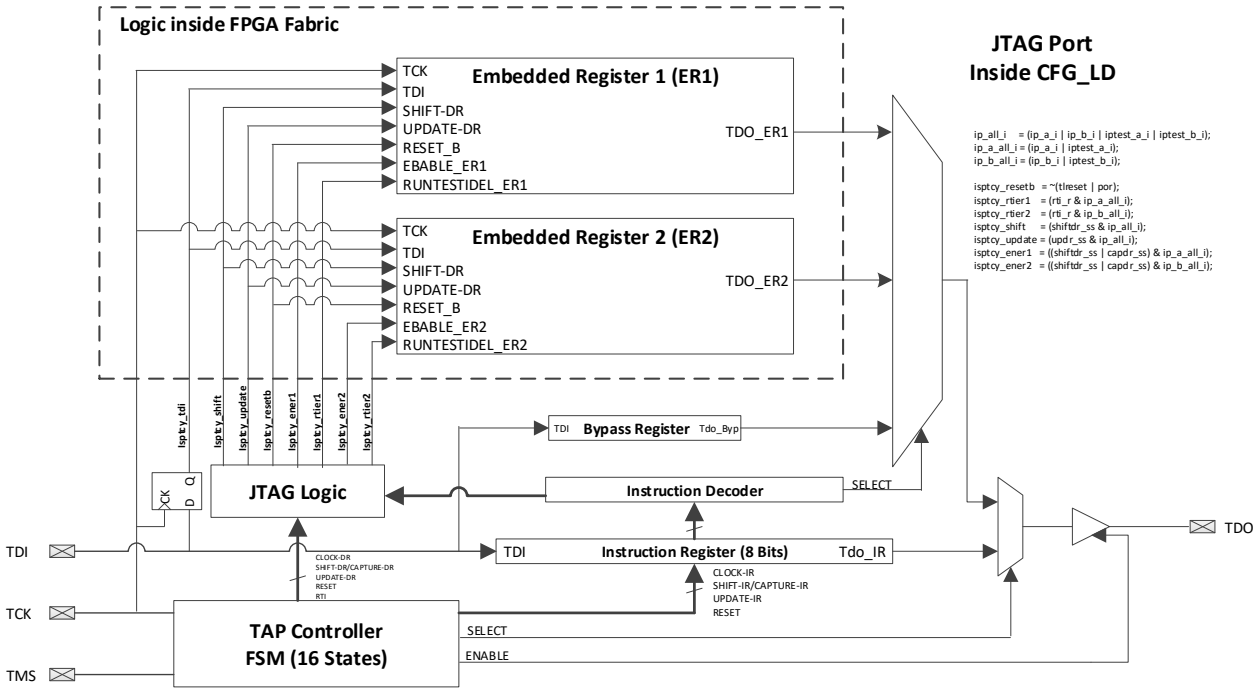


Figure 6.43. JTAG ispTracy Interface

In user functional mode, the four JTAG pins located in I/O BANK1 can have the functionality changed from user function to JTAG function by bringing JTAG_EN pin to VCCIO1. Hence, the TMS/TDI/TCK pins need to be configured as INPUT or BI-DIRECTIONAL in user mode.

The JTAG driver to Nexus needs to match the JTAG pins I/O type. For example, if four JTAG pins in Bank 1 use LVCMOS33 then the JTAG driver needs to be 3.3 V.

When JTAG_EN is high in user mode, the four signals need to be HIGHZ from other devices beside the JTAG driver.

After the JTAG_EN goes low in user mode, the drive strength of these four pins remains 8 mA until 200 ms later and need dynamic switching to return to user mode drive strength. If these four pins stay static after JTAG_EN goes low, they remain 8 mA drive strength until switching.

6.5. Device Status Register

The Nexus device has a 64-bit device status register, which indicates the status of the device. The READ_STATUS command shifts out this 64-bit internal status of the device. The device status register can only be read by the slave configuration port, such as JTAG port, Slave SPI port, Slave I^2C port, Slave I3C port, or LMMI interface.

Table 6.13. 64-bit Device Status Register

Bit	Function	Status Value			Comments
		Reset Value	0	1	
0	TRAN Mode (Transparent Mode)	0	No	Yes	Device is currently in Transparent mode.
[3:1]	CONFIG Target Selection	000	—	—	000 SRAM array.
					001 EFUSE Normal; Program input data to EFUSE Memory directly, Read back data from Shadow register.
					010 EFUSE Pseudo; Program input data to Shadow Register, Read back data from Shadow register.
					011 EFUSE Safe; Program Shadow Register data into EFUSE memory, Read back data from Shadow register.
4	JTAG Active	0	No	Yes	JTAG ISC Machine is currently active.
5	PWD Protection	0	No	Yes	Configuration logic is password protected.
6	Erase Enable	0	No	Yes	When set to 1, it indicates that erase is enabled.
7	Reserved	0	—	—	—
8	DONE	0	No	Yes	Done bit has been set.
9	ISC Enable	0	No	Yes	JTAG instructions are being executed with ISC Enabled.
10	Write Enable	0	Not Writable	Writable	Selected configuration target is write-protected from at least one of the following setup: • Selected configuration target security bit is set. • Password protection is enabled and password is mismatched.
11	Read Enable	0	Not Readable	Readable	Read-protected from at least one of the following setup: • Security bit is set. • Password protection is enabled and password is mismatched.
12	Busy Flag	0	No	Yes	Configuration logic is busy.
13	Fail Flag	0	No	Yes	Last instruction/command execution failed.
14	WDT Reboot	0	No	Yes	Status bit indicates that the watchdog timer triggered a refresh.
15	Decrypt Only	0	No	Yes	Only encrypted data are accepted.
16	PWD Enable	0	No	Yes	Password protection is enabled for EFUSE.
17	PWD All	0	No	Yes	Password protection is extended for SRAM.
18	UID EN	0	No	Yes	Customer ID is enabled for IDCODE_PUB command.
19	SDM Enable	0	No	Yes	Status bit indicates that the internal non-volatile memory has been fully programmed with the FPGA configuration pattern and is ready to be downloaded for the next booting event.

Bit	Function	Status Value			Comments
		Reset Value	0	1	
20	Lattice PreAmble	0	—	—	Lattice Encrypt-Preamble (for encrypted bitstream file) is detected. This bit is not a valid indicator of preamble status if the bitstream is loaded through the MSPI.
21	Encrypt PreAmble	0	No	Yes	Encrypted Preamble is detected. This bit is not a valid indicator of preamble status if the bitstream is loaded through the MSPI.
22	STD PreAmble	0	No	Yes	Standard Preamble is detected. This bit is not a valid indicator of preamble status if the bitstream is loaded through the MSPI.
23	SPIm Fail	0	No	Yes	Failed to configure from the primary pattern.
[27:24]	BSE Error Code	0000	—	—	0000 No error.
					0001 ID error.
					0010 CMD error – illegal command detected.
					0011 CRC error.
					0100 PRMB error – preamble error.
					0101 ABRT error – configuration is aborted.
					0110 OVFL error – data overflow error.
					0111 SDM error – bitstream pass the size of the SRAM array.
					1000 Authentication Error[1]
					1001 Authentication Setup Error[1]
					1010 Bitstream Engine Timeout Error.
28	EXEC Error	0	No	Yes	Error occurs during execution.
29	ID Error	0	No	Yes	ID mismatch with Verify_ID command.
30	Invalid Command	0	No	Yes	Invalid command received.
31	WDT Busy	0	No	Yes	Watch Dog Timer is busy.
32	UDS Programmed	0	No	Yes	Internal use.
33	Dry Run DONE	0	No	Yes	Bit to indicate the pseudo done bit been set by Dry Run activity.
[37:34]	BSE Error 1 Code for previous bitstream execution	0000	—	—	0000 No error.
					0001 ID error.
					0010 CMD error – illegal command detected.
					0011 CRC error.
					0100 PRMB error – preamble error.
					0101 ABRT error – configuration is aborted.
					0110 OVFL error – data overflow error.
					0111 SDM error – bitstream pass the size of the SRAM array.
					1000 Authentication Error[1]
					1001 Authentication Setup Error[1]
					1010 Bitstream Engine Timeout Error.
38	Bypass Mode	0	No	Yes	Device currently in Bypass mode.
39	Flow Through Mode	0	No	Yes	Device currently in Flow Through Mode.
40	Version	0	No	Yes	Version = 1 for production device.
41	Reserved	0	—	—	—
42	BSE Timeout	0	No	Yes	Indicates MSPI mode has timed out when trying to read the signature from flash.
43	Key Destroy Pass[1]	0	No	Yes	Key destroy function passed.

Bit	Function	Status Value			Comments
		Reset Value	0	1	
44	INITN	0	Low	High	INITN Pin state.
45	I3C Parity Error 2	0	No	Yes	I3C received a parity error type S2.
46	INIT Bus ID Error	0	No	Yes	Initialization Bus ID Error occurred.
47	I3C Parity Error 1	0	No	Yes	I3C received a parity error type S1.
[49:48]	Authentication Mode[1]	00	—	—	Indicates what authentication mode is set. 00 – No Auth 01 – ECDSA 10 – HMAC 11 – No Auth
50	Authentication Done[1]	0	No	Yes	Indicates that authentication has completed.
51	Dry Run Authentication Done[1]	0	No	Yes	Indicates that Dry Run authentication has completed.
52	JTAG Locked	0	No	Yes	Indicates that the JTAG port is locked.
53	SSPI Locked	0	No	Yes	Indicates that the Slave SPI port is locked.
54	I^2C/I3C Locked	0	No	Yes	Indicates that the I^2C/I3C port is locked.
55	PUB Read Lock[1]	0	No	Yes	ECDSA Public key read lock is enabled.
56	PUB Write Lock[1]	0	No	Yes	ECDSA Public key write lock is enabled.
57	FEA Read Lock	0	No	Yes	Feature Row Read Lock is enabled.
58	FEA Write Lock	0	No	Yes	Feature Row Write Lock is enabled.
59	AES Read Lock	0	No	Yes	AES Key Read Lock is enabled.
60	AES Write Lock	0	No	Yes	AES Key Write Lock is enabled.
61	PWD Read Lock	0	No	Yes	Password Read Lock is enabled.
62	PWD Write Lock	0	No	Yes	Password Write Lock is enabled.
63	Global Lock	0	No	Yes	Global Lock is enabled for OTP rows.

Note:

1. For Certus-NX, CrossLink-NX, and CertusPro-NX devices only.

6.6. Control Register 0 (CR0)

The 32 bits Control Register 0 is used to control configuration logic behavior during and after configuration. Write the CR0 through LSC_PROG_CNTRL0 command and read back the CR0 content through LSC_READ_CNTRL0 command. Also, the CR0 could be written through the LSC_PROG_CNTRL0 command with the Bitstream file. The bit definition of the CR0 is shown in Table 6.14.

Table 6.14. Bit Definition for Control Register 0

BIT	Description	Default	Note
[31:30]	Reserved	00	—
29	Wake Up Trans	0	Control bit transparent reconfiguration for the event of PROGRAMN pin toggle or REFRESH command execution. When this bit is set to *0,* the wake-up signals GOE, GWE and GSRN go low during reconfiguration. When this bit is set to *1*, the wake-up signals GOE, GWE and GSRN are set according to the user setting of NDR at the time of PROGRAMN pin toggle or REFRESH command execution.
28	NDR(TransFR)	0	Non-Disturbing Re-configure: Control bit for supporting Leave Alone I/O for reconfiguration. If set, the I/O output maintains previous value after entering ISC ACCESS state for configuration instead being tri-stated when device enters transparent access mode from PROGRAMN pin toggle or REFRESH command execution.
27	Reserved	0	—
26	No CDM	0	Control bit to bypass CDM downloading for REFRESH or PROGRAMN pin toggle. Control Information Download Mode (CDM) – The stage to download critical device control information from non-volatile memory to shadow registers.

BIT	Description	Default	Note
25	TranCRAM	0	Control bit to enable the write operation to the SRAM array in transparent mode.
24	TranEBR	0	This bit provides user option for EBR control through the INIT Bus in transparent access mode. If set, configuration logic takes over the control of EBR in transparent access mode, user access to EBR is suspended.
23	TranIP	0	This bit provides user option for Hard IP control in transparent access mode. If set, configuration logic takes over the control of the Initialization Bus in transparent access mode, user access to Hard Ips is suspended.
22	TranHSE	0	This bit provides the user the option for Hardware Security Engine/Module (HSE) control in transparent access mode. If set, the configuration logic takes over control of the HSE module in transparent access mode and user access to the HSE is suspended.
21	No Boot	0	This bit disables the automatic booting activity for the next PROGRAMN pin toggle or REFRESH command execution event.
20	SRME	0	The Slow Response Mode Enable (SRME) bit enables the automatic insertion of Lattice-specific protocol to handle the issue caused by slow response of non-volatile operation for high-speed slave SPI read. If set to 1, an all-zero byte (8'H00) is transmitted right before the actual data as an indicator for data validation. If the data is not yet available, a dummy value (not equal to 8'H00) is transmitted repeatedly until data becomes available for transmitting. Upon data availability, the flag byte (8'H00) is transmitted, followed by the actual data. It is up to the SPI master device to continue to toggle the SPI clock (SCLK) and maintain Slave Select (SCSN) low until the valid data is transmitted. In cases where the read data is immediately available, if this control bit is set, then no dummy bytes are transmitted. However, valid data is still preceded by the flag byte (8'H00). If this control bit is clear (1'b0), then no dummy byte or flag byte is transmitted. However, data from reads that might conflict with certain internal resources is not guaranteed. Data from reading of device ID is always guaranteed as this process will not conflict with slower internal resources.
19	SPIM	0	Control bit for Multiple Boot Enable for next refresh event (PROGRAMN pin toggle or REFRESH command execution). 0 – Use hard coded boot address (H000000 for first boot; HFFFF00 for second boot). 1 – Use boot address from CIB SRAM bits depend on bitstream setting
[18:17]	P_DONE CRTL	00	Overload control for PROGRAM_DONE command in bitstream for configuration daisy chain setup: 10 – Overload with BYPASS 11 – Overload with FLOW_THROUGH 0X – No Overload (Default)
[16:15]	INIT OPT	00	Reserved for internal testing.
[14:13]	DONE OPT	00	Reserved for internal testing.
[12:10]	STX DUM	000	Control bits determine the number of dummy bytes padding to add before the first frame is read out during incremental bitstream read back when using the slave configuration port. 000 – 0 001 – 3 010 – 4 011 – 8 100 – 16 101 – 32 110 – 64 111 – 128
9	Reserved	0	—
8	Config IO Voltage	0	Must be set to 0.
[7:6]	Slew Rate Control (for Config output pins).	00	Slew rate control for configuration output pins defined as: 00 – Slow

BIT	Description	Default	Note
			01 – Medium 10 – Fast 11 – Fast
[5:0]	Master Clock Frequency (On-chip Flash Clock Select)	000000	Control bits that set the divide ratio to derive the Master SPI clock from the on-chip oscillator. The master clock frequency can be set through the sysCONFIG option MCCLK_FREQ.

6.7. Control Register 1 (CR1)

The 32 bits Control Register 1 is used to control EFUSE/OTP access control logic behavior during and after configuration. User could Write the CR1 through LSC_PROG_CNTRL1 command and read back the CR1 content through LSC_READ_CNTRL1 command. The CR1 can also be written through the LSC_PROG_CNTRL1 command with the Bitstream file. The bit definition of the CR1 is shown in Table 6.15.

Table 6.15. Bit Definition for Control Register 1

Bit	Definition	Default	Note
[31:29]	Core CLK Sel	000	Control bits to set the configuration engine clock frequency with respect to the internal oscillator frequency. 000 – Divide by 3 001 – Divide by 4 010 – Divide by 5 011 – Divide by 6 100 – Divide by 7 101 – Divide by 8 110 – Divide by 9 111 – Divide by 10
[28:26]	Master Signature Timer Count	000	Number of clock cycles to get a valid LSCC/SFDP signature. 000 – 600000 (default) 001 – 400000 010 – 200000 011 – 40000 100 – 20000 101 – 4000 110 – 2000 111 – 400
25	I3C FILT	0	Enables I2C filter in I3C.
24	DPA Enable	0	DPA enable for bitstream decryption.
23	CPHA	0	This control bit is used to select SPI clock format. 0 – Sampling of data occurs at the rising edge of the clock 1 – Sampling of data occurs at the falling edge of the clock
22	CPOL	0	This control bit selects an inverted or non-inverted clock. 0 – Active high clocks selected. In idle state clock is low 1 – Active low clocks selected. In idle state clock is high
21	TX Edge	0	Control bit to adjust TX clock edge by half clock cycle.
20	RX Edge	0	Control bit to adjust RX clock edge by half clock cycle.
19	LSBF	0	Control bit for MSPI to send out commands and address from least significant bit first.
18	SSPI Auto	0	Enables the SSPI auto function for downstream daisy chain.
17	32-bit MSPI Address	0	Enables 32-bit MSPI addressing mode, default 24-bit addressing.
16	Bulk Erase Enable	0	Enables auto SRAM bulk erase for REFRESH command or PROGRAMN pin toggle.
15	SFDP Enable	0	Enables checking the SFDP signature during master SPI booting. Should be enabled for flash devices that support SFDP.

Bit	Definition	Default	Note
			0 – Enable reading LSCC signature from flash starting at address 0x000. 1 – Enable reading SFDP signature from flash SFDP header.
14	32-bit MSPI Commands	0	Sends 32-bit command set on MSPI, default 24-bit command set.
13	Disable I/O Glitch	0	Disables I/O Glitch Filter during configuration.
12	CFG Noise	0	If set to 1, CFG can enable HSE noise generation.
[11:8]	Slave Idle Timer Count	0000	Slave configuration mode hangup prevention idle timer value. 0000 – Disabled 0001 – 25000 ms 0010 – 10000 ms 0011 – 7500 ms 0100 – 5000 ms 0101 – 2500 ms 0110 – 1000 ms 0111 – 750 ms 1000 – 500 ms 1001 – 250 ms 1010 – 100 ms 1011 – 75 ms 1100 – 50 ms 1101 – 25 ms 1110 – 10 ms 1111 – 5 ms
[7:5]	Master Timer Count	000	Number of clock cycles to get a valid preamble. 000 – 600000 001 – 400000 010 – 200000 011 – 40000 100 – 20000 101 – 4000 110 – 2000 111 – 400
4	Signature Disable	0	If SFDP Enable = 0 do not read LSCC signature from flash, go directly to read preamble. 0 – Signature enabled (default 0) 1 – Skip checking signature when SFDP Enable = 0
[3:2]	Master Retry Count	00	Number of times the MSPI mode preamble detection should be retried. 00 – No retry 01 – Retry 1 time 10 – Retry 3 times 11 – Retry forever
1	Signature Infinite Retry	0	Retry the SFDP/LSCC signature an infinite number of times.
0	I/O Ready Disable	0	When set to 1, it disables waiting for I/O ready for device wakeup.

6.8. TransFR Operation

The CertusPro-NX device provides the TransFR™ capability. For more information about the TransFR operation, refer to the Minimizing System Interruption During Configuration Using TransFR Technology (FPGA-TN-02198).

6.9. Slave Command Support

6.9.1. Slave Command Format

Non-JTAG commands consist of one byte mandatory OPCODE, three bytes mandatory OPERANDs (except for ISC_NOOP command, which is OPCODE only), and optional data bytes and dummy bytes which are defined on a per command basis as shown in Table 6.16.

Table 6.16. Slave Command Support

OPCODE (1 Byte)	OPERAND1 (1 Byte)	OPERAND2 (1 Byte)	OPERAND3 (1 Byte)	DATA (X Bytes)	DUMMY (4 Bytes)	CRC CHKSUM (2 Bytes)
[7:0]	[7:0] [15:8] [23:16]			—	—	—
Mandatory	Mandatory	Mandatory	Mandatory	Optional	Optional	Optional

6.9.2. Slave Command Set

The ISC and programming commands for all non-JTAG configuration interfaces such as SPI, I^2C, I3C, and LMMI are shown in Table 6.17.

Table 6.17. Slave Configuration Interface Command

Command	OPCODE		Operand (Bytes)	Data (Bytes)	ISC Qualify	Class[4]	Execution Time	Description
	Binary	Hex						
ISC_NOOP	11111111	FF	0	0	—	C	None	Non-operation.
ISC_NOOP0	00000000	00	0	0	—	C	None	Non-operation.
READ_ID	11100000	E0	3	4	—	A	None	Read out the 32-bit IDCODE of the device.
READ_UNIQUEID	00011001	19	3	8	—	A	None	Reads the 64-bit UNIQUE ID, 56 bits set by Lattice, 8 bits set by customer.
USERCODE	11000000	C0	3	4	—	A	None	Read 32-bit user code.
LSC_READ_STATUS	00111100	3C	3	8	—	A	None	Read out internal status.
LSC_CHECK_BUSY	11110000	F0	3	1	—	A	None	Read 1 bit busy flag to check the command execution status.
LSC_DEVICE_CTRL	01111101	7D	3	0	—	B	[3]	An 8-bit field in Operand 1 allows various device control, such as GSR, Sleep, Dry Run, and others. Bit 0: Cause a Global Set/Reset to occur on the device. Bit 1: Launch a One-time check of SED. Bit 2: Reserved Bit 3: Cause a reset on configuration logic related FSM and flags. Bit 4: Reserved for internal dry-run. Bit [6:5]: Launch dry-run event. 2'b01 –Dry-run for primary bitstream 2'b10 – Dry-run for secondary bitstream 2'b11 – Slave dry-run 2'b00 – No dry-run Bit 7: CRC check bit
LSC_REFRESH	01111001	79	3	0	—	D	[3]	Equivalent to toggle PROGRAMN pin.

Command	OPCODE		Operand (Bytes)	Data (Bytes)	ISC Qualify	Class[4]	Execution Time	Description
	Binary	Hex						
ISC_ENABLE	11000110	C6	3	0	—	C	None	Enable the Offline configuration mode.
ISC_ENABLE_X	01110100	74	3	0	—	C	None	Enable the Transparent configuration mode.
ISC_DISABLE	00100110	26	3	0	—	C	None	Disable the configuration operation.
LSC_BITSTREAM_BURST	01111010	7A	3	BSF[5]	Yes	B	[3]	Configure the Device by burst in bitstream file in slave mode.
ISC_PROGRAM_USERCODE	11000010	C2	3	4	Yes	B	None	Write the 32-bit new USERCODE data to USERCODE register.
ISC_ERASE	00001110	0E	3	0	Yes	D	[2]	Bulk erase the memory array based on the access mode and array selection.
ISC_PROGRAM_DONE	01011110	5E	3	0	Yes	D	None	Program the DONE bit if the device is in Configuration state.
ISC_ERASE_DONE	00100100	24	3	0	Yes	D	None	Clear the DONE bit if the device is in Configuration state.
ISC_PROGRAM_SECURITY	11001110	CE	3	0	Yes	C	None	Program the Security bit if the device is in Configuration state.
LSC_INIT_ADDRESS	01000110	46	3	0	Yes	C	None	Initialize the Address Shift Register.
LSC_WRITE_ADDRESS	10110100	B4	3	4	Yes	B	None	Write the 16-bit Address Register to move the address quickly.
LSC_PROG_INCR	10000010	82	3	DSR[6]	Yes	B	[1]	Write the configuration data to the configuration memory frame at current address and post increment the address.
LSC_PROG_INCR_CMP[9]	10111000	B8	3	DSR[6]	Yes	B	[1]	Decompress the configuration data; write the data to the configuration memory at current address and post increment the address.
LSC_READ_INCR	01101010	6A	3	DSR[6]	Yes	A	None	Read back the configuration memory data frames selected by the address register and post increment the address.
LSC_PROG_CNTRL0	00100010	22	3	8	Yes	B	None	Modify the Control Register 0.
LSC_READ_CNTRL0	00100000	20	3	4	Yes	A	None	Read the Control Register 0.
LSC_PROG_CNTRL1	00100011	23	3	8	Yes	B	None	Modify the Control Register 1.
							[1]	Modify the NV content when access NVM.
LSC_READ_CNTRL1	00100001	21	3	4	Yes	A	None	Read the Control Register 1.
LSC_RESET_CRC	00111011	3B	3	0	Yes	C	None	Reset 16-bit frame CRC register to 0x0000.
LSC_READ_CRC	01100000	60	3	2	Yes	A	None	Read 16-bit frame CRC register content.

Command	OPCODE		Operand (Bytes)	Data (Bytes)	ISC Qualify	Class[4]	Execution Time	Description
	Binary	Hex						
LSC_PROG_SED_CRC	10100010	A2	3	4	Yes	B	None	Program the calculated 32-bit CRC based on configuration bit values only into overall CRC register.
LSC_READ_SED_CRC	10100100	A4	3	4	Yes	A	None	Read the 32-bit SED CRC.
LSC_PROG_PASSWORD	11110001	F1	3	16	Yes	B	[1]	Program 128-bit password into the non-volatile memory (Efuse). See additional note[8].
LSC_READ_PASSWORD	11110010	F2	3	16	Yes	A	None	Read out the 128-bit password before activated for verification.
LSC_SHIFT_PASSWORD	10111100	BC	3	8	Yes	B	None	Shift in the 128 bits password to unlock for re-configuration (necessary when password protection feature is active).
LSC_PROG_CIPHER_KEY	11110011	F3	3	32	Yes	B	[1]	Program the 256-bit cipher key into the non-volatile memory. See additional note[8].
LSC_READ_CIPHER_KEY	11110100	F4	3	32	Yes	A	None	Read out the 256-bit cipher key before activated for verification.
LSC_PROG_FEATURE	11100100	E4	3	12	Yes	B	[1]	Program security related feature such as Decrypt Enable, etc. based on the selection from the operand.
LSC_READ_FEATURE	11100111	E7	3	12	Yes	A	None	Read security related feature such as Decrypt Enable, etc. based on the selection from the operand.
LSC_PROG_FEABITS	11111000	F8	3	2	Yes	B	[1]	Program security related feature bits. See additional note[8].
LSC_READ_FEABITS	11111011	FB	3	2	Yes	A	None	Read security related feature bits.
LSC_PROG_OTP	11111001	F9	3	4	Yes	B	[1]	Program security related One-Time-Programmable bits based on the selection from the operand. See additional note[8].
LSC_READ_OTP	11111010	FA	3	4	Yes	A	None	Read security related One-Time-Programmable bits based on the selection from the operand.
LSC_WRITE_COMP_DIC[9]	00000010	02	3	16	Yes	B	None	Loads the most frequently used patterns into the device for decompressing compressed bitstreams.
LSC_INIT_BUS_ADDR	11110110	F6	3	4	Yes	B	None	Write the INIT Bus ID and Address Registers.
LSC_INIT_BUS_WRITE	01110010	72	3	BYTES[7]	Yes	B	None	Write data through the INIT Bus.
LSC_INIT_BUS_READ	11110111	F7	3	BYTES[7]	Yes	A	None	Read back data through INIT Bus.
LSC_PROG_SPI	00111010	3A	3	—	—	B	[3]	Connect SPI Host Port to the SPI interface to program the Internal/External SPI Flash, for JTAG-MSPI and SSPI-MSPI bridging.
LSC_PROG_ECDSA_PUBKEY	01011001	59	3	64	Yes	B	[1]	Program the ECDSA Public Key. See additional note[8].
LSC_READ_ECDSA_PUBKEY	01011010	5A	3	64	Yes	B	None	Read Back the ECDSA Public Key.
LSC_PROG_SEC_BOOT	01111111	7F	3	4	Yes	B		Secondary boot address override; 32 bits data to replace the hard coded secondary boot address.

Command	OPCODE		Operand (Bytes)	Data (Bytes)	ISC Qualify	Class[4]	Execution Time	Description
	Binary	Hex						
LSC_READ_DR_UES	01011101	5D	3	4	Yes	A	None	Read the dry-run User Electronic Signature shadow register.

Notes:

1. Execution time depends on data frame size and clock frequency. Duration could be in seconds. Use LSC_CHECK_BUSY to check the status.
2. Refer to Table 6.8 for the X-amount of execution time. After the execution time, use LSC_READ_STATUS to check that the DONE(Bit 8) and Fail Flag(Bit 13) are not '1' before you proceed to the next operation.
3. Execution time depends predominantly on the bitstream size and clock frequency. Duration could be in seconds.
4. The different command types are described in the Command Waveforms section.
5. The BSF is the Data Length of the whole Bitstream File.
6. The DSR is the Data Shift Register Length in Bytes of the device.
7. The BYTES is the number of bytes in a frame times the number of frames.
8. Command operation programs the internal EFUSE memory (non-volatile memory), which is one-time programmable (OTP).
9. The commands LSC_WRITE_COMP_DIC and LSC_PROG_INCR_CMP are only supported within the compressed bitstream. These commands cannot be executed directly through any sysCONFIG ports (JTAG, SSPI, or I2C/I3C).

All commands that require ISC qualification are only valid after the device is ISC ENABLED. Any ISC qualified command issued while the device is BUSY (Status Register bit [12] is *1*) is nullified. You should only issue ISC qualified commands when the device is not busy.

If specified, command operands are required even if they are not defined. Undefined operands should be all zero.

6.9.3. Slave Command Details

6.9.3.1. LSC_NOOP and LSC_NOOP0

Place a device in *no-operation* mode without the device leaving the ISC accessed modal state. This is often used when programming multiple devices in parallel with one device needs fewer operations than another.

6.9.3.2. READ_ID

Read out the 32-bit hardware IDCODE of the device. If the *UID_EN* feature bit is set, this IDCODE is the customer defined IDCODE (also referred to as User ID Code) stored in the device non-volatile Feature Row.

6.9.3.3. READ_UNIQUEID

Read the 64-bit UNIQUE ID (also referred to as Manufacture Electronic Signature, MES), 56 bits are set by Lattice Semiconductor and 8 bits set by customer as defined in the Feature Row.

6.9.3.4. USERCODE

Read the 32-bit USER Electronic Signature (also referred to as UES) programmed in the register by the ISC_PROGRAM_USERCODE command.

6.9.3.5. LSC_READ_STATUS

Read out the internal status bits such as OTP/EFUSE status, internal busy, fail, etc. from a 64-bit user status register, as defined in Table 6.13.

6.9.3.6. LSC_CHECK_BUSY

The 1-bit busy flag is corresponding to bit 7 of the return data byte. This is the same status bit as defined in the 64-bit user status register, which can be accessed by the LSC_READ_STATUS command.

6.9.3.7. LSC_DEVICE_CTRL

This command performs real-time control of actions on the device through 8-bit OPERAND1. This allows external Master devices to control actions on the device. The example of the 8-bit OPERAND is shown in Table 6.18.

Table 6.18. OPERAND1 Definition for LSC_DEVICE_CTRL Command

	Bit[7]	Bit[6:5]	Bit[4]	Bit[3]	Bit[2]	Bit[1]	Bit[0]
LSC_DEVICE_CTRL	RESERVED	DRY_RUN[1:0]	RESERVED	CFG_RST	RESERVED	SRAM_CHECK	SYS_RST

- Bit 0: Cause a Global Set/Reset to occur on the device.
- Bit 1: Launch a One-time check of SED. If SED is already running (including continuous SED mode), ignore this command. This command allows an LSC_READ_CRC_SED command to be issued later in order to determine if an SED error occurred
- Bit 2: Reserved
- Bit 3: Cause a reset on configuration logic related FSM and flags.
- Bit 4: Reserved
- Bit [6:5]: Launch dry-run boot. This loads the bitstream and checks the CRC of the non-volatile bits without writing the bits to the configuration SRAM. This is done in the background during normal device operation. Usercode is programmed to dry-run usercode.
  - dry_run[1:0] – bit 0 is primary and bit 1 is secondary
    - 2'b01 – Dry-run for primary bitstream (offset 0x00)
    - 2'b10 – Dry-run for secondary bitstream (secondary offset)
    - 2'b11 – Slave dry-run
    - 2'b00 – No dry-run
- Bit 7: Reserved

6.9.3.8. LSC_REFRESH

The basic function of the command is equivalent to toggling the PROGRAMN pin. The enhancement is added to this command such that arguments in OPERAND1 could be utilized to redirect the booting mode within bitstream file execution. The operand options are shown in Table 6.19.

Table 6.19. OPERAND1 Definition for LSC_REFRESH Command

Bit[7]	Bit[6]	Bit[5]	Bit[4]	Bit[3]	Bit[2]	Bit[1]	Bit[0]
RESERVED	Override ENABLE	SQ	MSPI SEL [1:0]		RSVD		

The enhanced REFRESH command is backward compatible with previous product families if the Override ENABLE bit is not set. Once Override ENABLE is set, the REFRESH command can softly re-direct the booting to the operand specified. The bit definition of the operand is shown as following:

- Bit[2:0]: Reserved
- Bit[4:3]: 2'b00 --- SLOW serial read for external SPI FLASH
- 2'b01 --- FAST serial read for external SPI FLASH
- 2'b10 --- DUAL read for external SPI FLASH
- 2'b11 --- QUAD read for external SPI FLASH
- Bit[5]: Stacking QUAD read SPI FLASH support, 8-bit throughput support only
- Bit[6]: Override the default pin setting if set, otherwise ignore Bit[5:0]
- Bit[7]: Reserved

6.9.3.9. ISC_ENABLE

The mandatory instruction puts the device in the off-line programming mode. The device enters Access Modal State with wakeup signals on hold. The user logic ceases to function and the I/O are tri-stated by default. The OPERAND1 of the ISC_ENABLE command can direct the Access Modal State pointing to different internal memory target, as shown in Table 6.20.

Table 6.20. OPERAND1 Definition for ISC_ENABLE Command

Memory Target	OPERAND1							
	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
SRAM	—	X	X	X	X	0	0	0
EFUSE NORM	X	X	X	X	0	0	1	X
EFUSE PSEUDO	X	X	X	X	0	1	0	X
EFUSE SAFE	X	X	X	X	0	1	1	X
RESERVED	X	X	X	X	1	0	0	X
RESERVED	X	X	X	X	1	0	1	X
RESERVED	X	X	X	X	1	1	0	X
RESERVED	X	X	X	X	1	1	1	X

When accessing the NV EFUSE Memory, all read command executions capture the shadow register data for readout if the targeted NV EFUSE Memory content is associated with a shadow register. There are three sub-access modes: Normal, Pseudo, and Safe mode. These are provided for different non-volatile memory programming activities.

Normal Mode – In Normal NV Memory access mode, the program command execution directly writes the shifted-in data into the non-volatile memory, whether or not the NV Memory content is associated with a shadow register.

Pseudo Mode – In Pseudo NV Memory access mode, the program command execution writes the shifted-in data into the shadow register, instead of the NV Memory, if the targeted NV Memory content is associated with a shadow register.

Safe Mode – In Safe NV Memory access mode, the program command execution writes the shadow register content (instead of the shifted-in data) into the NV Memory, if the targeted NV Memory content has shadow register associated with it.

6.9.3.10. ISC_ENABLE_X

Similar to ISC_ENABLE. However, this command puts the device into the transparent mode. I/O remain to be governed by user logic if the device is already configured. The device remains in User mode and this allows for transparent full or partial reconfigurations.

6.9.3.11. ISC_DISABLE

Exit Access State (if device is in Access State) and enter Complete State. The wake-up sequence starts if the DONE bit is programmed. Exit Transparent Read State (if device is in Transparent Read State) and enter Operational State or Unprogrammed State depending on the DONE bit.

6.9.3.12. LSC_BITSTREAM_BURST

Program the device with the Serial mode bitstream sent through the configuration port. After the command byte and the three operands, you can start sending data bytes from the bitstream. The number of the data bytes can be as large as the entire bitstream.

There is a guideline for you to follow while you are doing bitstream burst transfer via Slave SPI configuration ports. If your host is not busy for other tasks during the bitstream bursting and the buffer size is allowed, you are recommended to use continuous bitstream burst. In this mode, the host asserts SCSN low, then sends the bitstream data continuously after the LSC_BITSTREAM_BURST command. The host de-asserts SCSN high at the end of the bitstream burst.

Alternatively, you can perform a segmented bitstream burst. Refer to the following steps to perform segmented bitstream burst. Failure to follow the required steps might cause the Bitstream Engine (BSE) to terminate resulting in an Abort Error.

1. Host asserts SCSN low.
2. Clock in the LSC_BITSTREAM_BURST command and operands.
3. Clock in the segmented bitstream. The segmented bitstream can be in any size. For example, you can split the full bitstream into 4096 bytes per data block and send them into the FPGA block by block. In between the loading of bitstream segments, SCLK must remain idle or stop toggling when a valid segmented bitstream is unavailable or the host buffer is in the process of loading the next valid segmented bitstream on the SSI interface (refer to Figure 6.44).
4. Host de-asserts SCSN high at the end of the last segmented bitstream.

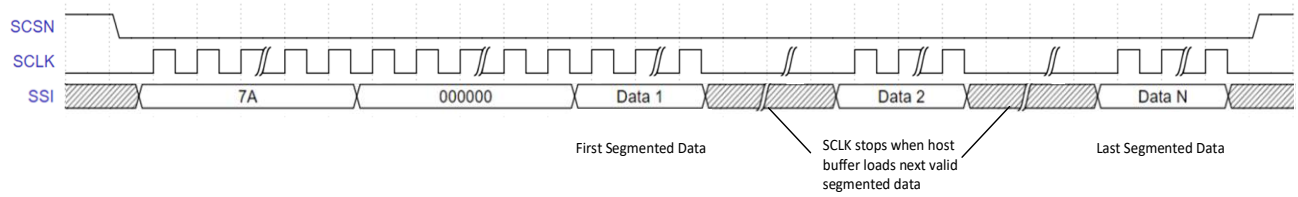


Figure 6.44. Segmented Bitstream Burst Timing Waveforms

There is a guideline for you to follow while you are doing bitstream burst transfer via I2C/I3C configuration ports. If your host is not busy for other tasks during the bitstream bursting and the buffer size is allowed, you are recommended to use continuous bitstream burst. In this mode, all the bitstream data are sent continuously after the LSC_BITSTREAM_BURST command, the host generates the STOP condition at the end of bitstream burst. Figure 6.45 illustrates the continuous bitstream burst operation.

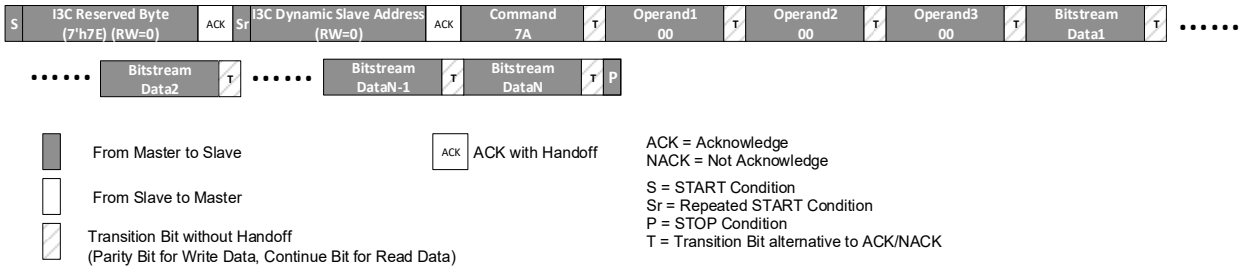


Figure 6.45. Continuous Bitstream Burst

If necessary, you can also perform a segmented bitstream burst (Figure 6.46). There should be No Stop condition between the segments, which makes the Bitstream Engine (BSE) terminated and causes the Abort Error. Neither should there be any extra clock edge on SCL, which can cause the bit alignment issues and generate illegal Command Error or CRC Error. Your host should hold onto the I3C bus between the segments to avoid accidental Stop condition and extra SCL edges.

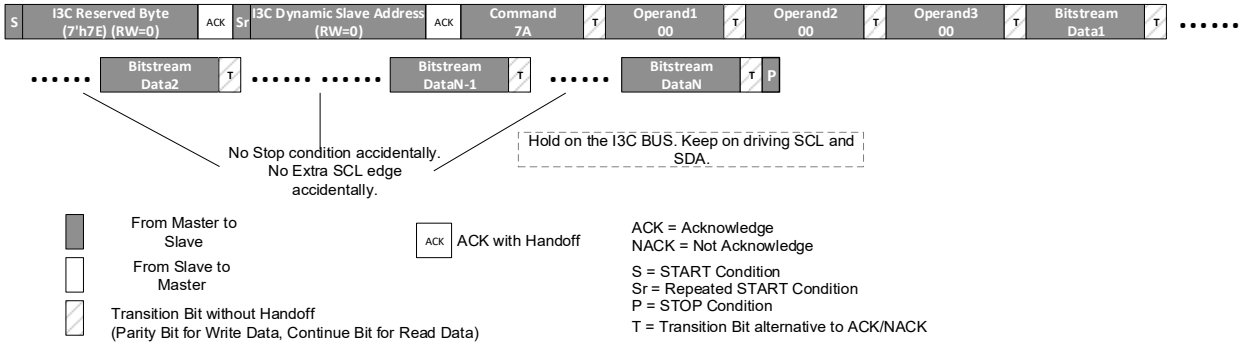


Figure 6.46. Segmented Bitstream Burst with No Extra Clock Edge on SCL

If your host must release the I^{2}C/I3C bus between the segments, there must be No Stop condition, which makes the BSE terminated and cause Abort Error. This means the host must release the SDL while SCL is LOW. If the extra SCL edge is unavoidable, the next segment should be started with normal I3C private write procedure, as shown in Figure 6.47.

