Lattice Sentry Demo Board for Mach-NX Devices

User Guide

FPGA-UG-02167-1.1

March 2024

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 REDUDANCIES, 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

Figures

Tables

Acronyms in This Document

A list of acronyms used in this document.

Acronym	Definition
BMC	Baseboard Management Controller
BSP	Board Support Package (File with address map and driver functions automatically generated in Propel SDK for firmware)
CFG	Configuration flash sector on Mach-NX. PFR Diamond SOC design jedec file (from Diamond) is programmed to this sector.
DICE	Device Identifier Composition Engine (Security certificate optionally used in Lattice PFR Solution)
ECDSA	Elliptic Curve Digital Signature Algorithm (Security algorithm optionally used in Lattice PFR Solution)
ECP5	Lattice FPGA device
FAM	Flash Address Map
FCH	Fusion Controller Hub (equivalent to PCH)
FPGA	Field Programmable Gate Array
FST	Firmware Signing Tool
FTDI	Future Technology Devices International (Chip on board to communicate with Programmer via USB/UART cable)
FW	Firmware
Gb	Gigabit
HMAC	Hash-based Message Authentication Code (Security algorithm optionally used in Lattice PFR Solution)
I2C	Inter-integrated circuit (communication protocol)
IP	Intellectual property
JED/JEDEC	Joint Electron Device Engineering Council (Type of file generated by Lattice Diamond and programmed into an FPGA)
JTAG	Joint Test Action Group
Mach-NX	Lattice FPGA device
PCH	Platform Controller Hub
PFR	Platform Firmware Resilience
PUBKEY	Public Key sector of Mach-NX device
QSPI	Quad Serial Peripheral Interface
RISC-V	Reduced Instruction Set Computer
RoT	Root of Trust
SFB	SoC Function Block
SH	Type of script file
SMBus	System Management Bus
SoC	System-on-Chip
SPI	Serial Peripheral Interface
SRAM	Static Random Access Memory
UART	Universal Asynchronous Receiver Transmitter
UFM	User Flash Memory
USB	Universal Serial Bus

1. Introduction

This document is a guide on how to set up and use the Lattice Sentry Demo Board for Mach™-NX Devices.

The topics covered are:

- Setting up the software environment and tools.
- Setting up the Lattice Propel™ and Lattice Diamond™ projects.
- Creating or importing a workspace in the Lattice Propel Software.
- Generating a public/private key pair and signing files using the Firmware Signing Tool.
- Programming the demo board.
- Troubleshooting common board programming issues.
- Testing the template design on the demo board.

2. Demo Requirements

2.1. Hardware Requirements

List of hardware requirements for the Lattice Sentry Demo Board for Mach-NX Devices.

- Mach-NX LFMNX-50-5FBG484C
- ECP5™ LFE5U-85F-8BG381C
- The demo board features two ECP5 devices:
  - ECP5 (BMC), which simulates the BMC (board management controllers) behavior.
  - ECP5 (PCH), which simulates the PCH (platform controller hub) behavior.
- USB connection for device programming.
- USB connection for UART communication to PFR firmware.
- Four 1-Gb SPI/QSPI flash devices for device configuration and firmware image storage (two flash devices per ECP5 device).
- Flexible expansion Raspberry Pi header.
- LEDs, switches, and push buttons for FPGAs.

2.2. Software Requirements

The following software are required for the steps outlined in this guide.

2.2.1. Lattice Software Tools

To ensure correct functionality between software, uninstall Lattice Diamond 3.12 and Lattice Propel 1.1 if they are already installed on your machine.

Install the software in the following order:

Lattice Diamond Software:

1. Lattice Diamond Software.
   a. Lattice Diamond ver. 3.12.0.240.2.
   b. Lattice Diamond Service Pack ver. 3.12.1.454.2.
   c. Lattice Diamond Patch 3.12.1.1006.0**.** This is only required for Intel Processors 11th gen onwards.
      - If the Lattice Diamond still shows error code 9 during synthesis, troubleshoot by replacing **lfmnx.v** file in C:\lscc\diamond\3.12\synpbase\lib\lucent
   d. Lattice Diamond Security Encryption Pack ver. 3.12.1.454.2.

Note that the Lattice Diamond Security Encryption Pack ver. 3.12.1.454.2 must be the last patch installed. If another patch is installed after it, it must be reinstalled.

2. Lattice Diamond Programmer.
   a. Lattice Diamond Programmer ver. 3.13.0.56.
   b. Lattice Diamond patch 3.13.0.56.2 131818.
   c. Lattice Diamond Programmer Encryption Security Patch ver. 3.13.0.56.

3. Lattice Propel Software.
   a. Lattice Propel ver. 1.1.
   b. Lattice Sentry Propel ver. 1.1 patch to enable the Mach-NX device.

Please contact Lattice FAE or Apps for a patch.

2.2.2. Firmware Signing Tool Software Requirements

The *Firmware Signing Tool (FST)* utility is available upon request. The FST utility is used to generate the signed software image for the RISCV and the signed config files, as outlined in Section 3.4 and in the Firmware Signing Tool User Guide.

Dependencies:

- Windows 10 or above or suitable Bash shell environment.
  - Cygwin64 Terminal can provide a shell environment for Windows: https://www.cygwin.com/install.html
    - This guide used Cygwin 3.4, x86_64-pc-cygwin
- Python 3.6 or above: https://www.python.org/downloads/
  - Install Python 3 during Cygwin64 installation.
- OpenSSL available in path (tested with OpenSSL 1.1.1f).
- OpenSSL 22.0.0
  - Install OpenSSL during Cygwin64 installation.

2.2.3. Firmware Signing Tool Files

The FST utility is a collection of Python scripts and bash scripts, which together are used to sign the firmware and bit file images. The Python scripts are not run by the user but are included in the two Bash scripts. The FST utility is used by running the two Bash scripts in a shell environment, such as Cygwin.

Table 2.1. Files Included in the FST Utility

File name	Type	Description
cfg_bin_sig_replace.py	Python script	Replace signature in full config bin and bit image
cfg_binjed_sig_replace.py	Python script	Convert insert bin signature to jed files
combine_fw_and_sig.py	Python script	Combine firmware bin files and signature
convert_bit_to_sign_bin.py	Python script	Convert bit files to binary files that can be signed
convert_fw_mem_to_bin.py	Python script	Convert mem files to binary files that can be signed
convert_jed_to_bin.py	Python script	Convert jed files to binary files that can be signed
openssl_DER_sig_to_raw.py	Python script	Convert OpenSSL output DER files to raw binary signature files
sign_sentry_firmware_ecdsa.sh	Bash script	Sign batch File for ECDSA
sign_sentry_firmware_hmac.sh	Bash script	Sign batch File for HMAC
README_ECDSA.md	Text readme	Readme File for ECDSA
README_HMAC.md	Text readme	Readme File for HMAC
FW Signing Tool User Guide	PDF	User Guide

2.3. Required Licenses

The required list of licenses are shown in Table 2.2.

Table 2.2. Required Licenses

License String	Description	OPN	Device
LSC_CTL_LFMNX-50	Device License	N/A	Mach-NX
LSC_SW_LFMNX_HARDIP_ESBA	Feature License	N/A	Mach-NX
LSC_SW_LFMNX_SECURITY_LOCK	Feature License	N/A	Mach-NX
LSC_SW_LFMNX_SECURITY_ENCRYPT	Feature License	N/A	Mach-NX
LSC_IP_sfb_if_pfr_mnx	SFB Interface IP License	SENTRY-PFR-LFMNX-U/ SENTRY-PFR-LFMNX-UT	Mach-NX
LSC_IP_i2c_filter_pfr_mnx	SMBus/I2C Filter IP License	SENTRY-PFR-LFMNX-U/ SENTRY-PFR-LFMNX-UT	Mach-NX

License String	Description	OPN	Device
LSC_IP_smbus_mb_pfr_mnx	SMBus Mailbox IP License	SENTRY-PFR-LFMNX-U/ SENTRY-PFR-LFMNX-UT	Mach-NX
LSC_CTL_PROPSDK_PFR	Feature License to enable Propel in PFR	N/A	All

2.4. Required Files

Several pre-generated files provided for the demo setup are shown in Table 2.3.

Table 2.3. Required Demo Files

File Name	Description
BMC_384_1GB.bit	Bitstream to configure ECP5 as test BMC[1] device
PCH_256_1GB.bit	Bitstream to configure ECP5 as test PCH[2] device
manifest.jed	Manifest file that contains keys, QSPI monitor setup information, SMBus filter setup information, and others[3]
machnx_jtagenset_hi_new.bit	Image to enable programming of a blank Mach-NX part
machnx_jtagenset_hi_signed.bit	Image to enable programming of an otherwise blank Mach-NX part with programmed ECDSA key
sfb_prod_secured.bit	Secure Function Block image
FAM.jed	Flash Address Map contains primary and secondary locations of SFB image and RISCV firmware image stored in external SPI flash

Notes:

1. BMC (Baseboard Management Controller) is an out-of-band server management device. It enables platform administrators to manage a platform without requiring the host operating system to be running. The BMC_384_1GB.bit file is not a full feature BMC, but acts as a placeholder to demonstrate how the Lattice Sentry solution interacts with a BMC in a server system.
2. PCH (Platform Controller Hub) is a controller in the server which can provide in-band updates. PCH is an Intel chip, whereas FCH (Fusion Controller Hub) is an AMD chip. Both provide control functions in the server. The PCH_256_1GB.bit file is not a full feature PCH/FCH, but acts as a placeholder to demonstrate how the Lattice Sentry solution interacts with a PCH/FCH in a server system.
3. The manifest contains information for configuring the I2C Filter and SPI Monitor Address Space with access controls for setting up the IP functionality accordingly. It can be created using the Lattice Manifest tool in Lattice Propel Software.

3. Setting up Projects

This section covers the setup of a Lattice Propel Workspace and a Lattice Diamond Project for the Lattice Sentry Demo Board for Mach-NX Devices. A complete design project contains three parts:

- Propel SoC project. This hardware portion of the project is used to define the peripheral components of the RISCV SoC that are instanced in the FPGA portion of the Mach-NX device.
- Propel RISCV project. This software portion of the project is used to develop and test code executed on the RISCV processor.
- Lattice Diamond project. This portion of the project is used for synthesis, fit and route, and generation of the programming file to be loaded into the FPGA portion of the Mach-NX device. The Diamond project is generated from the Propel SoC project.

Each of the project components can be imported from an existing project or created from a template design.

3.1. Setting up a Propel SoC Project

3.1.1. Creating a New SoC Project from Template

To create a New SoC project from a template:

1. Open Propel 1.1 by clicking **Start > Lattice Propel 1.1 >Lattice Propel 1.1**.
2. The **Lattice Propel Launcher** opens, as shown in Figure 3.1. Enter a directory or click **Browse** to indicate the folder or workspace where the project files will be created. For best results, the directory should be stored locally on your computer and not on a network. For example, use *C:\MachNX_workspace* for this demo.