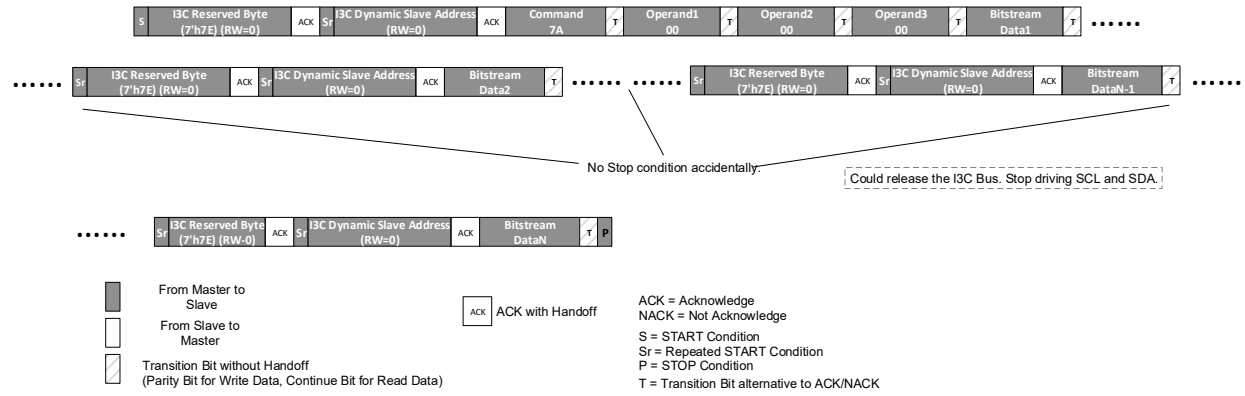


Figure 6.47. Segmented Bitstream Burst when Extra SCL Edge Unavailable

6.9.3.13. ISC_PROGRAM_USRCODE

Write the 32-bit USERCODE data to UES register. Since the USERCODE size is constant, the device must always expect a 4-byte data following the command. Therefore, the operands are ignored and discarded by the device.

6.9.3.14. ISC_ERASE

Bulk erase the configuration memory array including the DONE bit, security configuration bit, user code, configuration data, and persistence configuration bit. The command also resets the Address Shift Register to point to the first configuration frame.

6.9.3.15. ISC_PROGRAM_DONE

Program the DONE bit.

6.9.3.16. ISC_ERASE_DONE

Clear the DONE bit.

6.9.3.17. ISC_PROGRAM_SECURITY

Program the Security bit to stop the targeted FPGA SRAM memory from Read and Write. Once the Security bit is set, it remains set until the FPGA SRAM Memory Array is bulk erased by ISC_ERASE command or REFRESH event.

6.9.3.18. LSC_INIT_ADDRESS

Reset the content in the address counter to point to the first configuration data frame.

6.9.3.19. LSC_WRITE_ADDRESS

This is a 32-bit address for addressing FPGA SRAM array quickly. The MSB16-bits are reserved and the LSB 16-bits are used to set the Address Counter.

The Address counter is used to send addresses to Address Tree Decoder. The field definitions of the Address Counter are shown in Table 6.21.

Table 6.21. Address Counter Field Definitions

Address Counter Bits	Bit[15]	Bit[14:12]	Bit[11:7]	Bit[6:0]
Definition	MISC Frames Enable	Sector Count	PLC Count	Address Bit Count
Counting Range	1	*N_SECTORS-1*[1]	*N_PLCS_SEC[X]-1*[2]	*ADDR_PLC*[3]

Notes:

1. N_SECTORS is the number of sectors for the device minus 1. It is set by 3 bits tie high/tie low at full chip level for each device in the family.
2. N_PLCS_SEC[X] are the number of PLC for each sector, where the X is index for sectors, which is counting from 0 – 8, The eight 5-bits variable are set by tie high/tie low at full chip level for each device in the family. The effective counting range is automatically chosen based on the current Sector Count Field.
3. ADDR_PLC number of bits that are addressable in a PLC.

If Bit[15] – MISC_Frame_Enable_Bit, is reset (0), it is used for addressing the SRAM array. Else, if Bit[15] is set (1) and the Sector Count field and PLC Count field are reset *0*, then the Address Bit Count is used for addressing the MIB on the right, left, and the clock spine frames from right to left of the FPGA array.

6.9.3.20. LSC_PROG_INCR

The LSC_PROG_INCR command auto-increments the address after writing the configuration data into the targeted configuration memory frame(s). All 3 bytes command operands are utilized for the command execution. The operands definition is shown in Table below.

Table 6.22. OPERAND Definition for LSC_PROG_INCR Command

Bit 23	Bit 22	Bit 21	Bit 20	Bit [19:16]	Bit [15:0]
CRC Check Enable	CRC Check Last Frame	Exclude Dummy	Default Dummy Override	Number of Dummy Bytes per Frame	Number of Configuration Data Frames

- Bit[23]: Set to 1 to Enable CRC Check. CRC checksum should be provided.
- Bit[22]: Set to 1 to Only Check CRC at the end of the last frame.
- Bit[21]: Set to 1 to exclude the dummy bytes from input, the number of dummy cycles are still executed.
- Bit[20]: Set to 1 to override the default number of dummy bytes (4 bytes) per frame by using the value specified in Bit[19:16].
- Bit[19:16]: Indicate the number of configuration data frames included for this command.

The total data size in bytes covered by this command is the multiple of the configuration memory frame size plus all the padding bytes and optional 16 bit CRC Checksum fields.

6.9.3.21. LSC_PROG_INCR_CMP

Decompress the configuration data, write the data to the configuration memory at current address and post increment the address. Each uncompressed data frame must be first filled all 0s at the beginning to make it 64-bit bounded, then after compressed, the compressed data frame must be padded with all 0's at the end to make it byte bounded. The CRC is based on the compressed and padded frame, and the padding bytes of all 1's are not compressed after each frame.

6.9.3.22. LSC_READ_INCR

Read back the configuration memory data frames selected by the address register and post increment the address. Bit 7 of the OP1 is used to determine if the command is to be included into the CRC calculation. Bit 22-0 of the command operand indicates the number of frames of configuration data to be read out of the device.

Readback Size = (Frame Byte Count + 4-byte Padding) × Number of Frames + Device Specific Padding

Device Specific Padding = Bit[11:10] of Control Register 0

Table 6.23. OPERAND Definition for LSC_READ_INCR Command

To Device	LSC_READ_INCR Command	
From Device	Device Specific Padding	
	32-Bit Padding	Read back data frame
	32-Bit Padding	Read back data frame
	…	…
	32-Bit Padding	Read back data frame

On the read cycle, the device presents the four (4) bytes of padding data at the beginning of each Readback data frame. If the device requires additional cycles to prepare the read back data, the Device Specific Padding field shall be added. The size of the Device Specific Padding field shall be consistent across all LSC_READ_INCR and LSC_PROG_INC frames of the device. The padding bytes are all-1 bytes and their size is included in the operand information field. An additional 2-bytes of pipeline delay is also added at the beginning of each read.

6.9.3.23. LSC_PROG_CNTRL0

When OPERAND1 bit 0 is 0 the command behaves in legacy mode. The command format is:

Command opcode (8bit) + Command Operands (3 bytes) + data (4 bytes)

When OPERAND1 bit 0 is set this command takes 64-bits data, where the first 32-bits are data to program to control register 0, and the remaining 32-bits are mask bits. Bits with mask set to 1 retains their current values. The command format is:

Command opcode (8bit) + command Operands (3 bytes) + data (4 bytes) + mask (4 bytes)

Mask example:

Current Register Value = 0x000000AB

To change bit 0 to 0.

Data=0x00000000

Mask=0xFFFFFFFE

New Register Value = 0x000000AA

6.9.3.24. LSC_READ_CNTRL0

Read the Control Register Data from Control Register 0. In case of over read occurred, the device sends all '1' after the Control Register 0 content. This instruction does not check for CRC error.

6.9.3.25. LSC_PROG_CNTRL1

When OPERAND1 bit 0 is 0 the command behaves in legacy mode. The command format is:

Command opcode (8bit) + command info (3 bytes) + data (4 bytes)

When OPERAND1 bit 0 is set this command takes 64-bits data, where the first 32-bits are data to program to control register 1, and the remaining 32-bits are mask bits. Bits with mask set to 1 retains their current values. The command format is:

Command opcode (8bit) + command info (3 bytes) + data (4 bytes) + mask (4 bytes)

6.9.3.26. LSC_READ_CNTRL1

Read the Control Register Data from Control Register 1. In case of over read occurred, the device sends all '1' after the Control Register 1 content. This instruction does not check for CRC error.

6.9.3.27. LSC_RESET_CRC

Reset the CRC-16 engine to zero.

6.9.3.28. LSC_READ_CRC

Read the 16-bit CRC value back from the CRC engine.

6.9.3.29. LSC_PROG_SED_CRC

Store the calculated 32-bit CRC checksum based on the user configuration pattern into internal registers. The distributed RAM configuration bits are treated as '0' when the CRC is calculated, because these bits can be changed by the user so it may not have the same value for the SED event.

6.9.3.30. LSC_READ_SED_CRC

Verify the 32-bit SED CRC checksum.

6.9.3.31. LSC_PROG_PASSWORD

Program 128-bit password into the non-volatile memory.

6.9.3.32. LSC_READ_PASSWORD

Read out the 128-bit password from corresponding shadow register for verification immediately after programming.

6.9.3.33. LSC_SHIFT_PASSWORD

Shift in the password to unlock for reconfiguration if the part is locked by the password.

6.9.3.34. LSC_PROG_CIPHER_KEY

Program the 256-bit cipher key into the non-volatile memory.

6.9.3.35. LSC_READ_CIPHER_KEY

Read out the 256-bit cipher key from corresponding shadow register for verification immediately after programming.

6.9.3.36. LSC_PROG_FEATURE

Program user mode related feature such as Custom ID, I²C address, etc. into the non-volatile memory. There are 96 bits data required for this command, and data field definition is shown in Table 6.24.

Table 6.24. Data Fields Definition for LSC_PROG_FEATURE Command

LSC_PROG_FEATURE DATA				
Bit	**Definition**	**HW Default**	**SW Default**	**Description**
[95:81]	RESERVED	0	N/A	RESERVED
[80]	I3C Provisional ID Selection	0	N/A	Bit [32] of I3C Provisional ID. 1 – Use pseudo random number for Provisional ID. 0 – User Vender fixed value.
[79:72]	I3C Slave Address [7:0]	8'b00000000[1]	N/A	Default I3C Slave Address
[71:68]	I3C Instance ID [3:0]	4'b0000	N/A	I3C Instance ID Code
[67:64]	I3C Device Character [3:0]	4'b0000	N/A	MSB of 12-bit I3C Device Character Code
[63]	RESERVED	0	N/A	RESERVED
[62]	SDA_IN_DLY_DIS SDA_OUT_DLY_EN	0	N/A	SDA Input Delay Disable SDA Output Delay Enable
[61:56]	SDA_DEL_CAL [5:0]	0	N/A	SDA Delay adjustment.
[55:54]	MSPI_tI [1:0]	0	N/A	Extra delay for MCSN idle time in number of MCLK cycle from default 4 cycles.
[53:51]	MSPI_tT [2:0]	0	N/A	Extra delay from MCLK to MSCN high in number of MCLK cycle from default 1 cycle.
[50:48]	MSPI_tL [2:0]	0	N/A	Extra delay from MSCN low to MCLK in number of MCLK cycle from default 1 cycle.
[47:40]	I²C Slave Address [7:0]	8'b00000000[2]	N/A	I²C Slave Address
[39:32]	Trace ID MSB [7:0]	0	N/A	User Programmable MSB of the Trace ID
[31:0]	Customer ID [31:0]	0	N/A	Customer ID Code

Notes:

1. If the I3C Slave Address field is 0, hardware uses 8'b11110000 as default value to assemble the default I3C slave address as described in the Slave Address for I2C and I3C Ports section.
2. If the I²C Slave Address field is 0, hardware uses 8'b01110000 as default value to assemble the default I3C slave address as described in the Slave Address for I2C and I3C Ports section.

6.9.3.37. LSC_READ_FEATURE:

Read FEATURE setting from the corresponding shadow register, with the format as shown in Table 6.24.

6.9.3.38. LSC_PROG_FEABITS

Program security related feature bits such as Decrypt Enable, authentication control, etc. into non-volatile memory. *There are 16 bits data required for this command, and data field definition is shown in* Table 6.25*.*

Table 6.25. Data Fields Definition for LSC_PROG_FEABIT Command

LSC_PROG_FEABIT DATA				
Bit	**Definition**	**HW Default**	**SW Default**	**Description**
[15]	DONE_EN	0	N/A	Enable User IO on DONE pin.
[14]	INITN_EN	0	N/A	Enable User IO on INITN pin.
[13]	PROGN_EN	0	N/A	Enable User IO on PROGRAMN pin.
[12]	SLV_ONLY	0	N/A	Slave port only (only valid when bit 13 set).
[11]	I3C_ONLY	0	N/A	1 - I3C only (I^2C Disabled).
[10]	PWD_ALL	0	N/A	Extend password protection for FPGA SRAM array.
[9]	PWD_EN	0	N/A	Enable Password protection for EFUSE programming.
[8:6]	Boot Select	000	101	Boot_Select [2:0] 000 – N/A 001 – N/A 010 – N/A 011 – Single Boot 100 – N/A 101 – Dual Boot 110 – N/A 111 – No Boot
[5]	CID_EN	0	N/A	Customer ID Enable. If set, use Customer ID from FEATURE setting as device ID. Otherwise use Hardware specified device ID.
[4:3]	Decryption Enable	0	N/A	00 – OFF (Plain or Encrypted bitstream, controlled by preamble) 01 – Undefined 10 – ON (Encrypted bitstream only) 11 – OFF (Plain or Encrypted bitstream, controlled by preamble)
[2]	RESERVED	0	N/A	RESERVED
[1:0]	Authentication Control	00	N/A	00 – No Auth 01 – ECDSA 10 – HMAC 11 – No Auth

6.9.3.39. LSC_READ_FEABITS

Read the FEABIT setting from the corresponding shadow register, with the format as shown in Table 6.25.

6.9.3.40. LSC_PROG_OTP

Program security related One-Time-Programmable bits into the non-volatile memory. *There are 16 bits data required for this command, and data field definition is shown in* Table 6.26.

Table 6.26. Data Fields Definition for LSC_PROG_OTP Command

LSC_PROG_OTP DATA				
Bit	**Definition**	**HW Default**	**SW Default**	**Description**
[31:28]	RESERVED	0	N/A	RESERVED; Need to be set to 4'b0000.
[27]	ECDSA SEC_HLOCK	0	N/A	Hard Lock[2] EFUSE memory for ECDSA Authentication Key.
[26]	ECDSA SEC_READ	0	N/A	Read Lock[1] EFUSE memory for ECDSA Authentication Key.

LSC_PROG_OTP DATA				
Bit	**Definition**	**HW Default**	**SW Default**	**Description**
[25]	ECDSA SEC_PROG	0	N/A	Write Lock[1] EFUSE memory for ECDSA Authentication Key.
[24]	RESERVED	0	N/A	RESERVED.
[23]	AES SEC_HLOCK	0	N/A	Hard Lock[2] EFUSE memory for AES Decryption Key.
[22]	AES SEC_READ	0	N/A	Read Lock[1] EFUSE memory for AES Decryption Key.
[21]	AES SEC_PROG	0	N/A	Write Lock[1] EFUSE memory for AES Decryption Key.
[20]	RESERVED	0	N/A	RESERVED.
[19]	Feature SEC_HLOCK	0	N/A	Hard Lock[2] EFUSE memory for FEATURE, FEABIT and OTP.
[18]	Feature SEC_READ	0	N/A	READ Lock[1] EFUSE memory for FEATURE, FEABIT and OTP.
[17]	Feature SEC_PROG	0	N/A	Write Lock[1] EFUSE memory for FEATURE, FEABIT and OTP.
[16]	RESERVED	0	N/A	RESERVED.
[15]	RESERVED	0	N/A	RESERVED.
[14]	JTAG Port Test Only[5, 6]	0	N/A	Lock JTAG Configuration port for ISC operation. Only IEEE 1149 Test Instructions and ispTracy (TN1054) instructions are allowed.
[13]	JTAG Port Hard Lock[6]	0	N/A	Hard Lock[4] JTAG Configuration Port.
[12]	JTAG Port Soft Lock[6]	0	N/A	Soft Lock[3] JTAG Configuration Port.
[11]	I[2]C/I3C Port Hard Lock[6]	0	N/A	Hard Lock[4] Slave I[2]C/I3C Configuration Port.
[10]	I[2]C/I3C Port Soft Lock[6]	0	N/A	Soft Lock[3] Slave I[2]C/I3C Configuration Port.
[9]	SSPI Port Hard Lock[6]	0	N/A	Hard Lock[4] Slave SPI Configuration Port.
[8]	SSPI Port Soft Lock[6]	0	N/A	Soft Lock[3] Slave SPI Configuration Port.
[7]	SRAM SEC_HLOCK	0	N/A	Hard Lock2 FPGA SRAM Memory Array.
[6]	SRAM SEC_READ	0	N/A	Read Lock[1] FPGA SRAM Memory Array.
[5]	SRAM SEC_PROG	0	N/A	Write Lock[1] FPGA SRAM Memory Array.
[4]	SRAM SEC_ERASE	0	N/A	Erase Lock[1] FPGA SRAM Memory Array.
[3]	Password SEC_HLOCK	0	N/A	Hard Lock[2] EFUSE memory for Password Key.
[2]	Password SEC_READ	0	N/A	Read Lock[1] EFUSE memory for Password Key.
[1]	Password SEC_PROG	0	N/A	Write Lock[1] EFUSE memory for Password Key.
[0]	RESERVED	0	N/A	RESERVED.

Notes:

1. Memory Soft Lock for Read, Write or Erase correspondingly, which stops configuration access from external configuration ports but could not stop LMMI access from FPGA fabric if the corresponding Hard Lock is not set.
2. Memory Hard Lock, which stops configuration access from any interface, include LMMI access.
3. Port Soft Lock, which blocks the configuration port for configuration access but could be temporarily unlocked by user logic inside the FPGA fabric through LMMI interface by using the LSC_PROG_OTP command in EFUSE PSEUDO access mode.
4. Port Hard Lock. Once set, it blocks the configuration port for configuration access, and cannot be unlock by user logic through LMMI interface.
5. When bit 14 (JTAG Port Test Only) is set, only IEEE 1149.1 test instructions (refer to Table 6.31), commands related to the internal logic analyzer or debugger, and the sysCONFIG commands READ_ID, READ_UNIQUEID, and USERCODE can be used through the JTAG port. The sysCONFIG commands mentioned will return the expected data. All other sysCONFIG commands such as LSC_READ_STATUS and LSC_READ_OTP are blocked and will return 0s. The fast configuration operation via the JTAG port will not work when JTAG Port Test Only is set. Slave configuration ports such as SSPI/I2C/I3C are unaffected.
6. Bits 8 through 14 cannot be programmed through the CONFIG_LMMI interface for the LFD2NX-9/28, LFD2NX-17/40, and LIFCL-17/40 devices. Programming through sysCONFIG ports such as JTAG, SSPI, and I2C/I3C is not impacted.

6.9.3.41. LSC_READ_OTP

Read security related One-Time-Programmable bits from the corresponding shadow register, with same data field definitions as LSC_PROG_OTP command.

6.9.3.42. LSC_WRITE_COMP_DIC

Load the data compression dictionary into the internal registers for decompressing compressed data frame.

6.9.3.43. LSC_INIT_BUS_ADDR

Set the address for EBR or the Hard IP Config INIT Bus, to move the address quickly. This command is used to set the starting address when performing a read/write over the Config INIT Bus. The definitions of the operand and data following the command are listed below. The data packet following the command must be excluded from compression.

Table 6.27. Data Packet for LSC_INIT_BUS_ADDR

Commands	Operand/Data	Description
LSC_INIT_BUS_ADDR	Operand Bit [23:0]	Unused (default =0)
	Data[31:30]	Reserved
	Bus Width	0: 8-bit Hard IP 1: 10-bit Hard IP 2: 10-bit EBR /Large RAM 3: 32-bit Hard IP
	Data [27:17]	ID Code
	Data[16:0]	Starting address

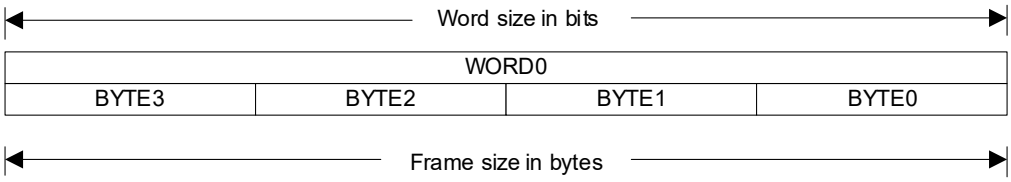


Figure 6.48. 32-bit Hard IP Example; Frame size =4, Word size = 32

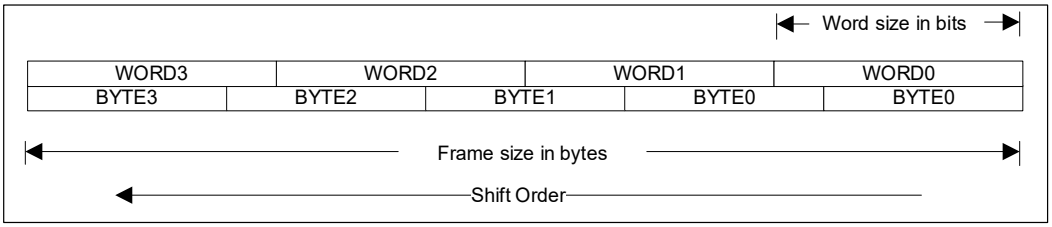


Figure 6.49. 10-bit EBR Example; Frame size = 5, Word size = 10

6.9.3.44. LSC_INIT_BUS_WRITE

The Config INIT Write command is applicable to the Config INIT Bus. For this instruction the dummy byte counts is ignored. The definition of the operand is shown below.

Table 6.28. Operand Definitions for LSC_INIT_BUS_WRITE

Commands	Operand (24 bits)	Description
LSC_INIT_BUS_WRITE	Bit [23:16]	Bit[23] = CRC Comparison Flag only effective inside Bitstream file Bit[22] = Check CRC at the last frame Bit[21] = Dummy Bytes excluded from Bit Stream Bit[20] = 0 Bit[19:16] = 0
	Bit [15:0]	Number of frames

6.9.3.45. LSC_INIT_BUS_READ

The INIT Bus Read command is applicable to the Config INIT Bus. The definition of the operand is shown below.

Table 6.29. Operand Definitions for LSC_INIT_BUS_READ

Commands	Operand (24 bits)	Description
LSC_INIT_BUS_WRITE	Bit [23:16]	Bit[23:22] = Reserved Bit[21] = Dummy Bytes excluded from Bit Stream Bit[20] = Dummy Byte Control override Enable Bit[19:16] = Dummy Byte count
	Bit [15:0]	Number of frames

6.9.3.46. LSC_PROG_SPI (SSPI to MSPI Bridge)

The LSC_PROG_SPI command unconditionally connects SCSN, SO, SI, SCLK of slave SPI respectively to the MCSN, MOSI, MISO, MCLK pin of the external SPI flash device.

The OPERAND1 definition is shown in Table 6.30.

Table 6.30. LSC_PROG_SPI Operand

SPI Command Type	Command (3A)	Operand 1	Operand 2	Operand 3
x1 read, x1 write	0011 1010	0xxx xxxx	xxxx xxxx	xxxx xxxx

6.9.3.47. LSC_PROG_ECDSA_PUBKEY

This command is to program the ECDSA 64-byte public key.

6.9.3.48. LSC_READ_ECDSA_PUBKEY

This command is to read the ECDSA 64-byte public key.

6.9.3.49. LSC_PROG_SEC_BOOT

This command is to set the 32-bit secondary boot offset to override the default boot offset, which is 0xFFFF00 for 24-bit flash mode and 0xFFFFFF00 for 32-bit flash mode.

6.9.3.50. LSC_READ_DR_UES

This command reads the dry run UES shadow register. When dry run booting is executed by writing to bit-2 of LSC_DEVICE_CTRL, the UES in the bitstream is redirected to the dry run UES shadow register. To read the UES of the currently loaded bitstream use the USERCODE command. Default reads 0x00000000 if dry run was not executed.

6.10. JTAG Instruction Support

The ISC and programming instructions are executed on the first rising edge of the TCK after entering Run-Test/Idle State. When an instruction is nullified, the instruction is not executed and no registers are updated or captured by the instruction. However, the associated register is still connected to TDI and TDO on Shift-DR state. For more details, refer to IEEE 1149 and IEEE 1532 standards.

Table 6.31. JTAG Instruction Table

JTAG Instruction	OPCODE		Source Data at Capture-DR	Target Data at Update-DR	Target Data at Runtest Idle	Shift Register	Note
	Binary	Hex					
EXTEST	00010101	15	Refer to IEEE1149.1 Specification				—
INTEST	00101100	2C	Refer to IEEE1149.1 Specification				—
CLAMP	01111000	78	Refer to IEEE1149.1 Specification				—
HIGHZ	00011000	18	Refer to IEEE1149.1 Specification				—
PRELOAD	00011100	1C	Refer to IEEE1149.1 Specification				—
SAMPLE	00011100	1C	Refer to IEEE1149.1 Specification				—
EXTEST_PULSE	00101101	2D	Refer to IEEE1149.1 Specification				—
EXTEST_TRAIN	00101110	2E	Refer to IEEE1149.1 Specification				—
ISC_NOOP	00110000	30	N/A	N/A	N/A	N/A	No Operation.
IDCODE	11100000	E0	IDCODE	N/A	N/A	32-bit shift register	Read out the 32-bit hardware IDCODE of the device. If the USRID feature bit is set, this IDCODE becomes the user-defined IDCODE.
UIDCODE	00011001	19	IDCODE	N/A	N/A	64-bit shift register	Read the 64-bit UNIQUE ID, 56 bits set by Lattice, 8 bits set by customer.
USERCODE	11000000	C0	USERCODE Register	N/A	N/A	32-bit shift register	Read the 32-bit USER Electronic Signature.
LSC_READ_STATUS	00111100	3C	Status register	N/A	N/A	64-bit shift register	Read out the internal status such as busy, fail, and others. (64-bit).
LSC_CHECK_BUSY	11110000	F0	BUSY FLAG	N/A	N/A	1-bit Shift Register	Read 1 bit busy flag to check the command execution status.

JTAG Instruction	OPCODE		Source Data at Capture-DR	Target Data at Update-DR	Target Data at Runtest Idle	Shift Register	Note
	Binary	Hex					
LSC_DEVICE_CTRL	01111101	7D	N/A	N/A	N/A	8-bit shift register	An 8-bit field in Operand 1 allows various device control, such as GSR, Sleep, Dry Run and so on. Bit 0: Cause a Global Set/Reset to occur on the device. Bit 1: Launch a One-time check of SED. Bit 2: Reserved Bit 3: Cause a reset on configuration logic related FSM and flags. Bit 4: Reserved for internal dry-run. Bit [6:5]: Launch Dry-Run event. 2'b01 – Dry-run for primary bitstream 2'b10 – Dry-run for secondary bitstream 2'b11 – Slave dry-run 2'b00 – No dry-run Bit 7: CRC check bit
LSC_REFRESH	01111001	79	N/A	N/A	N/A	Bypass	Equivalent to toggle the PROGRAMN pin.
ISC_ENABLE	11000110	C6	CONFIG_INFO	ISC_CONFIG	N/A	8-bit shift register	Enable the device for configuration. The device enters Access State. The I/O are tri-stated with a weak pull down. Array selection and modal state setup effective after stay at RTI for more than 2

JTAG Instruction	OPCODE		Source Data at Capture-DR	Target Data at Update-DR	Target Data at Runtest Idle	Shift Register	Note
	Binary	Hex					
							clock cycles.
ISC_ENABLE_X	01110100	74	CONFIG_INFO	ISC_CONFIG	N/A	8-bit shift register	Allow the transparent read in JTAG. The device enters Transparent mode and the I/O remain to be governed by user logic if the device is already being configured. Array selection and modal state setup effective after stay at RTI for more than two clock cycles.
ISC_DISABLE	00100110	26	N/A	N/A	N/A	Bypass	If the device is in Access State, exit Access State and enter the Complete State. The wake-up sequence starts if the DONE bit is programmed. If the device is in Transparent Read State, exit the Transparent Read State and enter the Operational State or Unprogrammed State dependent on the DONE bit.
LSC_BITSTREAM_BURST	01111010	7A	N/A	N/A	N/A	Bypass	Program the device with the bitstream sent in through the JTAG port.
ISC_PROGRAM_USERCODE	11000010	C2	N/A	N/A	User Code Register	32-bit Shift Register	Write the 32-bit user code into the device.
ISC_ERASE	00001110	0E	N/A	N/A	ISC_SECTIR Register	16-bit Shift Register	Bulk erase the SRAM memory array based on the access mode and array selection.

JTAG Instruction	OPCODE		Source Data at Capture-DR	Target Data at Update-DR	Target Data at Runtest Idle	Shift Register	Note
	Binary	Hex					
ISC_PROGRAM_DONE	01011110	5E	N/A	N/A	N/A	Bypass	Program the DONE bit if the device is in Configuration state.
ISC_PROGRAM_SECURITY	11001110	CE	N/A	N/A	N/A	Bypass	Program the Security Bit if the device is in Configuration state
LSC_INIT_ADDRESS	01000110	46	N/A	N/A	N/A	Bypass	Initialize the Address Shift Register.
LSC_WRITE_ADDRESS	10110100	B4	N/A	32-bit address register	N/A	32-bit shift Register	Write the 16-bit Address Register to move the address quickly. Note this is 32-bit but for CRAM address only the lower 16 bits are used.
LSC_PROG_INCR	10000010	82	N/A	N/A	Configuration Data Frame	DSR	Write the configuration data to the configuration memory frame at current address and post increment the address. Byte 2 to byte 0 of the opcode indicates number of frames included in the operand field.
LSC_PROG_INCR_CMP[2]	10111000	B8	N/A	N/A	Configuration Data Frame	128-bit shift Register	Decompress the configuration data; write the data to the configuration memory at current address and post increment the address. Byte 2 to byte 0 of the opcode indicates number of frames included in the operand field.

JTAG Instruction	OPCODE		Source Data at Capture-DR	Target Data at Update-DR	Target Data at Runtest Idle	Shift Register	Note
	Binary	Hex					
LSC_READ_INCR	01101010	6A	N/A	N/A	N/A	N/A	Read back the configuration memory data frames selected by the address register and post increment of the address.
LSC_PROG_CNTRL0	00100010	22	N/A	Control Register0	N/A	32-bit shift Register	Modify the Control Register 0.
LSC_READ_CNTRL0	00100000	20	Control register0	N/A	N/A	32-bit shift Register	Read the Control Register 0.
LSC_PROG_CNTRL1	00100011	23	N/A	Control Register1	N/A	32-bit shift Register	Modify the Control Register 1.
LSC_READ_CNTRL1	00100001	21	Control register0	N/A	N/A	32-bit shift Register	Read the Control Register 1.
LSC_RESET_CRC	00111011	3B	N/A	N/A	N/A	Bypass	Reset 16-bit CRC register to 0x0000.
LSC_READ_CRC	01100000	60	CRC Checksum	N/A	N/A	16-bit Shift Register	Read 16-bit CRC register content.
LSC_PROG_SED_CRC	10100010	A2	N/A	SED CRC	N/A	32-bit shift Register	Store the calculated 32-bit CRC based on the configuration bit values only.
LSC_READ_SED_CRC	10100100	A4	SED CRC	N/A	N/A	32-bit shift Register	Verify the 32-bit SED CRC
LSC_PROG_PASSWORD	11110001	F1	N/A	Password	N/A	128-bit shift Register	Program 128-bit password into the non-volatile memory (Internal EFUSE Memory). See additional note[1].
LSC_READ_PASSWORD	11110010	F2	Password	N/A	N/A	128-bit shift Register	Read out the 128-bit password before activated for verification.
LSC_SHIFT_PASSWORD	10111100	BC	N/A	Password Match Flag	N/A	128-bit shift Register	Shift in the password to unlock for re-configuration if the part is locked by the password.

JTAG Instruction	OPCODE		Source Data at Capture-DR	Target Data at Update-DR	Target Data at Runtest Idle	Shift Register	Note
	Binary	Hex					
LSC_PROG_CIPHER_KEY	11110011	F3	N/A	N/A	N/A	256-bit shift register	Program the 256-bit AES key into the non-volatile memory if 256-bit key is selected. See additional note[1].
LSC_READ_CIPHER_KEY	11110100	F4	N/A	N/A	N/A	256-bit shift register	Read out the 256-bit AES key before activated for verification if 256-bit key is selected.
LSC_PROG_FEATURE	11100100	E4	N/A	N/A	N/A	96-bit shift register	Program security related feature such as Decrypt Enable, etc. based on the selection from the operand.
LSC_READ_FEATURE	11100111	E7	N/A	N/A	N/A	96-bit shift register	Read security related feature such as Decrypt Enable, and others based on the selection from the operand.
LSC_PROG_FEABITS	11111000	F8	N/A	N/A	N/A	16-bit shift register	Program security related feature bits. See additional note[1].
LSC_READ_FEABITS	11111011	FB	N/A	N/A	N/A	16-bit shift register	Read security related feature bits.
LSC_PROG_OTP	11111001	F9	N/A	N/A	N/A	32-bit shift register	Program security related One-Time-Programmable bits based on the selection from the operand. See additional note[1].
LSC_READ_OTP	11111010	FA	N/A	N/A	N/A	32-bit shift register	Read security related One-Time-Programmable bits based on the selection from the operand.
LSC_WRITE_COMP_DIC[2]	00000010	02	N/A	Compression Dictionary	N/A	128-bit shift register	Loads the most frequently used patterns into the

JTAG Instruction	OPCODE		Source Data at Capture-DR	Target Data at Update-DR	Target Data at Runtest Idle	Shift Register	Note
	Binary	Hex					
							device for decompressing compressed bitstreams.
LSC_INIT_BUS_ADDR	11110110	F6	N/A	Bus Address	N/A	32-bit Shift Register	Write INIT Bus Address and ID Code Registers to move the address quickly.
LSC_INIT_BUS_WRITE	01110010	72	N/A	N/A	INIT Data Frame	Frame Shift Register	Write Data to INIT Bus.
LSC_INIT_BUS_READ	11110111	F7	N/A	N/A	N/A	Frame Shift Register	Read back Data from INIT Bus.
LSC_PROG_SPI	00111010	3A	N/A	N/A	N/A	SPI Flash PROM	Connect the TDI, TDO, Shift-DR-N and TCK signals to BUSY, SPID, DI, and MCLK pin respectively. The length of the register is not defined for this instruction.
LSC_IO_CONTROL	01010100	54	N/A	N/A	N/A	8-bit shift register	Control the release of the I/O Banks during configuration for early I/O release Bit [1]=1; Release right I/O BANK (Bank 1 and Bank 2) Bit [0]=1; Release left I/O BANK (Bank 6 and Bank 7)
LSC_PROG_ECDSA_PUBKEY	01011001	59	N/A	N/A	N/A	512-bit shift register	Program the ECDSA Public Key. See additional note[1].
LSC_READ_ECDSA_PUBKEY	01011010	5A	N/A	N/A	N/A	512-bit shift register	Read Back the ECDSA Public Key.
LSC_READ_DR_UES	01011101	5D	Dry Run UES	N/A	N/A	32-bit shift Register	Read the dry-run User Electronic Signature shadow register.

Notes:

1. JTAG instruction operation programs the internal EFUSE memory (non-volatile memory), which is one-time programmable (OTP).
2. The commands LSC_WRITE_COMP_DIC and LSC_PROG_INCR_CMP are only supported within the compressed bitstream. These commands cannot be executed directly through any sysCONFIG ports (JTAG, SSPI, or I2C/I3C).

7. Software Selectable Options

The operation of the Nexus device configuration logic is managed by options selected in the Lattice Radiant design software. The Nexus device uses the built-in arbitration logic, as described in the Configuration Ports Arbitration section, to select the configuration port. You can set the configuration port persistence after device enters user function mode, and the system Configuration Pins (PROGRAMN pin, INITN pin, and DONE pin) persistence using the Lattice Radiant Device Constraint Editor.

The configuration logic preferences are accessed using Device Constraint Editor. Click on the Global tab, and look for the sysCONFIG tree. The sysCONFIG section is shown in Figure 7.1.

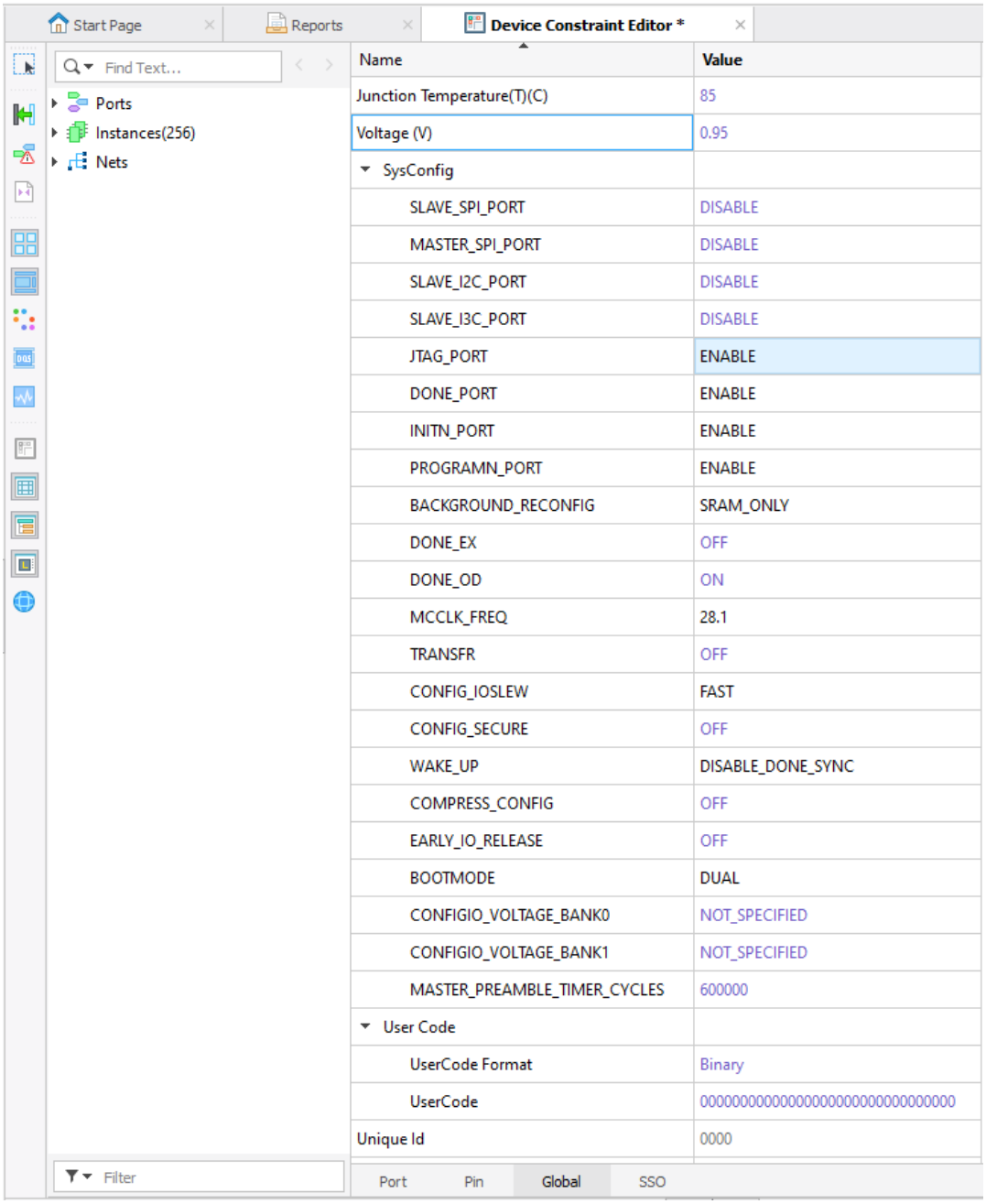


Figure 7.1. sysCONFIG Preferences in Global Tab, Lattice Radiant Device Constraint Editor

Table 7.1. sysCONFIG Options[1]

Option Name	Default Setting	All Settings
SLAVE_SPI_PORT	DISABLE	DISABLE, SERIAL, DUAL, QUAD
MASTER_SPI_PORT	DISABLE	DISABLE, SERIAL, DUAL, QUAD
SLAVE_I2C_PORT	DISABLE	DISABLE, ENABLE
SLAVE_I3C_PORT	DISABLE	DISABLE, ENABLE
JTAG_PORT	ENABLE[2]	DISABLE, ENABLE
DONE_PORT	ENABLE[2]	DISABLE, ENABLE
INITN_PORT	ENABLE[2]	DISABLE, ENABLE
PROGRAMN_PORT	ENABLE[2]	DISABLE, ENABLE[3]
BACKGROUND_RECONFIG	OFF	OFF, ON, SRAM_EBR, SRAM_ONLY
DONE_EX	OFF	OFF, ON
DONE_OD	ON	OFF, ON
MCCLK_FREQ	3.5	3.5, 7.0, 14.1, 28.1, 56.2, 75, 90, 112.5, 150
TRANSFR	OFF	OFF, ON
CONFIGIO_VOLTAGE_BANK0	NOT_SPECIFIED	NOT_SPECIFIED, 2.5, 1.2, 1.5, 1.8, 3.3
CONFIGIO_VOLTAGE_BANK1	NOT_SPECIFIED	NOT_SPECIFIED ,2.5, 1.2, 1.5, 1.8, 3.3
CONFIG_IOSLEW	SLOW	SLOW, MEDIUM, FAST
CONFIG_SECURE	OFF	OFF, ON
WAKE_UP	DISABLE_DONE_SYNC	ENABLE_DONE_SYNC, DISABLE_DONE_SYNC
COMPRESS_CONFIG	OFF	OFF, ON
EARLY_IO_RELEASE	OFF	OFF, ON
BOOTMODE	DUAL	DUAL, SINGLE, NONE
MASTER_PREAMBLE_TIMER_CYCLES	600000	600000, 400, 2000, 4000, 20000, 40000, 200000, 400000
USERCODE Format	Binary	Binary, Hex, ASCII, Auto
USERCODE	0 (32 binary zeros)	32-bit
UNIQUE_ID	0000	Upper 16-bit user-defined code for above Auto USERCODE.

Notes:

1. The CONFIG_MODE option in the Lattice Radiant Global Preference is for the software to preserve appropriated pins to help customer design flow. It does not serve the purpose of enabling SPI port nor setting appropriate bits in the PROM file.
2. The software default for this setting is DISABLED in the Lattice Radiant software version 3.0 and earlier.
3. The CrossLink-NX-33 device only supports the ENABLE setting for the PROGRAMN_PORT option.

7.1. SLAVE_SPI_PORT

The SLAVE_SPI_PORT allows you to preserve the Slave SPI configuration port after the Nexus device enters user mode. There are four states to which the SLAVE_SPI_PORT preference can be set:

- DISABLE – This setting disconnects the SPI port pins from the configuration logic.
- SERIAL – This setting preserves the standard serial SPI port I/O (SCLK, SCSN, SI, SO) when the Nexus device is in user mode. When the pins are preserved, an external SPI master controller can interact with the configuration logic. The preference also prevents you from over-assigning I/O to those pins.
- DUAL – This setting preserves the SPI port I/O (SCLK, SCSN, SI/SD0, SO/SD1) in DUAL mode when the Nexus device is in user mode. When the pins are preserved, an external SPI master controller can interact with the configuration logic. The preference also prevents you from over-assigning I/O to those pins.
- QUAD – This setting preserves the SPI port I/O (SCLK, SCSN, SI/SD0, SO/SD1, SD2, SD3) in QUAD mode when the Nexus device is in user mode. When the pins are preserved, an external SPI master controller can interact with the configuration logic. The preference also prevents you from over-assigning I/O to those pins.

The SLAVE_SPI_PORT could be enabled at the same time as the MASTER_SPI_PORT, because the Slave SPI port and Master SPI port are in different I/O banks, this makes the Nexus device possible to support the Slave SPI to Master SPI bridging functionality. The default setting for this preference is DISABLE.

7.2. MASTER_SPI_PORT

The MASTER_SPI_PORT allows you to preserve the Master SPI configuration port after the Nexus device enters user mode. There are four states to which the MASTER_SPI_PORT preference can be set:

- DISABLE – This setting disconnects the Master SPI port pins from the configuration logic. The Master SPI pins (MCLK, MCSN, MOSI/MD0, MISO/MD1, MD2 and MD3) could be general purpose user I/O.
- SERIAL – This setting preserves the SPI port I/O (MCLK, MCSN, MOSI/MD0, MISO/MD1) in serial mode when the Nexus device is in user mode. The preference also prevents you from over-assigning I/O to those pins.
- DUAL – This setting preserves the SPI port I/O (MCLK, MCSN, MOSI/MD0, MISO/MD1) in dual mode when the Nexus device is in user mode. The preference also prevents you from over-assigning I/O to those pins.
- QUAD – This setting preserves the SPI port I/O (MCLK, MCSN, MOSI/MD0, MISO/MD1, MD2 and MD3) in quad mode when the Nexus device is in user mode. The preference also prevents you from over-assigning I/O to those pins.

The default setting for this preference is DISABLE.

7.3. SLAVE_I2C_PORT/SLAVE_I3C_PORT

The SLAVE_I2C_PORT/SLAVE_I3C_PORT allows you to preserve the I^2C/I3C configuration port after the Nexus device enters user mode. There are two states to which SLAVE_I2C_PORT/SLAVE_I3C_PORT preference can be set:

- ENABLE – This setting preserves the I^2C/I3C port pins (SD2/SCL, SD3/SDA) when the Nexus device is in user mode. It allows you to read all the configuration data, except the EBR and the distributed RAM contents in user mode. The preference also prevents you from over-assigning I/O to those pins.
- DISABLE – This setting disconnects the I^2C/I3C port pins (SD2/SCL, SD3/SDA) from the configuration logic. It allows I^2C/I3C ports pins to be general-purpose I/O, if they are not persisted for Slave SPI Quad mode as well.

The default setting for this preference is DISABLE.

7.4. JTAG_PORT

The JTAG_PORT allows you to preserve the JTAG configuration port after the Nexus device enters user mode. There are two states to which JTAG_PORT preference can be set:

- ENABLE – This setting preserves the JTAG port pins (TCK, TMS, TDI, and TDO) when the Nexus device is in user mode. The preference also prevents you from over-assigning I/O to those pins.
- DISABLE – This setting disconnects the JTAG port pins (TCK, TMS, TDI, and TDO) from the configuration logic. It allows those ports pins to be general purpose I/O, if they are not persisted for Slave SPI port as well.

The default setting for this preference is ENABLE.

7.5. DONE_PORT

The DONE_PORT allows you to preserve the DONE pin as sysCONFIG pin for both configuration mode and user functional mode. There are two states to which DONE_PORT preference can be set:

- ENABLE – This setting preserves the DONE pins as the sysCONFIG pin. This preference setting also prevents you from over-assigning I/O to this pin.
- DISABLE – This setting frees the DONE pin from sysCONFIG functionality. It allows this pin to be a general purpose I/O.

The default setting for this preference is ENABLE.

7.6. INITN_PORT

The INIT_PORT allows you to preserve the INITN pin as sysCONFIG pin for both configuration mode and user functional mode. There are two states to which INITN_PORT preference can be set:

- ENABLE – This setting preserves the INITN pins as the sysCONFIG pin. This preference setting also prevents you from over-assigning I/O to this pin.
- DISABLE – This setting frees the INITN pin from sysCONFIG functionality. It allows this pin to be a general purpose I/O.

The default setting for this preference is ENABLE.

7.7. PROGRAMN_PORT

The PROGRAMN_PORT allows you to preserve the PROGRAMN pin as sysCONFIG pin for both configuration mode and user functional mode. There are two states to which PROGRAMN_PORT preference can be set:

- ENABLE – This setting preserves the PROGRAMN pins as the sysCONFIG pin. This preference setting also prevents you from over-assigning I/O to this pin.
- DISABLE – This setting frees the PROGRAMN pin from sysCONFIG functionality. It allows this pin to be a general purpose I/O.

The default setting for this preference is ENABLE.

Note: The CrossLink-NX-33 device only supports the ENABLE setting for the PROGRAMN_PORT option.

7.8. BACKGROUND_RECONFIG

The BACKGROUND_RECONFIG preference allows you to setup the device in transparent access mode. The device can enter transparent access mode by:

- Executing the LSC_ENABLE_X command.
- Toggling PROGRAMN pin low or executing the REFRESH command when the TRANSFR preference is set to ON in the design.

The BACKGROUND_RECONFIG preference works in combination with the TRANSFR preference. When TRANSFR is set to ON, BACKGROUND_RECONFIG must also be set to ON. This is because full bitstream reconfiguration requires the configuration engine to access the CRAM, EBR, hard IP, and HSE in transparent access mode. If TRANSFR is set to ON without BACKGROUND_RECONFIG set to ON, a potential scenario in which bitstream reconfiguration is attempted with authentication and/or decryption enabled will fail. This failure occurs because the configuration engine is unable to access the HSE to authenticate and/or decrypt the secure bitstream. If you do not use TRANSFR, you might still enable BACKGROUND_RECONFIG. For example, when using the soft error injection (SEI) tool to inject error into the CRAM, you need to set the TranCRAM bit in CR0 to 1.

There are four states to which BACKGROUND_RECONFIG preference can be set as shown in Table 7.2. The default setting for this preference is OFF.

Table 7.2. BACKGROUND_RECONFIG Options

Option	Configuration Engine Access in Transparent Access Mode				CR0 Bit Settings in Bitstream				
	CRAM (Write)[1]	EBR (Write)	Hard IP (Write)	HSE	No CDM	TranCRAM	TranEBR	TranIP	TranHSE
OFF	No	No	No	No	0	0	0	0	0
ON	Yes	Yes	Yes	Yes	1	1	1	1	1
SRAM_EBR	Yes	Yes	No	No	1	1	1	0	0
SRAM_ONLY	Yes	No	No	No	1	1	0	0	0

Note:

1. CRAM read/write access can be overwritten by the CONFIG_SECURE setting.

7.9. DONE_EX

You can select whether the device should wake up on its own after the Done Bit is set or wait for an external DONE signal to drive the DONE pin high. The DONE_EX preference determines if the wake-up sequence is driven by an external DONE signal. The DONE_EX preference takes the setting you entered—ON or OFF. ON if you want to delay wake up until the DONE pin is driven high by an external signal and synchronous to the clock. Select OFF if you want to synchronously wake up the device based on the internal DONE bit status and ignore any external driving of the DONE pin. The default setting is OFF.

7.10. DONE_OD

The DONE pin is used in sysCONFIG to indicate that configuration is done. The DONE pin can be linked to other DONE pins and used to initiate the wake-up process. The DONE pin can be open collector or active drive for the Nexus device. The default setting is ON. This ensures that you make a conscious decision to drive the pin.

7.11. MCLK Frequency

The MCCLK_FREQ preference allows you to alter the MCLK frequency used to retrieve data from an external SPI flash when using EXTERNAL or Dual Boot configuration modes. The Nexus device uses a nominal 3.5 MHz (±7%) clock frequency to begin retrieving data from the external SPI Flash. The MCCLK_FREQ value is stored in the incoming configuration data. The Nexus device reads a series of padding bits, a *start of data* word (0xBDB3) and a control register value. The control register contains the new MCCLK_FREQ value. The Nexus device switches to the new clock frequency shortly after receiving the MCCLK_FREQ value. The MCCLK_FREQ has a range of possible frequencies available from 3.5 MHz up to 150 MHz. Ensure that you do not exceed the maximum clock rate of the SPI flash, or of the printed circuit board.

7.12. TRANSFR

The TRANSFR preference allows you to enable or disable the TRANSFR capability. TRANSFR allows the device to enter the transparent reconfiguration mode in the event of PROGRAMN pin toggling or REFRESH command execution. In a TRANSFR operation, I/O outputs maintain their previous values during reconfiguration instead of being tri-stated. I/O outputs transition to their user logic values after configuration is done. When TRANSFR is set to ON, the Wake Up Trans, NDR, and TranCRAM bits in CR0 are set to 1.

The TRANSFR preference works in combination with the BACKGROUND_RECONFIG preference. When TRANSFR is set to ON, BACKGROUND_RECONFIG must also be set to ON to allow the configuration engine to access the CRAM, EBR, hard IP, and HSE in transparent access mode. See the TransFR Operation section and Minimizing System Interruption During Configuration Using TransFR Technology (FPGA-TN-02198) for more information on the TransFR operation.

7.13. CONFIGIO_VOLTAGE_BANK0/CONFIGIO_VOLTAGE_BANK1

Because the sysCONFIG pins used for configuration may or may not be used in the design, you can declare the voltage interface for the sysCONFIG banks (bank 0 and bank 1). Setting this attribute informs the software which voltage is required in this bank to satisfy the sysCONFIG requirements. DRC errors can then be generated based on CONFIGIO_VOLTAGE_BANK0/CONFIGIO_VOLTAGE_BANK1 and usage of the dual-purpose sysCONFIG pins. The default setting is NOT_SPECIFIED.

CONFIGIO_VOLTAGE_BANK0/CONFIGIO_VOLTAGE_BANK1 = NOT_SPECIFIED/1.2V/1.5V/1.8V/2.5V/3.3V (NOT_SPECIFIED = Default)

Notes:

- If the Device Constraint Editor Global Bank VCCIO setting for *Bank0(V)/Bank1(V)* is set to *Auto* then the default voltage is expected to be 3.3 V. However, CONFIGIO_VOLTAGE_BANK0/CONFIGIO_VOLTAGE_BANK0 is also following the voltage values set on the Bank VCCIO setting.
- To avoid any issue during configuration/programming, you must assign the appropriate CONFIGIO_VOLTAGE_BANK0/CONFIGIO_VOLTAGE_BANK0 from the Device Constraint Editor Global sysCONFIG setting selection that matches the actual hardware voltage setup for all sysCONFIG pins, especially on JTAG pins.

7.14. CONFIG_IOSLEW

The CONFIG_IOSLEW preference provides the option for different I/O slew rates for configuration related I/O, which makes the Nexus devices easily adaptable to different board environments. There are three options for the CONFIG_IOSLEW preference, SLOW, MEDIUM, and FAST. The default setting for the CONFIG_IOSLEW preference is SLOW.

Note: When MCCLK_FREQ is greater than 25 MHz, it is mandatory by design that CONFIG_IOSLEW be set to FAST. This guarantees proper operation of the flash. Failure to comply might cause configuration related issues resulting in the device not being able to boot from flash.

7.15. CONFIG_SECURE

When the CONFIG_SECURE preference is set to ON, the read-back function of the SRAM memory is blocked. The device must be reprogrammed to reset the security setting. Once the security fuses are reset, the device can be programmed again. The default setting for the CONFIG_SECURE preference is OFF.

7.16. WAKE_UP

The WAKE_UP preference provides options for you to choose different wake-up sequences for releasing the global control signals when the device is entering the user functional mode.

- ENABLE_DONE_SYNC – Select the wakeup sequence synchronize with external DONE pin. For this option, the DONE_EX preference must be ON.
- DISABLE_DONE_SYNC – Select the wakeup sequence does not synchronize with external DONE pin. For this option, the DONE_EX preference should be OFF.

The default selection for WAKE_UP preference is set to DISABLE_DONE_SYNC.

See the Device Wake-Up Sequence section for more detail about the Nexus device wake-up sequence.

7.17. COMPRESS_CONFIG

The COMPRESS_CONFIG preference alters the way files are generated with compressed FPGA data frames. The default setting of COMPRESS_CONFIG is OFF.

7.18. EARLY_IO_RELEASE

The EARLY_IO_RELEASE preference enables early I/O release feature on I/O Bank1, I/O Bank2, I/O Bank6, and I/O Bank7. The default setting of EARLY_IO_RELEASE is OFF.

7.19. BOOTMODE

The BOOTMODE preference allows you to select the booting mode at power up (POR), PROGRAMN pin toggle or REFRESH command execution.

- DUAL – Perform the dual boot for booting event. In case the primary booting image (bitstream) is failed, the secondary (Golden) booting image is automatically invoked to boot up the device.
- SINGLE – Perform the single boot only. If failed, device go to unprogrammed mode directly.
- NONE – No Master SPI booting at POR, PROGRAMN pin toggle or REFRESH command execution, wait for Slave Configuration Port to configure the device.

The default BOOTMODE selection is DUAL mode.

7.20. USERCODE_FORMAT

The USERCODE_FORMAT preference selects the format for the data field used to assign a value in the USERCODE preference. The USERCODE_FORMAT has four options:

- Binary – USERCODE is set using 32 *1* or *0* characters.
- Hex – USERCODE is set using eight hexadecimal digits, that is 0-9A-F.
- ASCII – USERCODE is set using up to four ASCII characters.
- Auto – USERCODE is automatically created by the software. The upper 16-bit is Unique ID and the lower 16 bits are sequentially increased automatically for every bitstream generation.

7.21. USERCODE

The Nexus device contains a 32-bit register for storing a user-defined value. The default value stored in the register is 0x00000000. Using the USERCODE preference, you can assign any value to the desired register. Suggested uses include the configuration data version number, a manufacturing ID code, date of assembly, or the JEDEC file checksum.

The format of the USERCODE field is controlled using the USERCODE_FORMAT preference with Hex format as default. Data entry can be performed in either Binary, Hex, ASCII, or Auto formats.

7.22. UNIQUE_ID

The Nexus device contains a 16-bit register for storing a user-defined value. The default value stored in the register is 0x0000. Unique ID can only be set when USERCODE_FORMAT is Auto.

8. Device Wake-Up Sequence

When configuration is complete (the SRAM has been loaded), the device wakes up in a predictable fashion. If the Nexus device is the only device in the chain, or the last device in a chain, the wake-up process should be initiated by the completion of configuration. Once configuration is complete, the internal DONE bit is set and then the wake-up process begins. Figure 8.1 shows the wake-up sequence with DISABLE_DONE_SYNC option.

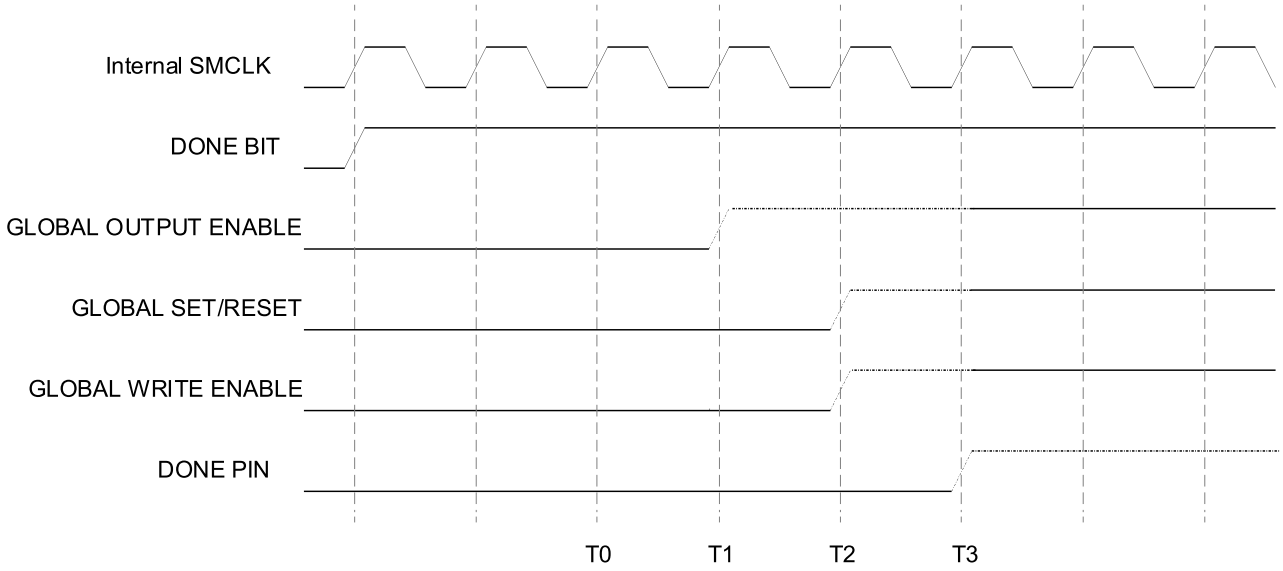


Figure 8.1. Wake-up Sequence Using Internal Clock

8.1. Wake-Up Signals

Three internal signals GSRN, GWE, and GOE determine the wake-up sequence.

- GSRN is used to set and reset the core of the device. GSRN is asserted (low) during configuration and de-asserted (high) in the wake-up sequence.
- When the GWE signal is low, it safeguards the integrity of the RAM Blocks and LUTs in the device. This signal is low before the device wakes up. This control signal does not control the primary input pin to the device but controls specific control ports of EBR and LUTs.
- When low, GOE prevents the device's I/O buffers from driving the pins. The GOE only controls *output* pins. Once the internal DONE is asserted the Nexus device responds to input data.
- When high, the DONE pin indicates that configuration is complete and that no errors were detected.
- Before DONE pin goes high, all signals going to EBR should hold steady, otherwise, it might impact the EBR initialization.

The available wake-up sequences are shown in Table 8.1. A wake-up sequence is the order in which the signals change. The phase transition based on a wakeup clock, as discussed below. The exact timing relationship between the internal signals and the wakeup clock varies and is not specified.

Table 8.1. Wake-Up Sequences

Sequence	Phase T0	Phase T1	Phase T2	Phase T3
ENABLE_DONE_SYNC	DONE	GOE	GWE, GSRN	—
DISABLE_DONE_SYNC	—	GOE	GWE, GSRN	DONE

8.2. Wake-Up Clock

The Nexus device always use the internal clock to perform the device wake up sequence. Once the Nexus device is configured, it enters the wake-up state, which is the transition from the configuration mode to user mode. This sequence is synchronized to the internal SMCLK, which is derived from the internal oscillator.

9. Daisy Chaining

Typically, there is one configuration bitstream per FPGA in a system. Today's systems often have several FPGA devices. To save PCB space and use external storage device more efficiently, several different FPGA bitstreams from various configuration devices and designs can share a single configuration mechanism by using the daisy chain method.

The Nexus device supports only the Flow Through chaining method with the first FPGA being master or slave. To enable synchronization to the external DONE pin, DONE_EX must be set to ON and WAKE_UP must be set to ENABLE_DONE_SYNC in the Lattice Radiant Device Constraint Editor. Based on the configuration ports arrangement, Nexus devices can only be the first device inside a configuration daisy chain.

Note: When DONE_EX=ON and ENABLE_DONE_SYNC is set, once the last device finishes configuration and after the DONE pin is released by the last device, the wake-up sequence will complete in 200 µs.

9.1. Flow Through Option

The flow through option for the Nexus device can only be used with serial daisy chains and supports either master SPI or slave SPI configuration mode on the leading FPGA device while all downstream devices in the chain are set to Auto SSPI mode.

To configure in flow through daisy chain mode, wire together the SPI data lines, PROGRAMN, DONE, and INITN pins of all the devices in the daisy chain. The SCSN of a downstream FPGA device is connected to the MCSNO of the FPGA device before it. In this daisy chain configuration, the devices are configured sequentially. When a device completes configuration, the completed device is placed in a flow through state and the MCSNO pin is driven low thereby selecting the next device in the chain. Once the flow through option starts, the device remains in flow through until the wake-up sequence completes.

Flow through daisy chain configuration uses either the MSPI or SSPI configuration mode on the first device. All downstream devices in the chain are configured using SSPI mode and must be set in Auto SSPI mode. Downstream devices can only accept bitstreams in Auto SSPI mode. There is no readback capability for downstream devices.

Note: The Refresh command cannot be used to reconfigure all FPGAs in the daisy chain mode. Instead, toggle PROGRAMN.

The following table shows the control bit settings for an FPGA device in a daisy chain.

Table 9.1. Control Bit Settings for FPGA Device in a Daisy Chain

FPGA	P_DONE CTRL Bits (CR0[18:17])[1]	SSPI Auto Bit (CR1[18])[2]
First FPGA	11	0
Between First and Last FPGA	11	1
Last FPGA	00	1

Notes:

1. To set P_DONE CTRL=11, chain mode must be set to Flowthrough in the Lattice Radiant Device Constraint Editor.
2. To set SSPI Auto=1, set SSPI_AUTO to ENABLED in the Lattice Radiant Device Constraint Editor (default setting is DISABLED). Once SSPI_AUTO is enabled, the SSPI Auto bit in the generated .fea file is set as 1 and you may proceed to program the SSPI Auto bit to 1 for the downstream FPGA in the daisy chain. To permanently set the SSPI Auto bit, refer to Appendix C for guides on programming the Nexus Feature Row using the Feature Row Editor. Note that feature row bits are one time programmable (OTP). These bits can only be modified once.

9.2. Using Radiant Deployment Tool to Create a sysCONFIG Daisy Chain Hex File

The following steps provide the procedure for generating a sysCONFIG daisy chain hex file using the Radiant Deployment Tool.

1. Invoke the Lattice Radiant Deployment Tool by selecting **Tools > Deployment Tool** from the Lattice Radiant Programmer.
2. In the Radiant Deployment Tool window, select **External Memory** as the **Function Type** and select **sysCONFIG Daisy Chain** as the **Output File Type** (Figure 9.1).
3. Click **OK**.

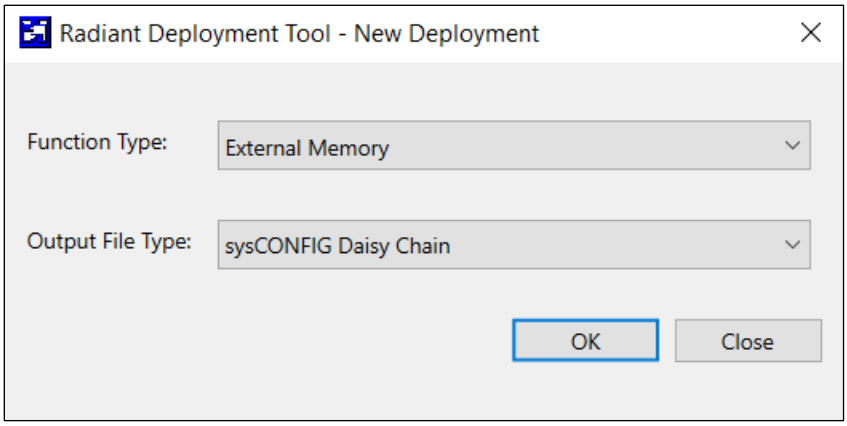


Figure 9.1. Creating New Deployment for sysCONFIG Daisy Chain Hex File

Step 1 of 4: Select Input File(s) window (Figure 9.2)

- Click on the **…** button in the **File Name** fields to browse and select the bitstream files to be used to create the hex file.
- The **Device Family** and **Device** fields automatically populate based on the bitstream files selected.
- Click **Next**.

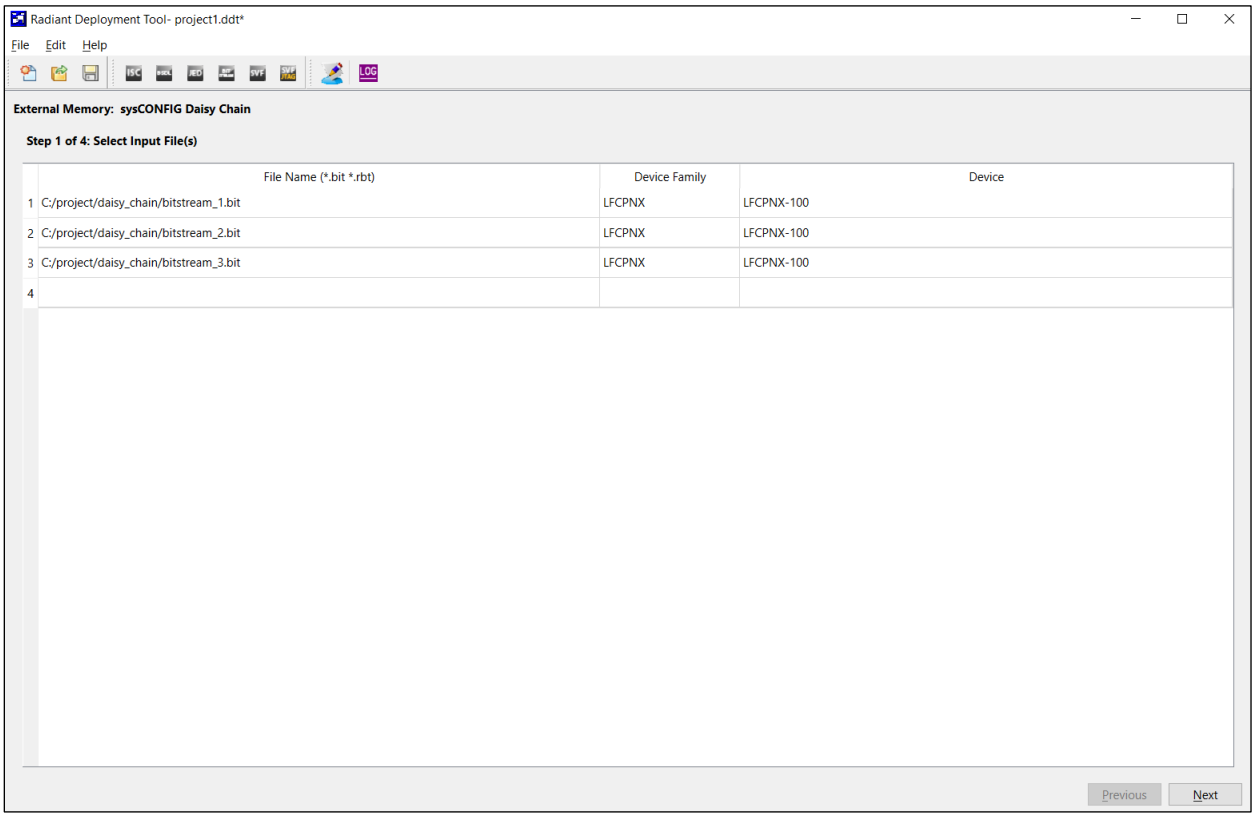


Figure 9.2. Select Input File(s) Window

Step 2 of 4: sysCONFIG Daisy Chain Options window (Figure 9.3)

- Select the **Format** (Intel Hex, Motorola Hex, or Extended Tektronix Hex).
- Select the **Merge Format** (Intelligent Merge, Combine Merge).
  - Intelligent Merge: Merge all the bitstreams provided in step 1 of 4 and set the P_DONE CTRL bits in each of the bitstreams accordingly.
  - Combine Merge: Merge all the bitstreams provided in step 1 of 4.
- Select the **Daisy Chain Mode** (**Flowthrough**).
- Select the **Frequency** (Default, 3.5 MHz, 7.0 MHz, 14.1 MHz, 28.1 MHz).
- Select the **Byte Wide Bit Mirror** option as required – Flips each byte in Intel, Extended Tektronix, or Motorola hexadecimal data files. For example, 0xCD (b1100 1101) becomes 0xB3 (b1011 0011) when this is selected.
- Click **Next**.

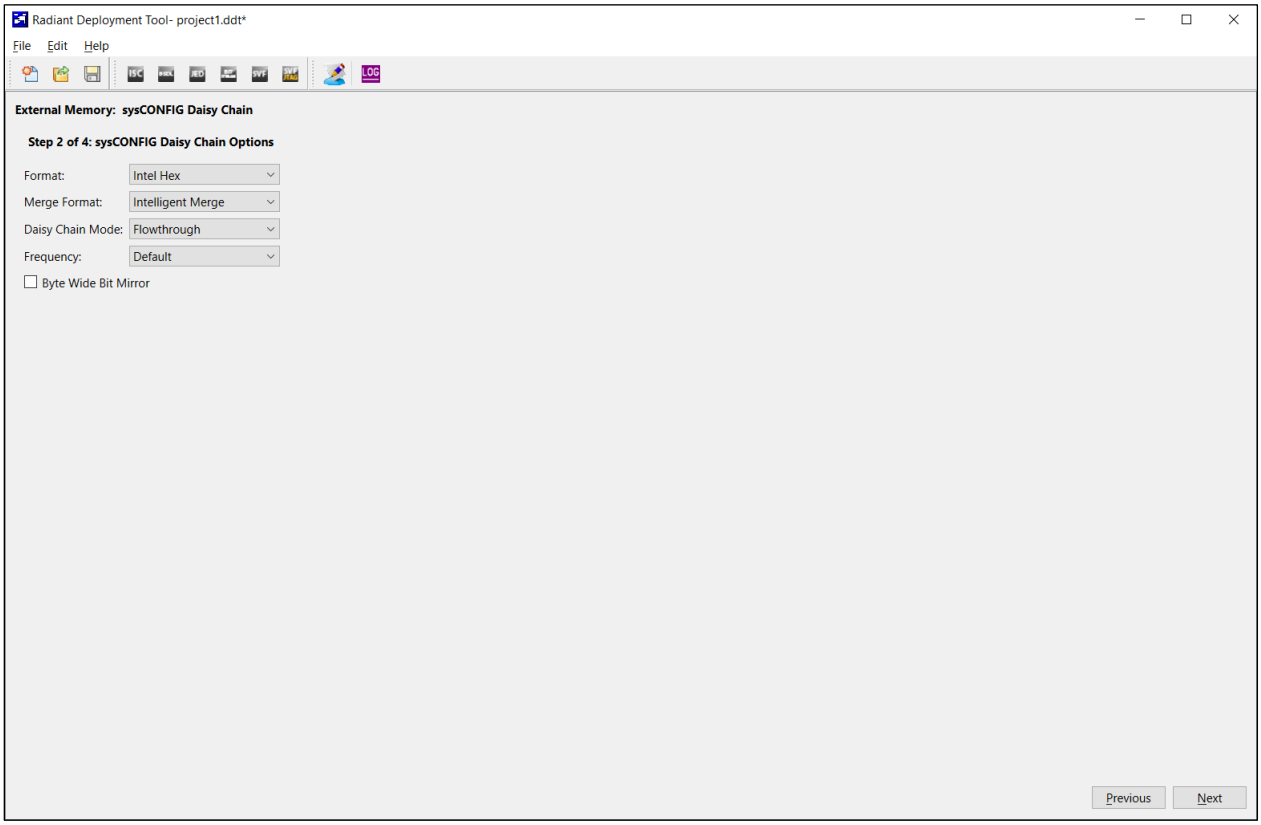


Figure 9.3. sysCONFIG Daisy Chain Options Window

Step 3 of 4: Select Output File(s) window (Figure 9.4)

- Specify the name and location of the output hex file in the **Output File 1** field.
- Click **Next**.

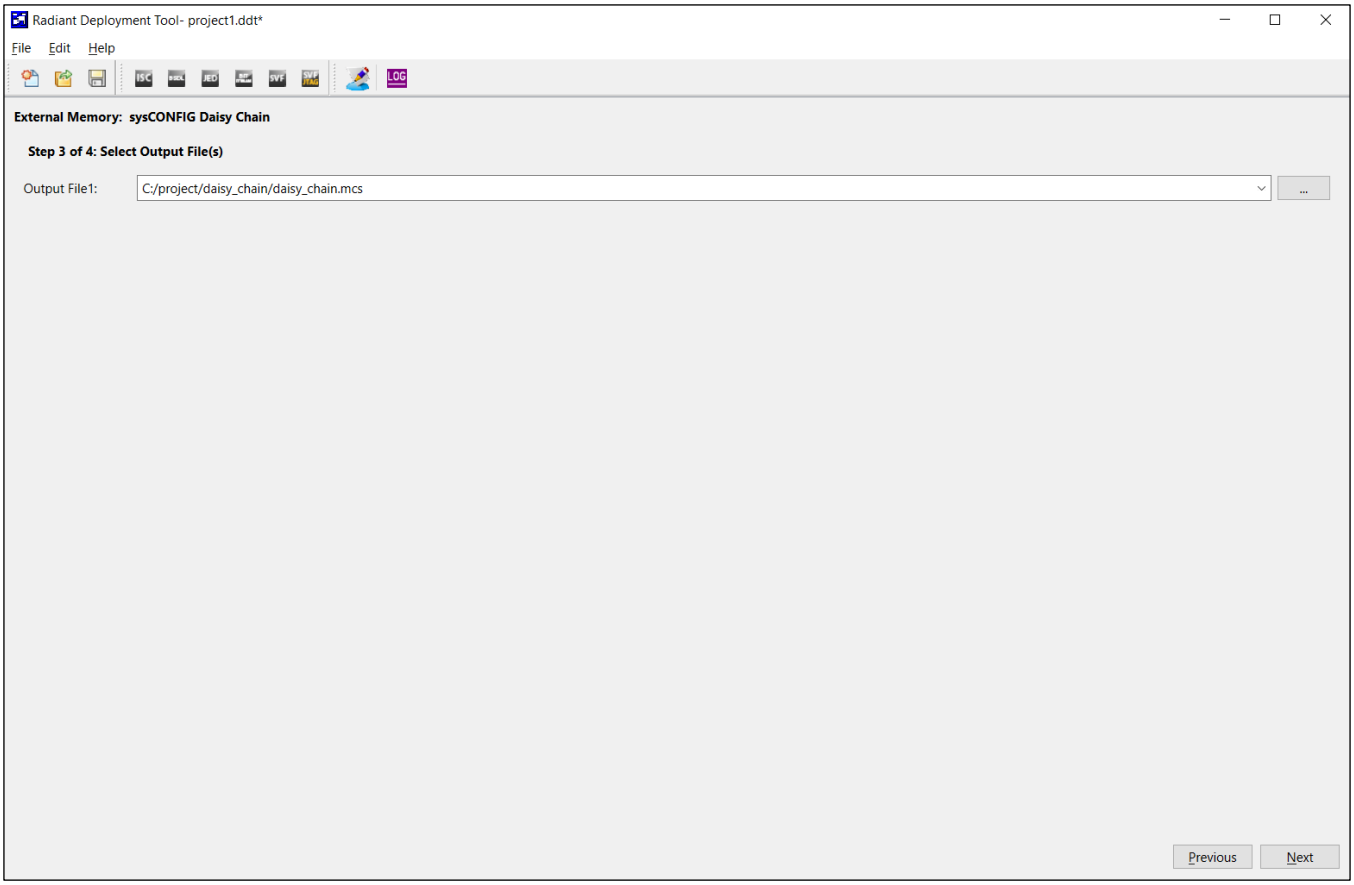


Figure 9.4. Select Output File(s) Window

Step 4 of 4: Generate Deployment window (Figure 9.5)

- Review the summary information.
- If everything is correct, click the **Generate** button.
- The **Generate Deployment** pane should indicate that the file was generated successfully.
- The generated .mcs file is located in the same directory as the bitstream files.
- Save the deployment settings by selecting **File > Save**.
- To exit, select **File > Exit**.

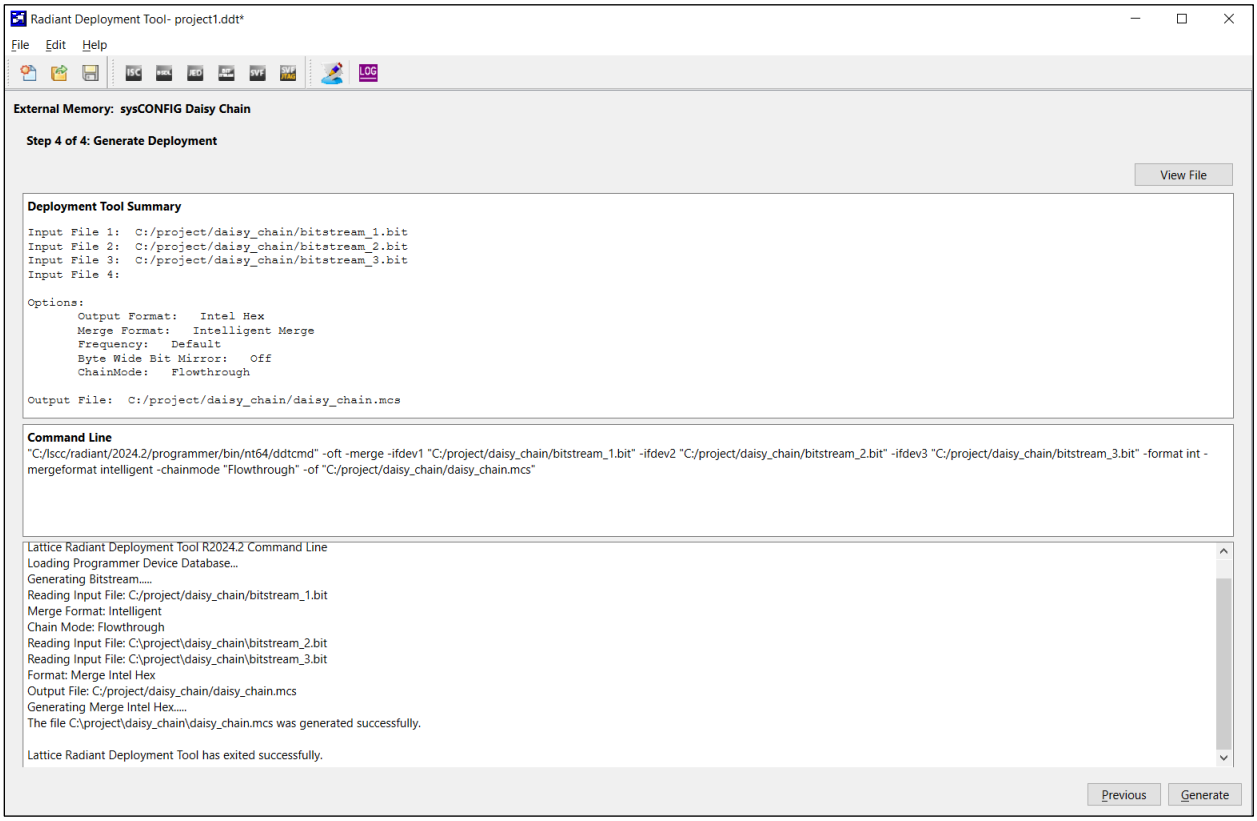
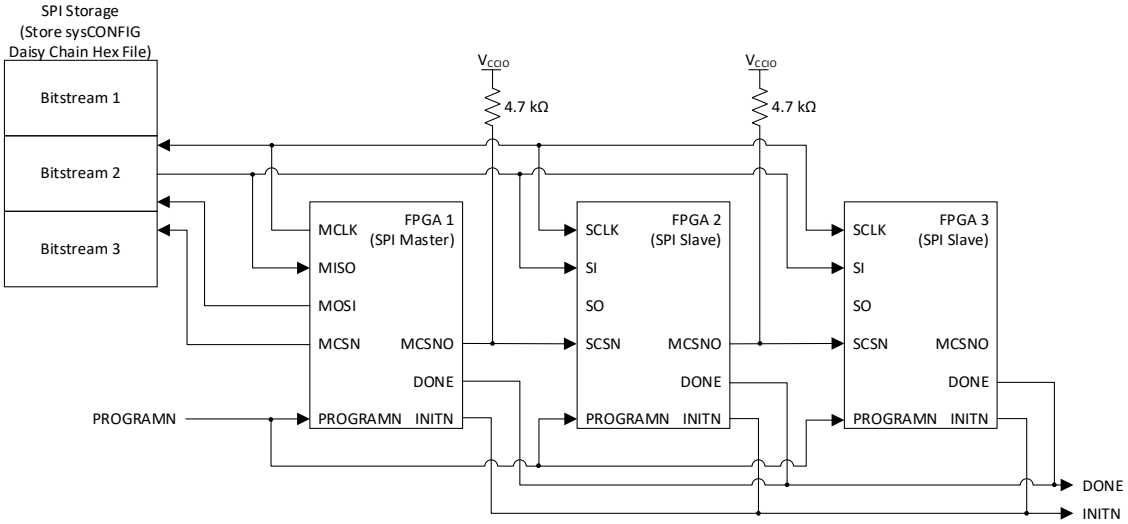


Figure 9.5. Generate Deployment Window

You may proceed to program the .mcs or hex file into SPI storage.