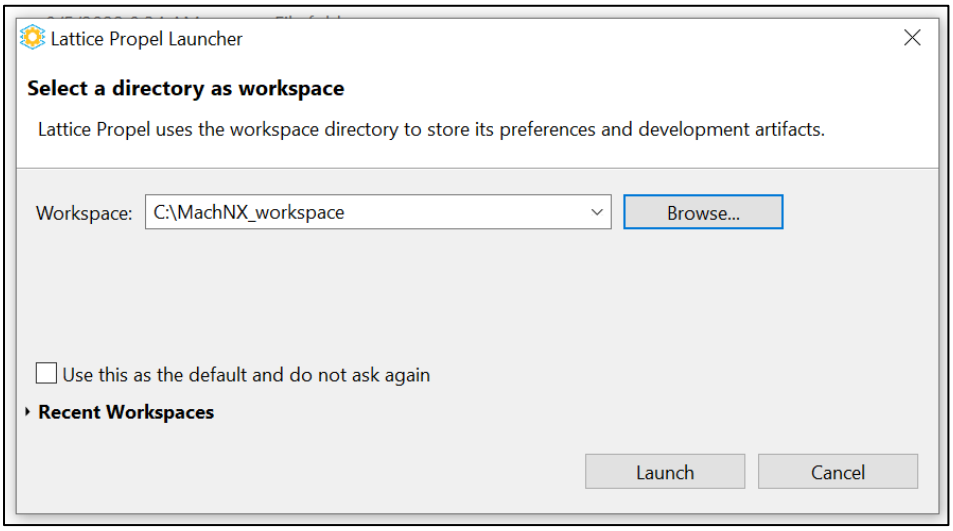

Figure 3.1. Lattice Propel Launcher

3. Click **Launch**. The workspace window opens, as shown in Figure 3.2.

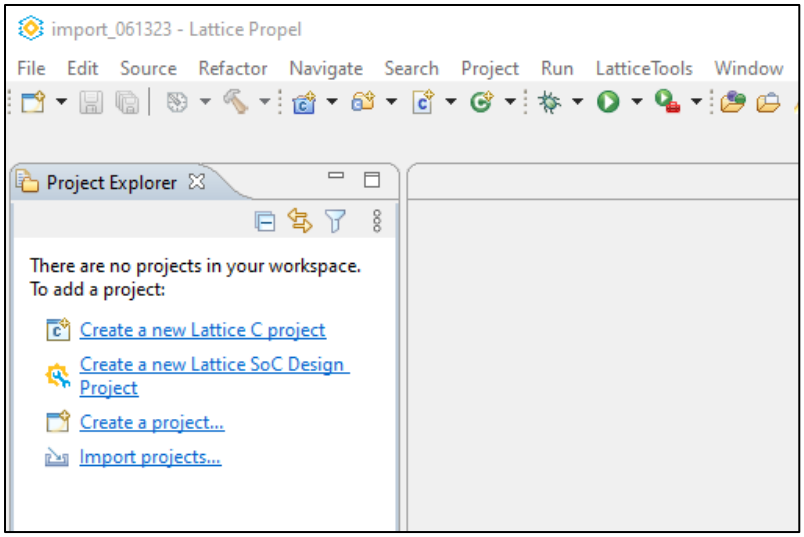

Figure 3.2. Propel Workspace

4. Click **File -> New -> Lattice SoC Design Project**.
5. The **Soc Project** wizard opens, as shown in Figure 3.3.

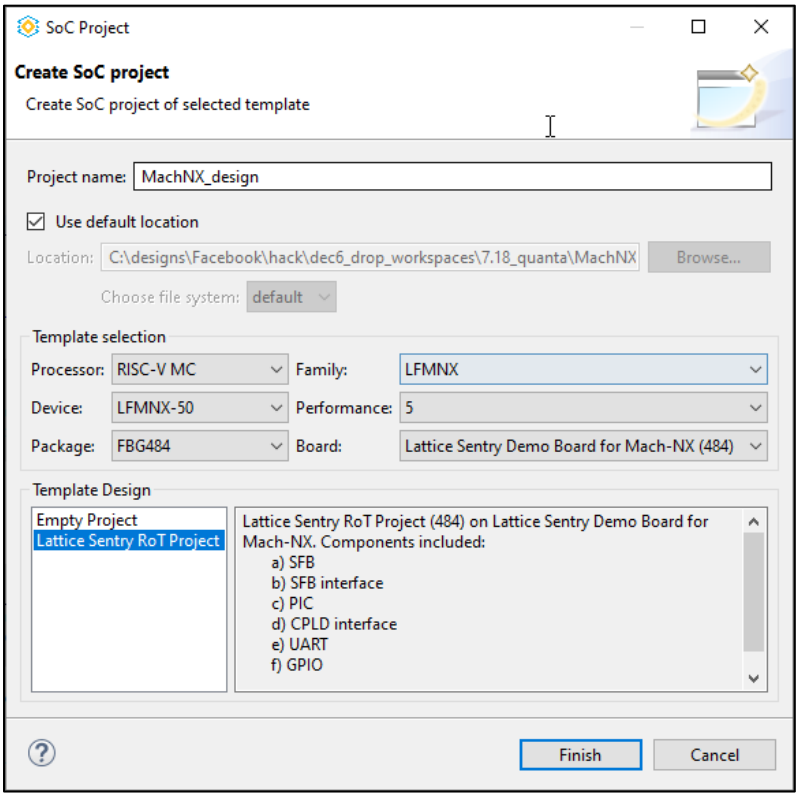

Figure 3.3. Create SoC Project Page

6. In the **Create Soc Project** page, enter a project name.

   Note: Do not include periods, colons, or spaces in the project name. For this demo, use ***MachNX_design***.

7. Select the following options:
   - **Processor** – **RISC-V MC**
   - **Family** – **LFMNX**
   - **Device** – **LFMNX-50**
   - **Performance** – **5**
   - **Package** – **FBG484**
   - **Board** – **Lattice Sentry Demo Board for Mach-NX**
   - **Template Design** – **Lattice Sentry RoT Project**
8. Click **Finish**. The SoC design project is created on the workbench, and its design is opened and displayed in Propel Builder, as shown in Figure 3.4.

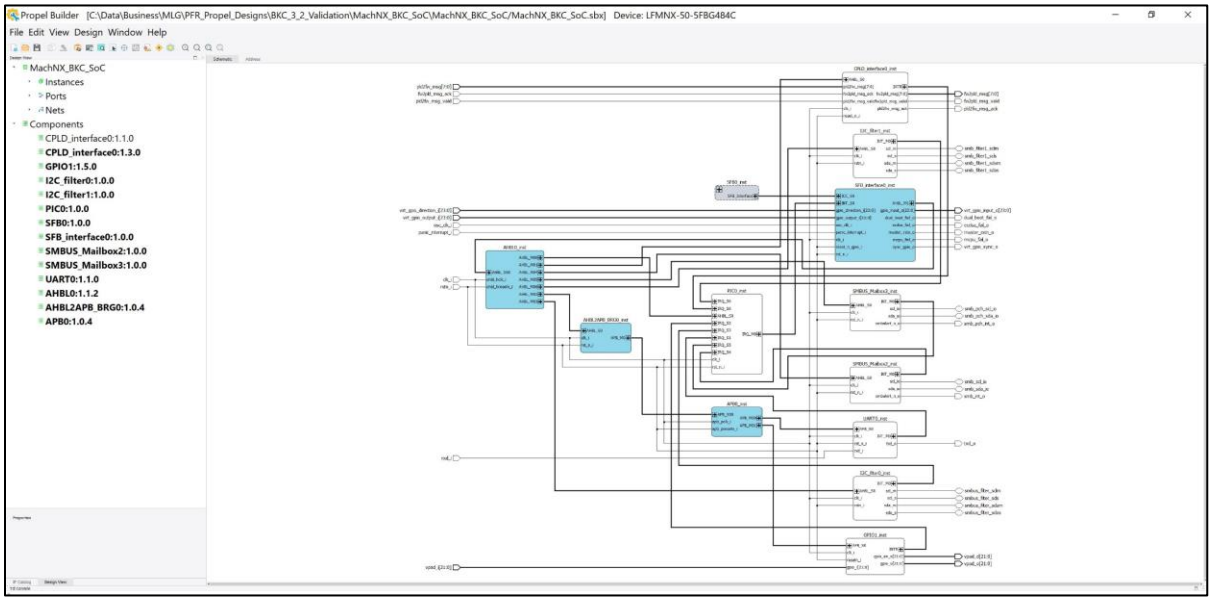

Figure 3.4. Propel Builder

9. Click **File > Save** to save the project.
10. Click **Design > Generate** to generate the project files.

3.1.2. Importing an Existing Propel SoC Project

To reuse or modify an existing project that you did not create, you need to import the workspace. Lattice provides workspaces in .zip format that contains two projects, the SoC and the Software. These projects need to be imported and the Propel tool needs to generate new metadata for the working directory.

To import an existing SoC project:

1. Extract the contents of the workspace .zip file.

   Note: The workspace .zip contains the SoC project folder and the Software project folder. There may also be a *metadata* directory. This metadata is not needed since new metadata is generated during the import process.

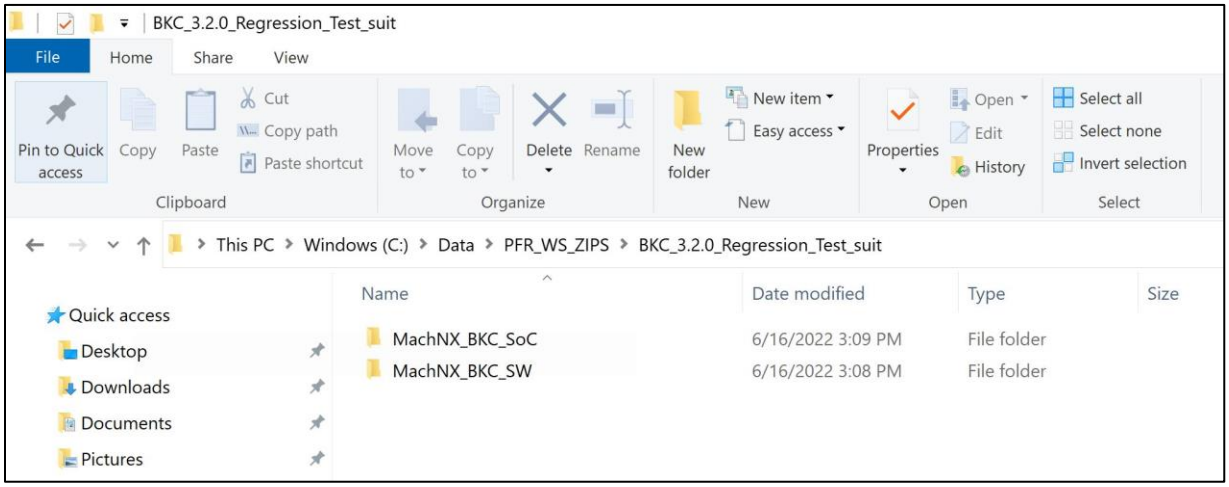

Figure 3.5. Existing Propel Project

2. Open Propel 1.1 by clicking **Start > Lattice Propel 1.1 > Lattice Propel 1.1**.
3. The **Lattice Propel Launcher** opens. Enter a directory or click **Browse** to indicate the folder or workspace where the project files will be created. For best results, the directory should be stored locally on your computer and not on a network. For example, use *C:\MachNX_workspace* for this demo. See Figure 3.1.
4. Click **Launch**.
5. In the workspace window, click **File > Import**.
6. The **Import** wizard opens, as shown in Figure 3.6.
7. In the Select page, select **Existing Projects into Workspace** under the **General** group.

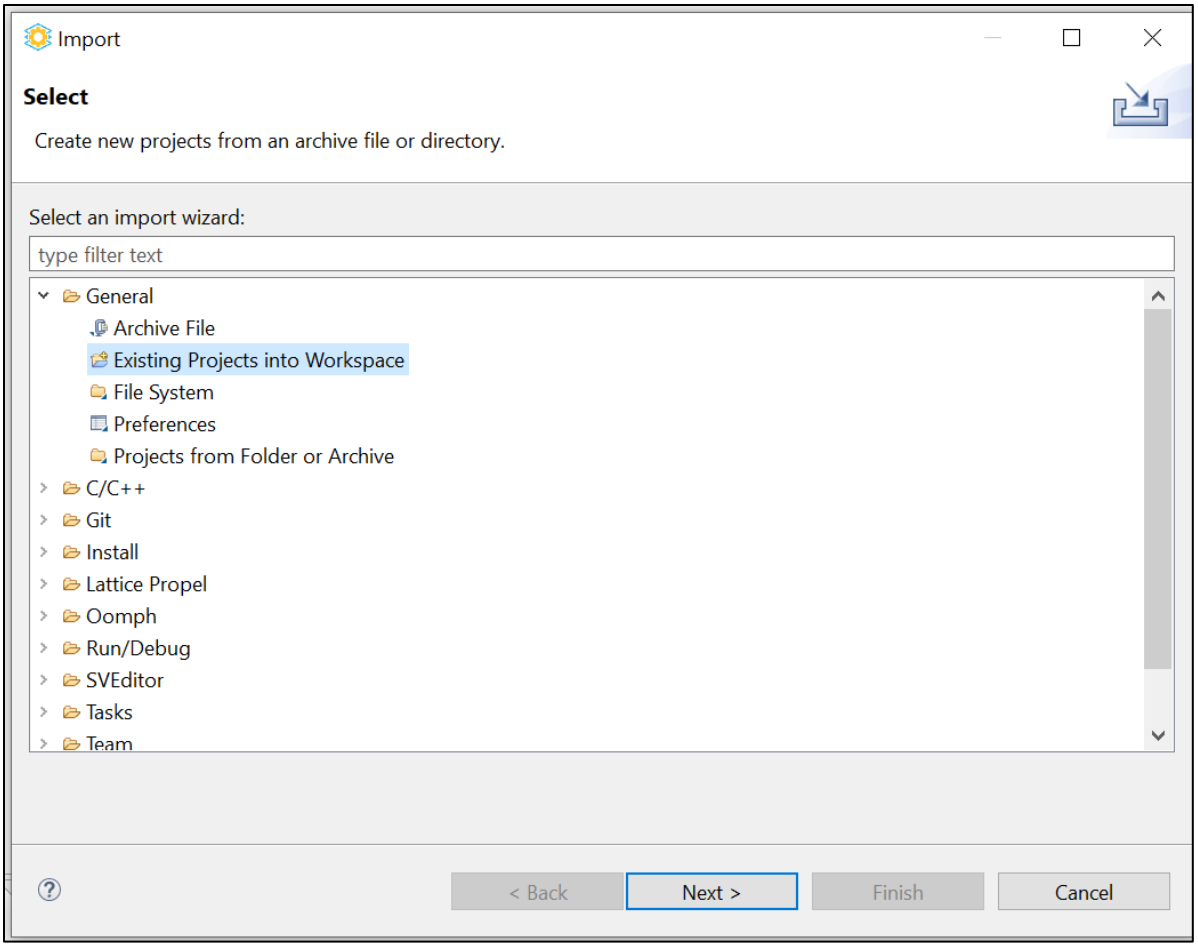

Figure 3.6. Select Page

8. Click **Next**.
9. The **Import Projects** page opens. In **Select root directory**, click **Browse** and select the folder where the SoC project folder and the Software project folder are located.