9.3. MSPI Configuration Mode in Flow Through Mode

When using MSPI configuration mode on the first device, the maximum supported MCLK frequency is 28.1 MHz. The following figure shows FPGAs connected in a daisy chain with the first FPGA being the master.



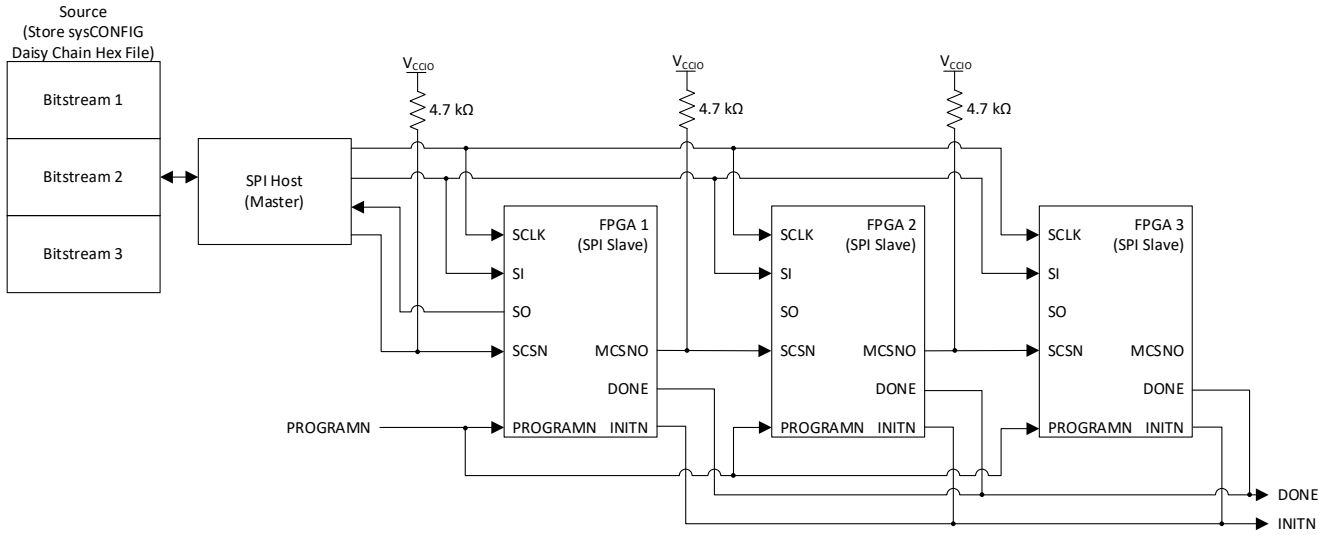
Note: Add a 4.7-kΩ pull-up resistor on the chip select signal between devices in the daisy chain.

Figure 9.6. Daisy Chain Configuration with Master SPI in Flow Through Mode

The first FPGA acts as the master device and configures from SPI flash. After the first device completes configuration, the MCLK clock continues running to read the image/pattern for the next downstream device from SPI flash. The configured device MCSNO is driven low to select the next downstream device in the chain to configure with the next image/pattern. Daisy chain configuration continues until the last downstream FPGA in the chain is successfully configured. After the DONE pin is released by the last device and the wake-up sequence completes, all FPGAs enter user mode. The external flash stores the merged bitstreams.

9.4. SSPI Configuration Mode in Flow Through Mode

When using SSPI configuration mode on the first device, the maximum supported SCLK frequency is 28.1 MHz. The following figure shows FPGAs connected in a daisy chain with the first FPGA being the slave.

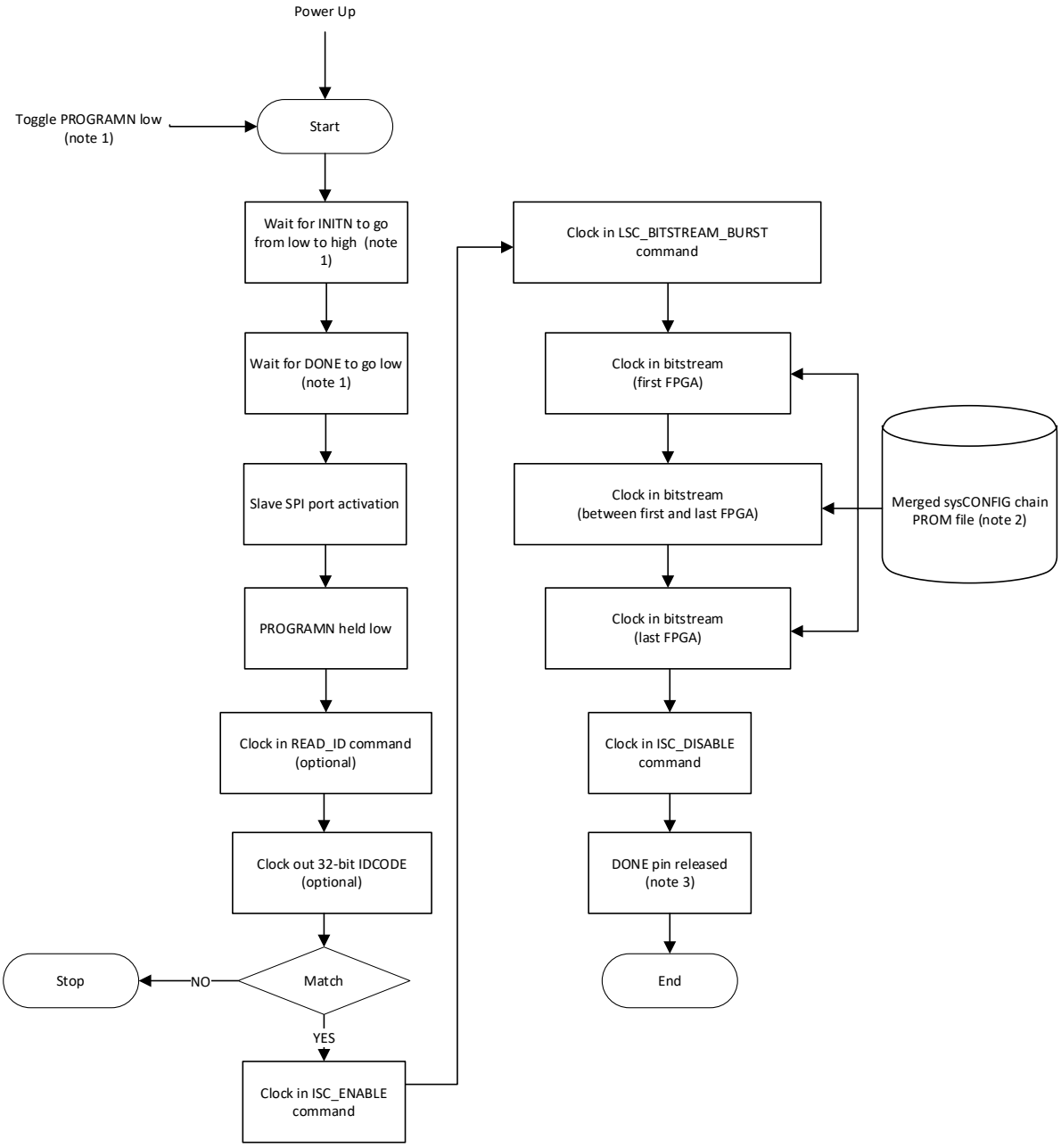


Note: Add a 4.7-kΩ pull-up resistor on the chip select signal between devices in the daisy chain.

Figure 9.7. Daisy Chain Configuration with Slave SPI in Flow Through Mode

The host CPU reads the merged bitstreams from the source and sends the bitstream data as a continuous stream with commands to configure the first device. After the first device completes configuration, the host CPU continues to send the bitstream for the next downstream device. The completed device MCSNO is driven low to select the next downstream device in the chain to configure. Daisy chain configuration continues until the last downstream FPGA in the chain is successfully configured. After the DONE pin is released by the last device and the wake-up sequence completes, all FPGAs enters user mode.

The following figure shows the sequence for slave SPI daisy chain operation.



Notes:

1. PROGRAMN toggle can be used to reconfigure all the FPGAs in the daisy chain. For more information about the behavior of the pin, refer to the PROGRAMN, INITN, and DONE section. For more information about the timing specifications, refer to the CertusPro-NX Family Data Sheet.
2. For a sysCONFIG chain of devices, the input file can be a merged PROM file.
3. When DONE_EX=ON and ENABLE_DONE_SYNC is set, once the last device finishes configuration and after the DONE pin is released by the last device, the wake-up sequence will complete in 200 µs.

Figure 9.8. Slave SPI Daisy Chain Operation Sequence

Appendix A. Nexus Slave SPI Programming Guide

The SPI port of the Nexus device can be used for device configuration. If these pins are not used by user logic, they are tri-stated with a weak pull-up. The Slave SPI port must be enabled to support device configuration from an external host or download cable using SPI protocol. This is done by setting the SLAVE_SPI_PORT preference to ENABLE in the bitstream through the Lattice Radiant Device Constraint Editor. Slave SPI mode supports single device configuration.

The Lattice Radiant Programmer supports the Slave SPI programming mode as one of the device access options. Selecting this option allows you to perform device erase, program, verify, readback, refresh, and more. The connections of Slave SPI pins to the Lattice programming cable are:

- TDI -> SISPI
- ISPEN -> SN
- TCK -> SCLK
- TDO -> SPISO

The Slave SPI chip select pin (SCSN) is held low during the command sequences. You can generate a SVF file from the bitstream (.bit) with the Lattice Deployment Tool software to show the details of the command sequences for Slave SPI programming mode.

The *Program, Erase and Verify* flow in Radiant Programmer for from any slave configuration port only targets the FPGA SRAM array, which does not handle the Hard IP and EBR initialization. *Fast Program* option is preferred for Full device configuration.

Appendix B. Nexus Bitstream File Format

B.1. Configuration Bitstream Format

The base binary file format is the same for all non-encrypted, non-1532 configuration modes. Different file types (hex, binary, ASCII, and so on) may ultimately be used to configure the device, but the data in the file is the same. Table B.1 lists the format of a non-encrypted and compressed bitstream. Table B.2 and Table B.3 list the formats of non-encrypted, uncompressed bitstreams without and with early I/O release, respectively. Each bitstream consists of a comment string, a header, the preamble, and the configuration setup and data. Note that when the bitstream is encrypted, the second 16 bits of the 32-bit preamble is as follows:

- Encrypted with IP evaluation and without user AES key: 0xFFFFBEB3.
- Encrypted without IP evaluation and with user AES key: 0xFFFFBFB3.
- Encrypted with IP evaluation and user AES key: 0xFFFFBCB3.

Only the frame data can be compressed. The EBR data cannot be compressed. The data frame padding for compression is as follows:

- Before compression, pad the leftmost bits of the frame data with zeroes (0) to make the data frame 64-bit bounded. For example, if the original uncompressed data frame length is 125-bit, such as 01.....10, the data frame must be padded with all 0 (3-bit 0s) at leftmost to make it 64-bit bounded (128-bit), 00001.....10.
- After compressing the frame data, pad the rightmost bits of the frame data with zeroes (0) to make the data frame byte bounded. For example, if the 128-bit data frame is compressed to become 101-bit (not byte bounded), such as 10.....01, the compressed data frame must be padded with all 0 (3-bit 0s) at the rightmost to make it byte bounded (104-bit), 10……01000

Table B.1. Nexus Compressed Bitstream Format

Frame	Contents[D7…D0]	Description	CRC
LSCC signature	4C534343	32 bits of the LSCC Signature (LSCC = 0x4C534343)	None
Comments	(Comment String)	ASCII Comment (Argument) String and Terminator	
Header	(LSB) 1111111111111111	First 16 bits of the 32-bit Preamble	None
	1011110110110011	Second 16 bits of the 32-bit Preamble (0xFFFFBDB3)	
	11111111…11111111	32-bit Dummy	
Frame (Reset CRC)	00111011	LSC_RESET_CRC Command	Exclude
	0	CRC Comparison Flag (0 = no)	Exclude
	0000.....000000	23-bit Command Information (23 zeroes)	Exclude
Dummy Frame	1111….111111	Dummy byte (32 bits of ones)	Exclude
Frame (Verify ID) OPTIONAL DATA But Mandatory Frame	11100010	LSC_VERIFY_ID command (if the Verify ID data is not there, then Frame contains all 1s)	Start Include
	0	CRC Comparison Flag (0 = no) for Verify ID command	Include
	0000.....000000	23-bit Command Information (23 zeroes)	Include
	(Device ID[31..0])	32-bit Device Id	Include
Frame (Control Register 0)	00100010	LSC_PROG_CNTRL0 command	Include
	0	CRC Comparison Flag (0 = no)	Include
	0000.....000000	23-bit Command Information (23 zeroes).	Include
	(CtlReg0[31..0])	Control Register 0 Data definition[2]	Include
Frame (Control Register 1) OPTIONAL FRAME	00100011	LSC_PROG_CNTRL1 command	Include
	0	CRC Comparison Flag (0 = no)	Include
	0000.....000000	23-bit Command Information (23 zeroes).	Include
	CtlReg1[31..0])	Control Register 1 Data definition[3]	Include

Frame	Contents[D7…D0]	Description	CRC
Frame (Store Compress)[8]	00000010	LSC_WRITE_COMP_DIC Command.	Include
	0	CRC Comparison Flag (0 = no) for Verify ID command	Include
	0000…..000000	23-bit Command Information (23 zeroes).	Include
	(Pattern16[7..0])	Next most frequently occurring 8-bit pattern in the Frame Data.	Include
	Pattern15[7..0])	Next most frequently occurring 8-bit pattern in the Frame Data.	Include
	==========	==========	Include
	Pattern1[7..0])	Next most frequently occurring 8-bit pattern in the Frame Data.	Include
Frame (Write Address for Loading MIB Frames)	10110100	LSC_WRITE_ADDRESS Command	Include
	0	CRC Comparison Flag (1 = yes).	Include
	0	CRC Comparison at End Flag (0 = Execute CRC comparison at the end of each data frame)	Include
	0	—	Include
	0	—	Include
	0000	—	Include
	0000…..000000	16-bit Command Information (23 zeroes).	Include
	000…00000	16-bit zeroes	Include
	1000000000000000	16-bit address to load	Include
Frame (Auto Increment For next 32 Frames)	10111000	LSC_PROG_INCR_CMP Command.	Include
	1	CRC Comparison Flag (1 = yes)	Include
	1	CRC Comparison at End Flag (0 = Execute CRC comparison at the end of each data frame)	Include
	0	Include dummy bits.	Include
	1	Dummy definition (1 = use Bit [3:0] as dummy byte count)	Include
	0004	Dummy definition.	Include
	0000000000100000	Number of MIB Frames Always 32 (16-bit Frame Size)	Include
Frame (Data 0)	(Frame 0 Data)	*x* Data bits for Frame 0 (14 bits Parity + Data)[5]	Include
	11111111…1111	Dummy byte (32 bits of ones)	Include
Frame (Data 1)	(Frame 1 Data)	*x* Data bits for Frame 1 (14 bits Parity + Data)[5]	Include
	11111111…11111	Dummy byte (32 bits of ones)	Include
==========	==========	==========	==========
Frame (Data 31)	(Frame 31 Data)	*x* Data bits for Frame 31 (14 bits Parity + Data)[5]	Include: End
	(CRC[15..0])	16-bit CRC	CRC
	11111111…1111	Dummy byte (32 bits of ones)	Include: Start
Frame (Reset Address)	01000110	LSC_INIT_ADDRESS Command	Include
	0	CRC Comparison Flag (0 = no)	Include
	0000…..000000	23-bit Command Information (23 zeroes).	Include

Frame	Contents[D7…D0]	Description	CRC
Frame (Write Increment)	10111000	LSC_PROG_INCR_CMP Command.	Include
	1	CRC Comparison Flag (1 = yes)	Include
	1	CRC Comparison at End Flag (0 = Execute CRC comparison at the end of each data frame)	Include
	0	Include dummy bits.	Include
	1	Dummy definition (1 = use Bit [3:0] as dummy byte count)	Include
	0004	Dummy definition	Include
	(16-bit = Number of Frames)	Number of configuration data frames included in operand[5] For Example, CrossLink-NX A. No. of Actual Data Frames = 4096(Total ASR Size) – 24(MIB Frames) – 12 (TAP Frames)	Include
Frame (Data 0)	(Frame 0 Data)	*x* Data bits for Frame 0 (14 bits Parity + Data)[5]	Include
	11111111…1111	Dummy byte (32 bits of ones)	Include
Frame (Data 1)	(Frame 1 Data)	*x* Data bits for Frame 1 (14 bits Parity + Data)[5]	Include
	11111111…1111	Dummy byte (32 bits of ones)	Include
=========	==========	==========	==========
Frame (Data n-1)	(Frame n-1 Data)	*x* Data bits for Frame 0 (14 bits Parity + Data)[5]	Include: End
	(CRC[15..0])	16-bit CRC	CRC
	11111111…1111	Dummy byte (32 bits of ones)	Start: Include
Frame (Write address for Loading TAP Frames)	10110100	LSC_WRITE_ADDRESS Command	Include
	0	CRC Comparison Flag (1 = yes).	Include
	0	CRC Comparison at End Flag (0 = Execute CRC comparison at the end of each data frame)	Include
	0	—	Include
	0	—	Include
	0000	—	Include
	0000…..000000	16-bit Command Information (23 zeroes).	Include
	000…00000	16-bit Zeroes	Include
	1000000000100000	16-bit address to load	Include
Frame (Write Increment)	10111000	LSC_PROG_INCR_CMP Command.	Include
	1	CRC Comparison Flag (1 = yes)	Include
	1	CRC Comparison at End Flag (0 = Execute CRC comparison at the end of each data frame)	Include
	0	Include dummy bits	Include
	1	Dummy definition (1 = use Bit [3:0] as dummy byte count)	Include
	0004	Dummy definition	Include
	(16-bit = Frame Number)	Number of TAP frames[5]	Include
Frame (Data 0)	(Frame 0 Data)	*x* Data bits for Frame 0 (14 bits Parity + Data)[5]	Include
	11111111…1111	Dummy byte (32 bits of ones)	Include
Frame (Data 1)	(Frame 1 Data)	*x* Data bits for Frame 1 (14 bits Parity + Data)[5]	Include
	11111111…1111	Dummy byte (32 bits of ones)	Include
==========	==========	==========	==========
Frame (Data m-1)	(Frame m-1 Data)	*x* Data bits for Frame (m-1) (14 bits Parity + Data)[5]	Include: End
	(CRC[15..0])	16-bit CRC	CRC
	11111111…1111	Dummy byte (32 bits of ones)	Start: Include
Frame (Power Control)	01010110	LSC_POWER_CTRL Command	Include
	0	CRC Comparison Flag (0 = no)	Include

Frame	Contents[D7…D0]	Description	CRC
	000….0000	Operand 22 zeroes	Include
	X	Enable the VCCPG Power Domain (1= yes)	Include
Frame (INIT Bus Address)	11110110	LSC_INIT_BUS_ADDR Command	Include
	0	CRC Comparison Flag (0 = no)	Include
	0000…..000000	23-bit Command Information (23 zeroes)	Include
	00	Default 2 zeroes (Reserved)	Include
	Data[29:28]	Bus Width[4] 0: 8-bit Hard IP 1: 10-bit Hard IP 2: 10-bit EBR /Large RAM 3: 32-bit Hard IP	Include
	Data [27:17]	ID CODE	Include
	Data[16:0]	Starting address	Include
Frame (Write data through INIT Bus)	01110010	LSC_INIT_BUS_WRITE Command	Include
	1	CRC Comparison Flag (1 = yes)	Include
	1	CRC Comparison at End Flag (1 = Execute CRC comparison at the end of all operands)	Include
	0	Exclude dummy bytes from bit stream	Include
	1	Dummy definition (1 = use the next 4-bit Dummy Definition settings)	Include
	0000	Dummy definition	Include
	(16-bit Size)	Number of 72-bit frames in the operand. One = 0000000000000001	Include
	(Data)	Data Frame	Include: End
	(CRC[15..0])	16-bit CRC	CRC
Frame (SED CRC) OPTIONAL FRAME	10100010	LSC_PROG_SED_CRC Command	Start : Include
	0	CRC Comparison Flag (0 = no)	Include
	0000…..000000	23-bit Command Information (23 zeroes)	Include
	(CRC[31..0])	32-bit SED CRC	Include
Frame (Program Security) OPTIONAL FRAME	11001110	ISC_PROGRAM_SECURITY Command	Include
	0	CRC Comparison Flag (0 = no)	Include
	0000…..000000	23-bit Command Information (23 zeroes)	Include
Frame (USERCODE)	11000010	ISC_PROGRAM_USRCODE Command	Include
	1	CRC Comparison Flag (1 = yes)	Include
	0000…..000000	23-bit Command Information (23 zeroes)	Include
	(U[31..0])	32-bit User code Data	Include: End
	(CRC[15..0])	16-bit CRC	CRC
Frame (Program Done)	01011110	ISC_PROGRAM_DONE	Exclude
	0	CRC Comparison Flag (0 = no)	Exclude
	00000….0000	Operand 21 zeroes	Exclude
	XX	Behavior 00 = No overload (default), 01 = Serial System Configuration Daisy Chain Support in Bit stream, 10 = Parallel System Configuration Daisy Chain Support in bit stream, 11 = No overload(default)	Exclude
Frame (DUMMY)	1111…1111	4 bytes DUMMY command (all ones).	Exclude

Notes:

1. Data is byte bounded and must be padded with ones (1). For example, a 6-bit data frame of b111010 is written in the bitstream as b11111010 to make it byte bounded.
2. If SED option is not chosen, place command DUMMY (NOOP) (32 bits of 1) on the command space. NOOP command is not included in CRC calculation. This is necessary to make the file size consistent with or without the security option chosen. It must be replaced with the LSC_PROG_SED_CRC frame by Bitgen if the user implements SED. Include in CRC calculation if LSC_PROG_SED_CRC command. Do not include in CRC calculation if NOOP.
3. The CRC must be flipped so that CRC[0] is the LSB and CRC[15] is the MSB. For example, CRC = 0x0823 (CRC[15]…CRC[0]) in the bit file above is b1100010000010000. Refer to Appendix Generic Configuration Specification for CRC calculation details.
4. Insert a Write EBR Frame for each initialized EBR, or for each group of EBR Frames that is initialized to the same value. Omit if there is no EBR initialization data.
5. By default, this frame must be the Verify ID command. Bitgen shall have a switch to allow the Verify ID frame to be replaced with NOOP (all 1s). Include in CRC calculation if Verify ID command. Do not include in CRC calculation if NOOP.
6. If security option is not chosen, place command NOOP (32 bits of 1) on the command space. NOOP command is not included in CRC calculation. This is necessary to make the file size consistent with or without security option chosen. Include in CRC calculation if Program Security command. Do not include in CRC calculation if NOOP.
7. The Control Register 0 must be set to the SW Default value, setting Control Register 0 bits [31..30] = [0..0].

Table B.2. Nexus Uncompressed Bitstream Format – Without Early I/O Release

Frame	Contents[D7…D0]	Description	CRC
LSCC signature	4C534343	32 bits of the LSCC Signature (LSCC = 0x4C534343)	None
Comments	(Comment String)	ASCII Comment (Argument) String and Terminator	
Header	(LSB) 1111111111111111	First 16 bits of the 32-bit Preamble	None
	1011110110110011	Second 16 bits of the 32-bit Preamble (0xFFFFBDB3)	
	11111111…11111111	32-bit Dummy	
Frame (Reset CRC)	00111011	LSC_RESET_CRC Command	Exclude
	0	CRC Comparison Flag (0 = no)	Exclude
	0000…..000000	23-bit Command Information (23 zeroes)	Exclude
Dummy Frame	1111….111111	Dummy byte (32 bits of ones)	Exclude
Frame (Verify ID) OPTIONAL DATA But Mandatory Frame	11100010	LSC_VERIFY_ID command (if the Verify ID data is not there, then Frame contains all 1s)	Start Include
	0	CRC Comparison Flag (0 = no) for Verify ID command	Include
	0000…..000000	23-bit Command Information (23 zeroes)	Include
	(Device ID[31..0])	32-bit Device Id	Include
Frame (Control Register 0)	00100010	LSC_PROG_CNTRL0 command	Include
	0	CRC Comparison Flag (0 = no)	Include
	0000…..000000	23-bit Command Information (23 zeroes)	Include
	(CtlReg0[31..0])	Control Register 0 Data definition[2]	Include
Frame (Control Register 1) OPTIONAL FRAME	00100011	LSC_PROG_CNTRL1 command	Include
	0	CRC Comparison Flag (0 = no)	Include
	0000…..000000	23-bit Command Information (23 zeroes)	Include
	CtlReg1[31..0])	Control Register 1 Data definition[3]	Include

Frame	Contents[D7…D0]	Description	CRC
Frame (Write Address for Loading MIB Frames)	10110100	LSC_WRITE_ADDRESS Command	Include
	0	CRC Comparison Flag (1 = yes)	Include
	0	CRC Comparison at End Flag (0 = Execute CRC comparison at the end of each data frame)	Include
	0	—	Include
	0	—	Include
	0000	—	Include
	0000…..000000	16-bit Command Information (23 zeroes).	Include
	000…00000	16-bit zeroes	Include
	1000000000000000	16-bit address to load[6]	Include
Frame (Auto Increment For next 32 Frames)	10000010	LSC_PROG_INCR command	Include
	1	CRC Comparison Flag (1 = yes)	Include
	0	CRC Comparison at End Flag (0 = Execute CRC comparison at the end of each data frame)	Include
	0	Include dummy bits	Include
	1	Dummy definition (1 = use Bit [3:0] as dummy byte count)	Include
	0001	Dummy definition.	Include
	0000000000100000	(MIB Frames size 32) (16-bit Frame Size)[5]	Include
Frame (Data 0)	(Frame 0 Data)	*x* Data bits for Frame 0 (14 bits Parity + Data)[5]	Include: End
	(CRC[15..0])	16-bit CRC	CRC
	11111111	Dummy byte (8 bits of ones)	Start: Include
Frame (Data 1)	(Frame 1 Data)	*x* Data bits for Frame 1(14 bits Parity + Data)[5]	Include: End
	(CRC[15..0])	16-bit CRC	CRC
	11111111	Dummy byte (8 bits of ones)	Start: Include
==========	==========	==========	==========
Frame (Data 31)	(Frame 31 Data)	*x* Data bits for Frame 31 (14 bits Parity + Data)[5]	Include: End
	(CRC[15..0])	16-bit CRC	CRC
	11111111	Dummy byte (8 bits of ones)	Start: Include
Frame (Reset Address)	01000110	LSC_INIT_ADDRESS Command	Include
	0	CRC Comparison Flag (0 = no)	Include
	0000…..000000	23-bit Command Information (23 zeroes)	Include
Frame (Write Increment)	10000010	LSC_PROG_INCR command	Include
	1	CRC Comparison Flag (1 = yes)	Include
	0	CRC Comparison at End Flag (0 = Execute CRC comparison at the end of each data frame)	Include
	0	Include dummy bits	Include
	1	Dummy definition (1 = use Bit [3:0] as dummy byte count)	Include
	0001	Dummy definition.	Include
	(16-bit = Number of Frames)	Number of configuration data frames included in operand[6] For Example, CrossLink-NX A. No. of Actual Data Frames = 4072(Total ASR Size) – 32(MIB Frames) – 12 (TAP Frames)	Include
Frame (Data 0)	(Frame 0 Data)	*x* Data bits for Frame 0 (14 bits Parity + Data)[5]	Include: End
	(CRC[15..0])	16-bit CRC	CRC
	11111111	Dummy byte (8 bits of ones)	Start: Include
Frame (Data 1)	(Frame 1 Data)	*x* Data bits for Frame 1 (14 bits Parity + Data)[5]	Include: End
	(CRC[15..0])	16-bit CRC	CRC
	11111111	Dummy byte (8 bits of ones)	Start: Include
==========	==========	==========	==========

Frame	Contents[D7…D0]	Description	CRC
Frame (Data n-1)	(Frame n-1 Data)	*x* Data bits for Frame 0 (14 bits Parity + Data) [5]	Include: End
	(CRC[15..0])	16-bit CRC	CRC
	11111111	Dummy byte (8 bits of ones)	Start: Include
Frame (Write address for Loading TAP Frames)	10110100	LSC_WRITE_ADDRESS Command	Include
	0	CRC Comparison Flag (1 = yes)	Include
	0	CRC Comparison at End Flag (0 = Execute CRC comparison at the end of each data frame)	Include
	0	—	Include
	0	—	Include
	0000	—	Include
	0000…..000000	16-bit Command Information (23 zeroes)	Include
	000…00000	16-bit Zeroes	Include
	1000000000100000	16-bit address to load[6]	Include
Frame (Write Increment)	10000010	LSC_PROG_INCR command	Include
	1	CRC Comparison Flag (1 = yes)	Include
	0	CRC Comparison at End Flag (0 = Execute CRC comparison at the end of each data frame)	Include
	0	Include dummy bits	Include
	1	Dummy definition (1 = use Bit [3:0] as dummy byte count)	Include
	0001	Dummy definition	Include
	(16-bit = Frame Number)	Number of TAP frames[5]	Include
Frame (Data 0)	(Frame 0 Data)	*x* Data bits for Frame 0 (14 bits Parity + Data)[5]	Include: End
	(CRC[15..0])	16-bit CRC	CRC
	11111111	Dummy byte (8 bits of ones)	Start: Include
Frame (Data 1)	(Frame 1 Data)	*x* Data bits for Frame 1 (14 bits Parity + Data)[5]	Include: End
	(CRC[15..0])	16-bit CRC	CRC
	11111111	Dummy byte (8 bits of ones)	Start: Include
==========	==========	==========	==========
Frame (Data m-1)	(Frame m-1 Data)	*x* Data bits for Frame (m-1) (14 bits Parity + Data)[5]	Include: End
	(CRC[15..0])	16-bit CRC	CRC
	11111111	Dummy byte (8 bits of ones)	Start: Include
Frame (Power Control)	01010110	LSC_POWER_CTRL Command	Include
	0	CRC Comparison Flag (0 = no)	Include
	000….0000	Operand 22 zeroes	Include
	X	Enable the VCCPG Power Domain (1= yes)	Include
Frame (INIT Bus Address)	11110110	LSC_INIT_BUS_ADDR Command	Include
	0	CRC Comparison Flag (0 = no)	Include
	0000…..000000	23-bit Command Information (23 zeroes).	Include
	00	Default 2 zeroes (Reserved)	Include
	Data[29:28]	Bus Width[4] 0: 8-bit Hard IP 1: 10-bit Hard IP 2: 10-bit EBR /Large RAM 3: 32-bit Hard IP	Include
	Data [27:17]	ID CODE	Include
	Data[16:0]	Starting address	Include

Frame	Contents[D7…D0]	Description	CRC
Frame (Write data through INIT Bus)	01110010	LSC_INIT_BUS_WRITE Command	Include
	1	CRC Comparison Flag (1 = yes)	Include
	1	CRC Comparison at End Flag (1 = Execute CRC comparison at the end of all operands)	Include
	0	Exclude dummy bytes from bit stream.	Include
	1	Dummy definition (1 = use the next 4-bit Dummy Definition settings)	Include
	0000	Dummy definition.	Include
	(16-bit Size)	Number of 72-bit frames in the operand. One = 0000000000000001	Include
	(Data)	Data Frame	Include: End
	(CRC[15..0])	16-bit CRC	CRC
Frame (SED CRC) OPTIONAL FRAME	10100010	LSC_PROG_SED_CRC Command	Start : Include
	0	CRC Comparison Flag (0 = no)	Include
	0000…..000000	23-bit Command Information (23 zeroes)	Include
	(CRC[31..0])	32-bit SED CRC	Include
Frame (Program Security) OPTIONAL FRAME	11001110	ISC_PROGRAM_SECURITY Command	Include
	0	CRC Comparison Flag (0 = no)	Include
	0000…..000000	23-bit Command Information (23 zeroes)	Include
Frame (USERCODE)	11000010	ISC_PROGRAM_USRCODE Command	Include
	1	CRC Comparison Flag (1 = yes)	Include
	0000…..000000	23-bit Command Information (23 zeroes)	Include
	(U[31..0])	32-bit User code Data	Include: End
	(CRC[15..0])	16-bit CRC	CRC
Frame (Program Done)	01011110	ISC_PROGRAM_DONE	Exclude
	0	CRC Comparison Flag (0 = no)	Exclude
	00000….0000	Operand 21 zeroes	Exclude
	XX	Behavior 00 = No overload(default) , 01 = Serial System Configuration Daisy Chain Support in Bit stream , 10 = Parallel System Configuration Daisy Chain Support in bit stream, 11 = No overload(default)	Exclude
Frame (DUMMY)	1111…1111	4 bytes DUMMY command (all ones)	Exclude

Notes:

1. Data is byte bounded and must be padded with ones (1). For example, a 6-bit data frame of b111010 is written in the bitstream as b11111010 to make it byte bounded.
2. If SED option is not chosen, place command DUMMY (NOOP) (32 bits of 1) on the command space. NOOP command is not included in CRC calculation. This is necessary to make the file size consistent with or without security option chosen. It must be replaced with the LSC_PROG_SED_CRC frame by Bitgen if the user implements SED. Include in CRC calculation if LSC_PROG_SED_CRC command. Do not include in CRC calculation if NOOP.
3. The CRC must be flipped so that CRC[0] is the LSB and CRC[15] is the MSB. For example, CRC = 0x0823 (CRC[15]…CRC[0]) in the bit file above is b1100010000010000. Refer to Appendix Generic Configuration Specification for CRC calculation details.
4. Insert a Write EBR Frame for each initialized EBR, or for each group of EBR Frames that is initialized to the same value. Omit if there is no EBR initialization data.
5. By default, this frame must be the Verify ID command. Bitgen shall have a switch to allow the Verify ID frame to be replaced with NOOP (all 1s). Include in CRC calculation if Verify ID command. Do not include in CRC calculation if NOOP.
6. If security option is not chosen, place command NOOP (32 bits of 1) on the command space. NOOP command is not included in CRC calculation. This is necessary to make the file size consistent with or without the security option chosen. Include in CRC calculation if Program Security command. Do not include in CRC calculation if NOOP.
7. The Control Register 0 must be set to the SW Default value, setting Control Register 0 bits [31..30] = [0..0].

Table B.3. Nexus Uncompressed Bitstream Format – With Early I/O Release

Frame	Contents[D7…D0]	Description	CRC
LSCC signature	4C534343	32 bits of the LSCC Signature (LSCC = 0x4C534343)	None
Comments	(Comment String)	ASCII Comment (Argument) String and Terminator	
Header	(**LSB**) 1111111111111111	First 16 bits of the 32-bit Preamble.	None
	1011110110110011	Second 16 bits of the 32-bit Preamble (0xFFFFBDB3).	
	11111111…11111111	32-bit Dummy	
Frame (Reset CRC)	00111011	LSC_RESET_CRC Command	Exclude
	0	CRC Comparison Flag (0 = no).	Exclude
	0000…..000000	23-bit Command Information (23 zeroes).	Exclude
Dummy Frame	1111….111111	Dummy byte (32 bits of ones)	Exclude
Frame (Verify ID) OPTIONAL DATA But Mandatory Frame	11100010	LSC_VERIFY_ID command (if the Verify ID data is not there, then Frame contains all 1s)	Start: Include
	0	CRC Comparison Flag (0 = no) for Verify ID command	Include
	0000…..000000	23-bit Command Information (23 zeroes).	Include
	(Device ID[31..0])	32-bit Device Id	Include
Frame (Control Register 0)	00100010	LSC_PROG_CNTRL0 command	Include
	0	CRC Comparison Flag (0 = no)	Include
	0000…..000000	23-bit Command Information (23 zeroes).	Include
	(CtlReg0[31..0])	Control Register 0 Data definition[2]	Include
Frame (Control Register 1) OPTIONAL FRAME	00100011	LSC_PROG_CNTRL1 command	Include
	0	CRC Comparison Flag (0 = no)	Include
	0000…..000000	23-bit Command Information (23 zeroes).	Include
	CtlReg1[31..0])	Control Register 1 Data definition[3]	Include
Frame (Write Address for Loading MIB Frames)	10110100	LSC_WRITE_ADDRESS Command	Include
	0	CRC Comparison Flag (1 = yes).	Include
	0	CRC Comparison at End Flag (0 = Execute CRC comparison at the end of each data frame)	Include
	0	—	Include
	0	—	Include
	0000	—	Include
	0000…..000000	16-bit Command Information (23 zeroes).	Include
	000…00000	16-bit zeroes	Include
	100000000000000	16-bit address to load[6]	Include
Frame (Auto Increment For next 32 Frames)	10000010	LSC_PROG_INCR command	Include
	1	CRC Comparison Flag (1 = yes)	Include
	0	CRC Comparison at End Flag (0 = Execute CRC comparison at the end of each data frame)	Include
	0	Include dummy bits.	Include
	1	Dummy definition (1 = use Bit [3:0] as dummy byte count)	Include
	0001	Dummy definition.	Include
	00000000000100000	Number of MIB Frames Always 32 (16-bit Frame Size[5, 6]	Include
Frame (Data 0)	(Frame 0 Data)	*x* Data bits for Frame 0 (14 bits Parity + Data)[5]	Include: End
	(CRC[15..0])	16-bit CRC	CRC
	11111111	Dummy byte (8 bits of ones)	Start: Include
Frame (Data 1)	(Frame 1 Data)	*x* Data bits for Frame 1(14 bits Parity + Data)[5]	Include: End
	(CRC[15..0])	16-bit CRC	CRC

Frame	Contents[D7…D0]	Description	CRC
	11111111	Dummy byte (8 bits of ones)	Start: Include
==========	==========	==========	==========
Frame (Data 31)	(Frame 31 Data)	*x* Data bits for Frame 31(14 bits Parity + Data)[5]	Include: End
	(CRC[15..0])	16-bit CRC	CRC
	11111111	Dummy byte (8 bits of ones)	Start: Include
Frame (Release I/O) OPTIONAL FRAME (BOTH BANK A and B)	01010100	LSC_ IO_CONTROL Command	Include
	0	CRC Comparison Flag (0 = no)	Include
	00000	5 zeroes	Include
	X	release I/O bank B	Include
	X	release I/O bank A	Include
	000….0000	all 16 zeroes in Operand	
Dummy Frame	0xFFFFF…FFFF	1000 Bytes dummy	Include
Frame (Reset Address)	01000110	LSC_INIT_ADDRESS Command	Include
	0	CRC Comparison Flag (0 = no)	Include
	0000…..000000	23-bit Command Information (23 zeroes).	Include
Frame (Write Increment)	10000010	LSC_PROG_INCR command	Include
	1	CRC Comparison Flag (1 = yes)	Include
	0	CRC Comparison at End Flag (0 = Execute CRC comparison at the end of each data frame)	Include
	0	Include dummy bits.	Include
	1	Dummy definition (1 = use Bit [3:0] as dummy byte count)	Include
	0001	Dummy definition.	Include
	(16-bit = Number of Frames)	Number of configuration data frames included in operand[5] For Example, CrossLink-NX A. No. of Actual Data Frames = 4096(Total ASR Size) – 24(MIB Frames) – 12 (TAP Frames)	Include
Frame (Data 0)	(Frame 0 Data)	*x* Data bits for Frame 0 (14 bits Parity + Data)[5]	Include: End
	(CRC[15..0])	16-bit CRC	CRC
	11111111	Dummy byte (8 bits of ones)	Start: Include
Frame (Data 1)	(Frame 1 Data)	*x* Data bits for Frame 1 (14 bits Parity + Data)[5]	Include: End
	(CRC[15..0])	16-bit CRC	CRC
	11111111	Dummy byte (8 bits of ones)	Start: Include
==========	==========	==========	==========
Frame (Data n-1)	(Frame n-1 Data)	*x* Data bits for Frame 0 (14 bits Parity + Data)[5]	Include: End
	(CRC[15..0])	16-bit CRC	CRC
	11111111	Dummy byte (8 bits of ones)	Start: Include
Frame (Write address for Loading TAP Frames)	10110100	LSC_WRITE_ADDRESS Command	Include
	0	CRC Comparison Flag (1 = yes).	Include
	0	CRC Comparison at End Flag (0 = Execute CRC comparison at the end of each data frame)	Include
	0	—	Include
	0	—	Include
	0000	—	Include
	0000…..000000	16-bit Command Information (23 zeroes).	Include
	000…00000	16-bit Zeroes	Include

Frame	Contents[D7…D0]	Description	CRC
	1000000000100000	16-bit address to load	Include
Frame (Write Increment)	10000010	LSC_PROG_INCR command	Include
	1	CRC Comparison Flag (1 = yes)	Include
	0	CRC Comparison at End Flag (0 = Execute CRC comparison at the end of each data frame)	Include
	0	Include dummy bits.	Include
	1	Dummy definition (1 = use Bit [3:0] as dummy byte count)	Include
	0001	Dummy definition.	Include
	(16-bit = Frame Number)	Number of TAP frames[5]	Include
Frame (Data 0)	(Frame 0 Data)	*x* Data bits for Frame 0 (14 bits Parity + Data)[5]	Include: End
	(CRC[15..0])	16-bit CRC	CRC
	11111111	Dummy byte (8 bits of ones)	Start: Include
Frame (Data 1)	(Frame 1 Data)	*x* Data bits for Frame 1 (14 bits Parity + Data)[5]	Include: End
	(CRC[15..0])	16-bit CRC	CRC
	11111111	Dummy byte (8 bits of ones)	Start: Include
==========	==========	==========	==========
Frame (Data m-1)	(Frame m-1 Data)	*x* Data bits for Frame (m-1) (14 bits Parity + Data)[5]	Include: End
	(CRC[15..0])	16-bit CRC	CRC
	11111111	Dummy byte (8 bits of ones)	Start: Include
Frame (Power Control)	01010110	LSC_POWER_CTRL Command	Include
	0	CRC Comparison Flag (0 = no)	Include
	000….0000	Operand 22 zeroes	Include
	X	Enable the VCCPG Power Domain (1= yes)	Include
Frame (INIT Bus Address)	11110110	LSC_INIT_BUS_ADDR Command	Include
	0	CRC Comparison Flag (0 = no)	Include
	0000…..000000	23-bit Command Information (23 zeroes).	Include
	00	Default 2 zeroes (Reserved)	Include
	Data[29:28]	Bus Width[4] 0: 8-bit Hard IP 1: 10-bit Hard IP 2: 10-bit EBR /Large RAM 3: 32-bit Hard IP	Include
	Data [27:17]	ID CODE	Include
	Data[16:0]	Starting address	Include
Frame (Write data through INIT Bus)	01110010	LSC_INIT_BUS_WRITE Command	Include
	1	CRC Comparison Flag (1 = yes)	Include
	1	CRC Comparison at End Flag (1 = Execute CRC comparison at the end of all operands).	Include
	0	Exclude dummy bytes from bit stream.	Include
	1	Dummy definition (1 = use the next 4-bit Dummy Definition settings).	Include
	0000	Dummy definition.	Include
	(16-bit Size)	Number of 72-bit frames in the operand. One = 0000000000000001.	Include
	(Data)	Data Frame	Include: End
	(CRC[15..0])	16-bit CRC	CRC
Frame (SED CRC) OPTIONAL FRAME	10100010	LSC_PROG_SED_CRC Command	Start : Include

Frame	Contents[D7…D0]	Description	CRC
	0	CRC Comparison Flag (0 = no)	Include
	0000…..000000	23-bit Command Information (23 zeroes).	Include
	(CRC[31..0])	32-bit SED CRC	Include
Frame (Program Security) OPTIONAL FRAME	11001110	ISC_PROGRAM_SECURITY Command	Include
	0	CRC Comparison Flag (0 = no)	Include
	0000…..000000	23-bit Command Information (23 zeroes).	Include
Frame (USERCODE)	11000010	ISC_PROGRAM_USRCODE Command.	Include
	1	CRC Comparison Flag (1 = yes)	Include
	0000…..000000	23-bit Command Information (23 zeroes).	Include
	(U[31..0])	32-bit User code Data.	Include: End
	(CRC[15..0])	16-bit CRC.	CRC
Frame (Program Done)	01011110	ISC_PROGRAM_DONE.	Exclude
	0	CRC Comparison Flag (0 = no).	Exclude
	00000….0000	Operand 21 zeroes.	Exclude
	XX	Behavior 00 = No overload(default) , 01 = Serial System Configuration Daisy Chain Support in Bit stream , 10 = Parallel System Configuration Daisy Chain Support in bit stream, 11 = No overload(default)	Exclude
Frame (DUMMY)	1111…1111	4 bytes DUMMY command (all ones).	Exclude

Notes:

1. Data is byte bounded and must be padded with ones (1). For example, a 6-bit data frame of b111010 is written in the bitstream as b11111010 to make it byte bounded.
2. If SED option is not chosen, place command DUMMY (NOOP) (32 bits of 1) on the command space. NOOP command is not included in CRC calculation. This is necessary to make the file size consistent with or without the security option chosen. It must be replaced with the LSC_PROG_SED_CRC frame by Bitgen if the user implements SED. Include in CRC calculation if LSC_PROG_SED_CRC command. Do not include in CRC calculation if NOOP.
3. The CRC must be flipped so that CRC[0] is the LSB and CRC[15] is the MSB. For example, CRC = 0x0823 (CRC[15]…CRC[0]) in the bit file above is b1100010000010000. Refer to Appendix Generic Configuration Specification for CRC calculation details.
4. Insert a Write EBR Frame for each initialized EBR, or for each group of EBR Frames that is initialized to the same value. Omit if there is no EBR initialization data.
5. By default, this frame must be the Verify ID command. Bitgen shall have a switch to allow the Verify ID frame to be replaced with NOOP (all 1s). Include in CRC calculation if Verify ID command. Do not include in CRC calculation if NOOP.
6. If security option is not chosen, place command NOOP (32 bits of 1) on the command space. NOOP command is not included in CRC calculation. This is necessary to make the file size consistent with or without security option chosen. Include in CRC calculation if Program Security command. Do not include in CRC calculation if NOOP.
7. The Control Register 0 must be set to the SW Default value, setting Control Register 0 bits [31..30] = [0..0].