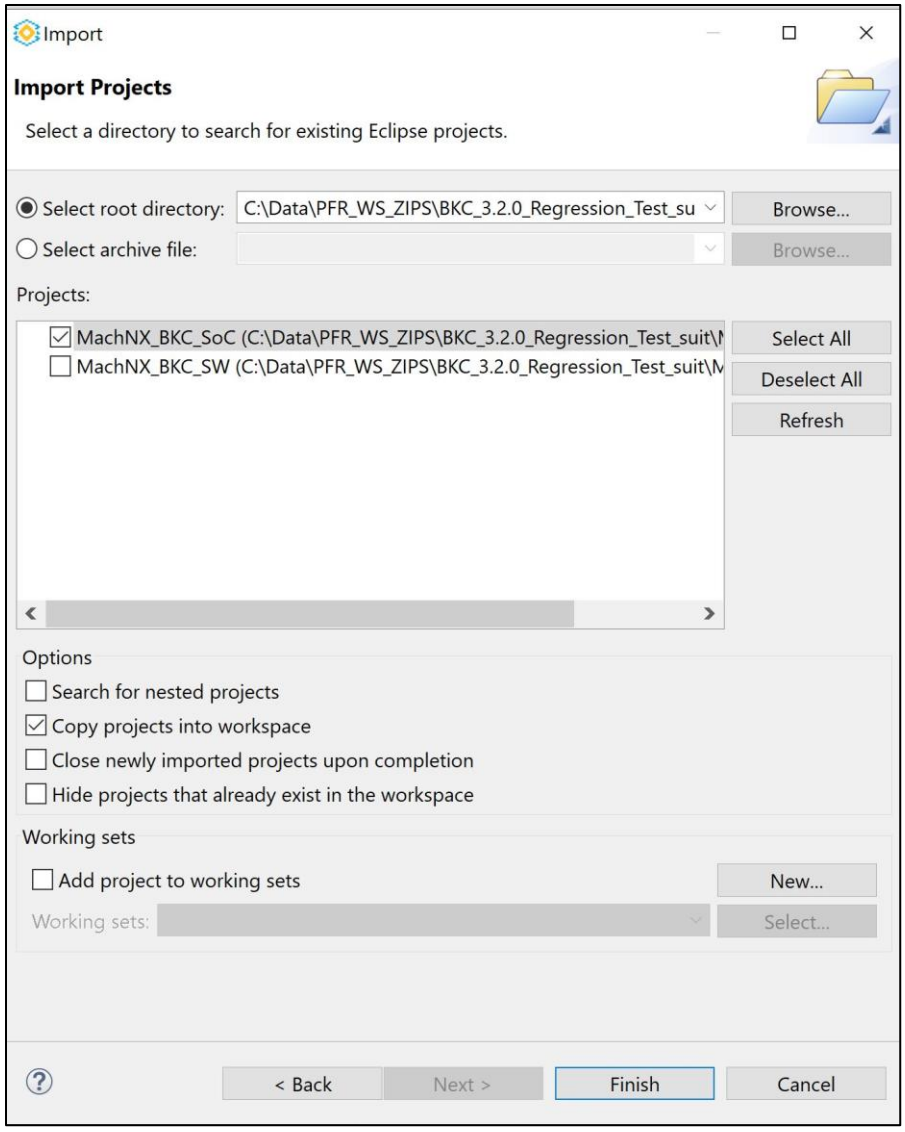

Figure 3.7. Import Projects Page

10. Check the **Copy Projects into Workspace** check box so that the source files remain unchanged.
11. Click **Finish**. The Propel project contains the SoC project, as well as a *metadata* folder generated for your system.
12. Click **Launch** to create the empty workspace.
13. In Propel, highlight the SoC project in the **Project Explorer** tab.
14. Click **Lattice Tools > Open Design in Propel Builder**. This opens the SoC project in Propel Builder.

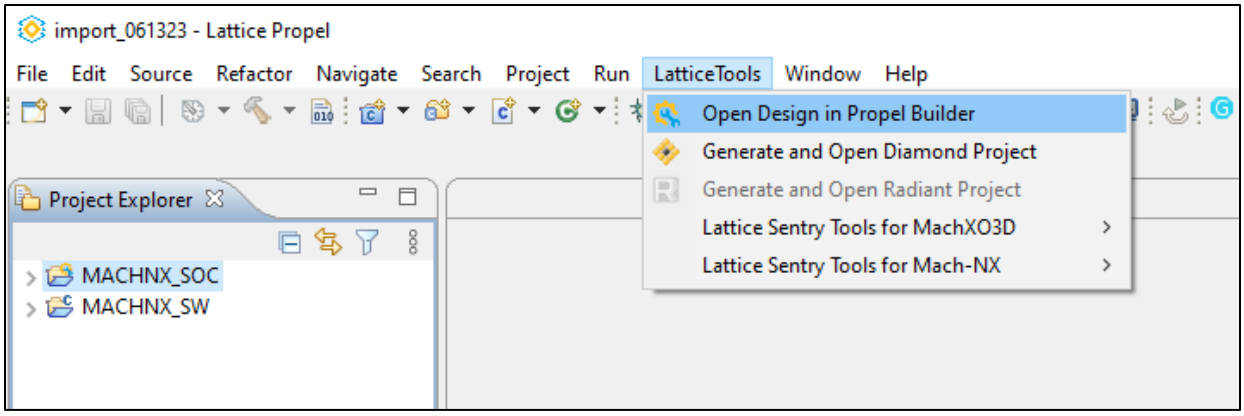

Figure 3.8. Lattice Tools Options – Open Design

15. Click **File > Save**, to save the project.
16. Click **Design > Generate** to generate the project files.

3.2. Generating Diamond Project

To generate a Diamond project:

1. In Propel interface, click **LatticeTools > Generate and Open Diamond Project**.
   Note: You can also click **Design > Run Diamond** from the Propel Builder interface.

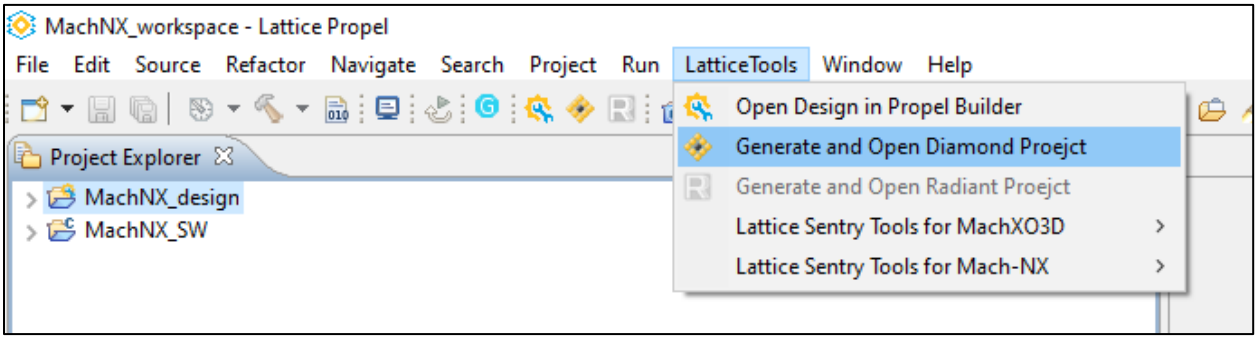

Figure 3.9. Lattice Tools Options – Generate

2. The project is open in Diamond. The **File List** tab shows the Verilog files generated by Propel, defining the SoC.

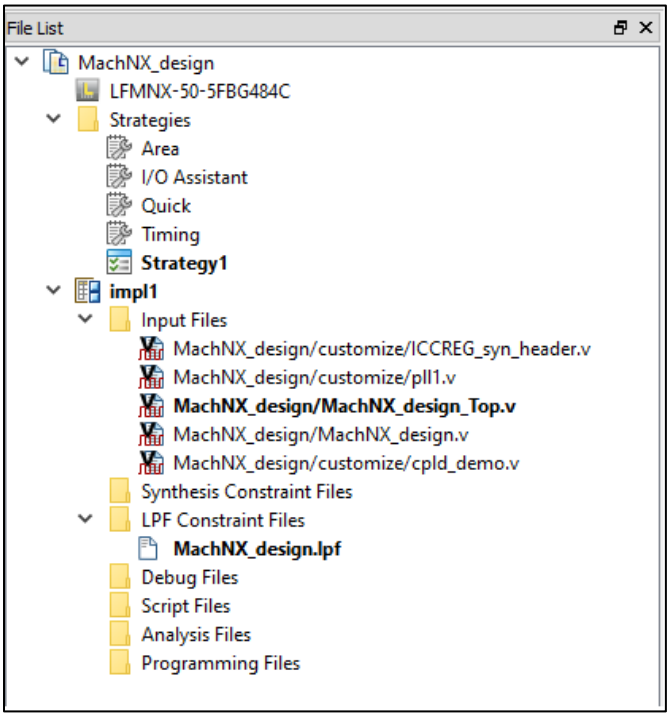

Figure 3.10. Diamond File List

3. Click **Tools > Spreadsheet View** and select the **Global Preferences** tab. Make sure that your project settings are the same as the settings shown in Figure 3.11.

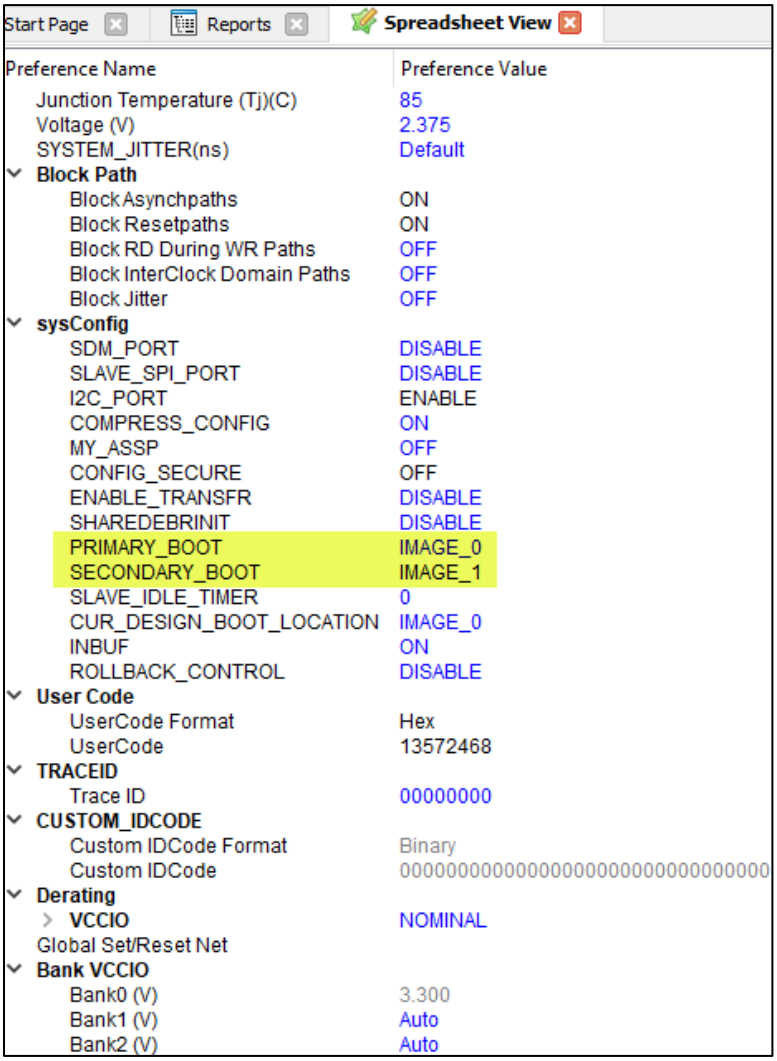

Figure 3.11. Global Preferences

4. Note that the UserCode is an optional user-accessible field that can be used for version number data. It does not have to match Figure 3.10, as it will depend on the design requirements.
5. Select **Security Setting** from the **Tools** menu.

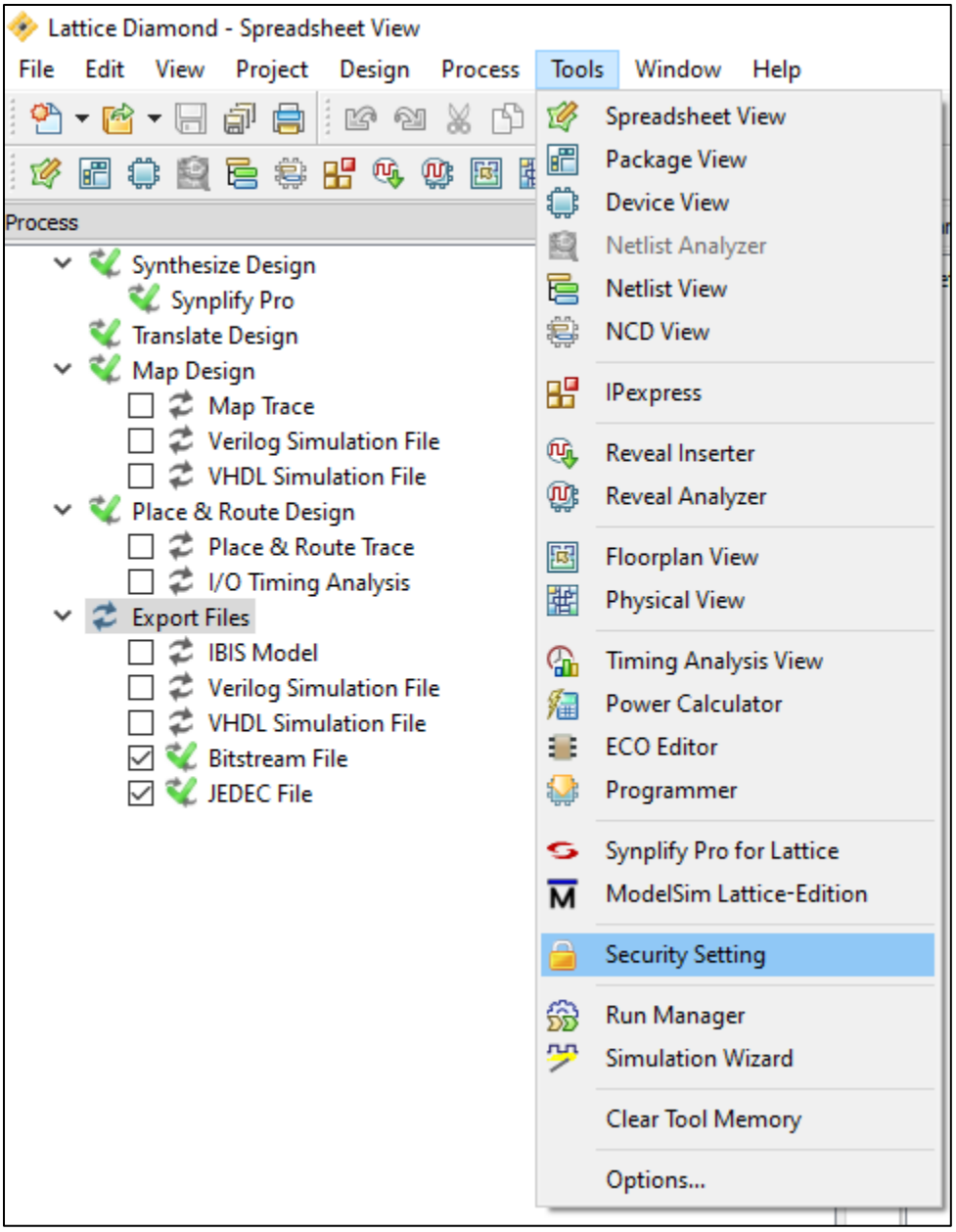

Figure 3.12. Security Settings

6. In the **Security Settings** dialog box, click the checkbox for **ECDSA Authentication**.
7. Load the keys:
   a. If a key pair does not yet exist (this will be the case for a design generated from a template), click the **Auto Generated Key Pair** button.
   b. If a key pair already exists, select Load From File and select the appropriate keys for public and private keys. This may be the case for an imported design.
8. Click **OK**. The selected key pair is used to sign the **.jed** and **.bit** files as a placeholder signature. Later, the FST tool will re-sign these files with the customer key pair, as described in Section 3.4.