B.2. Read Back Bitstream Format

If required, the Lattice Radiant software generates a binary read back file. The read back file contains an ASCII comment string, device ID, control register 0 data, frame data, dummy bits, LUT, EBR, and usercode data. It does not contain a header, commands, or CRC. The data is ready to be shifted into the devices as required by the programming flow. The data is listed from MSB to LSB, where the MSB is the first bit shifted into TDI and the LSB the last bit. The MSB is also the first bit shifted out of TDO. The data for each frame is byte bounded. The byte must be padded with all ones (1). For example, a 6-bit data frame of b111010 would be written in the readback file as b11111010. The format is described in Table B.4.

Table B.4. Nexus Read Back File Format

Frame	Contents (D0..D7)	Description
Comments	(Comment String)	ASCII Comment (Argument) String.
Comment Terminator	11111110	Read Back File Comment Terminator = 0xFE (binary file (.rbk) only).
Device ID	(ID[0..31])	Expected 32-bit Device ID.
Control Register 0	(CtlReg0[0..31])	Control Register 0 Data.
Device Specific Padding	1111…1111	Terminator bytes (all 1s). See Table B.5 for number of bytes.

Frame	Contents (D0..D7)	Description
Frame 0	1111…1111	Dummy (Stop) bytes (all 1s). See Table B.5 for number of bytes.
	(Frame 0 Data)	Data for Frame 0 (byte bounded, padded with zeros).
Frame 1	1111…1111	Dummy (Stop) bytes (all 1s). See Table B.5 for number of bytes.
	(Frame 1 Data)	Data for Frame 1 (byte bounded, padded with zeros).
•	•	•
•	•	•
•	•	•
Frame n-1	1111…1111	Dummy (Stop) bytes (all 1s). See Table B.5 for number of bytes.
	(Frame n-1 Data)	Data for Frame n-1 (byte bounded, padded with zeros)
Usercode	(U[0..31])	32-bit Usercode Data.

B.3. Nexus Device Specific

Table B.5. Nexus Device Specifics

Device Name	Configuration Frames (n)	Byte Bound Padding Bits	Actual Bits per Frame (w)	Parity Bits (ECC)	CRC Bits	Dummy	Total Data Bits per Frame (x)
LIFCL-17 LFD2NX-9 LFD2NX-17	7900	0	338	14	16	8	376
LIFCL-33	5344	0	986	14	16	8	1024
LIFCL-33U	5344	0	986	14	16	8	1024
LIFCL-40 LFD2NX-28 LFD2NX-40	9172	4	662	14	16	8	704
LFCPNX-50	16822	4	878	14	16	8	920
LFCPNX-100	16822	4	878	14	16	8	920
LFD2NX-15	6622	4	662	14	16	8	704
LFD2NX-25	6622	4	662	14	16	8	704
LFD2NX-35	10444	0	986	14	16	8	1024
LFD2NX-65	10444	0	986	14	16	8	1024

Table B.6. Nexus Device ID

Device Family	Device Name	32-bit IDCODE
CrossLink-NX	LIFCL-17	0x010F0043
	LIFCL-40	0x110F1043
CrossLink-NX-33	LIFCL-33	0x010FA043
CrossLinkU-NX	LIFCL-33U	0x010FB043
Certus-NX	LFD2NX-9	0x710F0043
	LFD2NX-15	0x190F1043
	LFD2NX-17	0x310F0043
	LFD2NX-25	0x090F1043
	LFD2NX-28	0x710F1043
	LFD2NX-35	0x790F1043
	LFD2NX-40	0x310F1043
	LFD2NX-65	0x390F1043
CertusPro-NX	LFCPNX-50	0x010F2043
	LFCPNX-100	0x010F4043

Appendix C. Modifying the Nexus Feature Row

The Feature Row contains design-specific information for non-volatile device operations and behaviors. For example, you can specify the device's I^2C address. Radiant Programmer 2024.1 and onwards offer the flexibility to program the Feature Row OTP EFUSE or pseudo EFUSE, which is also referred to as the shadow register. Note that in this document, the terms pseudo EFUSE and shadow register are used interchangeably.

Programming the OTP EFUSE is non-volatile and non-reversible. Programming the pseudo EFUSE or shadow register is volatile, which means that the settings are cleared after a power cycle or REFRESH event. To avoid clearing the shadow register after a REFRESH event, set the No CDM bit in CR0 before triggering the REFRESH event. Refer to the Control Register 0 (CR0) section for information on the No CDM bit.

C.1. Breakdown of Feature Row Bits

The Feature Row has a total of 144 bits:

- 32-bit NV CR1 bits
- 96-bit FEATURE bits
- 16-bit FEABITS

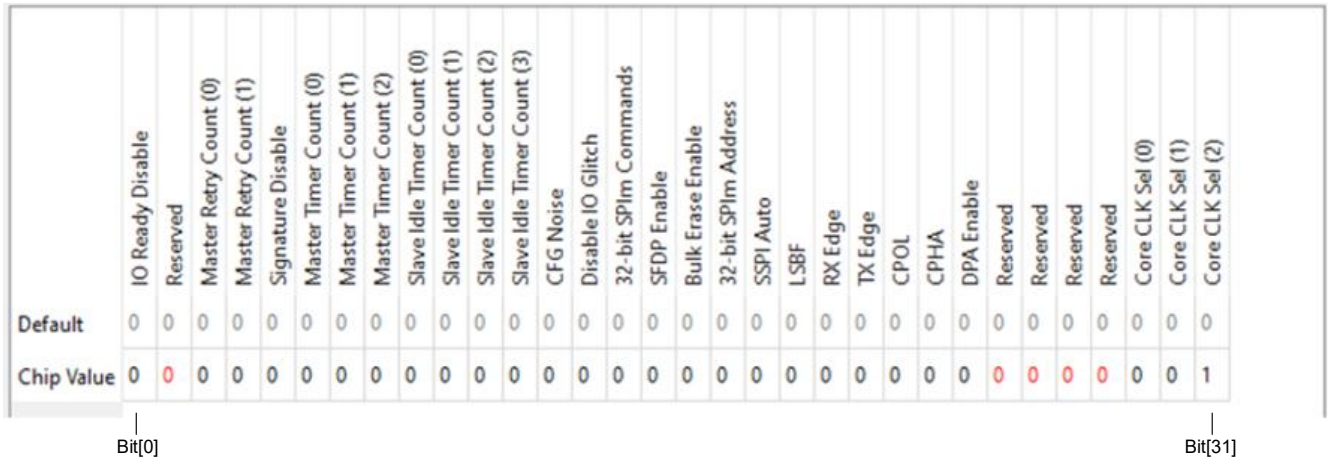


Figure C.1. 32-bit NV CR1 Bits

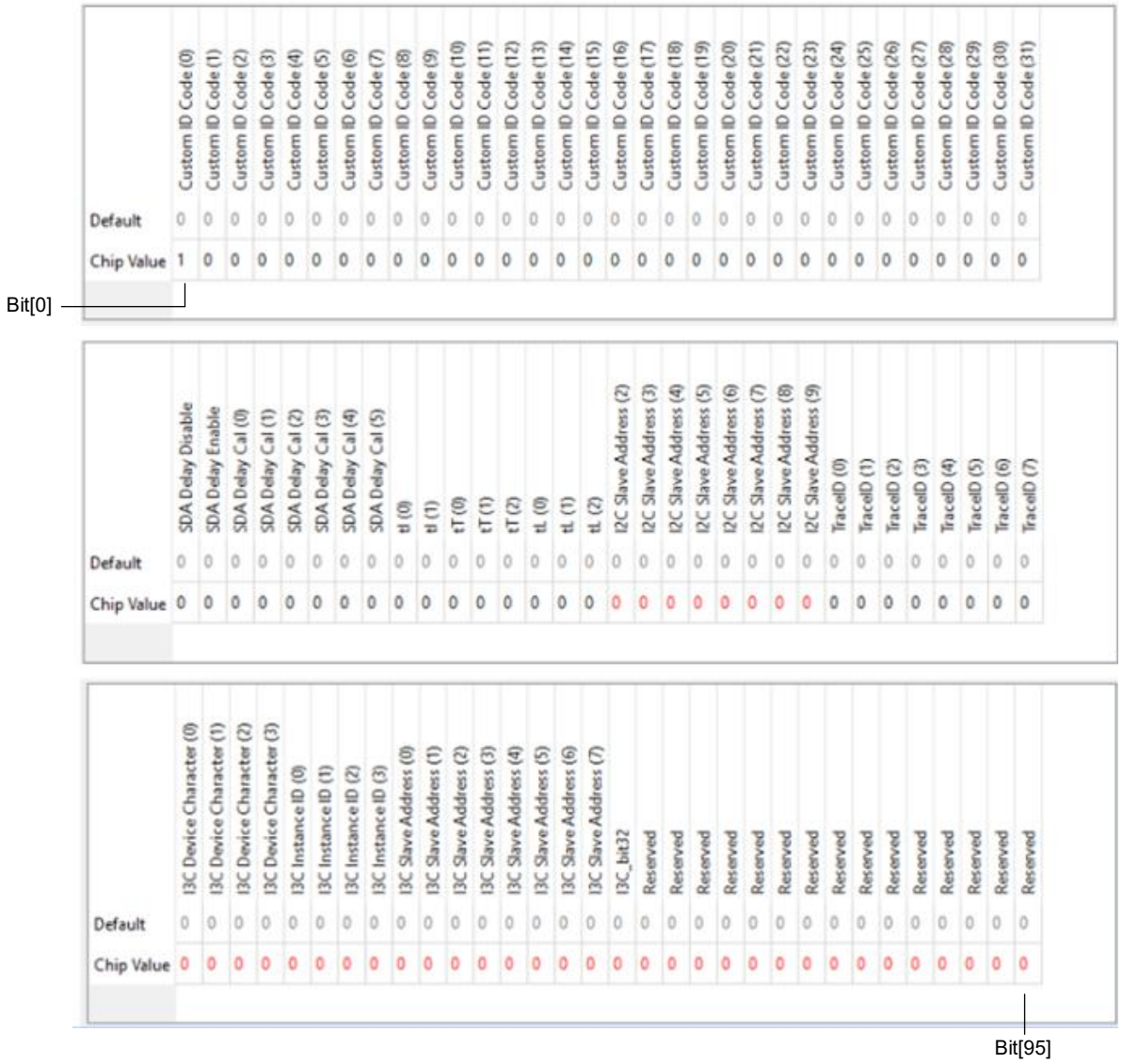


Figure C.2. 96-bit FEATURE Bits

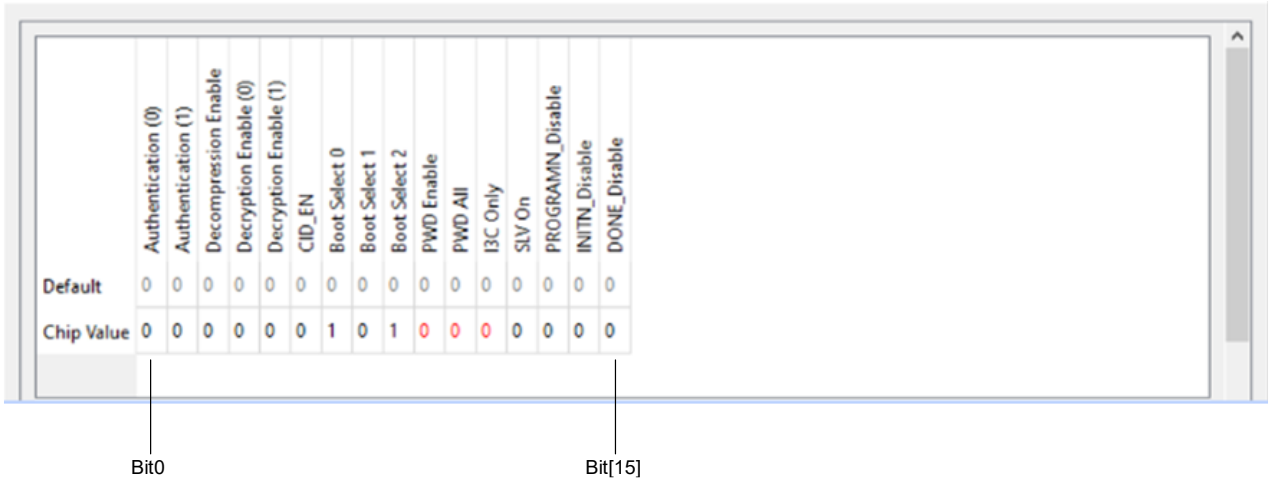


Figure C.3. 16-bit FEABITS

C.2. Modifying the Feature Row Programming File

You can use the Feature Row Editor available in the *Programming File Utility* to enable or disable silicon features in the Lattice FPGA devices to modify the Feature Row fuse settings in the data file (*.fea*).

The Programming File Utility tool can be accessed by using either of the following method:

- The *fileutility.exe* is located inside the *programmer* folder from the installation path of the Lattice Radiant software or a Lattice Radiant standalone programmer.
- Accessing through the Lattice Radiant Programmer's *Tools* menu.

Before modifying the feature row, you must create a project first in the Lattice Radiant software and run it through *Export Files*. The *.fea* file is generated inside the *Implementation* folder.

The next step is to edit the feature row bits.

To edit the feature row bits, follow these steps:

1. From the Programming File Utility, select **Tools > Feature Row Editor**. The Feature Row Editor dialog box opens.

   Using the **Browse** button, select the data (*.fea*) file to modify. Click **Read** to display the current data file settings. The software displays the default values in the top row. The second row shows the feature row bits in black that can be modified. The feature row bits that cannot be modified are shown in red.
2. Select the desired bit cell value to change and then toggle the value from 1 to 0, or from 0 to 1.
3. Click **Save** to overwrite the existing data file (*.fea*).
4. To create a new data file (*.fea*) with a different name, select **Save As**.

C.3. Modifying the Feature Row of a Programmed Device

C.3.1. Modifying the Feature Row Using the .fea Programming File

To modify the feature row using the .fea file, perform the following steps:

1. Run the Radiant Programmer.
2. Using the Run menu, select **Scan Device**.
3. Once the scanning of the device is done, select **Edit > Device Properties**.
4. Change the Device Operation using the following settings:
   a. **Target Memory** – **Non Volatile Configuration Memory**
   b. **Port Interface** – **JTAG** (or any of the available interfaces: **Slave SPI**, **I²C**, or **I3C Bridge**)
   c. **Access Mode:**
      - **Feature Rows Programming** – This setting targets OTP EFUSE, which can only be modified once.
      - **Feature Rows Pseudo Programming** – This setting targets pseudo EFUSE or shadow register. The settings are cleared after a power cycle or REFRESH event if the No CDM bit in CR0 is not set.
   d. **Operation** – **Program Feature Row**

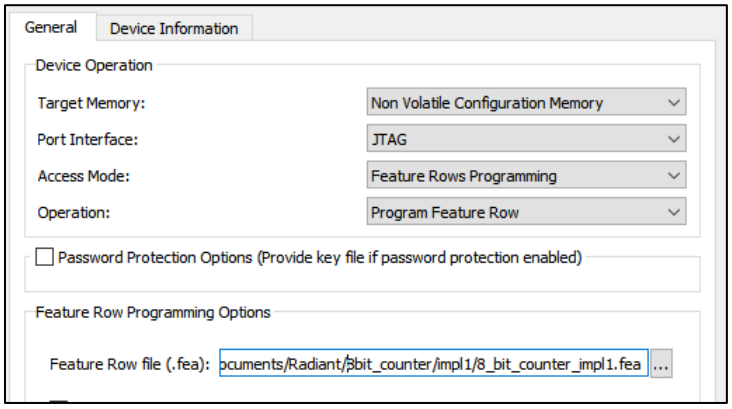


Figure C.4. Feature Row Programming Setup

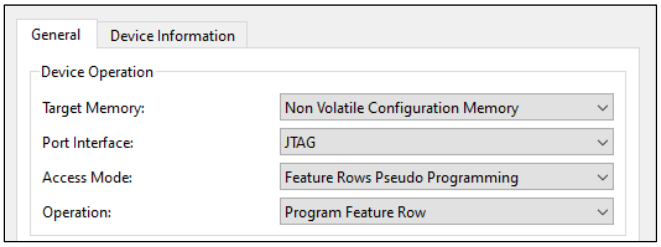


Figure C.5. Feature Row Pseudo Programming Setup

5. Under **Feature Row Programming Options**, select the .fea file from the Implementation folder.
6. Click **OK**, and then select **Run > Program Device**.

   Note: The **Feature Rows Programming** setting enables programming of the OTP EFUSE, which can only be modified once. OTP EFUSE programming is non-reversible.

C.3.2. Modifying the Feature Row without Using the .fea Programming File

Updating the feature row can be also done without the need of the Lattice Radiant project or generating the *.fea* file.

To update the feature row, perform the following steps:

1. In the Radiant Programmer, you must select the device feature row to modify.
2. Choose **Edit > Device Properties**.
3. Change the Device Properties using the following settings:
   a. **Target Memory** – **Non Volatile Configuration Memory**
   b. **Port Interface** – **JTAG** (or any of the available interfaces: **Slave SPI**, **I²C**, or **I3C Bridge**)
   c. **Access Mode**:
      - **Feature Rows Programming** – This setting targets OTP EFUSE, which can only be modified once.
      - **Feature Rows Pseudo Programming** – This setting targets pseudo EFUSE or shadow register. The settings are cleared after a power cycle or REFRESH event if the No CDM bit in CR0 is not set.
   d. **Operation** – **Update Feature Row**
4. Click **OK** to close the Device Properties dialog box.
5. Select **Design > Program**. The Feature Row dialog box displays.
6. Edit the fields in the dialog box as desired, and then click **Program**. The dialog box prompts you to overwrite the selected feature row bit or bits. If you select **Yes**, the device's feature row is programmed.

Note: Update Feature Row can only accept per-bit modification from 0 to 1. The modification is only applied to the values displayed in black.

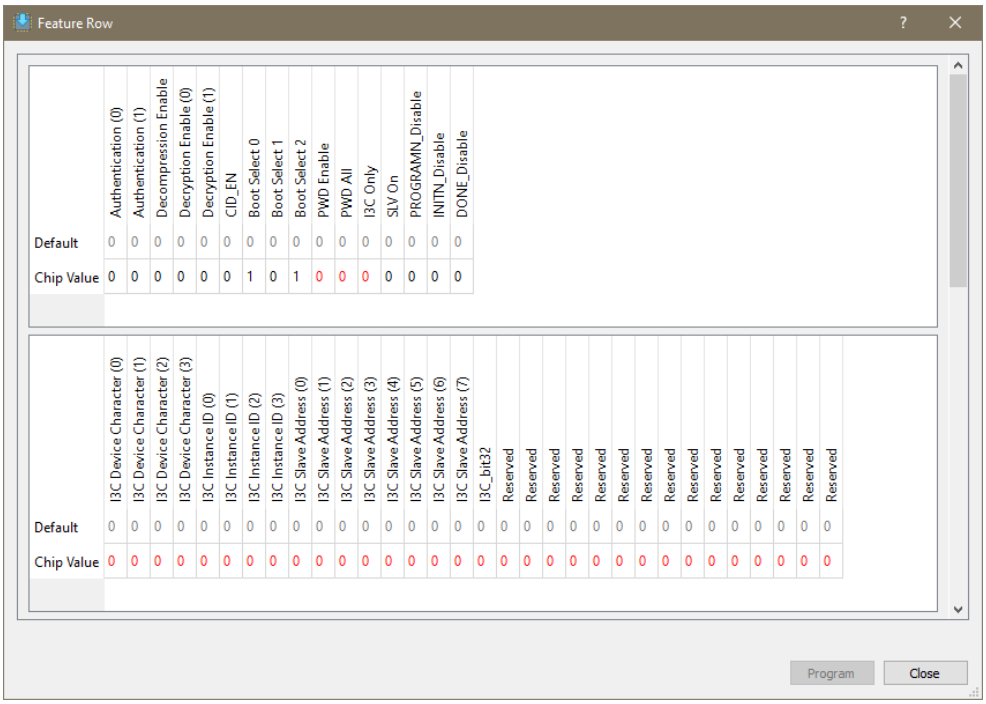


Figure C.6. Update Feature Row User Interface

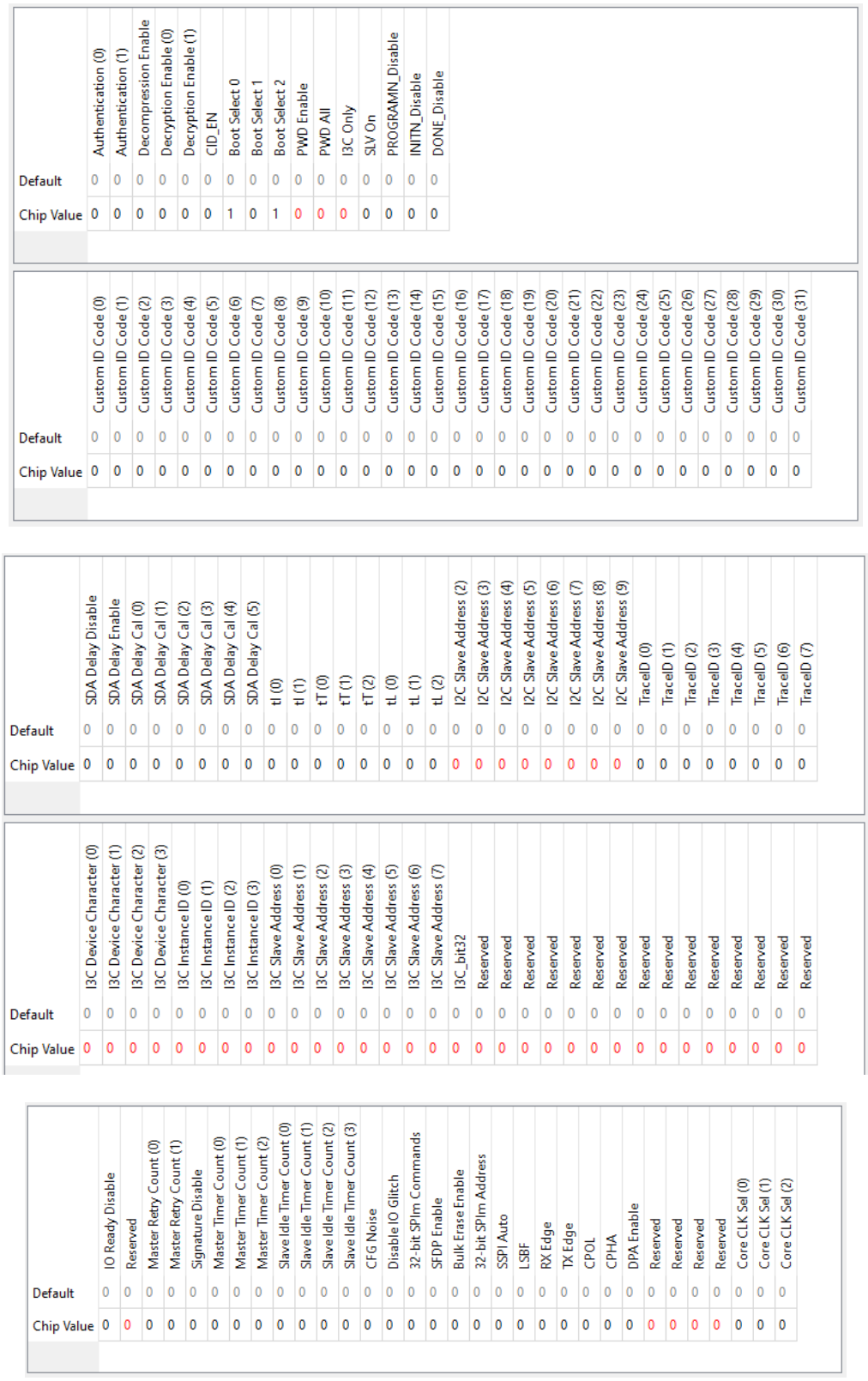


Figure C.7. Feature Row Bit Examples

C.4. Programming No CDM Bit in CR0

Shadow register programming is volatile. Settings are cleared after a power cycle or REFRESH event. To retain the shadow register settings, you need to set the No CDM bit in CR0 to 1 as follows:

1. Select **Static Random Access Memory** from the **Target Memory** drop-down menu.
2. In the **Port Interface** drop-down menu, select **JTAG**, **Slave SPI**, or **I2C**.
3. Select **Program Control Register 0** from the **Operation** drop-down menu.
4. Double-click on the chip value under the *No CDM* column. The chip value changes from 0 to 1.
5. Click **Program**.

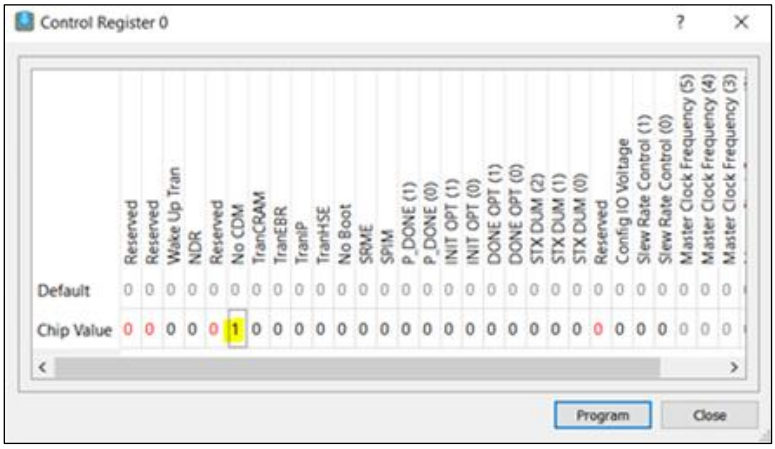


Figure C.8. Control Register 0 Programming for No CDM Bit

C.5. Comparison of Feature Row Programming File (.fea)

Figure C.9 shows the 1:1 bit alignment of the .fea file (generated by the Lattice Radiant software) with regards to the feature row readout through the Radiant Programmer. For the individual bit definition, refer to the following:

- 32-bit NV CR1 bits – refer to Control Register 1 (CR1) section.
- 96-bit FEATURE bits – refer to Table 6.24.
- 16-bit FEABITS – refer to Table 6.25.

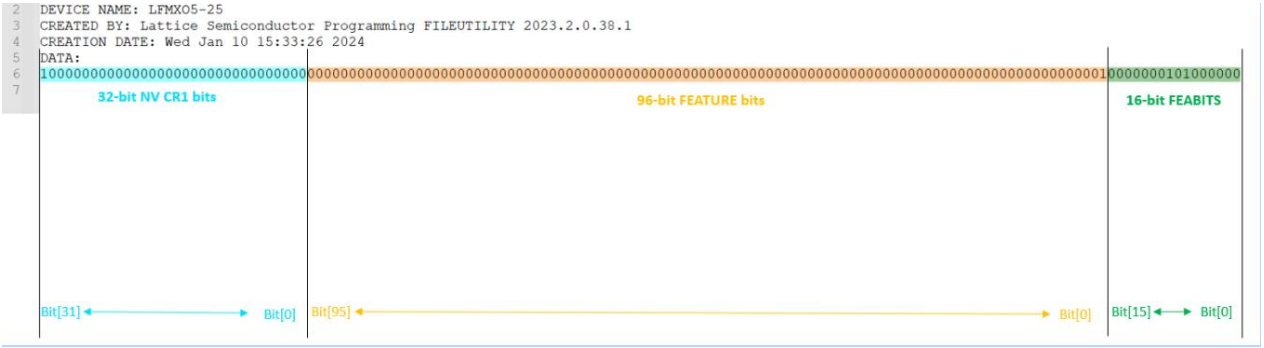


Figure C.9. .fea File Sample

Appendix D. Configuration Access from User Logic

Nexus configuration logic provides an LMMI interface to allow user logic residing inside the FPGA fabric to access the device configuration (CFG) functionalities. To achieve this, you must instantiate the CONFIG_LMMI primitive in the design. The following are the supported CONFIG_LMMI primitives:

- CONFIG_LMMI
- CONFIG_LMMIA
- CONFIG_LMMIB
- CONFIG_LMMIE
- CONFIG_LMMIF
- CONFIG_LMMIG
- CONFIG_LMMIH

For more details on the CONFIG_LMMI primitives supported by a device family, refer to the corresponding primitive library in the Lattice Radiant Software Help.

To make sure the clock for the configuration engine is enabled, you must instantiate the OSC or OSC for CRE IP in the design as well. Refer to the respective IP user guides from the Radiant IP catalog for more information. Be aware that when utilizing the CONFIG_LMMI to access configuration logic, all operations that may alter the FPGA fabric setting or logic are prohibited.

D.1. CONFIG_LMMI and CONFIG_LMMI Primitive

The pin diagram of CONFIG_LMMI primitive is shown in Figure D.1. The description of the I/O and parameters are shown in Table D.1 and Table D.2.

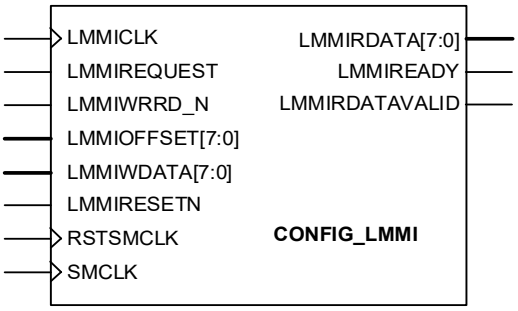


Figure D.1. CONFIG_LMMI Primitive Pin Diagram

Table D.1. CONFIG_LMMI Primitive Input/Output Ports1

Pin Name	Direction	Description
LMMICLK[2]	In	Clock for LMMI Interface
LMMIREQUEST	In	LMMI Start transaction
LMMIWRRD_N	In	LMMI Write/Read control; 1=Write; 0=Read
LMMIOFFSET [7:0]	In	LMMI Register Offset
LMMIWDATA [7:0]	In	LMMI Write Data
LMMIRESETN	In	LMMI Interface reset
RSTSMCLK	In	Reset SMCLK
SMCLK	In	Configuration Block Clock
LMMIRDATA [7:0]	Out	LMMI Read Data
LMMIREADY	Out	LMMI Ready Signal
LMMIRDATAVALID	Out	LMMI Read transaction is complete and LMMIRDATA contains valid data

Notes:

1. There is no simulation model supported for CONFIG_LMMI primitive.
2. The maximum clock frequency usable for LMMICLK is 50 MHz.

Table D.2. CONFIG_LMMI Primitive Parameters[1]

Pin Name	Value	Description
LMMI_EN	"DIS" (default) "EN"	Enable/disable the LMMI interface. Must be set to "EN".

Note:

1. There is no simulation model supported for CONFIG_LMMI primitive.

D.2. CONFIG_LMMI Primitive and OSC/OSC for CRE IP Connection and Instantiation

In the user design, the CONFIG_LMMI primitive and OSC IP should be connected as shown in Figure D.2.

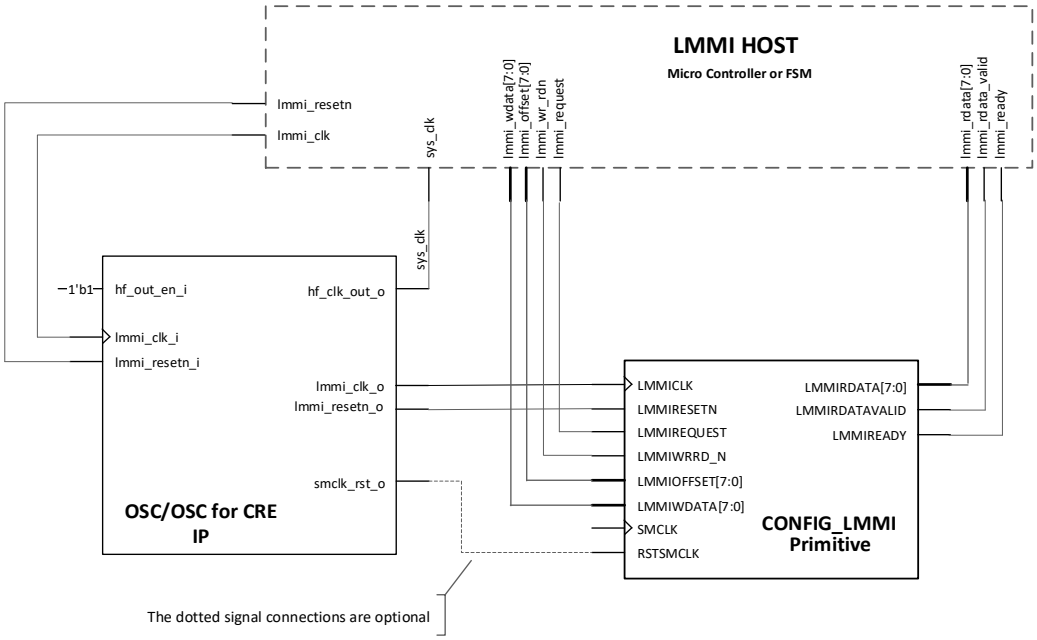


Figure D.2. CONFIG_LMMI Primitive and OSC/OSC for CRE IP Connection (hf_clk_out_o as Clock Source)

The following shows a sample RTL design for instantiating the CONFIG_LMMI primitive and OSC IP.

```
wire       sys_clk;
wire       LMMI_LRST_N;
wire       smclk_rst;
wire       smclk;

wire       lmmi_clk;
wire       lmmi_resetn;
wire       lmmi_request;
wire       lmmi_wr_rdn;
wire [7:0] lmmi_offset;
wire [7:0] lmmi_wdata;
wire [7:0] lmmi_rdata;
wire       lmmi_ready;
wire       lmmi_rdata_valid;

// ********************************************************************************************

Implement LMMI Host Logic here.

// ********************************************************************************************
```

```
osc_ip osc_ip_inst(
    .hf_out_en_i(1'b1),             // I -> 0 = disable HF clock, 1 = enable HF clock
    .lmmi_clk_i(sys_clk),           // I -> clock source for CONFIG_LMMI primitive
    .lmmi_resetn_i(LMMI_LRST_N),    // I -> Reset CONFIG_LMMI primitive
    .hf_clk_out_o(sys_clk),         // O -> HF clock out
    .smclk_rst_o(),                 // O -> CFG Clock (Hardware inferred)
    .lmmi_clk_o(lmmi_clk),          // O -> connect to CONFIG_LMMI priitive clock
    .lmmi_resetn_o(lmmi_resetn));   // O -> connect to CONFIG_LMMI primitive reset


CONFIG_LMMI #(.LMMI_EN ("EN"))
config_lmmi (// Inputs
             .LMMICLK       (lmmi_clk),             // I -> Clock for LMMI
             .LMMIREQUEST   (lmmi_request),         // I -> Start Transaction
             .LMMIWRRD_N    (lmmi_wr_rdn),          // I -> 1 = Write ; 0 = Read
             .LMMIOFFSET    (lmmi_offset),          // I -> Register Offset
             .LMMIWDATA     (lmmi_wdata),           // I -> Write Data
             .LMMIRESETN    (lmmi_resetn),          // I -> Reset the LMMI
             .RSTSMCLK      (smclk_rst),              // I -> Reset SMCLK (Hardware inferred)
             .SMCLK         (smclk),         // I -> CFG Clock (Hardware inferred)
             // Outputs
             .LMMIRDATA     (lmmi_rdata),           // O -> Read data
             .LMMIREADY     (lmmi_ready),           // O -> Ready Flag
             .LMMIRDATAVALID (lmmi_rdata_valid));  // O -> Read complete. LMMIRDATA is valid
```

Alternatively, you can choose not to use hf_clk_out_o as the clock source to drive the LMMI host and CONFIG_LMMI primitive. This allows hf_clk_out_o to be used to drive other components for a more flexible design implementation.

Figure D.3 shows an alternative connection for the CONFIG_LMMI primitive and OSC IP.

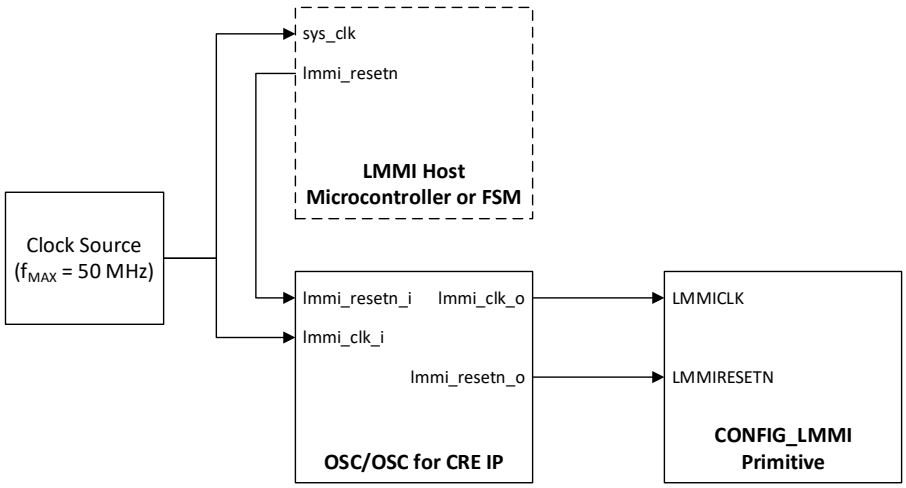


Figure D.3. CONFIG_LMMI Primitive and OSC/OSC for CRE IP Connection (Alternative Clock Source)

Apart from hf_clk_out_o, you can use any clock source to drive the LMMICLK port of the CONFIG_LMMI primitive provided the following conditions are met:

- f_{MAX} is 50 MHz.
- OSC/OSC for CRE IP and CONFIG_LMMI are connected as follows:
  - lmmi_clk_o to LMMICLK
  - lmmi_resetn_o to LMMIRESETN

Note: lmmi_clk_o is a passthrough of lmmi_clk_i. lmmi_resetn_o is a passthrough of lmmi_resetn_i.