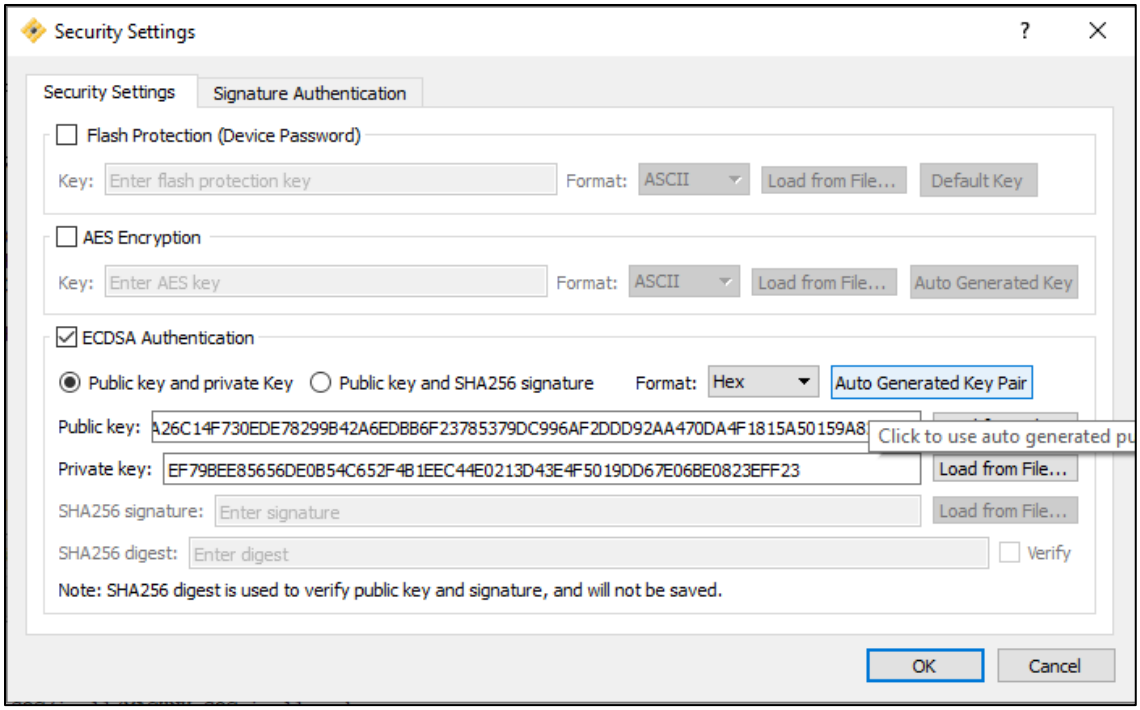

Figure 3.13. Security Settings Dialog Box

9. Select the **Process** tab, as shown in Figure 3.14.
10. Select the **Place & Route Trace** and the **JEDEC File** options.
11. Right-click on the text and select **Run**. The *MachNX_design_impl1_a.jed* is generated.

    Note: This .jed file will be signed by the FST utility, described in Section 3.4, and then that signed file is programmed into the CFG0 of Mach-NX device.

    Note: The file path is *<Workspace file path>/<SoC Project>/impl1/MachNX_design_impl1_a.jed*.

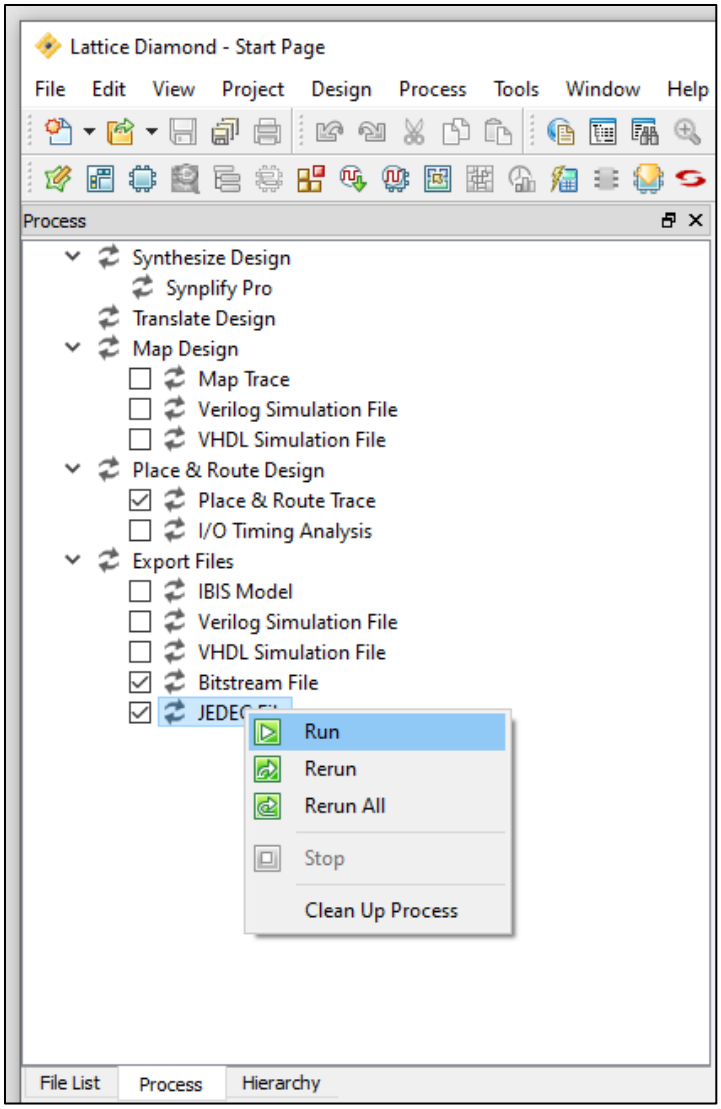

Figure 3.14. Process Tab

3.3. Setting up a Propel SW Project

3.3.1. Creating a New SW Project from Template

This section provides the procedure in creating the software project to accompany the SoC project.

To create a new software project from template:

1. In the Lattice Propel 1.1 workspace, click **File -> New -> Lattice C Project**. This opens the C Project wizard **Load system and bsp** page, as shown in Figure 3.15.

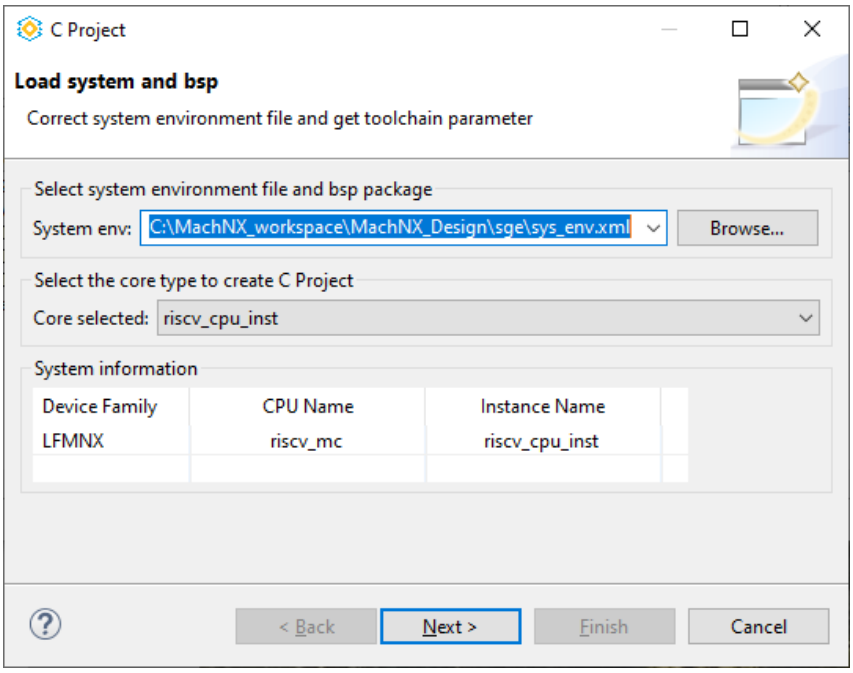

Figure 3.15. C Project Options

2. The **System env** field shows the *sys_env.xml* file in *<Workspace location>\<SoC project folder>\sge\sys_env.xml*. Enter this directory path if needed.
3. Click **Next**. This opens the **C Project** page.

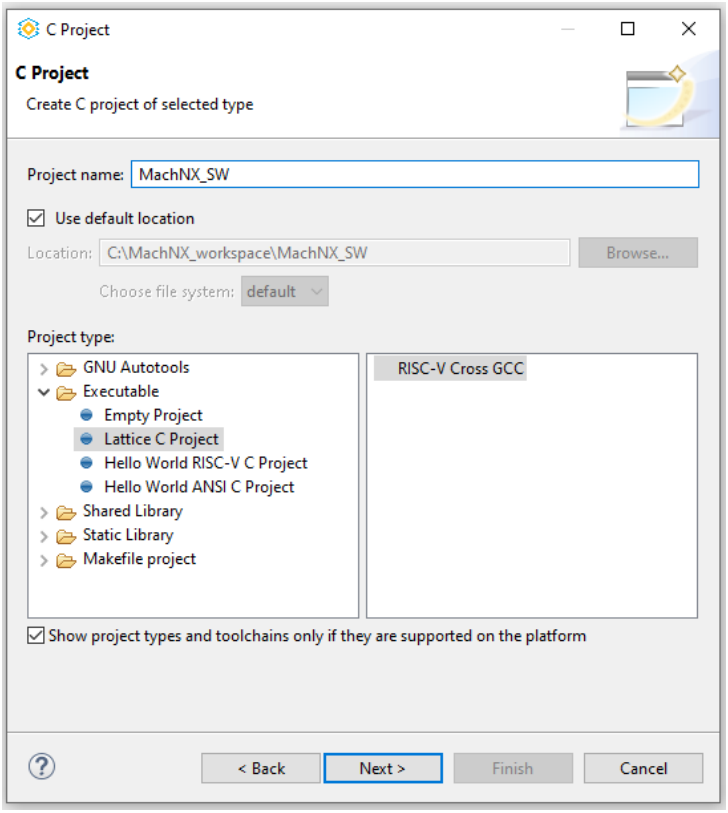

Figure 3.16. C Project Page

4. Enter a name for your project in **Project name**. For this demo, use **MachNX_SW**.
5. Under **Project type**, select **Lattice C Project** and **RISC-V Cross GCC** for toolchain.
6. Click **Next**. This opens the **Lattice toolchain setting** page.

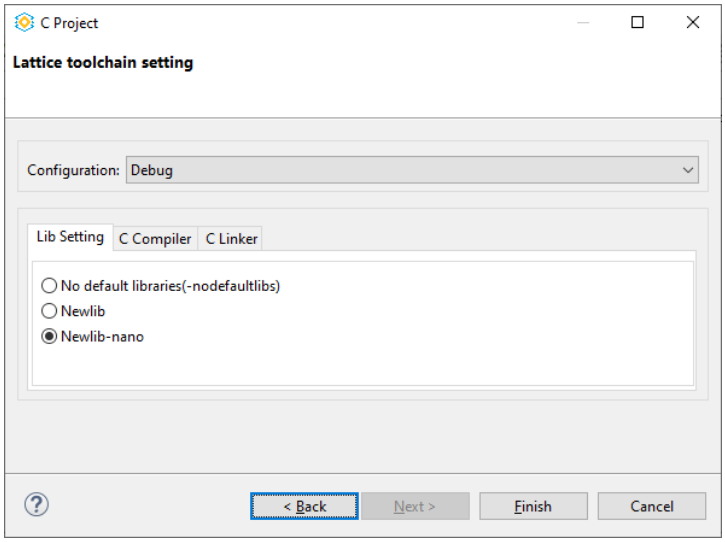

Figure 3.17. Lattice Toolchain Setting Page

7. Select **Debug** in **Configuration**.
8. Select **Newlib-nano** for **Lib setting**.
9. Click **Finish**. A second project folder opens in the **Project Explorer** tab.

3.3.2. Importing an Existing SW Project

To import an existing SW project:

1. In the Lattice Propel 1.1 workspace, click **File > Import**.
2. Select **Existing Projects into Workspace** under the **General** group, as shown in Figure 3.18.
3. Click **Next**.

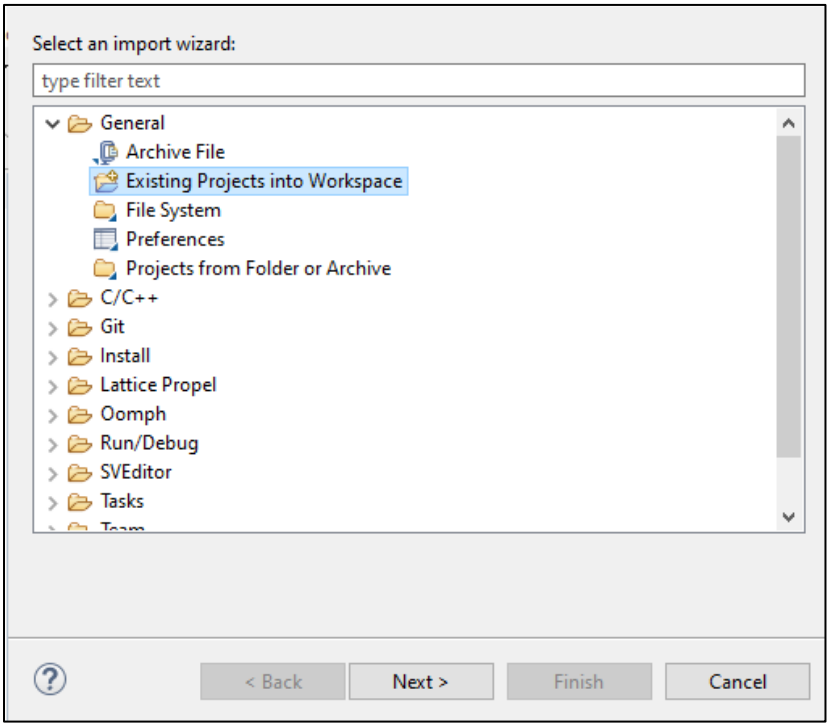

Figure 3.18 Select Import Wizard

4. The **Import Projects** page opens. In **Select root directory**, click **Browse** and select the folder where the unzipped Source and Project Files are located. See Figure 3.19.

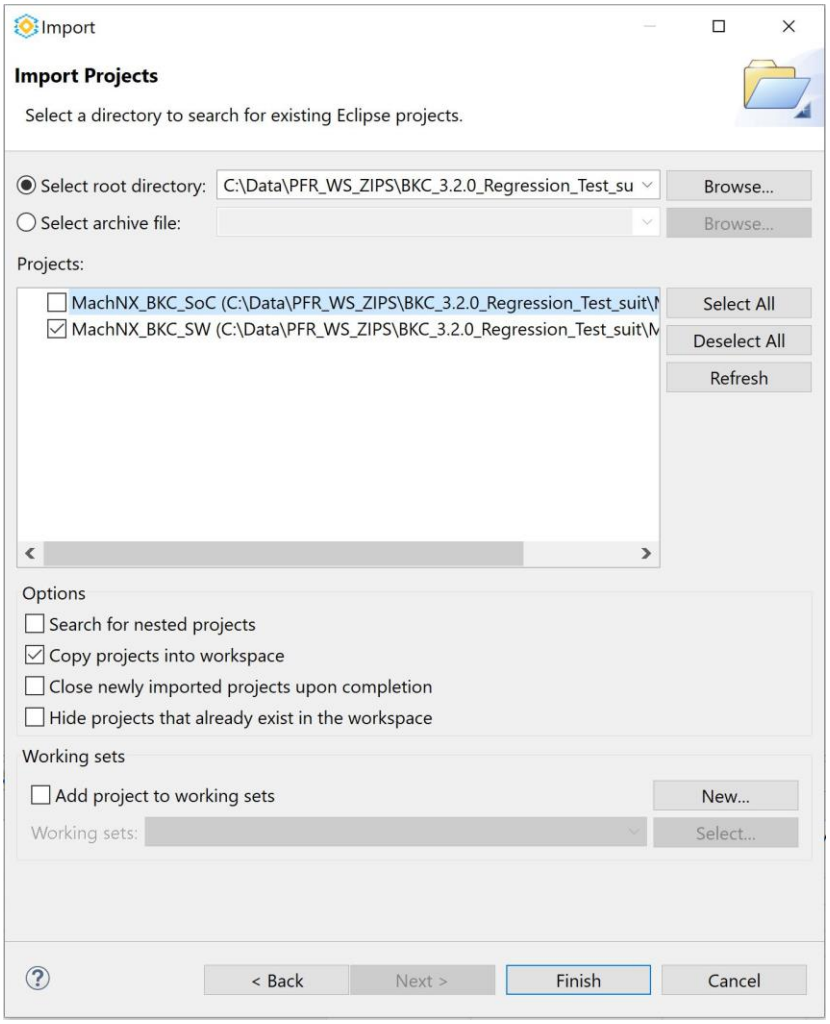

Figure 3.19. Import Projects

5. Under **Options**, select the **Copy projects into workspace**. This provides a new copy of the source files that can be modified.
6. Click **Finish**. The Propel project contains the SoC project, as well as a *metadata* folder generated for your system.

3.3.3. Building and Updating the Software Project

After creating or importing a software project, this section describes how to build the project, and how to associate new hardware changes with the project.

1. Build the project. Right-click the software project in the **Project Explorer** tab and select **Build Project.** The console window at the bottom of the screen displays the output, as shown in Figure 3.20. This indicates a successful compilation.

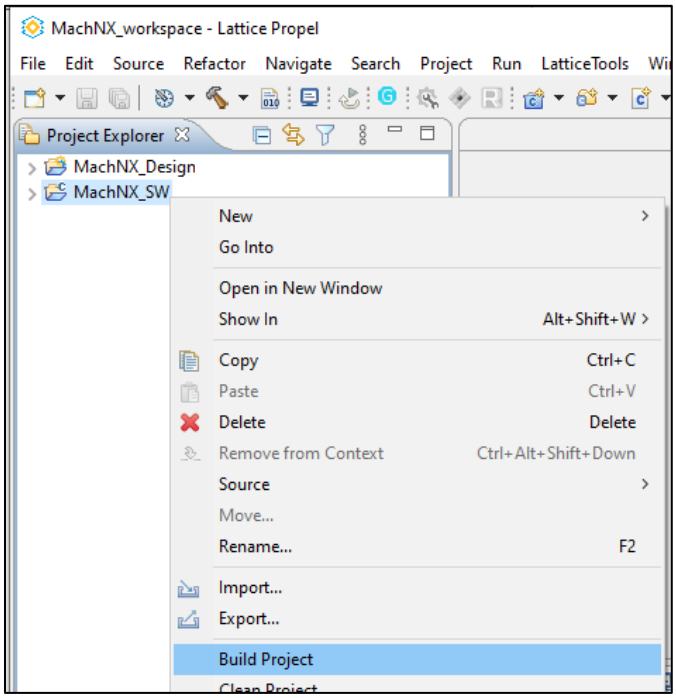

Figure 3.20. Build Project Option

2. The .mem file is created when the project is built. The directory path is *<Workspace file path>/<SW Project>/Debug/MachNX_SW.mem.*

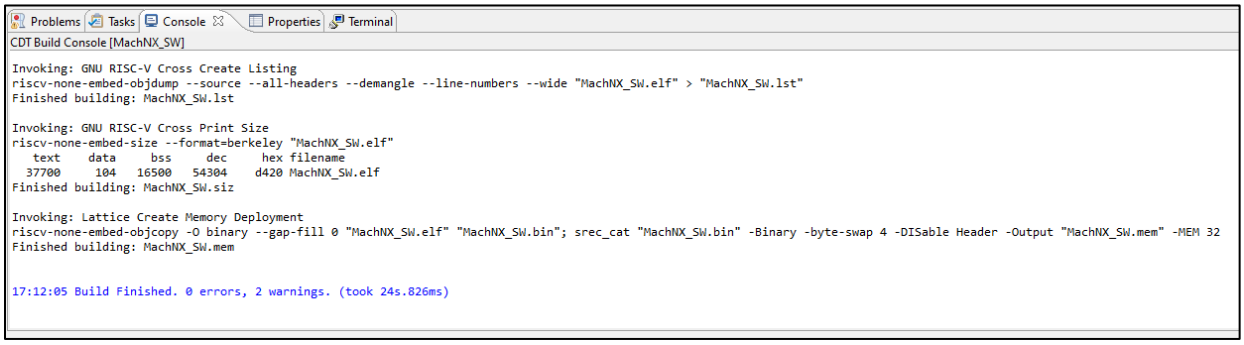

Figure 3.21 Console Output

3.3.3.1. Updating the software project

If changes have been made to the project hardware in Propel Builder, such as changes to the memory map or deleting/adding new IP, these changes must be re-associated with the software project with the following steps:

1. Highlight the software project in the **Project Explorer** tab, right click, and choose **Update Lattice C Project**.

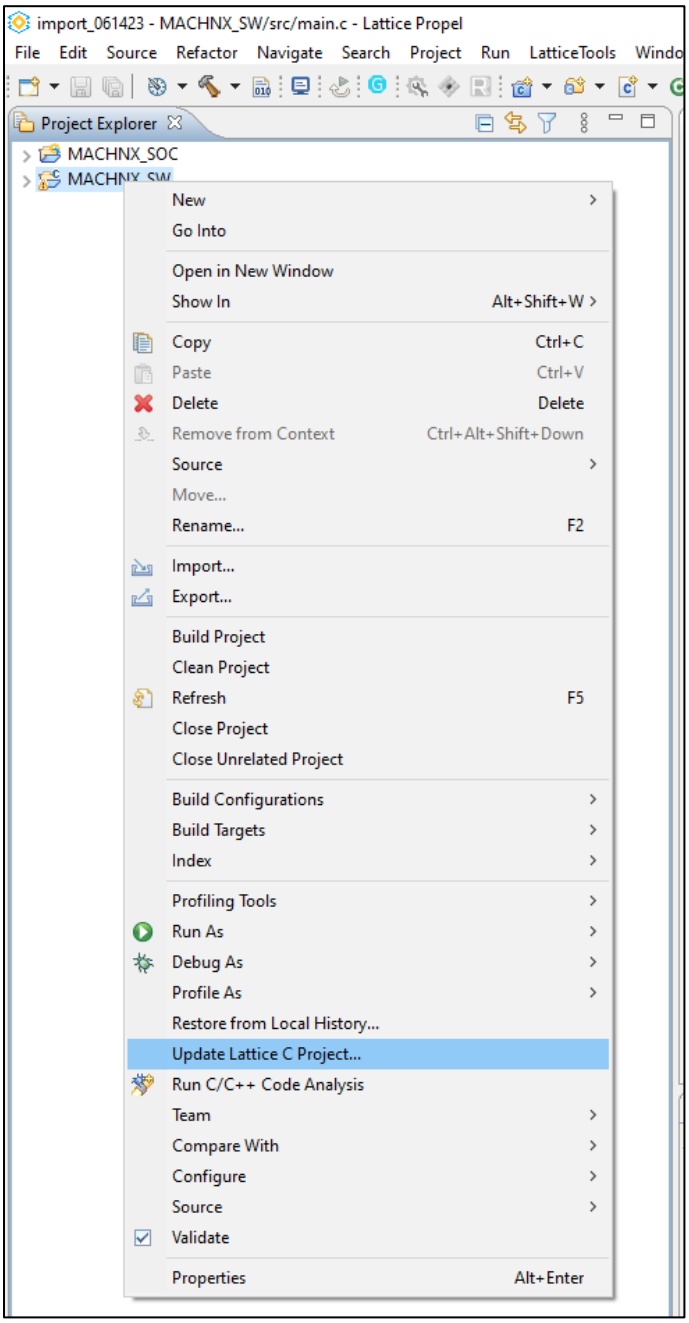

Figure 3.22. Update Lattice C Project

2. Browse to the **sge** directory generated by Propel Builder and select **sys_env.xml.**

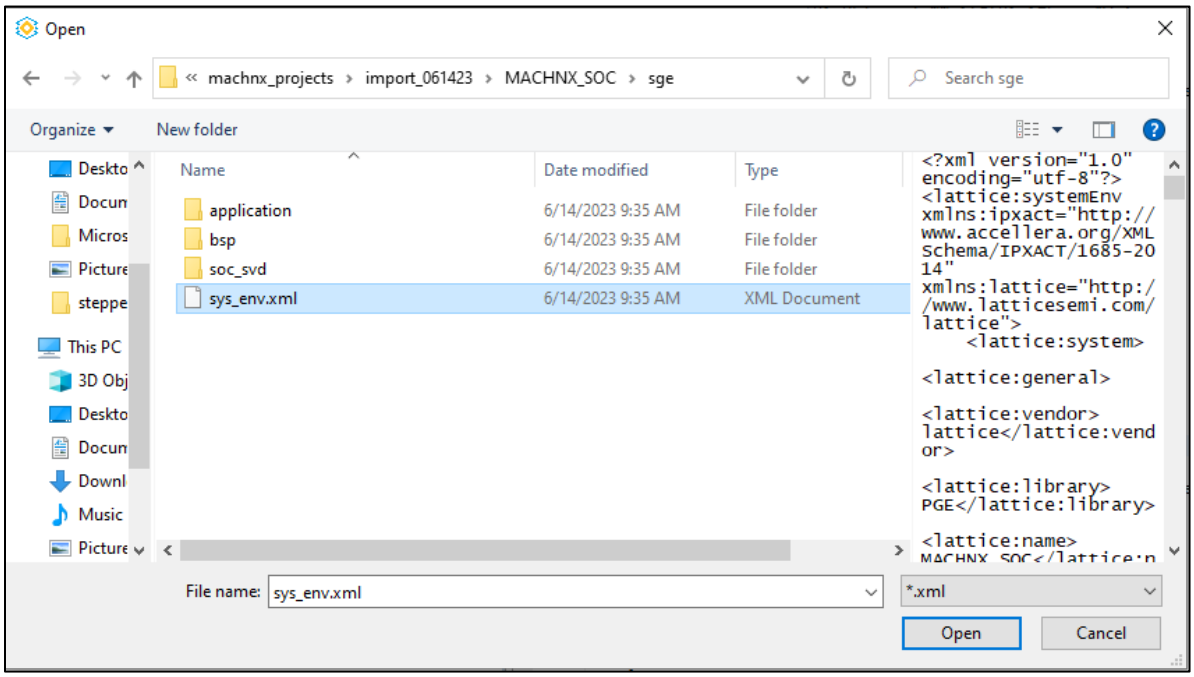

Figure 3.23. sys_env.xml File in Directory

3. Check **Regenerate toolchain parameters and linker script** checkbox.
4. Do **NOT** check **Update bsp package** checkbox. Checking this box will revert all drivers to released versions, which will overwrite any software updates that have been made.