It is mandatory to follow the connection scheme due to physical routing restrictions and for proper handling of the CONFIG_LMMI primitive LMMICLK and LMMIRESETN ports. If these conditions are not met, the software generates an error preventing the design flow from proceeding.

D.3. CONFIG_LMMI Access Protocol

To access the CONFIG_LMMI primitive, the LMMI host logic should drive the LMMI interface by following the LMMI protocol. Refer to the Lattice Memory Mapped Interface and Lattice Interrupt Interface (FPGA-UG-02039) for details on the LMMI specification. When accessing any CFG internal register or any other CFG data, the non-JTAG slave command format should be followed. The supported commands are listed in the section.

The CONFIG_LMMI primitive has a hardware assigned lmmi_offset with ADDR_CFG=0x01. To execute slave configuration command, LMMI host should perform four back-to-back write transfers: 1 byte Opcode (OC), 3 bytes Operands, which are OPERAND1 – OPERAND3 (OR1 –OR3), followed by the optional data bytes. The Class A command, Class B command, and Class C command execution are demonstrated in Figure D.4, Figure D.5, and Figure D.6.

The only supported Class D command is the LSC_REFRESH command, which should be treated as Class C command for CONFIG_LMMI execution. Once the LSC_REFRESH command is executed, the LMMI host inside the FPGA fabric no longer exists.

The following are important guidelines for using the CONFIG_LMMI primitive:

- LMMI Request for Command Data Transmission
  The lmmi_request signal must be asserted high when transmitting command data. This ensures that the command is recognized and processed correctly by the receiving system.
- LMMI Request for Read Operation
  During a read operation, after the command data has been sent, the lmmi_request signal must remain high for a specific number of clock cycles, denoted as N. The value of N corresponds to the number of expected data bytes to be returned. For instance, if the expected return data is 4 bytes, when using the CONFIG_LMMI primitive, the lmmi_request signal must be kept high for four clock cycles to ensure proper data retrieval.
- Clock Cycles between LMMI Commands
  After issuing a command, wait at least four LMMI clock cycles after lmmi_ready goes high before issuing the next command. This prevents the bus from becoming unresponsive.

Failure to adhere to these guidelines can lead to unexpected behavior in the LMMI block. One such issue is the lmmi_ready signal getting stuck at low indefinitely. This situation requires toggling of the lmmi_reset signal to recover and resume normal operation. Ensuring the lmmi_request signal is managed correctly is crucial to avoid such disruptions.

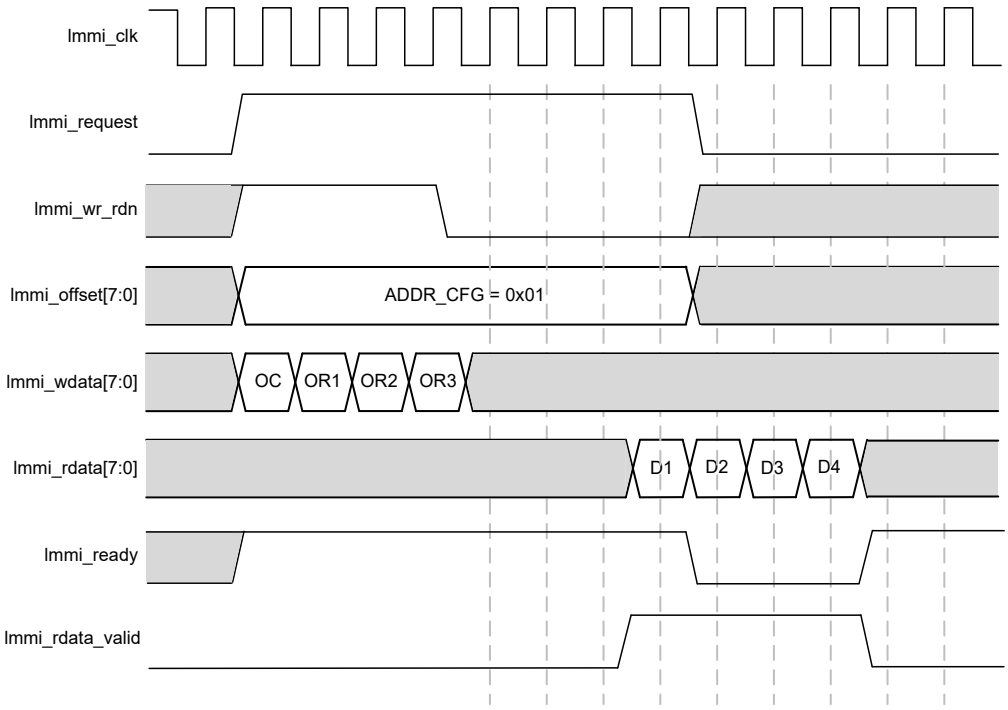


Figure D.4. LMMI Class A Commands Execution

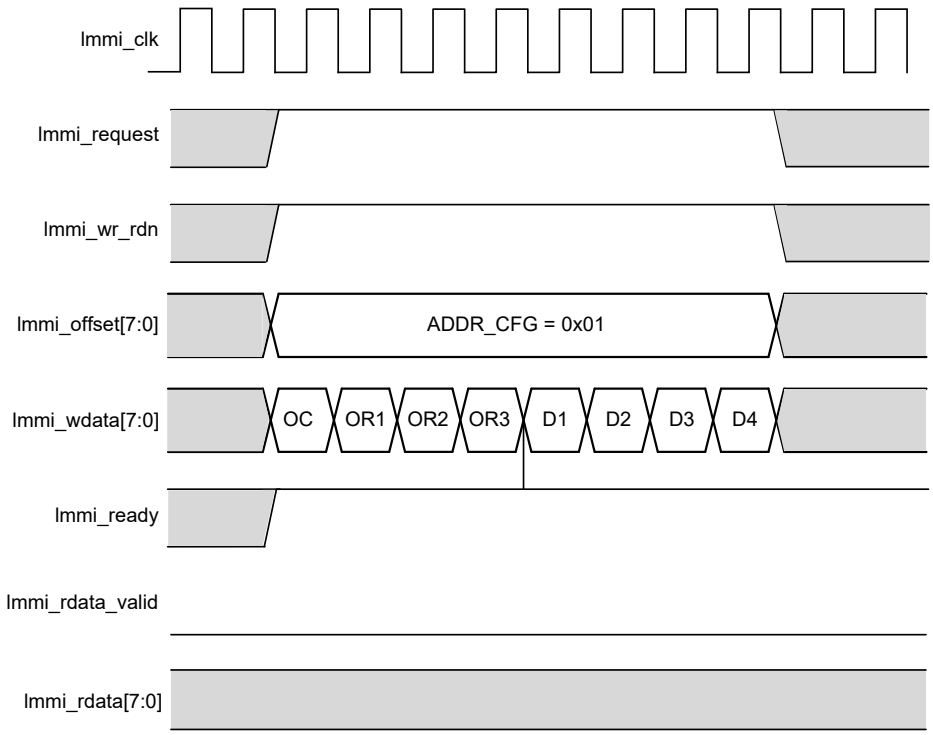


Figure D.5. LMMI Class B Commands Execution

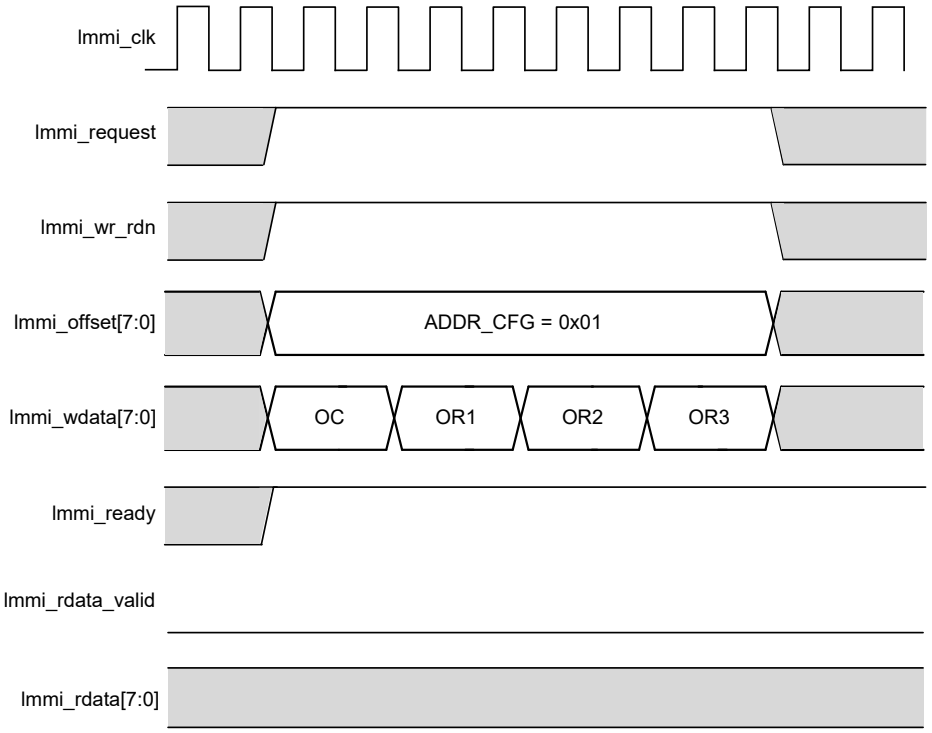


Figure D.6. LMMI Class C Commands Execution

D.4. CONFIG_LMMI Supported Commands

Due to the nature of CONFIG_LMMI access, only selected slave configuration commands can be supported by CONFIG_LMMI. The supported commands are listed in Table D.3, which are the subset of the slave command set in the Slave Command Support section.

Table D.3. CONFIG_LMMI Supported Command List

Command	Binary	Hex	Operand (Bytes)	Data (Bytes)	Class[3]	Execution Time	Description
READ_ID	11100000	E0	3	4	A	None	Read out the 32-bit IDCODE of the device.
READ_UNIQUEID	00011001	19	3	8	A	None	Reads the 64-bit UNIQUE ID, 56 bits set by Lattice, 8 bits set by customer.
USERCODE	11000000	C0	3	4	A	None	Read 32-bit usercode.
LSC_READ_STATUS	00111100	3C	3	8	A	None	Read out internal status.
LSC_CHECK_BUSY	11110000	F0	3	1	A	None	Read 1-bit busy flag to check the command execution status.
LSC_DEVICE_CTRL	01111101	7D	3	0	B	[2]	An 8-bit field in Operand 1 allows various device control, such as GSR, Sleep, Dry Run, and others. Bit 0: Cause a Global Set/Reset to occur on the device. Bit 1: Launch a One-time check of SED. Bit 2: Reserved Bit 3: Cause a reset on configuration logic related FSM and flags. Bit 4: Reserved for internal dry-run. Bit [6:5]: Launch dry-run event. 2'b01 – Dry-run for primary bitstream 2'b10 – Dry-run for secondary bitstream 2'b11 – Slave dry-run 2'b00 – No dry-run Bit 7: CRC check bit
LSC_REFRESH	01111001	79	3	0	D	—	Equivalent to toggle PROGRAMN pin
ISC_ENABLE_X	01110100	74	3	0	C	None	Enable the Transparent configuration mode.
ISC_DISABLE	00100110	26	3	0	C	None	Disable the configuration operation
ISC_PROGRAM_USERCODE	11000010	C2	3	4	B	None	Write the 32-bit new USERCODE data to USERCODE register.
ISC_PROGRAM_SECURITY	11001110	CE	3	0	C	None	Program the Security bit if the device is in Configuration state.
LSC_PROG_CNTRL0	00100010	22	3	8	B	None	Modify the Control Register 0.
LSC_READ_CNTRL0	00100000	20	3	4	A	None	Read the Control Register 0.
LSC_PROG_CNTRL1	00100011	23	3	8	B	None	Modify the Control Register 1.
						1	Modify the NV content when access NVM.
LSC_READ_CNTRL1	00100001	21	3	4	A	None	Read the Control Register 1.
LSC_RESET_CRC	00111011	3B	3	0	C	None	Reset 16-bit frame CRC register to 0x0000
LSC_READ_CRC	01100000	60	3	2	A	None	Read 16-bit frame CRC register content.
LSC_PROG_SED_CRC	10100010	A2	3	4	B	None	Program the calculated 32-bit CRC based on configuration bit values only into overall CRC register.
LSC_READ_SED_CRC	10100100	A4	3	4	A	None	Read the 32-bit SED CRC.
LSC_PROG_PASSWORD	11110001	F1	3	16	B	1	Program 128-bit password into the non-volatile memory (Efuse). See additional note[4].

Command	Binary	Hex	Operand (Bytes)	Data (Bytes)	Class[3]	Execution Time	Description
LSC_READ_PASSWORD	11110010	F2	3	16	A	None	Read out the 128-bit password before activated for verification.
LSC_SHIFT_PASSWORD	10111100	BC	3	8	B	None	Shift in the 128 bits password to unlock for re-configuration (necessary when password protection feature is active).
LSC_PROG_CIPHER_KEY	11110011	F3	3	32	B	1	Program the 256-bit cipher key into the non-volatile memory. See additional note[4].
LSC_READ_CIPHER_KEY	11110100	F4	3	32	A	None	Read out the 256-bit cipher key before activated for verification.
LSC_PROG_FEATURE	11100100	E4	3	12	B	1	Program security related feature such as Decrypt Enable, etc. based on the selection from the operand.
LSC_READ_FEATURE	11100111	E7	3	12	A	None	Read security related feature such as Decrypt Enable, etc. based on the selection from the operand.
LSC_PROG_FEABITS	11111000	F8	3	2	B	1	Program security related feature bits. See additional note[4].
LSC_READ_FEABITS	11111011	FB	3	2	A	None	Read security related feature bits.
LSC_PROG_OTP	11111001	F9	3	4	B	1	Program security related One-Time-Programmable bits based on the selection from the operand. See additional note[4].
LSC_READ_OTP	11111010	FA	3	4	A	None	Read security related One-Time-Programmable bits based on the selection from the operand.
LSC_PROG_ECDSA_PUBKEY	01011001	59	3	64	B	1	Program the ECDSA Public Key. See additional note[4].
LSC_READ_ECDSA_PUBKEY	01011010	5A	3	64	B	None	Read Back the ECDSA Public Key
LSC_PROG_SEC_BOOT	01111111	7F	3	4	B		Secondary boot address override; 32 bits data to replace the hard coded secondary boot address.
LSC_READ_DR_UES	01011101	5D	3	4	A	None	Read the dry-run User Electronic Signature shadow register.

Notes:

1. Execution time depends on data frame size and clock frequency. Duration could be in seconds. Use LSC_CHECK_BUSY to check the status.
2. Execution time depends predominantly on the bitstream size and clock frequency. Duration could be in seconds.
3. The different command types are described in the CONFIG_LMMI Access Protocol section.
4. Command operation programs the internal EFUSE memory (non-volatile memory), which is one-time programmable (OTP).

Refer to the Slave Command Details section for additional information.

Appendix E. Dual Boot Fail Safe Upgrade

When using Dual Boot and standard reprogramming flows, there can be *boot image uncertainty* if the bitstream update process is interrupted. That is, the device may boot up from either the primary bitstream image or secondary bitstream image depending upon where the interrupt occurs during the update process. This kind of uncertainty is undesirable in some use cases. This issue is magnified in systems with multiple FPGAs, where system coherence is required. That is, all FPGAs should boot from the same respective source, primary or secondary, regardless of regardless of which device's image update is interrupted. To substantially eliminate *boot image uncertainty*, a new *Fail Safe Upgrade* operation is supported in the Radiant Programmer (Revision 2023.2 or later) for Nexus devices.

The Fail Safe Upgrade process can support multiple devices configured individually, or chained together in a single JTAG chain. A typical sysCONFIG chain setup is shown in Figure E.1.

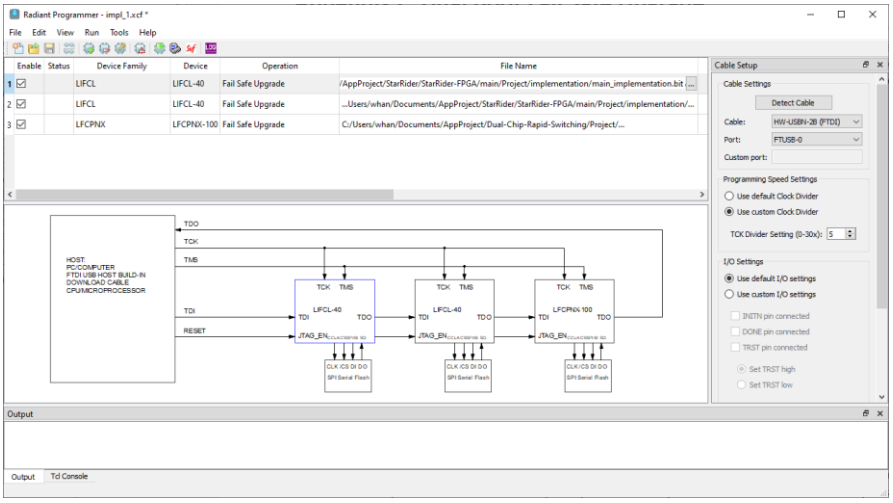


Figure E.1. SysCONFIG Chain Setup Example

E.1. Fail Safe Upgrade Process

The Fail Safe Upgrade operation is a carefully engineered process to dramatically reduce the boot image uncertainty for a device or devices. The Fail Safe process steps are shown below. Lattice Radiant Programmer performs the entire Fail Safe process automatically. Alternatively, you can perform the steps with a JTAG controller of their choice by following the steps and information below.

The Fail Safe Upgrade operation divides the bitstream update process into steps as shown in Table E.1. The example is shown for a primary bitstream update. (To update the secondary image, merely substitute the Secondary image in the process where the Primary image appears in the Process Action.)

Table E.1. Fail Safe Upgrade Process Steps

Process Step Index	Process Action	Process Duration	Safe or Vulnerable	Boot up Image (in case of interruption)	Note
A	Check the FPGA Device ID Clear SRAM in "Direct Programming" Mode	Short	Safe	Primary	Sanity checking
B	Write First Page of the primary bitstream with 256 bytes 0x00 for all device in the chain	Short	Vulnerable1	Primary or Secondary	Destroy the Preamble for the primary bitstream to make it not bootable.
C	Erase Primary Bitstream for all devices in the chain.	Long	Safe	Secondary	Erase the SPI flash memory contains the primary bitstream, which is a time consuming process.

Process Step Index	Process Action	Process Duration	Safe or Vulnerable	Boot up Image (in case of interruption)	Note
D	Program Fail Safe bitstream for new primary image with empty first page (first 256 bytes with 0xFF) for all devices in the chain.	Long	Safe	Secondary	Program the SPI flash memory for the primary bitstream, which is a time consuming process.
E	Verify new Primary Bitstream for all devices in the chain.	Long	Safe	Secondary	Verify the primary bitstream programming, which is a time consuming process.
F	Write the first page (256 Bytes) with valid Preamble for all devices in the chain.	Short	Vulnerable1	Primary or Secondary	Program the LSCC Signature and Preamble to make the updated bitstream bootable.
G	Update completed. Issue REFRESH command in “Direct Programming” Mode to wake up all devices in chain	Short	Safe	New Primary	Process Complete

Note: Process steps B and F are the steps which are sensitive to the unexpected interruption. All the other steps provide certainty regarding the boot image. Steps B and F only involve writing a single page (256 bytes) of data. The execution time is very short, greatly minimizing the risk to the system coherency.

The special first page of the bitstream is formatted as shown in Table E.2.

Table E.2. Bitstream First Page Format

Frame	Contents	Description
LSCC Signature	0x4C534343	32 bits of the LSCC Signature
Comments	(Comment String)	ASCII Comment String and Terminator
Preamble	0xFFFFBDB3	32 bits Preamble Code
Dummy	0xFFFFFFFF	32 bits Dummy (Required)

E.2. Fail Safe Upgrade with Radiant Programmer

To perform the Fail Safe Upgrade using Radiant Programmer (Revision 2023.2 or later), open the *Device Properties* user interface by click the *Operation* field for each device in the chain. Select *External SPI Flash Memory (SPI FLASH)* for *Target Memory*, and set the *Operation* for *Fail Safe Upgrade*, as shown in Figure E.2, then click *OK*.

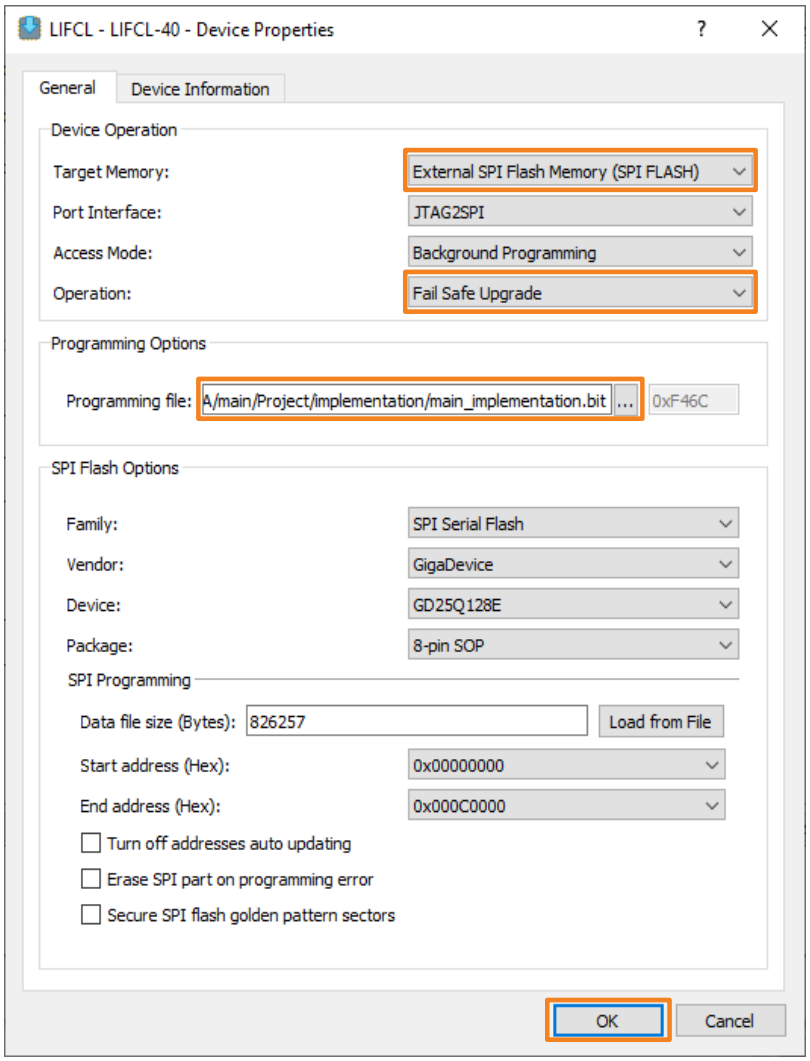


Figure E.2. Fail Safe Upgrade with Radiant Programmer

The Fail Safe Upgrade operation supported by Radiant Programmer automatically performs the process steps listed in Table E.1, even with original bitstream file.

E.3. Bitstream Generation for Fail Safe Upgrade

In case you handle the upgrade process without using Lattice Radiant Programmer, you can generate the Fail Safe Bitstream (bitstream file with empty preamble field) by using Lattice software tools.

E.3.1. Generate the Fail Safe Bitstream from Radiant

To setup the Radiant Bitstream Generator for Fail Safe Bitstream generation, you need to open the *Radiant Strategy*, and set the *Bitstream Mode* from *Normal* to *Fail Safe* as shown in below, then click *OK*.

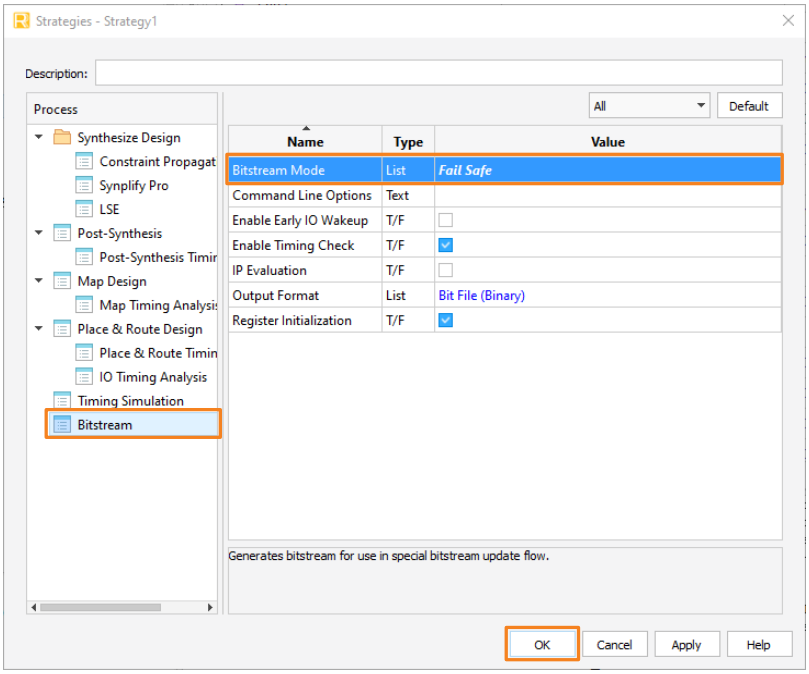


Figure E.3. Strategy Setup for Bitstream Generation

With this strategy setting, the Fail Safe Bitstream is generated when the user design is compiled.

E.3.2. Converting the Original Bitstream using Deployment Tool

To convert the Fail Safe bitstream from the existing bitstream with the header preamble,

1. Launch the Deployment Tool from Radiant Programmer.
2. Select **Function Type** as **External Memory** and the **Output File Type** as **Hex Conversion**, as shown in Figure E.4. Click **OK**.

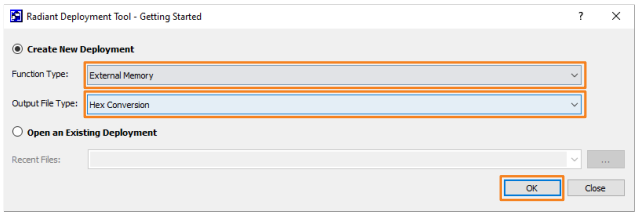


Figure E.4. Fail Safe Bitstream Conversion: Launch Deployment Tool

3. Input Primary bitstream file and clock **Next**.

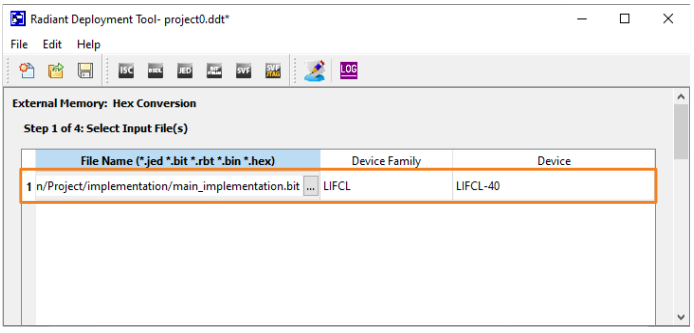


Figure E.5. Fail Safe Bitstream Generation: Input Files Setup

4. Select No Header Preamble on Primary Bitstream option and click **Next**.

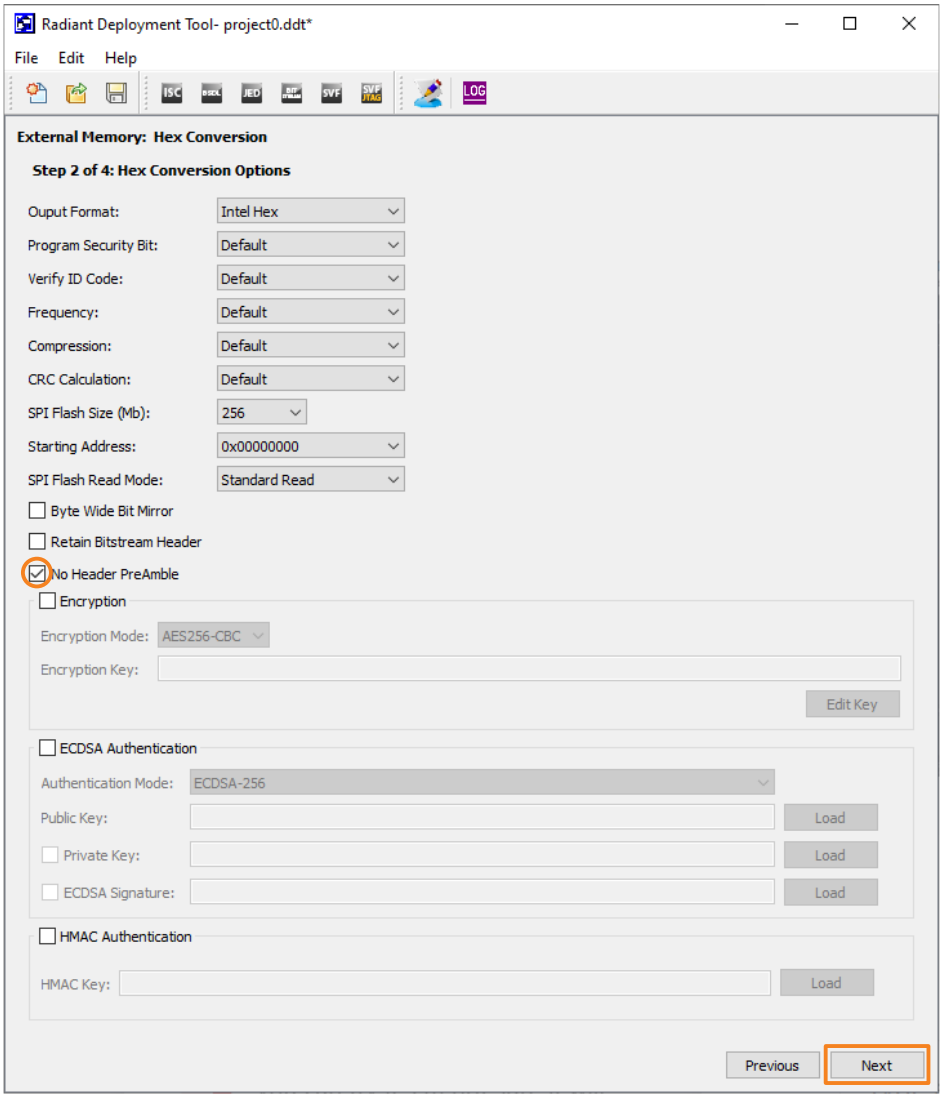


Figure E.6. Fail Safe Upgrade Bitstream Generation: Option Setup

5. Select the **Output File** name and click **Next**.

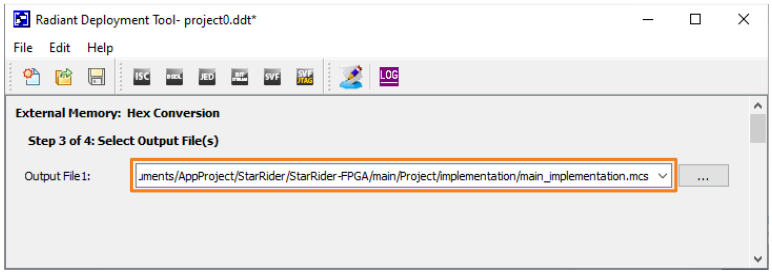


Figure E.7. Fail Safe Upgrade Bitstream Generation: Select Output File Name

6. Click the **Generate** button to generate the dual boot image with no Preamble for primary bitstream .mcs file.

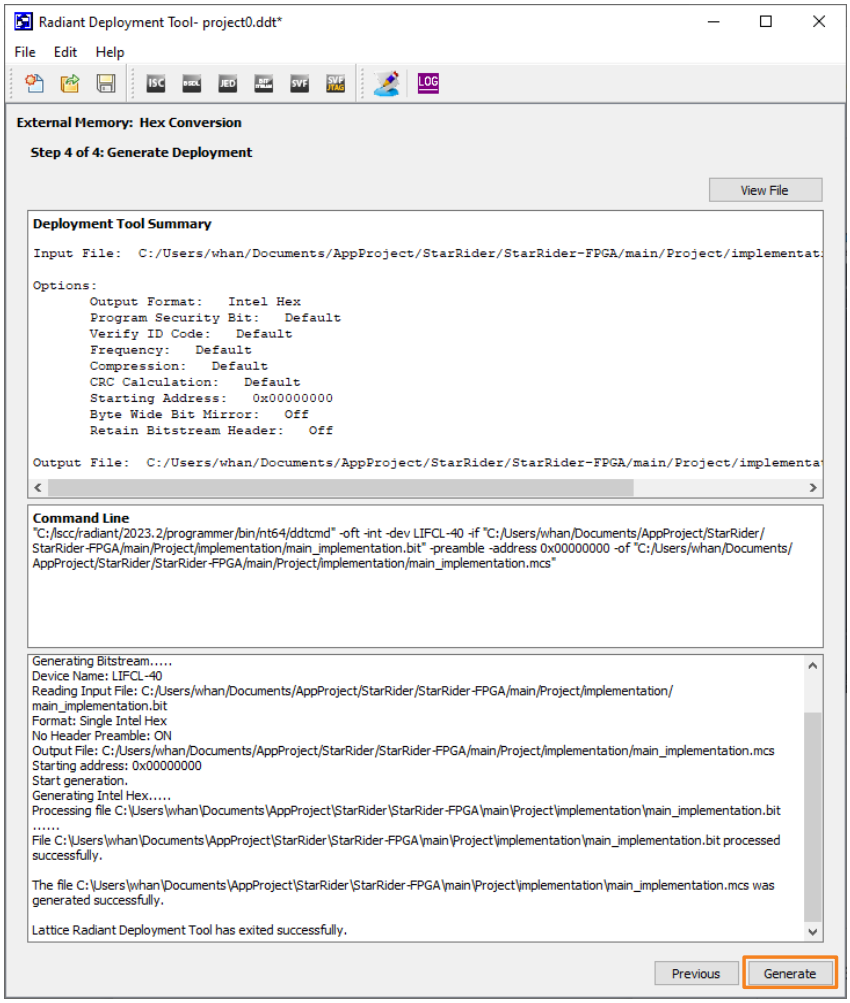


Figure E.8. Fail Safe Upgrade Bitstream Generation: Generate the mcs file

Appendix F. Enabling 32-bit (4-byte) MSPI Address and Command Mode (Dual-Boot/Multi-Boot/Ping-Pong Boot)

When using dual-boot, multi-boot, or ping-pong boot mode, the backup JUMP instruction is assigned and stored at the last 256 bytes of the configuration memory. Setting the 32-bit (4-byte) MSPI command and address option bits is essential for the MSPI controller to send the appropriate read command and byte-addressing to the configuration memory for reading the JUMP instruction contents thereby ensuring the fail-safe mechanism works properly. In addition, when attempting to boot the image/pattern stored at an address beyond 0xFF_FFFF, you must ensure that the MSPI controller is operating in 4-byte command and addressing mode in order for the device to load the pattern successfully.

By default, the 32-bit MSPI address and 32-bit MSPI commands option bits in CR1 are set to 0. The MSPI controller operates in 3-byte command and addressing mode, enabling the device to support dual-boot, multi-boot, or ping-pong boot mode with configuration memory sizes up to 128 Mb. If your configuration memory size is 256 Mb or larger, the 32-bit MSPI address and 32-bit MSPI commands option bits in CR1 must be set to 1.

Once the fail-safe mechanism or multi-boot feature to boot stored images/patterns in addresses beyond 0xFF_FFFF is tested to be working properly on hardware as covered in the Setting 32-bit MSPI Address and Command in CR1 Through Bitstream and Setting No CDM in CR0 Through Bitstream sections, refer to the Setting 32-bit SPI Address and Command in Non-Volatile Configuration Memory section for information on permanently setting the 32-bit MSPI Address and 32-bit MSPI Commands option bits.

F.1. Setting 32-bit MSPI Address and Command in CR1 Through Bitstream

This guide allows you to test and ensure that MSPI configuration from an address beyond 0xFF_FFFF can succeed and therefore the fail-safe mechanism can function properly. The option bits are set in volatile SRAM memory. You can perform this test before committing the settings permanently in non-volatile configuration memory.

The following are steps to set the 32-bit MSPI address and 32-bit MSPI commands option bits:

1. Run the Lattice Radiant Programmer.
2. Select **Tools > Programming File Utility**.
3. In the Programming File Utility, select **Tools > Control Register 1 Editor**. The Control Register 1 window appears.

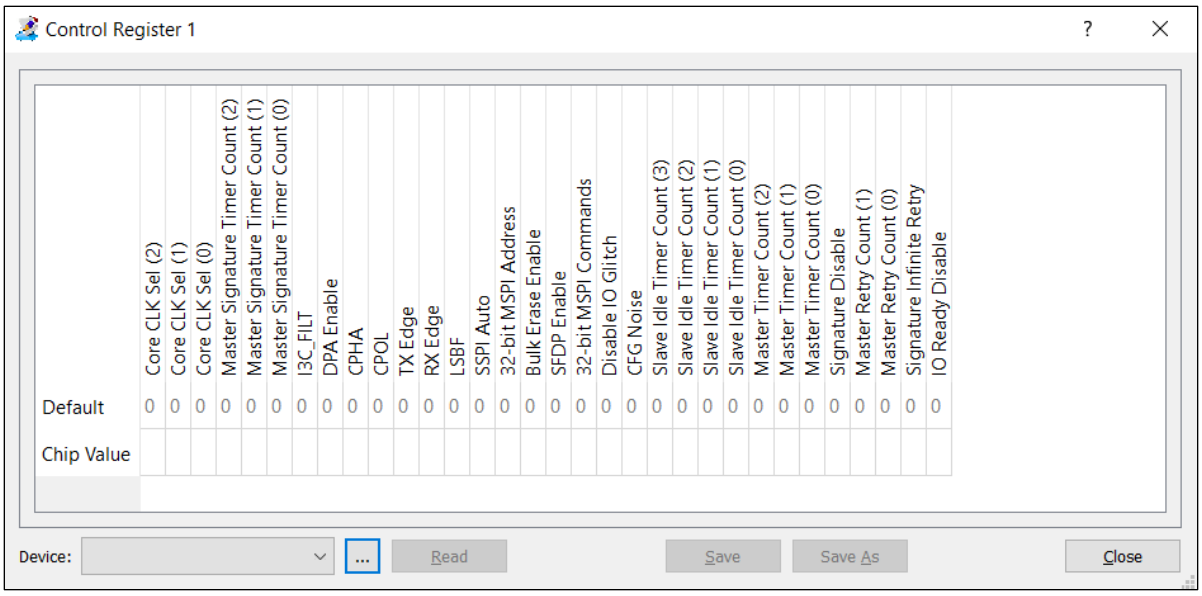


Figure F.1. Control Register 1 Window

4. Click on the **…** button. The Open Data File window appears.

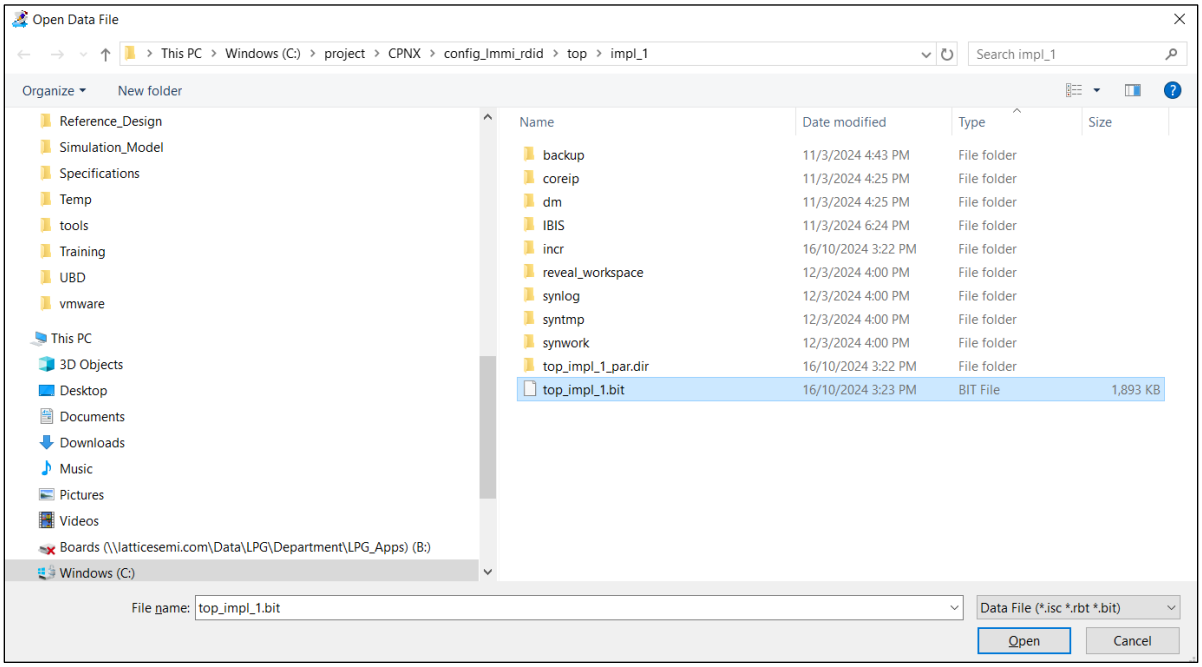


Figure F.2. Open Data File Window (Selecting .bit File)

5. Navigate to and select the .bit file, then click **Open**.
6. Click **Read** to display the Control Register 1 settings. By default, 32-bit MSPI Commands and 32-bit MSPI Address are set to 0.