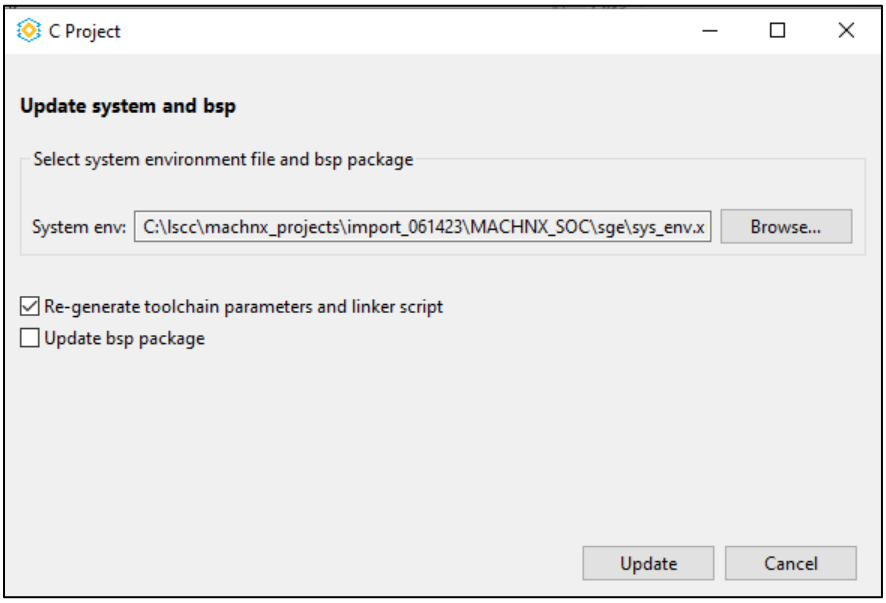

Figure 3.24. Update System Dialog Box

5. After making these changes, follow Steps 1-2 of Section 3.3.3 to rebuild the project and generate an updated **.mem** file.

3.4. Generating Board Files

The Firmware Signing Tool (FST) is used to sign the ***.mem*** file created by Propel and convert it to a ***.bin*** file, which will be programmed into an external SPI Flash as the desired FW address location.

The FST is also used to sign the ***.jed*** and ***.bit*** files created by Diamond with the private key from the user's key pair. Follow the steps below in generating board files.

1. Open the Cygwin terminal and navigate to the FST directory.
2. Set up the keys:
   a. To generate new keys.

```
./sign_sentry_firmware_ecdsa.sh -s key
```

   - This will produce.

```
keys/ECDSA_prv.pem and keys/ECDSA_pub.pem
```

   b. To use existing keys.
   - Create a directory named keys.

```
mkdir keys
```

   - Copy the pem files into that directory according to the names below:

```
p my_private.pem keys/ECDSA_prv.pem
cp my_public.pem keys/ECDSA_pub.pem
```

3. Copy the following Diamond and Propel generated files to the FST directory. Note that your file names may differ depending on how you named your project.
   - <Workspace file path>/<SW Project>/Debug/MachNX_SW.mem
   - <Workspace file path>/<SoC Project>/impl1/MachNX_design_impl1_a.jed
   - <Workspace file path>/<SoC Project>/impl1/MachNX_design_impl1.bit
4. Sign the generated files with your keys:

```
   ./sign_sentry_firmware_ecdsa.sh -s sign -m MachNX_SW.mem -j
MachNX_design_impl1_a.jed -b MachNX_design_impl1.bit
```

5. This produces three signed files in the result directory:

Table 3.1. Files Generated by FST Utility

File name	Description
result/CFG0_signed.bit	Signed Config File for Mach-NX (not used in this walkthrough)
result/CFG0_signed.jed	Signed Config File for Mach-NX to program into CFG0 (and optionally CFG1)
result/RiscVImage_signed.bin	Signed RiscV image to program into Flash.

4. Setting up the Board

This section covers programming the board with the required files. Table 4.1 shows the required files, how they are generated, and where they are to be programmed.

Table 4.1. Programming Files

File Name	Description/Purpose	File Source	Programming Location
CFG0_signed.jed	PFR hardware other than SFB, SFB interface, and customer logic	Generated and signed in Diamond, then re-signed by FST using the same key pair that is used to sign the FW image	CFG0 sector of Mach-NX
manifest.jed	Sets up White, Gray, and Black spaces	Included with demo files	UFM0 sector of Mach-NX
FAM.jed	SFB authentication information, SPI Flash settings, external SPI address locations, PFR FW bitstreams	Included with demo files	FAM sector of Mach-NX
BMC_384_1GB.bit	Baseboard Management Controller firmware	Included with demo files	0x00000000-0x00210000 of Flash C
RiscVImage_signed.bin	PFR Firmware	Generated by FST using .mem file that was generated by Propel SDK	0x03690000-0x036A0000 (primary) and 0x03B60000-0x03B70000 (backup) of Flash C
sfb_prod_secured.bit	SFB hardware	Included with demo files	0x036C0000-0x037D0000 (primary) and 0x03B90000-0x03CA0000 (backup) of Flash C
PCH_256_1GB.bit	Platform Controller Hardware firmware	Included with demo files	0x00000000-0x00200000 of Flash A

The demo board has several jumpers that connect one or more of the FPGAs on the board in a JTAG chain. Table 4.2 shows the jumper settings to connect each of the FPGAs to the JTAG programming port individually.

Table 4.2. JTAG Jumper Settings

	JP24	JP25	JP26	JP20	JP21	JP22	J20	J21	J23
Mach NX	*Closed*	Open	Open	*Closed*	Open	Open	*1:2*	*3:4*	Open
BMC	Open	*Closed*	Open	Open	*Closed*	Open	*3:4*	Open	*1:2*
PCH	Open	Open	*Closed*	Open	Open	*Closed*	Open	*1:2*	*3:4*

Table 4.3. Flash Jumper Settings

Target Flash Programming	JP19	JP17	JP18	JP42 and JP44	JP43	JP41
Flash A – PCH Primary	Open	Open	Open	*Closed*	*Closed*	Open
Flash B – PCH Backup	Open	Open	Open	*Closed*	Open	*Closed*
Flash C – BMC Primary	*Closed*	Open	Open	Open	Open	Open
Flash D – BMC Backup	*Closed*	*Closed*	*Closed*	Open	Open	Open
PFR Demo	Open	Open	Open	Open	Open	Open

Note that there are four SPI flash devices on the board. Two for the ECP5, acting as the BMC, and two for the ECP5, acting as the PCH. For this walkthrough, only Flash A for the PCH and Flash C for the BMC are used.

The Lattice Sentry Demo Board for Mach-NX Devices supports various JTAG chain options to scan individuals or certain combinations of devices. The jumper setting to scan all devices in the JTAG chain is shown in Figure 4.1. The configuration to scan the Mach-NX device is shown in Figure 4.2.

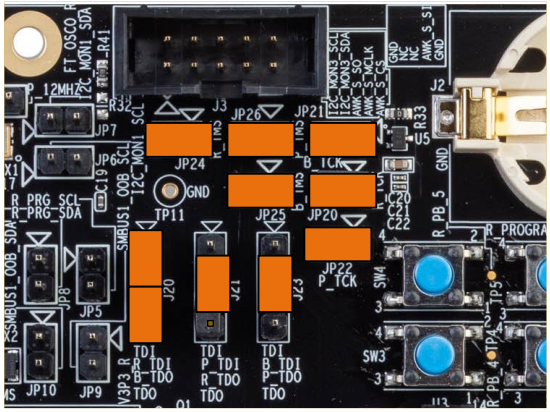

Figure 4.1. All Device JTAG Chain Jumper Setting

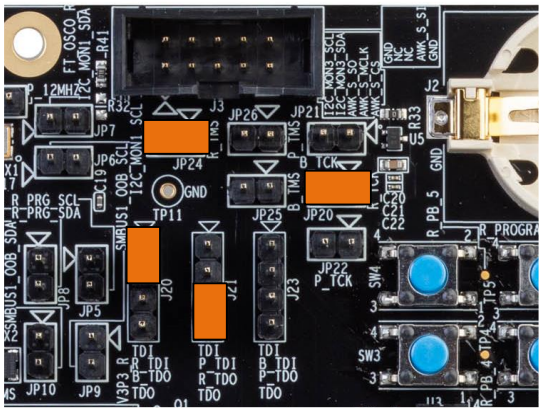

Figure 4.2. Mach-NX JTAG Chain Jumper Setting

Some useful JTAG jumper settings are provided in Figure 4.3 and Table 4.4.

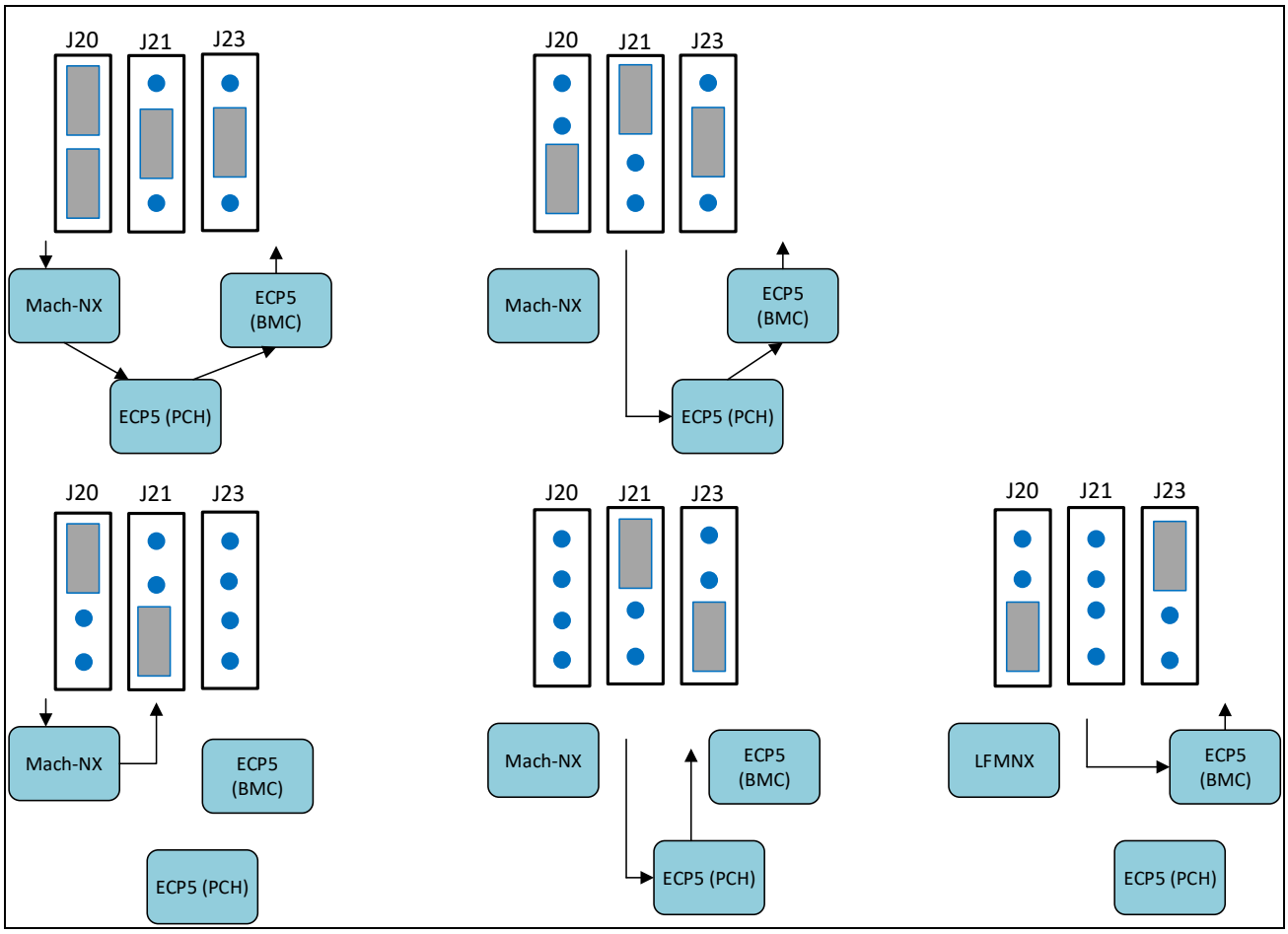

Figure 4.3. JTAG Chain Jumper Diagram

Table 4.4. JTAG Chain Jumper Settings

Jumper	Function	Description
J20	Mach-NX TCK	Placing jumper allows board TCK to drive Mach-NX TCK
J21	ECP5 (BMC) TCK	Placing jumper allows board TCK to drive BMC TCK
J22	ECP5 (PCH) TCK	Placing jumper allows board TCK to drive PCH TCK
J24	Mach-NX TMS	Placing jumper allows board TMS to drive Mach-NX TMS
J25	ECP5 (BMC) TMS	Placing jumper allows board TMS to drive BMC TMS
J26	ECP5 (PCH) TMS	Placing jumper allows board TMS to drive PCH TMS

4.1. Programming the Mach-NX Files

4.1.1. Programmable Sectors of the Mach-NX

The Mach-NX device contains the following programmable sectors:

- **CFG0**: Primary boot location if dual boot is used; only boot location if dual boot is not used. This image contains the SFB interface and user logic.
- **UFM0**: Manifest information, including white, gray, and black address spaces for SPI filtering, I2C whitelisting, etc.
- **CFG1**: Secondary boot location if dual boot is used. Same contents as CFG0 if dual boot is used. Empty or ignored if dual boot is not used.
- **UFM1**: Backup manifest location or unused.
- **UFM2**: Location of the UDS certificate used for DICE attestation. UFM2 with a dummy UDS certificate from Lattice is required. Optional, if UDS certificate or DICE is not used. This sector is also used to log the Mach-NX boot errors and PFR FW log data.
- **FAM**: SFB authentication information, SPI Flash settings, and external SPI address locations for SFB and PFR firmware images. Also called UFM3.

Note that the instructions in this section describe programming a basic use case and will work with the template design or the Sentry demo design. Refer to the details of your own design if you need to program other sectors of the Mach-NX device.

4.1.2. Connecting to the Mach-NX

To connect to the Mach-NX board in Diamond Programmer:

1. Connect the Mach-NX to the JTAG chain. Refer to Table 4.2 or Figure 4.4.

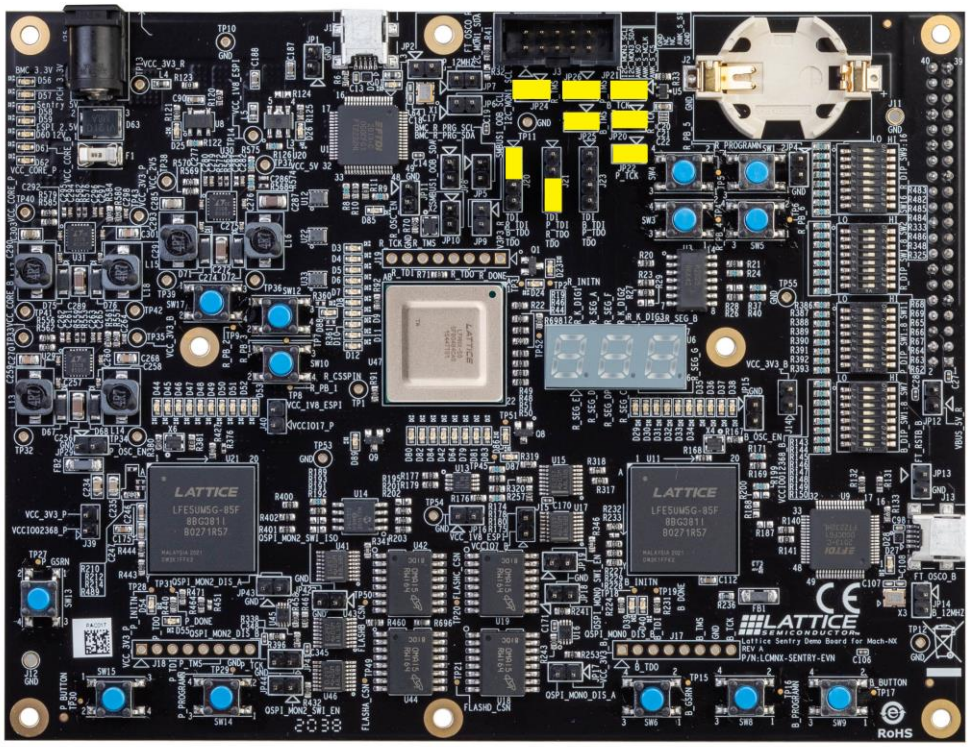

Figure 4.4. Mach-NX Jumper Settings

2. Connect power to the board at J25.
3. Connect the FTDI interface at the top of the board (J1) using a USB cable.
4. Open the Diamond Programmer. In the **Getting Started** dialog box, select **Create a new project from a JTAG scan**.

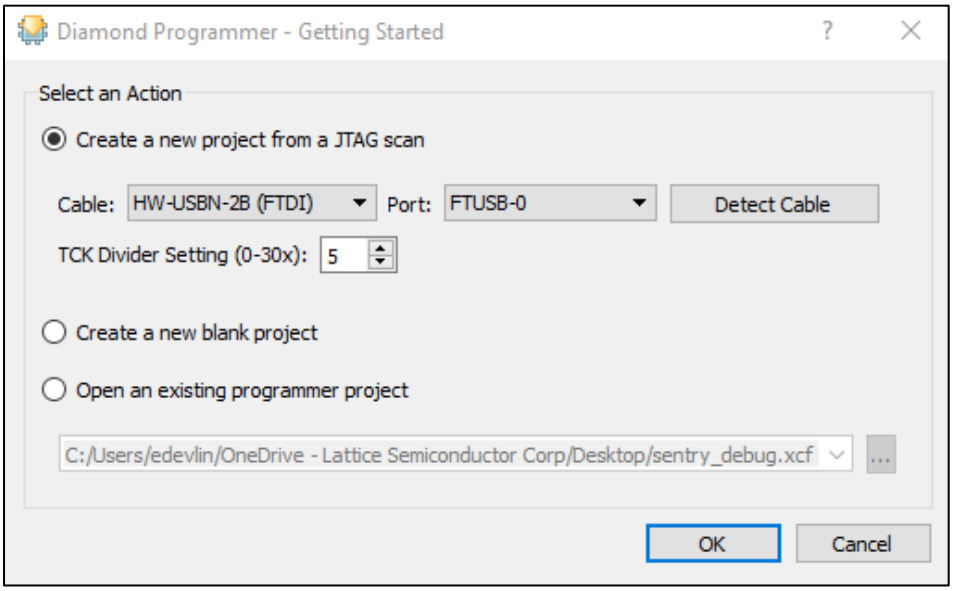

Figure 4.5. Getting Started

5. Click **OK**.
6. In the main interface, the scanned Mach-NX is listed, as shown in Figure 4.6.

Figure 4.6. Scanned Mach-NX Device

Note: If the LFMNX device does not appear upon scanning the board, see Section 6 for troubleshooting instructions.

4.1.3. Erasing the Mach-NX Public Key

The Lattice Sentry Demo Board for Mach-NX Devices is shipped with the factory **.pub** key programmed into the Mach-NX PUBKEY sector. This is the public key of the ECDSA key pair.

To change keys, the Lattice Diamond tools or the Lattice FST utility can be used to generate a new key pair. Please keep the **newkey.prv** (Diamond generated key pair) / **ECDSA_prv.pem** (FST generated key pair) secure and use it to sign both CFG and FW images. After programming CFG and FW images into the CFG (CFG0/1 internal flash sector) and the FW (external SPI flash), erase the PUBKEY sector and program the **newkey.pub** (Diamond generated key pair) or **ECDSA_pub.pem** (FST generated key pair) into the PUBKEY sector as the final step in the programming sequence.

To erase the factory key:

1. Follow the steps in Section 4.1.2 to connect to the Mach-NX board in Diamond Programmer.
2. Double-click on the **Operation** field to open the **Device Properties** dialog box.
3. In Device Properties, select **Advanced Security Keys Programming**, then **Security Erase Public Key.**

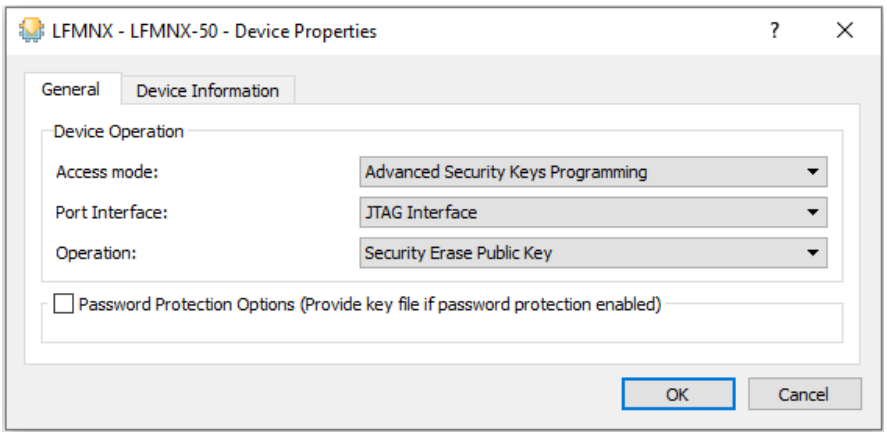

Figure 4.7. Mach-NX Erase Public Key

4. Click **OK.**
5. In the main interface, choose **Design -> Program.**
6. Check the output log for a success message.

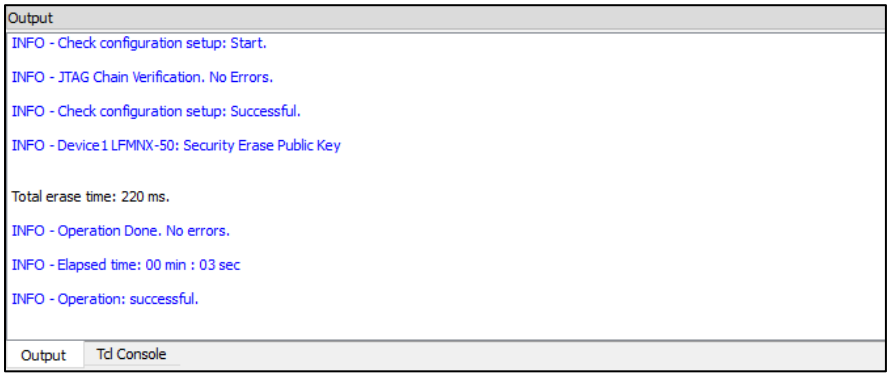

Figure 4.8. Mach-NX Erase Public Key Output Log Message

4.1.4. Programming the Mach-NX files

To program the Mach-NX internal flash:

1. Follow the steps in Section 4.1.2 to connect to the Mach-NX board in Diamond Programmer.
2. Double-click on the **Operation** field to open the **Device Properties** dialog box.
3. Select the following options under **Device Operation**
   - **Access mode – Flash Background Mode**
   - **Port Interface – JTAG Interface**
   - **Operation – XFLASH Erase, Program, Verify**

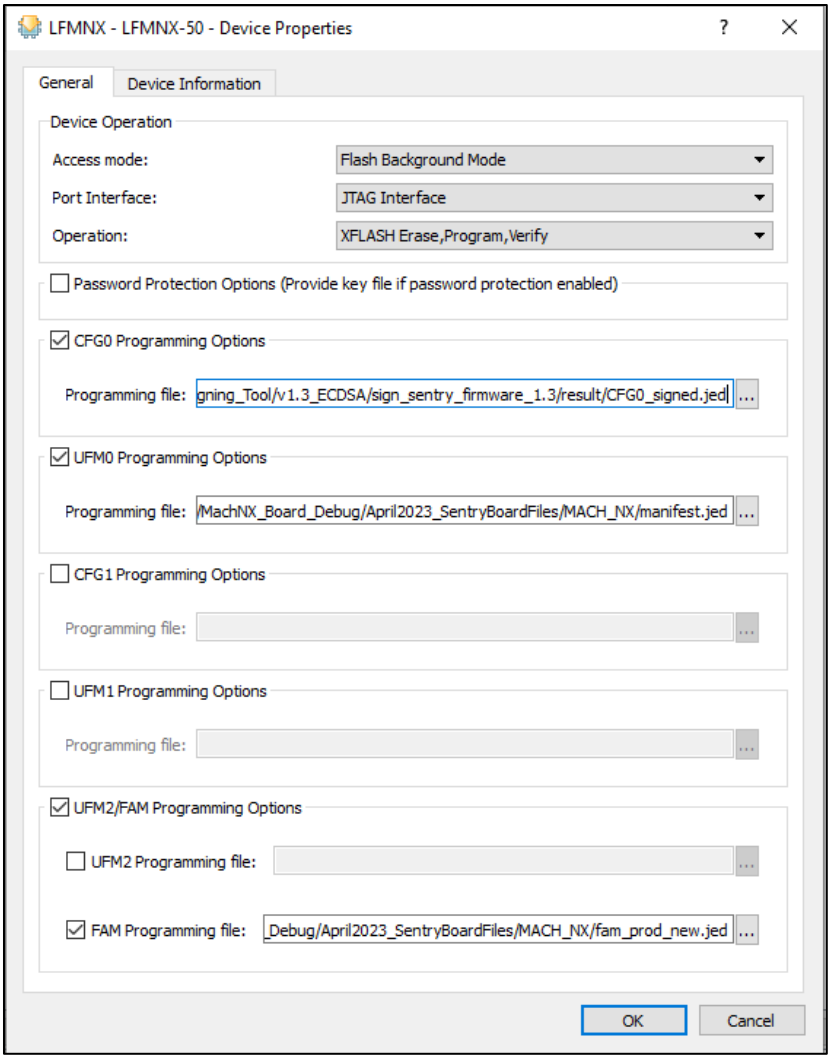

Figure 4.9. Mach-NX Erase, Program, Verify

4. Select **CFG0 Programming Options**. In **Programming file**, enter the **.jed** file created in Section 3.2 and then signed by the FST in Section 3.4.
5. Select **UFM0 Programming Options**. In **Programming file**, enter the **manifest.jed** file included in the demo package folder.
6. Select **UFM2/FAM Programming Options**. Select also **FAM Programming file**. Enter the **FAM.jed** file included in the demo package folder.
7. Click **OK**.
8. In the main interface, click **Design > Program**.
9. After the operation is completed, the log windows shows that the device has been programmed successfully.

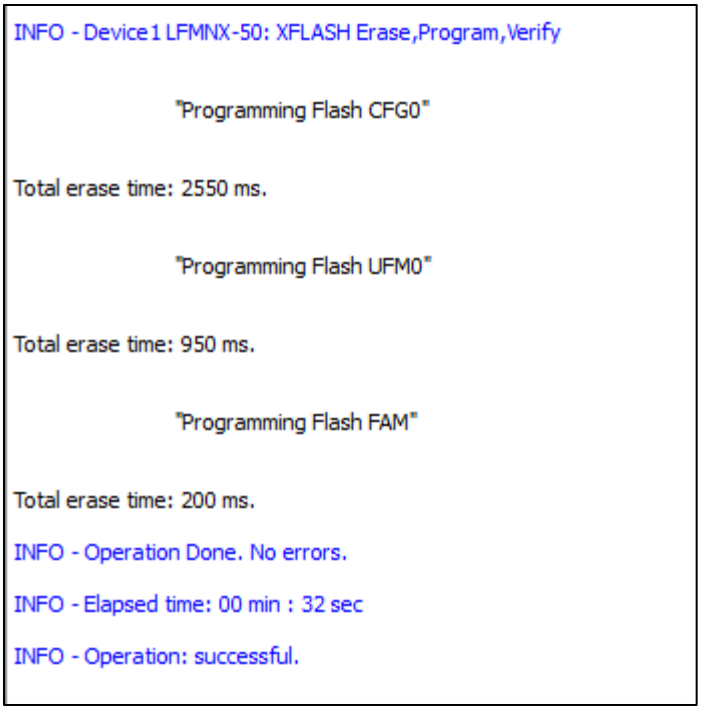

Figure 4.10. Log Output Success Message After Programming Mach-NX

4.2. Programming the BMC Flash

The BMC's configuration image is stored on an external SPI flash chip, labeled flash C, along with the RISC-V image and the SFB image.