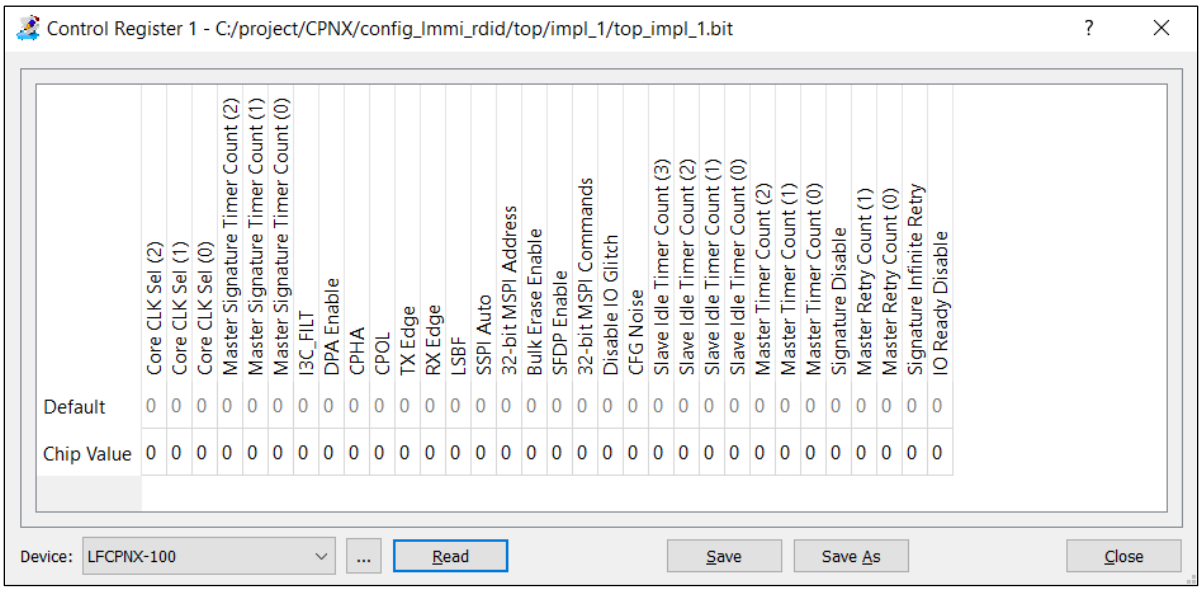


Figure F.3. Control Register 1 Settings

7. Click on the chip value associated with 32-bit MSPI Commands and change the value from 0 to 1.
8. Repeat step 7 for 32-bit MSPI Address.

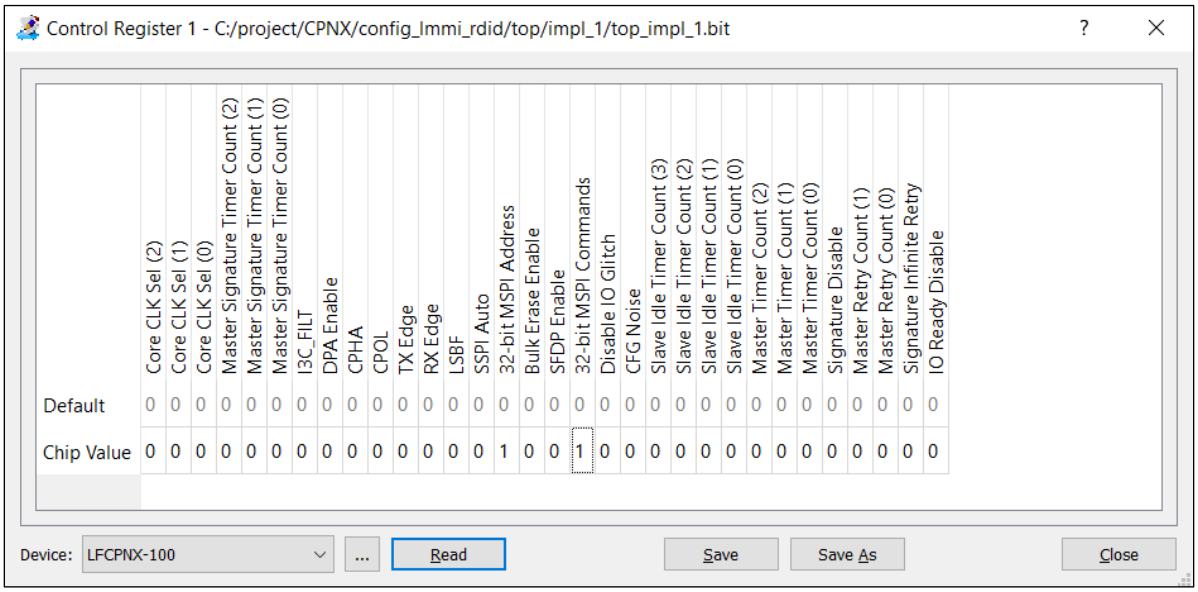


Figure F.4. Changing 32-bit MSPI Commands and 32-bit MSPI Address Option Bit Settings

9. Click **Save As**. The Save As window appears.

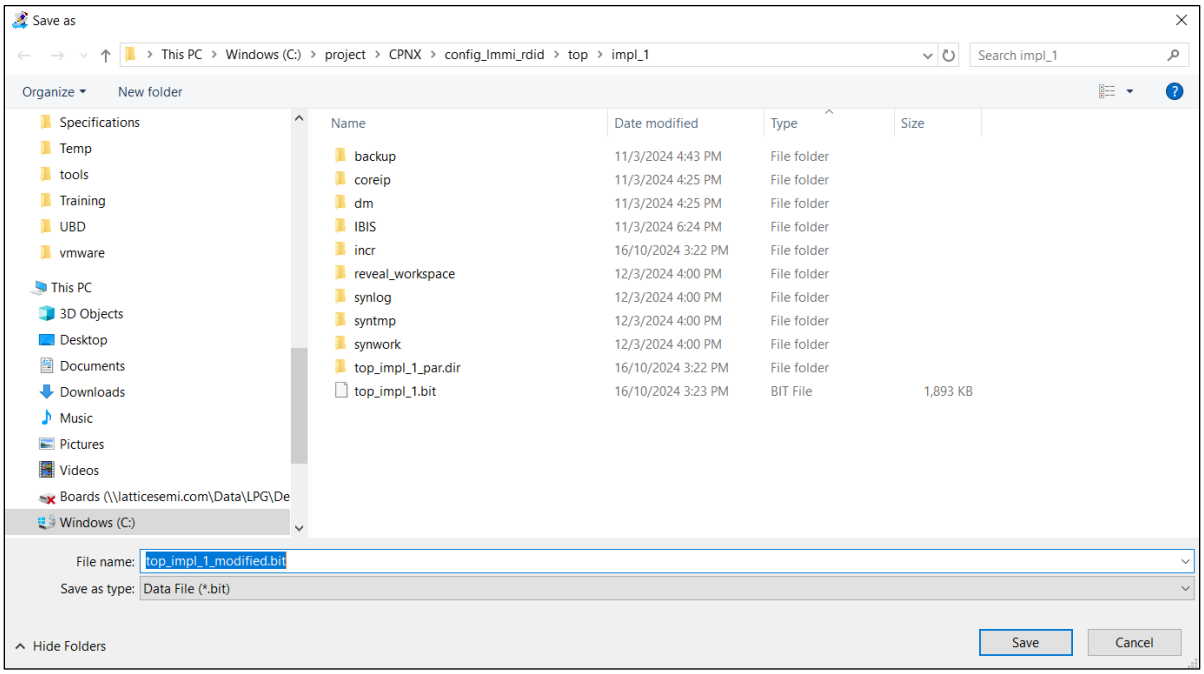


Figure F.5. Save As Window

10. Enter a new filename, for example top_impl_1_modified.bit, then click **Save**.

11. Click **Save** in the Control Register 1 window to generate a new .bit file with the 32-bit MSPI Commands and 32-bit MSPI Address option bits set to 1. A write to file successful dialog box appears.

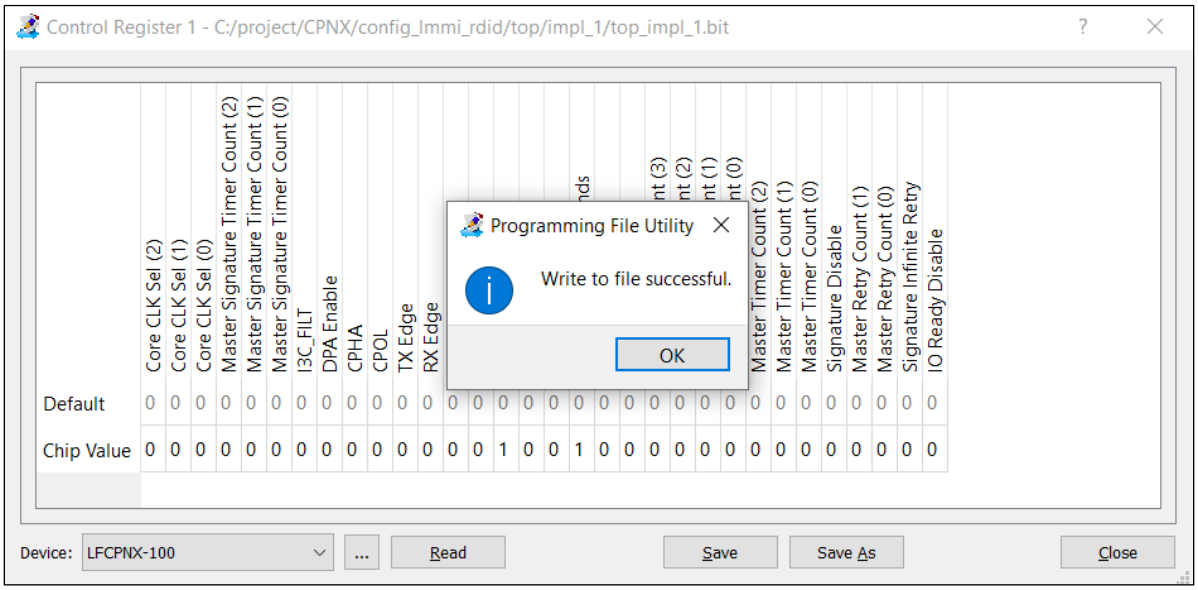


Figure F.6. Write to File Successful Dialog Box

12. Click **OK**.
    Note: You can repeat the step to read the new .bit file to confirm that the 32-bit MSPI Commands and 32-bit MSPI Address option bits are set to 1.
13. Repeat steps 4 to 12 for all the images/patterns to be used for dual-boot, multi-boot, or ping-pong boot. Once this step is done, proceed to generate the dual-boot, multi-boot, or ping-pong boot .mcs file, program the configuration memory with the generated .mcs file, and test the fail-safe mechanism or multi-boot feature to boot the images/patterns stored in addresses beyond 0xFF_FFFF.

F.2. Setting No CDM in CR0 Through Bitstream

Setting the No CDM option bit to 1 is essential to retain the CR1 settings in SRAM during reconfiguration.

The following are steps to set the No CDM option bit:

1. In the Programming File Utility, select **Tools > Control Register 0 Editor**. The Control Register 0 window appears.

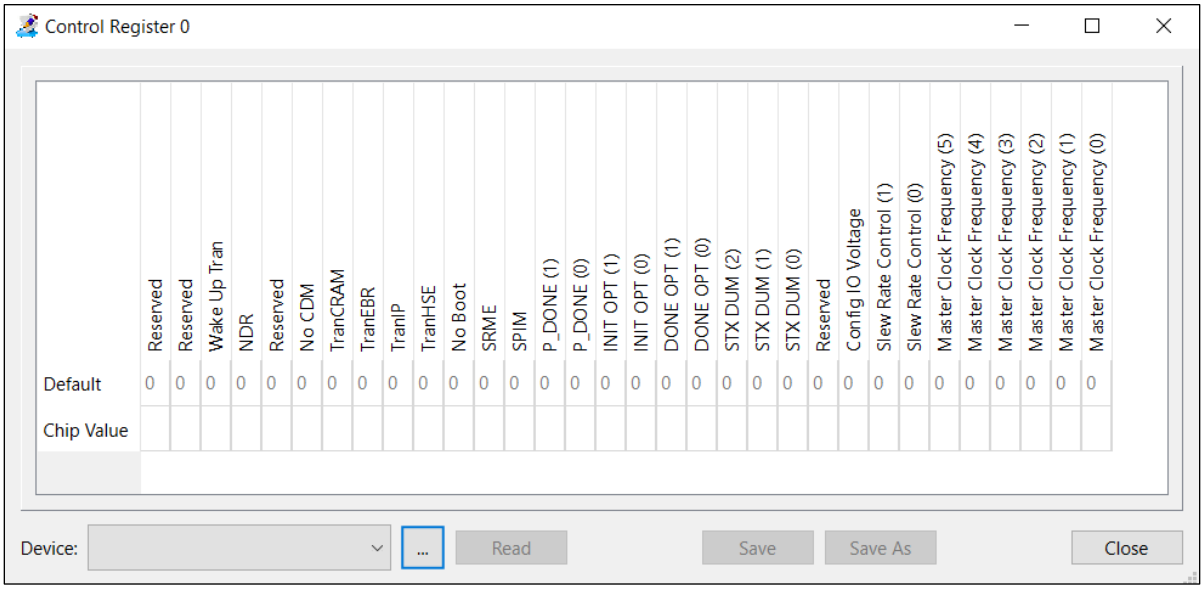


Figure F.7. Control Register 0 Window

2. Click on the **…** button. The Open Data File window appears.

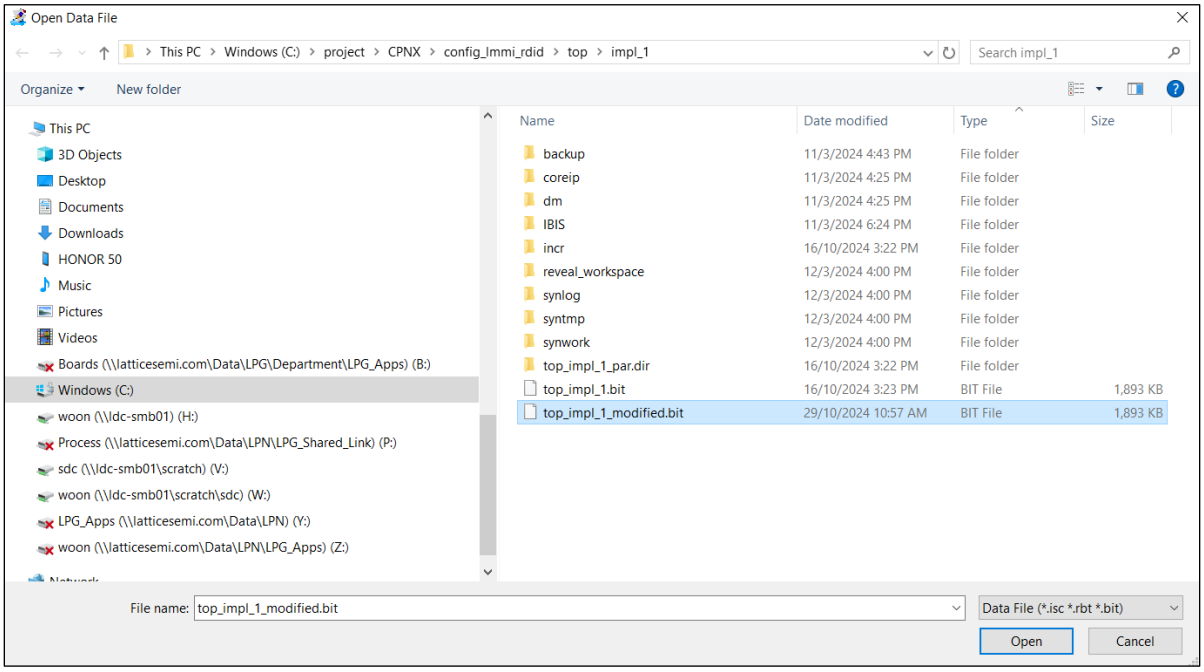


Figure F.8. Open Data File Window (Selecting Modified .bit File)

3. Navigate to and select the modified .bit file, for example top_impl_1_modified.bit as created in the Setting 32-bit MSPI Address and Command in CR1 Through Bitstream section, then click **Open**.

4. Click **Read** to display the Control Register 0 settings. By default, the No CDM option bit is set to 0.

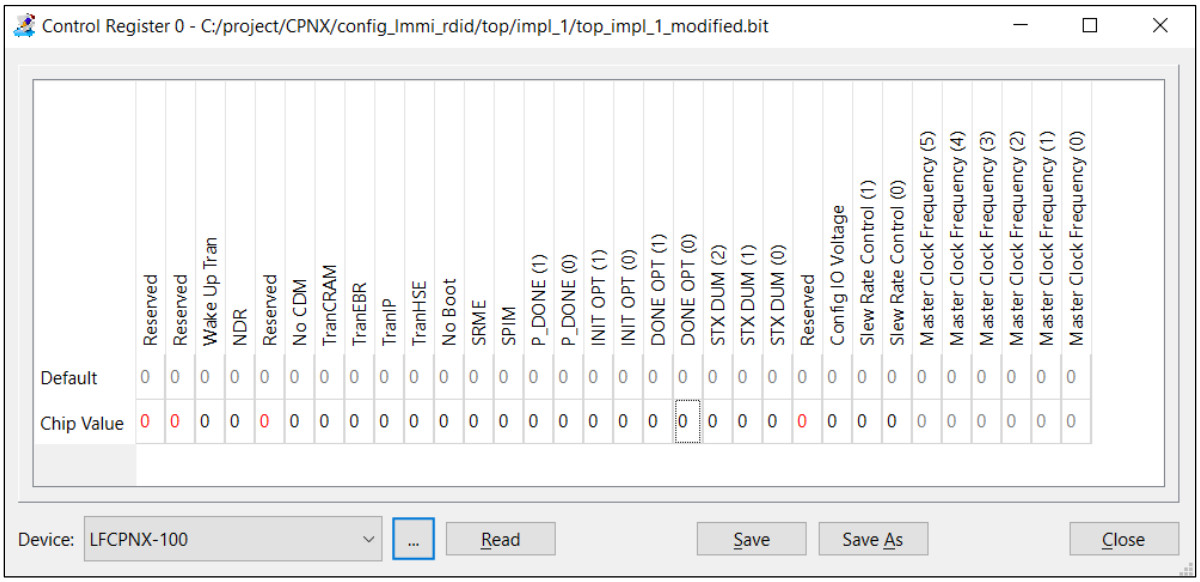


Figure F.9. No CDM Option Bit Setting

5. Click on the chip value associated with No CDM and change the value from 0 to 1.

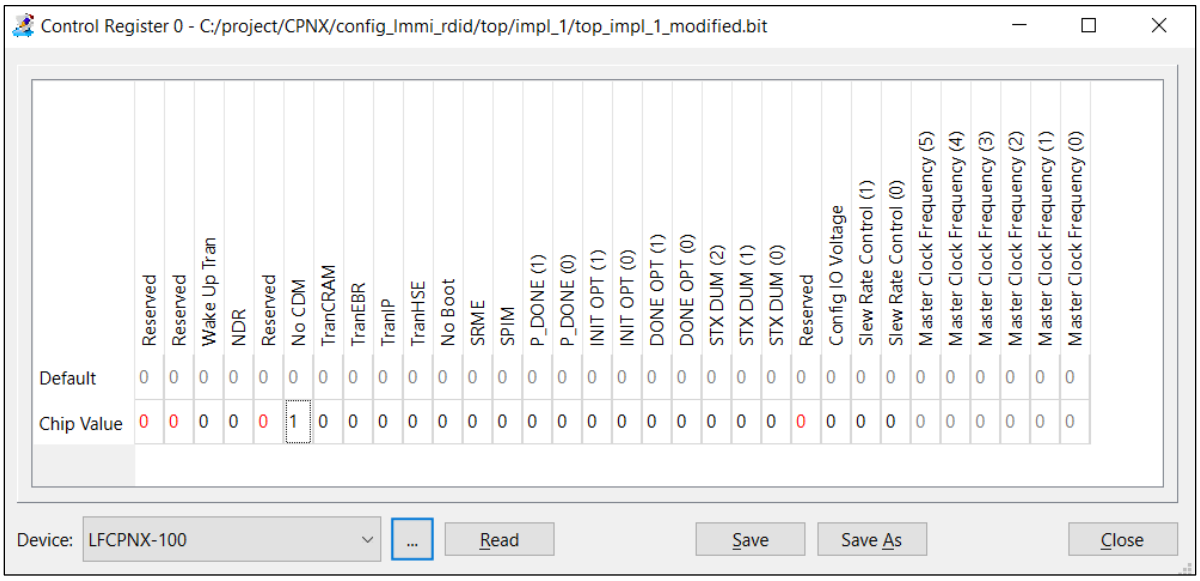


Figure F.10. Changing No CDM Option Bit Setting

6. Click **Save** in the Control Register 0 window. An Overwrite dialog box appears.

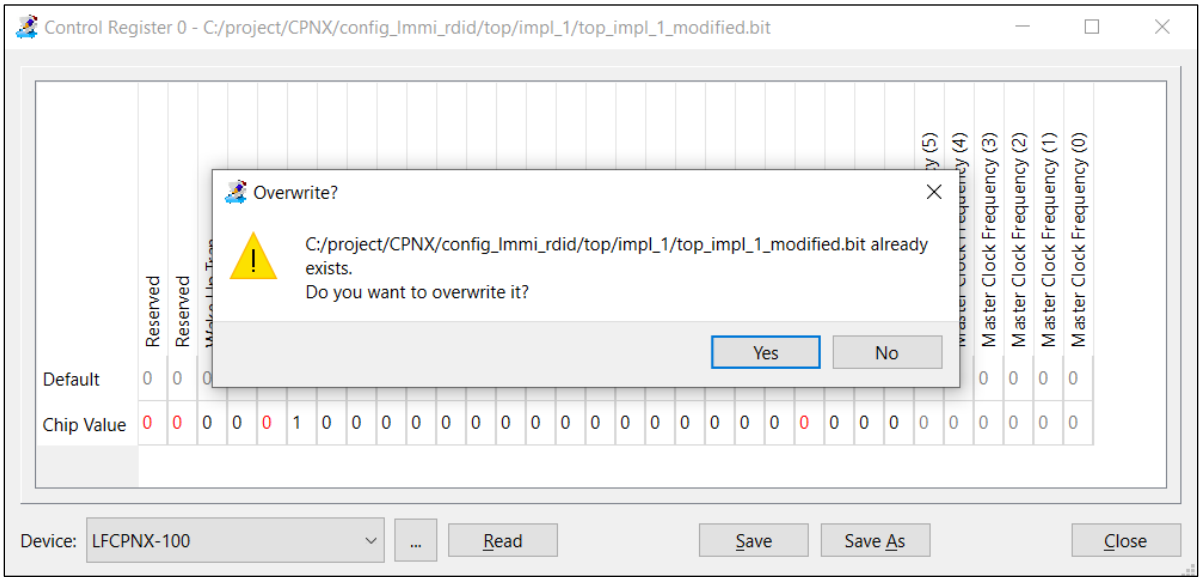


Figure F.11. Overwrite Dialog Box

7. Click **Yes** to overwrite the modified .bit file.
   Note: You can repeat the step to read the modified .bit file to confirm that the No CDM option bit is set to 1.
8. Repeat steps 2 to 7 for all the images/patterns to be used for dual-boot, multi-boot, or ping-pong boot. Once this step is done, proceed to generate the dual-boot, multi-boot, or ping-pong boot .mcs file, program the configuration memory with the generated .mcs file, and test the fail-safe mechanism or multi-boot feature to boot the images/patterns stored in addresses beyond 0xFF_FFFF.

F.3. Setting 32-bit SPI Address and Command in Non-Volatile Configuration Memory

To permanently set the 32-bit SPIM Address and 32-bit SPIM Commands option bits to 1, refer to Appendix C for the steps to modify the Nexus feature row by using the Feature Row Editor.

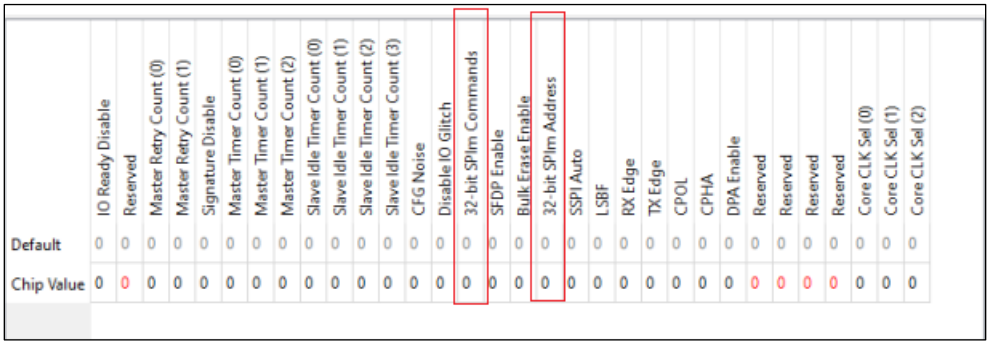


Figure F.12. 32-bit SPIM Address and 32-bit SPIM Commands Option Bits in Nexus Feature Row

Note: Feature row bits are one-time programmable (OTP). Each bit can only be modified one time. Once the 32-bit SPIM Address and 32-bit SPIM Commands option bits are set to 1, the FPGA can only work with configuration memory of sizes 256 Mb or larger to perform dual-boot, multi-boot, or ping-pong boot.

Appendix G. Modifying sysCONFIG Options in the Lattice Radiant Device Constraint Editor without Re-running Place and Route Design

The sysCONFIG attributes can be altered through the bitstream generation (bitgen) command. This can be done using any one of the following two methods: through the bitstream strategy settings command line and through Tcl command.

Note: The Tcl command method will not work if the sysCONFIG option is already defined in the .pdc file.

For more information on the bitgen command, refer to the Lattice Radiant Software User Guide.

G.1. Through Bitstream Strategy Settings Command Line

1. In the Lattice Radiant software, select **Project** > **Active Strategy** > **Bitstream Settings**.

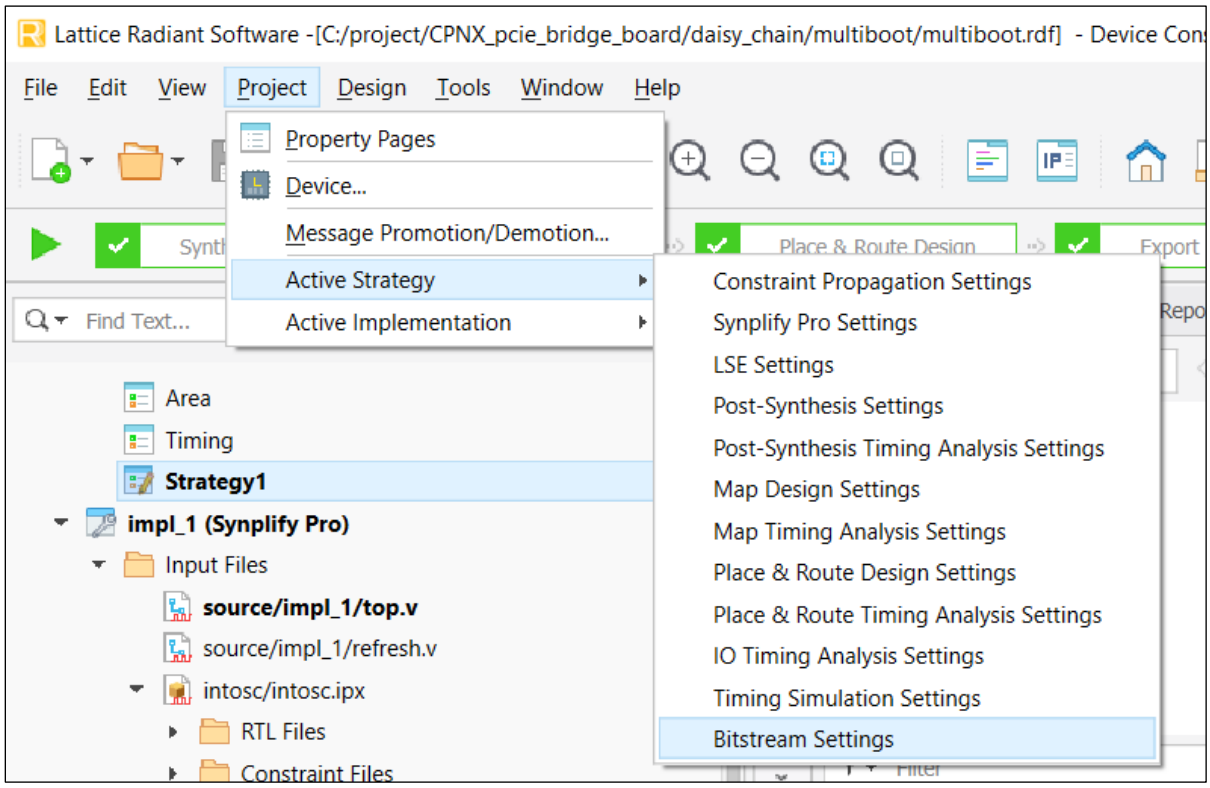


Figure G.1. Selecting Bitstream Settings Strategy

2. Add the bitgen option in the **Command Line Options** field with the syntax *-g <opt:val>* where opt is the sysCONFIG option name and val is an available setting. For example, the BACKGROUND_RECONFIG option has available settings OFF, ON, SRAM_EBR, and SRAM_ONLY. To set BACKGROUND_RECONFIG to ON, enter *-g BACKGROUND_RECONFIG:ON*.

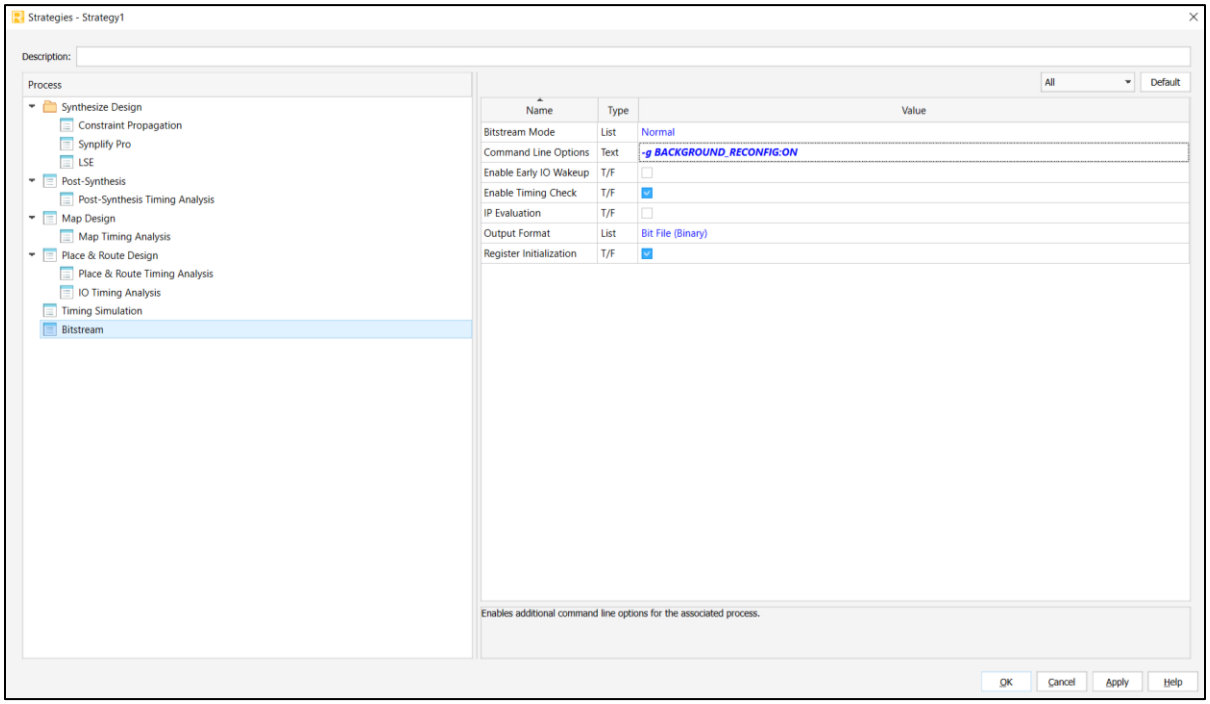


Figure G.2. Adding bitgen Option in Command Line Options Field

3. Click **Apply** followed by **OK**.
4. Run **Export Files** to generate the .bit file.

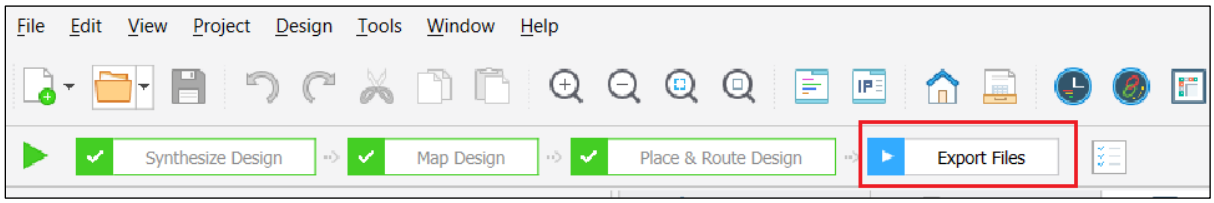


Figure G.3. Generating .bit File

G.2. Through Tcl Command

1. Navigate to and open the .bgn file, located in the implementation folder of your project, as a text file.

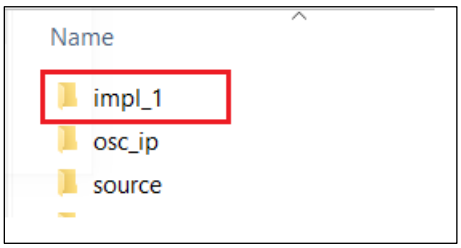


Figure G.4. Implementation Folder

2. Copy the bitgen command from the .bgn file.

```
BITGEN: Bitstream Generator Radiant Software (64-bit) 2024.1.1.259.1
Copyright (c) 1991-1994 by NeoCAD Inc. All rights reserved.
Copyright (c) 1995 AT&T Corp.   All rights reserved.
Copyright (c) 1995-2001 Lucent Technologies Inc.  All rights reserved.
Copyright (c) 2001 Agere Systems   All rights reserved.
Copyright (c) 2002-2024 Lattice Semiconductor Corporation,  All rights reserved.
Wed Dec 18 17:48:41 2024


Command: bitgen -w -gui -msgset C:/project/CPNX_pcie_bridge_board/daisy_chain/multiboot/promote.xml -g
TRANSFR:OFF -g REGISTER_INIT:ON multiboot_impl_1.udb
C:/project/CPNX_pcie_bridge_board/daisy_chain/multiboot/impl_1/multiboot_impl_1

Running DRC.
DRC detected 0 errors and 0 warnings.

Preference Summary:
+---------------------------------+---------------------------------+
|  Preference                     |  Current Setting                |
+---------------------------------+---------------------------------+
|                         DONE_EX |                              ON |
+---------------------------------+---------------------------------+
|                         DONE_OD |                             ON* |
+---------------------------------+---------------------------------+
|                  CONFIG_SECURE  |                            OFF* |
+---------------------------------+---------------------------------+
|               EARLY_IO_RELEASE  |                            OFF* |
+---------------------------------+---------------------------------+
|                  REGISTER_INIT  |                            ON** |
+---------------------------------+---------------------------------+
|   MASTER_PREAMBLE_TIMER_CYCLES  |                         600000* |
+---------------------------------+---------------------------------+
|                 SLAVE_SPI_PORT  |                       DISABLE** |
+---------------------------------+---------------------------------+
|                 SLAVE_I2C_PORT  |                       DISABLE** |
```

Figure G.5. Copying Bitgen Command from .bgn File

3. Modify the bitgen command by adding in the sysCONFIG option to modify. The syntax is *-g <opt:val>* where opt is the sysCONFIG option name and val is an available setting. The following example shows the bitgen command modified to set the BACKGROUND_RECONFIG option to ON.

   bitgen -w -gui -msgset C:/project/CPNX_pcie_bridge_board/daisy_chain/multiboot/promote.xml **-g BACKGROUND_RECONFIG:ON** -g TRANSFR:OFF -g REGISTER_INIT:ON multiboot_impl_1.udb C:/project/CPNX_pcie_bridge_board/daisy_chain/multiboot/impl_1/multiboot_impl_1

4. Copy the modified bitgen command into the Lattice Radiant software Tcl Console and press enter to generate the .bit file.

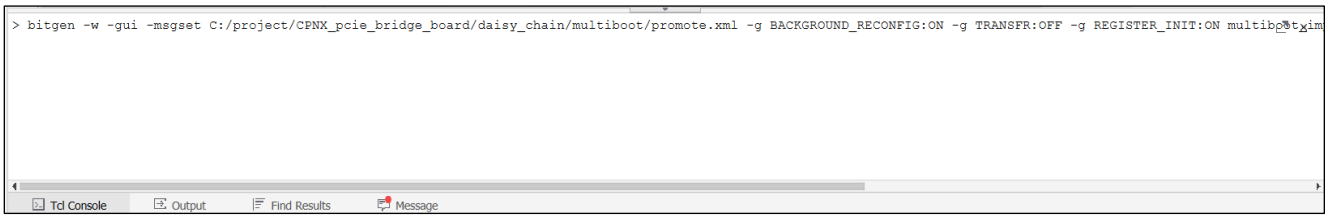


Figure G.6. Copying bitgen Command into Tcl Console

5. After bitstream generation completes, open the updated .bgn file to verify that the sysCONFIG option has been changed, for example BACKGROUND_RECONFIG is set to ON.

Preference	Current Setting
DONE_EX	OFF*
DONE_OD	ON*
CONFIG_SECURE	OFF*
EARLY_IO_RELEASE	OFF*
REGISTER_INIT	ON**
MASTER_PREAMBLE_TIMER_CYCLES	600000*
SLAVE_SPI_PORT	DISABLE**
SLAVE_I2C_PORT	DISABLE**
SLAVE_I3C_PORT	DISABLE**
JTAG_PORT	ENABLE**
DONE_PORT	ENABLE**
INITN_PORT	ENABLE**
PROGRAMN_PORT	ENABLE**
TRANSFR	OFF*
MASTER_SPI_PORT	DISABLE**
MCCLK_FREQ	3.5*
COMPRESS_CONFIG	OFF*
BACKGROUND_RECONFIG	ON
CONFIG_IOSLEW	SLOW*
WAKE_UP	ENABLE_DONE_SYNC*
BOOTMODE	DUAL*
CONFIGIO_VOLTAGE_BANK0	NOT_SPECIFIED*
CONFIGIO_VOLTAGE_BANK1	NOT_SPECIFIED*

Figure G.7. Verifying sysCONFIG Option in .bgn File

Appendix H. Dry Run User Image

For applications where system uptime is critical, it is important to ensure that the newly deployed user image can be loaded successfully during FPGA reconfiguration. This prevents system downtime caused by configuring the FPGA with a corrupted bitstream, an unauthorized user image, or an image encrypted with the wrong AES key. Lattice FPGAs allow you to perform a dry run before loading the new image into the FPGA. This section describes how to perform the dry run using the LSC_DEVICE_CTRL command. Refer to the LSC_DEVICE_CTRL section for more information.

You can store the bitstream in the external flash as the primary or secondary image. When dry run is initiated, the configuration engine will load the bitstream from the external flash. Alternatively, you can perform dry run by sending the bitstream from an external host using sysCONFIG ports such as JTAG, SSPI, or I2C/I3C.

H.1. Dry Run the Bitstream Stored in External Flash

To initiate a dry run and check the result from the dry run, perform the following steps:

Note: Commands can be executed from user logic through the CONFIG_LMMI interface or using syCONFIG ports such as JTAG, SSPI, and I2C/I3C.

1. Set the MCLK frequency to 3.5 MHz to mimic initial configuration during the dry run. Optionally, set the TranHSE bit in CR0. Execute the following commands in sequence:
   a. ISC_ENABLE_X (74h).
   b. LSC_PROG_CNTRL0 (22h) with operand1 set to 1. This accepts the new CR0 value with the mask bit.
      i. For plain bitstream, data is 0x0, and mask bit is 0xFFFFFFE0.
      ii. For secure bitstream, data is 0x400000, and mask bit is 0xFFBFFFE0. This sets the TranHSE bit in CR0 (bit[22]) to allow the configuration engine to access HSE to authenticate and decrypt the secure bitstream.
   c. ISC_DISABLE (26h).
2. Initiate dry run by executing the following commands in sequence:
   a. ISC_ENABLE_X (74h).
   b. LSC_DEVICE_CTRL (7Dh) with operand1 set to 0x20 for primary image or 0x40 for secondary image.
   c. Delay for 1 to 3 seconds. Do not execute any command during this period. This allows the dry run to complete without interruption to the configuration engine. The time to complete the dry run is dependent on the bitstream size and bitstream authentication. For a safe dry run operation, a longer delay is recommended.
   d. ISC_DISABLE (26h).
3. Read the Status Register by executing the LSC_READ_STATUS (3Ch) command. You can perform one of the following:
   - Read BSE Error Code from Status Register bit[27:24] to determine if the dry run is successful. A read back value of zero indicates that the dry run is successful.
   - Read Status Register bit[33]. This bit is set if the dry run is successful. However, this bit is sticky. Once set, any subsequent unsuccessful dry run will not reset it to 0. It can only be reset by a REFRESH event or power cycle. Therefore, reading BSE Error Code is recommended to determine if the dry run is successful.
4. Read the dry-run User Electronic Signature shadow register. This register stores the USERCODE from the last successful dry run image. The value of this register will not be updated until the next successful dry run, or will be cleared by a REFRESH event or power cycle. Note that this register is different from the USERCODE register, which is used to store the current running image USERCODE. Execute the following commands in sequence to read the UES shadow register:
   a. ISC_ENABLE_X (74h).
   b. LSC_READ_DR_UES (5Dh).
   c. ISC_DISABLE (26h).

H.2. Dry Run the Bitstream Sent through sysCONFIG Ports

To initiate dry run and check the result from the dry run, perform the following steps:
Note: Commands can be executed using sysCONFIG ports such as JTAG, SSPI, and I2C/I3C.

1. If your bitstream is signed or encrypted, set TranHSE bit in CR0. Otherwise, skip this step.
   a. ISC_ENABLE_X (74h).
   b. LSC_PROG_CNTRL0 (22h) with operand1 set to 1. This accepts the new CR0 value with the mask bit. Data is 0x400000 and mask bit is 0xFFBFFFFF. This sets the TranHSE bit in CR0 (bit[22]) to allow the configuration engine to access HSE to authenticate and decrypt the secure bitstream.
   c. ISC_DISABLE (26h).
2. To perform dry run with the bitstream sent through a sysCONFIG port, execute the following commands in sequence:
   a. ISC_ENABLE_X (74h).
   b. LSC_DEVICE_CTRL (7Dh) with operand1 set to 0x60. The external host shall burst the bitstream by using LSC_BITSTREAM_BURST (7Ah) after executing the LSC_DEVICE_CTRL command.
   c. ISC_DISABLE (26h).
3. Read the Status Register by executing the LSC_READ_STATUS (3Ch) command. You can perform one of the following:
   - Read BSE Error Code from Status Register bit[27:24] to determine if the dry run is successful. A read back value of zero indicates that the dry run is successful.
   - Read Status Register bit[33]. This bit is set if the dry run is successful. However, this bit is sticky. Once set, any subsequent unsuccessful dry run will not reset it to 0. It can only be reset by a REFRESH event or power cycle. Therefore, reading BSE Error Code is recommended to determine if the dry run is successful.
4. Read the dry-run User Electronic Signature shadow register. This register stores the USERCODE from the last successful dry run image. The value of this register will not be updated until the next successful dry run, or will be cleared by a REFRESH event or power cycle. Note that this register is different from the USERCODE register, which is used to store the current running image USERCODE. Execute the following commands in sequence to read the UES shadow register:
   a. ISC_ENABLE_X (74h).
   b. LSC_READ_DR_UES (5Dh).
   c. ISC_DISABLE (26h).

H.3. INITN Pin Behavior

The INITN pin behavior is described in the INITN section. It stays high after the FPGA is configured successfully. However, it will respond to the result from a dry run. If the dry run is successful, it remains high. If the dry run fails, INITN asserts low to indicate that the dry run is not successful. Any subsequent successful dry run will not de-assert INITN high. This behavior will not affect the function of the current running image. INITN will de-assert high upon the next successful FPGA reconfiguration.

H.4. Limitation

If you set the TranHSE bit (CR0[22]) to allow the configuration engine to access HSE for authentication and decryption of the secure bitstream during a dry run, the CRE Module IP is rendered non-functional after the dry run completes in any design that includes it. The IP remains non-functional until the FPGA is reconfigured by a REFRESH event or power cycle.

References

- CrossLink-NX Family Data Sheet (FPGA-DS-02049)
- Certus-NX Family Data Sheet (FPGA-DS-02078)
- CertusPro-NX Family Data Sheet (FPGA-DS-02086)
- CrossLink-NX-33 and CrossLinkU-NX Data Sheet (FPGA-DS-02104)
- CrossLink-NX web page
- Development Kits and Boards for CrossLink-NX
- IP and Reference Designs for CrossLink-NX
- Certus-NX web page
- Development Kits and Boards for Certus-NX
- IP and Reference Designs for Certus-NX
- CertusPro-NX web page
- Development Kits & Boards for CertusPro-NX
- IP and Reference Designs for CertusPro-NX
- Minimizing System Interruption During Configuration Using TransFR Technology (FPGA-TN-02198)
- Advanced Configuration Security User Guide for Nexus Platform (FPGA-TN-02176)
- Multi-Boot Usage Guide for Nexus Platform (FPGA-TN-02145)
- I3C Controller IP Core (FPGA-IPUG-02228)
- Lattice Memory Mapped Interface and Lattice Interrupt Interface (FPGA-UG-02039)
- CRE Module IP User Guide (FPGA-IPUG-02067)
- MIPI I3C Basic v1.0
- Lattice Radiant FPGA design software
- Lattice Insights 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 3.6, February 2026

Section	Change Summary
Definition of Terms	Updated command name in Dry-Run definition to *LSC_DEVICE_CTRL*.
Introduction	Added reference to MachXO5-NX device family and link to its programming and configuration user guide.
Configuration Details	In Table 4.3. Slave Configuration Port Activation Key: • Updated Note 1 to indicate that dummy bytes before the activation key are optional for the slave SPI port.

Configuration Modes	• In the Master SPI Modes section:
	  • Added discussion on use of 32-bit addressing and commands.
	• In Table 6.13. 64-bit Device Status Register:
	  • Updated bit 6 to *Erase Enable* and related comments.
	  • Added comment to bits 20, 21, and 22 regarding applicability if the bitstream is loaded from MSPI.
	  • Updated bit 41 to reserved.
	• In Table 6.17. Slave Configuration Interface Command:
	  • Updated command name to *LSC_DEVICE_CTRL* and made minor editorial changes in description.
	  • Removed note *For Certus-NX, CrossLink-NX, CertusPro-NX, and MachXO5-NX devices only* applied to commands LSC_PROG_ECDSA_PUBKEY and LSC_READ_ECDSA_PUBKEY.
	  • Added note on LSC_WRITE_COMP_DIC and LSC_PROG_INCR_CMP being commands supported within compressed bitstream only.
	• In the LSC_DEVICE_CTRL section:
	  • Removed statement referencing internal specifications.
	  • Removed description of method to test the dual boot setup.
	• In the LSC_PROG_CNTRL0 section:
	  • Updated values for mask example.
	• In Table 6.24. Data Fields Definition for LSC_PROG_FEATURE Command:
	  • Updated information for bits 62 and 63.
	• In Table 6.26. Data Fields Definition for LSC_PROG_OTP Command:
	  • Added Note 6 on programmability of bits 8 through 14 through JTAG, SSPI, I2C/I3C ports but not CONFIG_LMMI interface for selected devices.
	• In Table 6.31. JTAG Instruction Table:
	  • Updated JTAG instruction name to *LSC_DEVICE_CTRL*.
	  • Updated description of Bit [6:5] for LSC_DEVICE_CTRL.
	  • Removed note *For Certus-NX, CrossLink-NX, CertusPro-NX, and MachXO5-NX devices only* applied to commands LSC_PROG_ECDSA_PUBKEY and LSC_READ_ECDSA_PUBKEY.
	  • Added note on LSC_WRITE_COMP_DIC and LSC_PROG_INCR_CMP being commands supported within compressed bitstream only.

Software Selectable Options	• In Table 7.1. sysCONFIG Options1:
	  • Added Note 3 on unavailability of DISABLE setting for PROGRAMN_PORT option for a specific device.
	• In the PROGRAMN_PORT section:
	  • Added note on unavailability of DISABLE setting for PROGRAMN_PORT option for a specific device.

Appendix B. Nexus Bitstream File Format	• In Table B.5. Nexus Device Specifics:
	  • Removed MachXO5-NX devices.
	• In Table B.6. Nexus Device ID:
	  • Removed MachXO5-NX devices.

Appendix D. Configuration Access from User Logic	• In the CONFIG_LMMI Access Protocol section: • Added guideline on the required minimum number of clock cycles between commands. • In Table D.3. CONFIG_LMMI Supported Command List:

Section	Change Summary
	• Updated command name to *LSC_DEVICE_CTRL* and made minor editorial changes in description. • Removed note *For Certus-NX, CrossLink-NX, CertusPro-NX, and MachXO5-NX devices only* applied to commands LSC_PROG_ECDSA_PUBKEY and LSC_READ_ECDSA_PUBKEY.
Appendix H. Dry Run User Image	Added new section.
References	Added CRE Module IP User Guide.

Revision 3.5, July 2025

Section	Change Summary
Abbreviations in This Document	Added *FSM* and *RTL*.
Configuration Details	In Table 4.1. Maximum Configuration Bits: • Updated header to *Uncompressed*. • Updated Recommended SPI Flash Size and Dual Boot Recommended SPI Flash Size for LFCPNX-50 device with *No LRAM, No EBR* and *No LRAM, MAX EBR* scenarios. • Added LIFCL-33U, LFD2NX-15, LFD2NX-25, LFD2NX-35, and LFD2NX-65 devices.

Configuration Modes	• Updated Figure 6.4. One Quad SPI Flash Interface (Quad) by changing connection between MISO/MD1 and SO/SIO1 to bidirectional.
	• In Table 6.8. Execution Time for ISC_ERASE Command:
	  • Added LIFCL-33U, LFD2NX-15, LFD2NX-25, LFD2NX-35, and LFD2NX-65 devices.
	  • Removed LFMXO5-25, LFMXO5-55T, and LFMXO5-100T devices.

Appendix B. Nexus Bitstream File Format	• In Table B.5. Nexus Device Specifics:
	  • Updated Configuration Frames, Actual Bits per Frame, and Total Data Bits per Frame for LIFCL-17, LFD2NX-9, and LFD2NX-17 devices.
	  • Updated Actual Bits per Frame and Total Data Bits per Frame for LIFCL-33 and LIFCL-33U devices.
	  • Added values for LFCPNX-50 device.
	  • Updated Byte Bound Padding Bits and Actual Bits per Frame for LFCPNX-100 device.
	  • Added LFD2NX-15, LFD2NX-25, LFD2NX-35, LFD2NX-65, LFMXO5-15D, LFMXO5-25, LFMXO5-35, LFMXO5-35T, LFMXO5-55T, LFMXO5-55TD, LFMXO5-65, LFMXO5-65T, and LFMXO5-100T devices.
	• In Table B.6. Nexus Device ID:
	  • Removed LIFCL-40-ES device.
	  • Added LFD2NX-15, LFD2NX-25, LFD2NX-35, and LFD2NX-65 devices.
	  • Added LFMXO5-15D, LFMXO5-25, LFMXO5-35, LFMXO5-35T, LFMXO5-55T, LFMXO5-55TD, LFMXO5-65, LFMXO5-65T, and LFMXO5-100T devices.

Appendix D. Configuration Access from User Logic	• Removed references to *A/B* throughout section. • Updated description on use of CONFIG_LMMI primitives. • Updated Figure D.1. CONFIG_LMMI Primitive Pin Diagram and title by removing reference to *CONFIG_LMMIA/B* and *A/B*. • In Table D.1. CONFIG_LMMI Primitive Input/Output Ports1: • Added note on maximum clock frequency for LMMICLK. • Updated Figure D.2. CONFIG_LMMI Primitive and OSC/OSC for CRE IP Connection (hf_clk_out_o as Clock Source) by adding *lmmi_resetn_o* label, removing reference to *CONFIG_LMMIA/B,* and updating title. • Added description for alternative clock source. • Added Figure D.3. CONFIG_LMMI Primitive and OSC/OSC for CRE IP Connection (Alternative Clock Source).

Revision 3.4, April 2025

Section	Change Summary
Abbreviations in This Document	Added *CRAM, SDM*, and *SEI*.
Configuration Details	• Updated Figure 4.2. Configuration from PROGRAMN Timing by adding t_{INIT_HIGH} label.

Section	Change Summary
	• Updated Figure 4.3. Configuration Error Notification by adding t_{INIT_LOW} label.
Configuration Modes	Updated Figure 6.24. Slave SPI Configuration Flow by adding *enable dual/quad mode* to flow and Note 2 to figure.

Software Selectable Options	• In the BACKGROUND_RECONFIG section:
	  • Updated description.
	  • Added Table 7.2. BACKGROUND_RECONFIG Options.
	• In the MCLK Frequency section:
	  • Updated the confidence interval for clock frequency to ±7%.
	• Updated description in the TRANSFR section.
	• Added note on CONFIG_IOSLEW setting in relation to MCCLK_FREQ in the CONFIG_IOSLEW section.

Appendix B. Nexus Bitstream File Format	In the Configuration Bitstream Format section: • Corrected cross reference to Table B.1. Nexus Compressed Bitstream Format. • Added introducing sentence for Table B.2. Nexus Uncompressed Bitstream Format – Without Early I/O Release and Table B.3. Nexus Uncompressed Bitstream Format – With Early I/O Release. • Updated description with preamble values when the bitstream is encrypted.

Appendix C. Modifying the Nexus Feature Row	• Added information on feature row OTP EFUSE and pseudo EFUSE (shadow register) programming.
	• Added the Breakdown of Feature Row Bits section.
	• In the Modifying the Feature Row Using the .fea Programming File section:
	  • In Step 4:
	  • Updated description.
	  • Added description for Feature Rows Programming under Access Mode.
	  • Added Feature Rows Pseudo Programming under Access Mode.
	  • Added Figure C.5. Feature Row Pseudo Programming Setup.
	  • Updated note in Step 6 to emphasize that Feature Rows Programming for OTP EFUSE is non-reversible.
	• In the Modifying the Feature Row without Using the .fea Programming File section:
	  • In Step 3:
	  • Added description for Feature Rows Programming under Access Mode.
	  • Added Feature Rows Pseudo Programming under Access Mode.
	• Added the Programming No CDM Bit in CR0 section.
	• Added the Comparison of Feature Row Programming File (.fea) section.

Revision 3.3, January 2025

Section	Change Summary
All	Updated terms SPIM address and SPIM commands to MSPI address and MSPI commands, respectively, throughout document except in the Setting 32-bit SPI Address and Command in Non-Volatile Configuration Memory section.
Abbreviations in This Document	• Updated section title, description, and table header. • Added *CDM*, *HSE*, *MSI*, and *SRME*.

Configuration Details	• In Table 4.4. Default State of the sysCONFIG Pins:
	  • Updated note on SD[15:4].
	• In the MCLK section:
	  • Added 75 MHz to Table 4.5. Nexus MCLK Valid Frequencies.
	  • Added guidelines for selecting MCCLK_FREQ and guides for setting RX Edge.

Master Port Configuration Process and Flow	Updated description on signature verification process when loop timer expires.
Configuration Modes	• In the Master SPI Modes section: • Added Table 6.2. Bitstream Read Command Codes with MSPI Controller in 24-bit and 32-bit Addressing Modes and introducing sentence.

Section	Change Summary

	• In the Dual-Boot and Multi-Boot Configuration Modes section:
	  • Updated statement to mention jump table if the FPGA fails to configure.
	  • Added note on need to set 32-bit MSPI address and 32-bit MSPI commands to 1 if configuration memory is 256 Mb or larger.
	• In the Ping-Pong Boot section:
	  • Added note on need to set 32-bit MSPI address and 32-bit MSPI commands to 1 if configuration memory is 256 Mb or larger.
	• In the Slave SPI Configuration Flow Diagrams section:
	  • Updated Figure 6.24. Slave SPI Configuration Flow by adding note on reference to the LSC_BITSTREAM_BURST section and Figure 6.44. Segmented Bitstream Burst Timing Waveforms.
	• In Table 6.13. 64-bit Device Status Register:
	  • Updated default values to match the number of field bits.
	  • Updated field name for *TRAN Mode (Transparent Mode)*, *CONFIG Target Selection*, *UID EN*, *Encrypt PreAmble*, *STD PreAmble*, *SPIm Fail*, and *EXEC Error*.
	  • Added fields *WDT Reboot*, *SDM Enable*, *Lattice PreAmble*, *UDS Programmed*, *Version*, and *Slave SPI Timeout*.
	  • Updated field name and description for *BSE Timeout*.
	• In Table 6.14. Bit Definition for Control Register 0:
	  • Updated default values to match the number of field bits.
	  • Updated field name for *Wake Up Trans*, *TranCRAM*, *TranEBR*, *TranIP*, and *Slew Rate Control (for Config output pins)*.
	  • Added fields "No CDM", "TranHSE", "No Boot", "SRME", "INIT OPT", "DONE OPT", "STX DUM", and "Config IO Voltage".
	  • Updated field name and description for *Master Clock Frequency (On-chip Flash Clock Select)*.
	• In Table 6.15. Bit Definition for Control Register 1:
	  • Updated default values to match the number of field bits.
	  • Added fields *Core CLK Sel*, *Master Signature Timer Count*, *I3C FILT*, *DPA Enable*, *TX Edge*, *RX Edge*, *LSBF*, *SSPI Auto*, *Bulk Erase Enable*, *CFG Noise*, *Slave Idle Timer Count*, and *Signature Infinite Retry*.
	  • Updated field name for *32-bit MSPI Address*, *32-bit MSPI Commands*, *Disable IO Glitch*, and *Signature Disable*.
	  • Updated field name and description for *Master Timer Count* and *Master Retry Count*.

Software Selectable Options	• In Table 7.1. sysCONFIG Options1:
	  • Added *75* to MCCLK_FREQ settings.
	  • Updated default setting for WAKE_UP attribute to DISABLE_DONE_SYNC.
	• Updated default selection for WAKE_UP preference to DISABLE_DONE_SYNC in the WAKE_UP section.

Daisy Chaining	• Updated description.
	• In the Flow Through Option section:
	  • Updated description.
	  • Removed the Nexus in Configuration Daisy Chain with Slave SPI in Flow Through Mode figure.
	  • Added Table 9.1. Control Bit Settings for FPGA Device in a Daisy Chain.
	• Added the Using Radiant Deployment Tool to Create a sysCONFIG Daisy Chain Hex File, MSPI Configuration Mode in Flow Through Mode, and SSPI Configuration Mode in Flow Through Mode sections.

Appendix D. Configuration Access from User Logic	• Updated description and titles to include CONFIG_LMMIB primitive. • Added important guidelines for using the CONFIG_LMMI primitive in the CONFIG_LMMI(A/B) Access Protocol section.
Appendix F. Enabling 32-bit (4-byte) SPI Address and Command Mode (Dual-Boot/Multi-Boot/Ping-Pong Boot)	Added new section.

Section	Change Summary
Appendix G. Modifying sysCONFIG Options in the Lattice Radiant Device Constraint Editor without Re-running Place and Route Design	Added new section.

Revision 3.2, October 2024

Section	Change Summary
All	Minor editorial changes.
Configuration Details	Updated sysCONFIG pin name from SD[15:8] to SD[15:4] in Table 4.4. Default State of the sysCONFIG Pins.

Configuration Modes	• Added guideline for performing bitstream burst for slave SPI configuration ports in the LSC_BITSTREAM_BURST section.
	• Removed mention of dual and quad support for SSPI to MSPI bridging and updated description in the Slave SPI to Master SPI Bridge section.
	• In the LSC_PROG_SPI (SSPI to MSPI Bridge) section:
	  • Removed description on x2 and x4 modes support.
	  • Removed statement on default mode.
	  • Removed rows related to x2 and x4 modes in Table 6.30. LSC_PROG_SPI Operand.

Revision 3.1, July 2024

Section	Change Summary

Configuration Details	• In Table 4.1. Maximum Configuration Bits of the Bitstream/PROM Sizes section:
	  • Added LFD2NX-9 and LFD2NX-28 devices.
	  • Updated recommended flash sizes for LFCPNX-50 device with "No LRAM, No EBR" scenario.
	• In the INITN section of the sysCONFIG Pins section:
	  • Updated Figure 4.2. Configuration from PROGRAMN Timing.
	  • Updated Figure 4.3. Configuration Error Notification.
	  • Added discussion on restarting configuration after a configuration error as signaled by INITN going low during configuration.

Configuration Modes	• Updated Figure 6.22. Slave SPI Read Waveforms and Figure 6.23. Slave SPI Write Waveforms in the Specifications and Timing Diagrams – Slave SPI Port Waveforms section of the Slave SPI Mode section.
	• Updated Table 6.7. Execution Time for ISC_ERASE Command in the Slave SPI Configuration Flow Diagrams section to include LFD2NX-9 and LFD2NX-28 devices.
	• In the Class A Command Waveforms section of the Command Waveforms section:
	  • Updated description for bit shifted in first.
	  • Updated Figure 6.25. Class A Command Waveforms.
	• In the Class B Command Waveforms section of the Command Waveforms section:
	  • Updated description for bit shifted in first.
	  • Updated Figure 6.26. Class B Command Waveforms.
	• Updated note 3 to Figure 6.41. Slave I2C/I3C Configuration Flow in the Slave I2C/I3C Configuration Flow Diagrams section to add reference to LSC_BITSTREAM_BURST command and include Erase, Program, Verify operation for debug purpose.
	• Added note for LSC_PROG_PASSWORD, LSC_PROG_CIPHER_KEY, LSC_PROG_FEABITS, LSC_PROG_OTP, and LSC_PROG_ECDSA_PUBKEY to indicate that operation programs internal EFUSE memory, which is one-time programmable, in Table 6.16. Slave Configuration Interface Command of the Slave Command Set section.
	• Added note to Table 6.25. Data Fields Definition for LSC_PROG_OTP Command in the LSC_PROG_OTP section of the Slave Command Details section regarding usable instructions or commands when bit 14 (JTAG Port Test Only) is set.
	• Added note for LSC_PROG_PASSWORD, LSC_PROG_CIPHER_KEY, LSC_PROG_FEABITS, LSC_PROG_OTP, and LSC_PROG_ECDSA_PUBKEY to indicate that operation programs

Section	Change Summary
	internal EFUSE memory, which is one-time programmable, in Table 6.30. JTAG Instruction Table of the JTAG Instruction Support section.

Appendix B	• In Nexus Device Specific section:
	  • Updated Table B.5. Nexus Device Specifics to include LFD2NX-9 and LFD2NX-28 devices.
	  • Updated Table B.6. Nexus Device ID to include LFD2NX-9 and LFD2NX-28 devices.

Appendix C	• Added note to the *Target Memory* bullet of step 4 to indicate that operations under the *Non-Volatile Configuration Memory* category is one-time programmable in the Modifying the Feature Row Using the .fea Programming File section. • Added note to the *Target Memory* bullet of step 3 to indicate that operations under the *Non-Volatile Configuration Memory* category is one-time programmable in the Modifying the Feature Row without Using the .fea Programming File section.

Appendix D	• Updated description to reference OSC IP and OSC for CRE IP and user guides, and added LFD2NX-9 and LFD2NX-28 devices to list of devices for CONFIG_LMMI primitive.
	• Removed OSC Primitive section.
	• In CONFIG_LMMI(A) Primitive and OSC/OSC for CRE IP Connection and Instantiation section:
	  • Updated section title from *CONFIG_LMMI(A) Primitive and OSC Primitive Connection and Instantiation* to *CONFIG_LMMI(A) Primitive and OSC/OSC for CRE IP Connection and Instantiation*.
	  • Updated description.
	  • Updated Figure D.2. CONFIG_LMMI(A) Primitive and OSC/OSC for CRE IP Connection and title.
	  • Updated sample RTL design for instantiating the CONFIG_LMMI(A) primitive and OSC IP.
	• Added note for LSC_PROG_PASSWORD, LSC_PROG_CIPHER_KEY, LSC_PROG_FEABITS, LSC_PROG_OTP, and LSC_PROG_ECDSA_PUBKEY to indicate that operation programs internal EFUSE memory, which is one-time programmable, in Table D.3. CONFIG_LMMI(A) Supported Command List of the CONFIG_LMMI(A) Supported Commands section.

Revision 3.0, April 2024

Section	Change Summary
Configuration Details	• Added bitstream and recommended flash sizes for LFCPNX-50 device in Table 4.1. Maximum Configuration Bits in Bitstream/PROM Sizes section. • Updated recommended external pull-down resistor value for TCK/SCLK in TCK/SCLK section.

Configuration Modes	• Minor editorial changes.
	• Figure 6.24. Slave SPI Configuration Flow in Slave SPI Configuration Flow Diagrams section:
	  • Added note 5 regarding external DONE pin and wait time before checking the status register.
	  • Updated flow to include check for AUTH DONE=1 after bitstream clock in.
	  • Updated flow to include "Clock in LSC_READ_STATUS command" operations.
	  • Added note 6 to indicate that AUTH DONE=1 check only applicable if encrypted bitstream is used.
	• Updated CCLK waveform in Figure 6.25. Class A Command Waveforms in Class A Command Waveforms section.
	• Updated CCLK waveform in Figure 6.26. Class B Command Waveforms in Class B Command Waveforms section.
	• Updated CCLK waveform in Figure 6.27. Class C Command Waveforms in Class C Command Waveforms section.
	• Updated CCLK waveform in Figure 6.28. Class D Command Waveforms in Class D Command Waveforms section.
	• Figure 6.41. Slave I2C/I3C Configuration Flow in Slave I2C/I3C Configuration Flow Diagrams section:
	  • Added note 6 regarding external DONE pin and wait time before checking the status register.

Section	Change Summary
	• Updated flow to include check for AUTH DONE=1 after bitstream clock in. • Updated flow to include "Clock in LSC_READ_STATUS command" operations. • Added note 7 to indicate that AUTH DONE=1 check only applicable if encrypted bitstream is used. • Table 6.16. Slave Configuration Interface Command in Slave Command Set section: • Removed Flow Operation Number column. • Removed note on Flow Operation Number. • Updated description for LSC_PROG_SPI. • Updated cross refences for notes 5 to 8. • Updated note 5 to include MachXO5-NX. • Table 6.30. JTAG Instruction Table in JTAG Instruction Support section: • Fixed cross reference formatting for "LSC_PROG_ECDSA_PUBKEY" and "LSC_READ_ECDSA_PUBKEY". • Updated note 1 to include MachXO5-NX.
Appendix D	Table D.5. CONFIG_LMMI(A) Supported Command List in CONFIG_LMMI(A) Supported Commands section: • Removed Flow Operation Number column. • Removed note on Flow Operation Number. • Added note 4 on applicability to Nexus devices. • Updated cross references at "LSC_PROG_ECDSA_PUBKEY" and "LSC_READ_ECDSA_PUBKEY".

Revision 2.9, February 2024

Section	Change Summary
Configuration Details	Updated Table 4.1. Maximum Configuration Bits to add SPI flash size footnote.
Configuration Modes	Clarified default I^2C and I3C slave address in Slave Address for I2C and I3C Ports section.
Appendix E. Dual Boot Fail Safe Upgrade	Newly added section.

Revision 2.8, January 2024

Section	Change Summary
Disclaimers	Updated the disclaimer.
Acronyms in This Document	Added a new acronym definition for QE.
Configuration Details	• Updated Table 4.4. Default State of the sysCONFIG Pins to include a new pin *SD[15:4]* and table note 8. • Added the following description in the 4.4.3 DONE section: *After the internal DONE bit is asserted, the GPIO will wake-up and the external DONE pin will go high after 10 µs. This means that the GPIO pin setting will be applied before the DONE pin goes high.*
Master Port Configuration Process and Flow	• Updated the following sentence *During the power-up sequence phase, once the POR circuit is active, the POR circuit makes sure that the external I/O pins are in a high-impedance state.* in the 5.1 Power-Up Sequence section. • Updated the following sentence *This initialization phase starts after the power-up sequence. In the power-up sequence phase, the GPIO of the device* • *defaults to tri-stated outputs with active weak input pull-downs.* in the 5.2 Initialization section.
Configuration Modes	• Added the following new subsections: • 6.1.4.1 Enabling Dual and Quad Master SPI Read Mode • 6.1.4.2 Using the Radiant Programmer to Program the External SPI Flash via the JTAG Port • 6.1.4.3 Using the Radiant Programmer to Program Flash Quad Enable Bit via the JTAG Port • Updated the configuration interface command names in Table 6.16 Slave Configuration Interface Command.

Section	Change Summary
	• Updated note 1, 2, and 3 in Table 6.16. Slave Configuration Interface Command. • Improved Figure 6.24. Slave SPI Configuration Flow and added figure notes 1 and 2. • Updated note 2 to state an exception for the ISC_ERASE command in section 6.3.5.2 Slave I3C Configuration Access Flow. • Improved Figure 6.41. Slave I2C/I3C Configuration Flow and added figure notes 1, 2, 3, 4, and 5. • Added Table 6.7. Execution Time for ISC_ERASE Command. • Updated the following sentence *The 1-bit busy flag is corresponding to bit 7 of the return data byte* in section 6.9.3.6 LSC_CHECK_BUSY.
Appendix D. Configuration Access from User Logic	Corrected *Delay time* to *Execution time* in Table D.5. CONFIG_LMMI(A) Supported Command List.

All	• Updated the following:
	  • Replaced all LSC_PROG_CTRL0 with *LSC_PROG_CNTRL0*
	  • Replaced all LSC_PROG_CTRL1 with *LSC_PROG_CNTRL1*
	  • Replaced all LSC_READ_CTRL0 with *LSC_READ_CNTRL0*
	  • Replaced all LSC_READ_CTRL1 with *LSC_READ_CNTRL1*
	• Corrected all *wait time* to *execution time*.

Revision 2.7, September 2023

Section	Change Summary
Acronyms in This Document	Newly added BSE.
Introduction	Updated the referenced document title to the current CrossLink-NX-33 and CrossLinkU-NX Data Sheet (FPGA-DS-02104).
Features	Updated Ultrafast I/O to *Ultrafast Early I/O (EIO)*.
Configuration Details	Updated the document reference to the current CrossLink-NX-33 and CrossLinkU-NX Data Sheet (FPGA-DS-02104) in the following subsections: • 4.4.1 PROGRAMN • 4.5.1. MCLK • 4.6.1. TCK/SCLK
Master Port Configuration Process and Flow	• In Section 5.4 Wake-Up, updated FPGA's I/O to *FPGA I/O*. • Updated the title of Section 5.5 from Early I/O Release to *Early I/O Release (EIO)*.

Configuration Modes	• Added the document reference to the current CrossLink-NX-33 and CrossLinkU-NX Data Sheet (FPGA-DS-02104) in the following subsections:
	  • 6.2.3. Slave SPI Port AC Timing Requirements
	  • 6.3.7. Slave I2C/I3C AC Timing Requirements
	  • 6.4.2. JTAG Port AC Timing Requirements
	• Slave I2C/I3C Mode section: updated the table reference to Table 6.16 in the first paragraph.
	• Moved the Slave I3C Configuration Access Flow section under the Slave I2C/I3C Configuration Logic Access section.
	• Updated the LSC_BITSTREAM_BURST section including newly added Figure 6.44. Continuous Bitstream Burst, Figure 6.45. Segmented Bitstream Burst with No Extra Clock Edge on SCL, and Figure 6.46. Segmented Bitstream Burst when Extra SCL Edge Unavailable.
	• Corrected the table caption to the current one for Table 6.18. OPERAND1 Definition for LSC_REFRESH Command, Table 6.19. OPERAND1 Definition for ISC_ENABLE Command, Table 6.21. OPERAND Definition for LSC_PROG_INCR Command, and Table 6.22. OPERAND Definition for LSC_READ_INCR Command.

Appendix B. Nexus Bitstream File Format	Added the CrossLinkU-NX Device Information to Table B.5. Nexus Device Specifics and Table B.6. Nexus Device ID.
References	Added the following to this section: • CrossLink-NX Family Data Sheet (FPGA-DS-02049) • Certus-NX Family Data Sheet (FPGA-DS-02078)

Section	Change Summary
	• CertusPro-NX Family Data Sheet (FPGA-DS-02086) • CrossLink-NX-33 and CrossLinkU-NX Data Sheet (FPGA-DS-02104) • Development Kits & Boards for CrossLink-NX • IP & Reference Designs for CrossLink-NX • Development Kits & Boards for Certus-NX • IP and Reference Designs for Certus-NX • Development Kits & Boards for CertusPro-NX • IP and Reference Designs for CertusPro-NX • Minimizing System Interruption During Configuration Using TransFR Technology (FPGA-TN-02198) • Advanced Configuration Security User Guide for Nexus Platform (FPGA-TN-02176) • Multi-Boot Usage Guide for Nexus Platform (FPGA-TN-02145) • I3C Controller IP Core (FPGA-IPUG-02228) • Lattice Memory Mapped Interface and Lattice Interrupt Interface (FPGA-UG-02039) • MIPI I3C Basic v1.0 • Lattice Radiant FPGA design software • Lattice Insights for Lattice Semiconductor training courses and learning plans

Revision 2.6, June 2023

Section	Change Summary
All	Minor adjustments in formatting across the document.
Features	Updated document link and title of Advanced Configuration Security Usage Guide for Nexus Platform (FPGA-TN-02176) to Advanced Configuration Security User Guide for Nexus Platform (FPGA-TN-02176).
Configuration Details	Updated Table 4.1. Maximum Configuration Bits to change SPI Flash size of LFCPNX-100 to 32 and 64.
Configuration Modes	• Updated SCSN information to Release PROGRAMN, then wait for INITN to go high. After INITN goes high, drive the SCSN of the Slave SPI Port and shift to the Slave Configuration Port Activation Key in the Method to Enable the Slave SPI Port section. • Updated Figure 6.24. Slave SPI Configuration Flow to add Wait for INITN to go HIGH process. • Added reference to I3C Controller document in Slave I2C/I3C Command Format section.
References	Added this section.

Revision 2.5, April 2023

Section	Change Summary
Configuration Modes	Changed Fast-Read Dual Output (BBh) to Fast-Read Dual I/O (BBh) and Fast-Read Quad Output (EBh) to Fast-Read Quad I/O (EBh) in the Dual and Quad Master SPI Read Mode section.

Revision 2.4, March 2023

Section	Change Summary
Acronyms in This Document	Added CIB and PLC.
Configuration Modes	Enhanced the I3C Slave Port Capabilities section. • Added statement regarding compliance with the MIPI I3C (Improved Inter Integrated Circuit) standard v1.0. • Added PID, BCR, and DCR information.
	Enhanced the Slave Command Support section. • Added headings for Slave Command Format and Slave Command Set subsections. • Updated Table 6.16. Slave Configuration Interface Command. • Added ISC Qualify column for identifying command type, added ISC_ERASE_DONE,

Section	Change Summary
	removed JUMP, SSPI_PORT_CTRL, LSC_IO_CONTROL, LSC_AUTH_CTRL, and adjusted footnotes. • Added the Slave Command Details section.
Appendix D. Configuration Access from User Logic	• Updated the CONFIG_LMMI(A) Access Protocol section with new waveform for Class A, B, C commands. • Added the CONFIG_LMMI(A) Supported Commands for supported command list.
All	Minor adjustments in formatting and style.

Revision 2.3, February 2023

Section	Change Summary
Configuration Modes	• Changed section heading to Slave Command support. • Added Non-JTAG Command Format description at the beginning of the Slave Command Support section. • Updated Table 6.14. Slave Command Support. • Added Data (Bytes) column for data length information and modified Operand (Bytes) unit from bits to bytes.
Appendix D. Configuration Access from User Logic	Added this section.
All	Minor adjustments in formatting and style.

Revision 2.2, January 2023

Section	Change Summary
Configuration Modes	Added the ISC_ERASE command in Figure 6.9. Slave SPI Configuration Flow.

Revision 2.1, December 2022

Section	Change Summary
Configuration Details	Added pin description in the PROGRAMN section.
Technical Support Assistance	Added reference to the Lattice Answer Database on the Lattice website.
All	Minor adjustments in format and style.

Revision 2.0, October 2022

Section	Change Summary
Configuration Modes	Updated Reveal Debug bullet point in JTAG Mode to add reference to Appendix A – Reveal User Guide in the Lattice Radiant user guide.

Appendix B. Nexus Bitstream File Format	• Updated the following in Table B.1. Nexus Compressed Bitstream Format to Table B.3. Nexus Uncompressed Bitstream Format – With Early I/O Release:
	  • Removed the different gray shadings.
	  • Removed footnote that mentions “Only the data frame bits highlighted in yellow can be compressed” and adjusted/corrected footnote references in the tables.

Revision 1.9, August 2022

Section	Change Summary
All	Minor adjustments in formatting across the document.

Configuration Details	• Updated the following in Table 4.1. Maximum Configuration Bits:
	  • Changed LFCPNX-100 (No LRAM, No EBR) bitstream size from 15.005 to 17.293.
	  • Updated table note to include that the device supports the use of compression and encryption as long as these are not implemented in a single bitstream simultaneously.

Configuration Modes	Updated Slave I3C Configuration Access Flow section to add footnote 3 that the Nexus device is compliant to I3C CCC.

Section	Change Summary
Appendix C. Modifying the Nexus Feature Row	Added this section.

Revision 1.8, June 2022

Section	Change Summary
All	Added document reference to CrossLink-NX-33 Data Sheet (FPGA-DS-02104) across the document.
Configuration Details	Updated Table 4.1. Maximum Configuration Bits to add LIFCL-33 device.
Configuration Modes	Corrected document reference to Minimizing System Interruption During Configuration Using TransFR Technology (FPGA-TN-02198) in TransFR Operation section.
Software Selectable Options	Corrected document reference to Minimizing System Interruption During Configuration Using TransFR Technology (FPGA-TN-02198) in TRANSFR section.
Appendix B. Nexus Bitstream File Format	Updated Table B.5. Nexus Device Specifics and Table B.6. Nexus Device ID to add information for CrossLink-NX-33.

Revision 1.7, March 2022

Section	Change Summary
Software Selectable Options	Added notes for Device Constraint Editor Global Bank VCCIO setting in section 7.13.

Revision 1.6, January 2022

Section	Change Summary
All	Change document title from sysCONFIG Usage Guide for Nexus Platform to sysCONFIG User Guide for Nexus Platform.
Definition of Terms	Updated Refresh definition.
Configuration Details	• Updated Table 4.4 to add Hardware and Software Default columns and table notes 5 to 7. • Updated bullet list in PROGRAMN.
Configuration Modes	• Added Slave I^2C/I3C Configuration Flow Diagrams and Slave I^2C/I3C Command Format sections. • Updated information in Method to Enable the Slave SPI Port and Method to Enable the Slave I^2C/I3C Port. • Changed 4.7K to 4.7k in Figure 6.1, Figure 6.3, and Figure 6.4. • Updated reference to table note in Table 6.4.
Software Selectable Options	• Updated Figure 7.1. • Updated JTAG, DONE, and INITN_PORT default settings and added table note in Table 7.1. • Updated information in JTAG_PORT, DONE_PORT, INITN_PORT, and PROGRAMN_PORT.

Revision 1.5, September 2021

Section	Change Summary
Software Selectable Options	• Corrected the spelling of BOOTMODE in Table 7.1. sysCONFIG Options. • Corrected the spelling of four states in Section 7.8. BACKGROUND_RECONFIG.

Revision 1.4, July 2021

Section	Change Summary
Acronyms in This Document	Added definition for JEDEC and SFDP.
Master Port Configuration Process and Flow	Updated the Configuration section to add signature verification information.
Configuration Details	Updated Figure 4.1 and sysCONFIG Pins section to add I3C.

Section	Change Summary
Configuration Modes	Updated Table 6.8 to add I3C.

Revision 1.3, June 2021

Section	Change Summary
All	• Minor adjustment in formatting. • Added CertusPro-NX support, including reference to CertusPro-NX Family Datasheet, across the document. • Changed AC Timing to sysCONFIG Port Timing Specification, JTAG_ENABLE to JTAG_EN, GSR to GSRN, GWDIS to GWE, and SLAVE_I2C/I3C_PORT to SLAVE_I2C_PORT/SLAVE_I3C_PORT references across the document. • Changed MCLK_FREQ to MCCLK_FREQ across the document.
Acronyms in This Document	Added definition for MIB.
Features	Added CertusPro-NX and CrossLink-NX support in Bitstream Authentication HMAC and ECDSA.
Definition of Terms	Updated content to add I^2C and I3C in Port.
Configuration Details	• Updated Table 4.1. • Updated content in MCSNO/MSDO for MSDO. • Updated Master SPI Modes section to remove CCLK support.
Master Port Configuration Process and Flow	Updated conditions in Power-up Sequence.
Configuration Modes	• Updated content in Master SPI Modes. • Changed 32 to 64-bit in Table 6.11. • Updated Table 6.12, Table 6.13, Table 6.14, and Table 6.15.
Software Selectable Options	• Updated Figure 7.1 and Table 7.1. • Changed 112 to 150 MHz in MCLK Frequency. • Added NOT_SPECIFIED/1.2V/1.5V and changed CONFIG_IOVOLTAGE to CONFIGIO_VOLTAGE_BANK0/CONFIGIO_VOLTAGE_BANK1 in CONFIGIO_VOLTAGE_BANK0/CONFIGIO_VOLTAGE_BANK1 section.
Device Wake-up Sequence	Updated Table 8.1.
Daisy Chaining	Updated content, including Figure 9.1.
Appendix A. Nexus Slave SPI Programming Guide	Changed CCLK to SCLK.
Appendix B. Nexus Bitstream File Format	• Changed reference to CrossLink-NX in Table B.1 to Table B.3. • Updated Table B.5 and Table B.6.

Revision 1.2, March 2021

Section	Change Summary
All	Changed CSNO and SDO to *MCSNO* and *MSDO* across the document.
Features	Updated content to change document title, number, and link of Advanced Security Encryption Key Programming Guide for Nexus Platform (FPGA-TN-02126) to *Advanced Configuration Security Usage Guide for Nexus Platform (FPGA-TN-02176)*.
Daisy Chaining	Updated Figure 9.3 and Figure 9.4.

Revision 1.1, June 2020

Section	Change Summary
All	• Changed document name from CrossLink-NX sysCONFIG Usage Guide to *sysCONFIG Usage Guide for Nexus Platform*. • Added Nexus device across the document. • Fixed formatting across the document.
Features	Updated section content.

Section	Change Summary
Configuration Details	• Updated Figure 4.1 in Configuration Ports Arbitration. • Updated Table 4.1.
Master Port Configuration Process and Flow	Updated content of Early I/O Release.
Configuration Modes	Updated Table 6.11, Table 6.13, Table 6.14, and Table 6.15.
Software Selectable Options	• Updated heading numbers. • Updated Table 7.1 to remove 1.0 V, 1.2 V, and 1.5 V from the configuration port support. • Removed 1.0 V, 1.2 V, and 1.5 V in CONFIG_IOVOLTAGE section.

Revision 1.0, December 2019

Section	Change Summary
All	Initial release

www.latticesemi.com