To program the BMC flash:

1. Connect BMC to the JTAG chain. Refer to Figure 4.1 or Table 4.2.
   a. Connect the FTDI interface at the top of the board (J1) using a USB cable.
   b. With power disconnected, attach the yellow labeled jumpers (JP24, JP25, JP26, JP21, JP20, JP22, J20(3:4), J23(1:2), JP4.
   c. Power on the board by connecting J25 to power.
   d. Attach the orange labeled jumper, JP19.

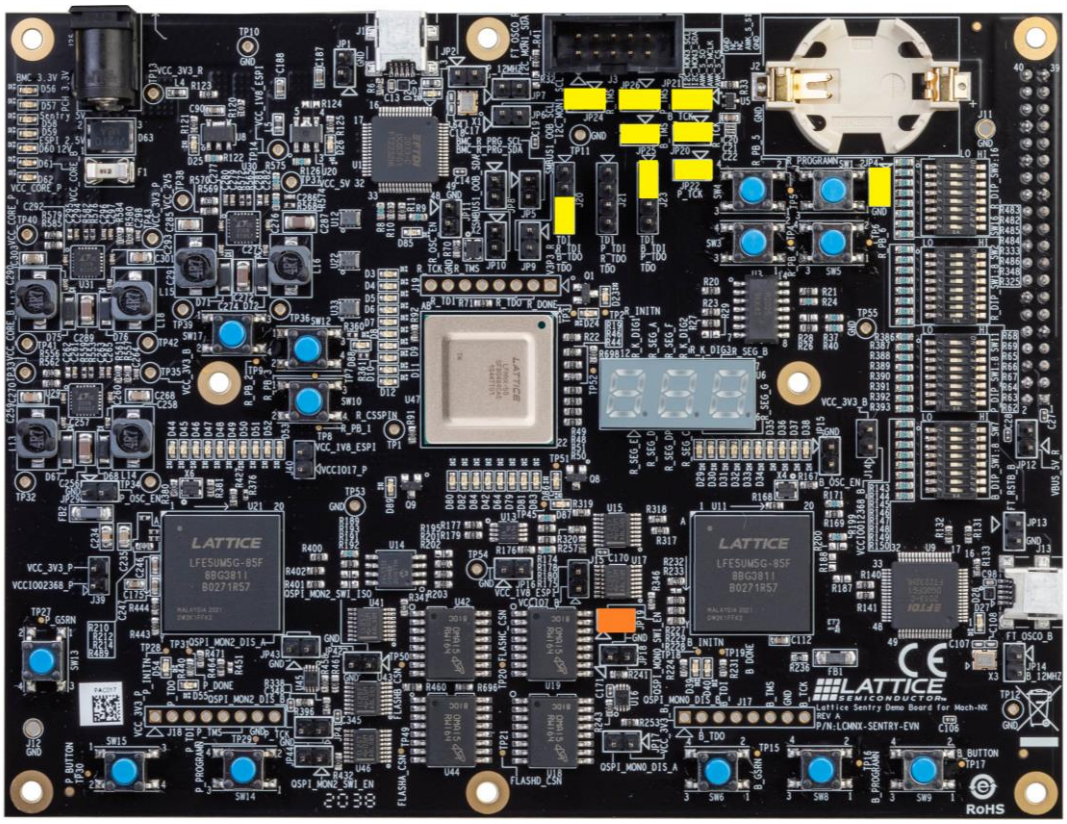

Figure 4.11. BMC Jumper Settings

2. Scan the board in Diamond Programmer. The **LFE5U-85F** device is highlighted in yellow.
3. Select **Device** as **LFE5U-85F** if it is not already selected.
4. Double-click on the **Operation** field to open the **Device Properties** dialog box, as shown in Figure 4.12.

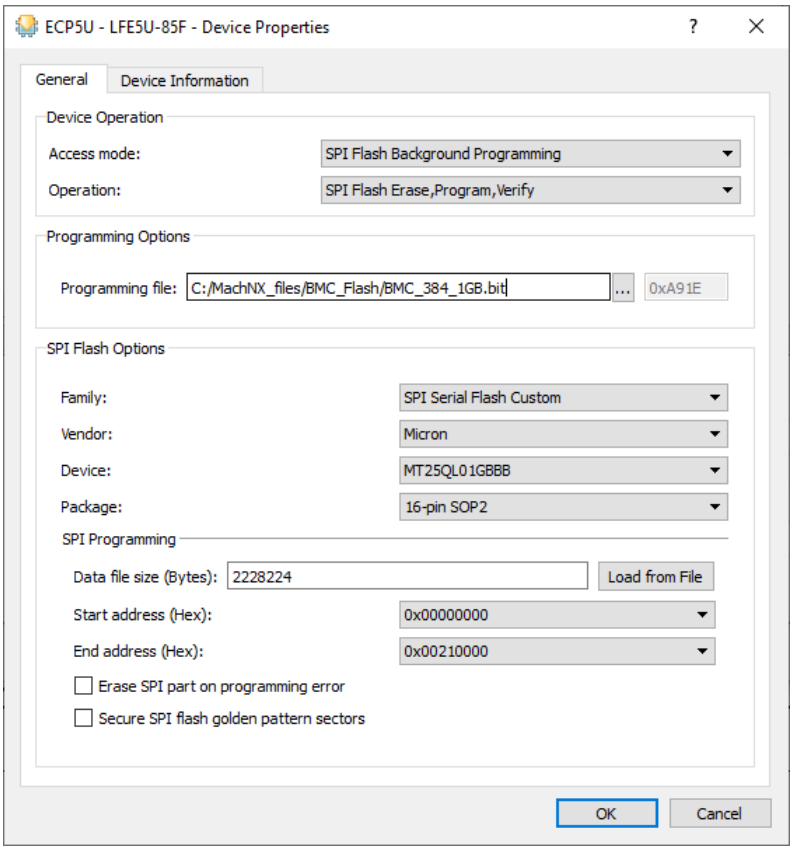

Figure 4.12. Device Properties for BMC

5. Select the following options under **Device Operation**.
   - **Access mode – SPI Flash Background Programming**
   - **Operation – SPI Flash Erase, Program, Verify**
6. Under **Programming Options** in the **Programming file**, select the BMC image file *BMC_384_1GB.bit* from the demo package.
7. Select the following options under **SPI Flash Options**.
   - **Family – SPI Serial Flash**
   - **Vendor – Micron**
   - **Device – MT25QL01GBBB**
   - **Package – 16-Pin SOP2**
8. Click **OK**.
9. In the main interface, click **Design > Program**.
10. Make sure the output log shows a success message.
11. Program the RISCV image into the flash. Reopen the **Device Properties** dialog box.

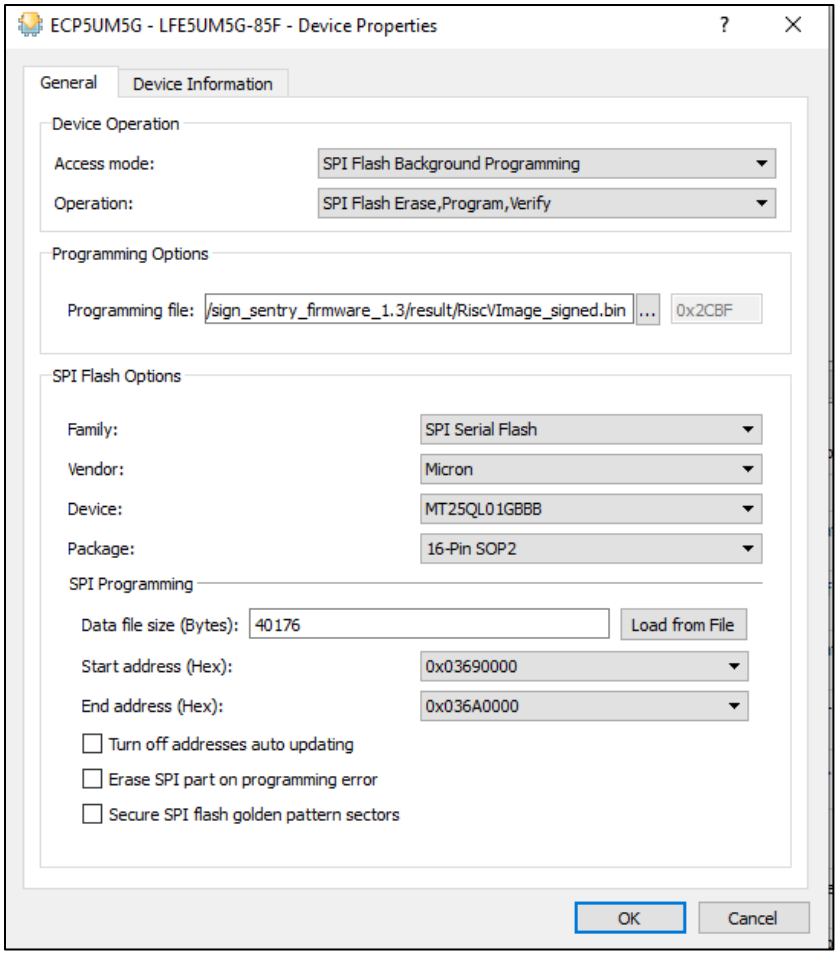

Figure 4.13. Programming Settings for RISC-V Image

12. In **Programming file**, select the signed RiscV image which was generated by the FST utility (*RiscVImage_signed.bin*).
13. In **Start Address (Hex)**, enter **0x03690000**.
14. In **End Address (Hex)**, enter **0x036A0000**.
15. Click **OK**.
16. In the main interface, click **Design > Program**.
17. Make sure the output log shows a success message.
18. Program the SFB image into the flash. Reopen the **Device Properties** dialog box.

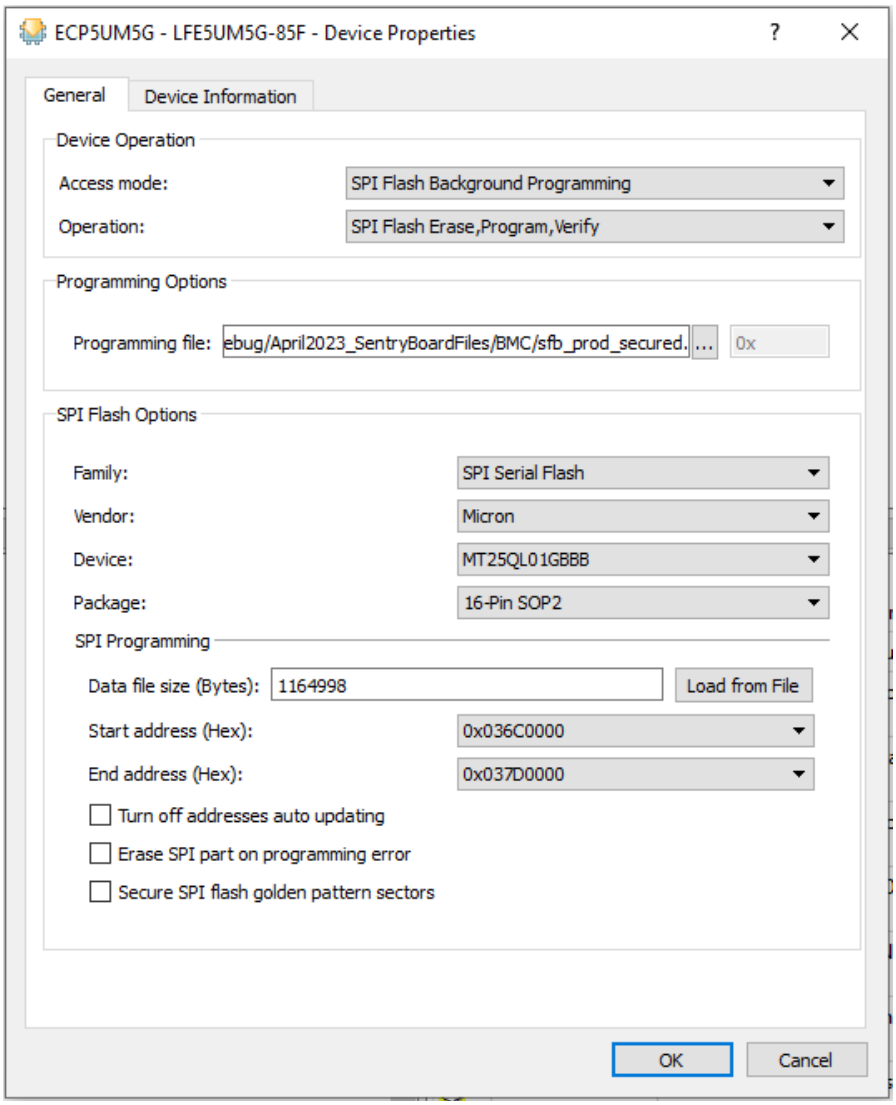

Figure 4.14. Programming Settings for SFB Image

19. In **Programming file**, select SFB image from the demo package (*sfb_prod_secured.bit*).
20. In **Start Address (Hex)**, enter **0x036C0000**.
21. In **End Address (Hex)**, enter **0x037D0000**.
22. Click **OK**.
23. In the main interface, click **Design > Program**.
24. Make sure the output log shows a success message.
25. Remove JP19.

4.3. Programming the PCH Flash

The configuration image for the PCH is stored on flash A.

To program the PCH flash:

1. Connect PCH to the JTAG chain. Refer to Figure 4.15 or Table 4.2.
   a. Connect the FTDI interface at the top of the board (J1) using a USB cable.
   b. With power disconnected, attach the yellow labeled jumpers (JP24, JP25, JP26, JP21, JP20, JP22, J21(1:2), J23(3:4), JP4.
   c. Power on the board by connecting J25 to power.
   d. Attach the orange labeled jumpers, JP42, JP43, JP44.

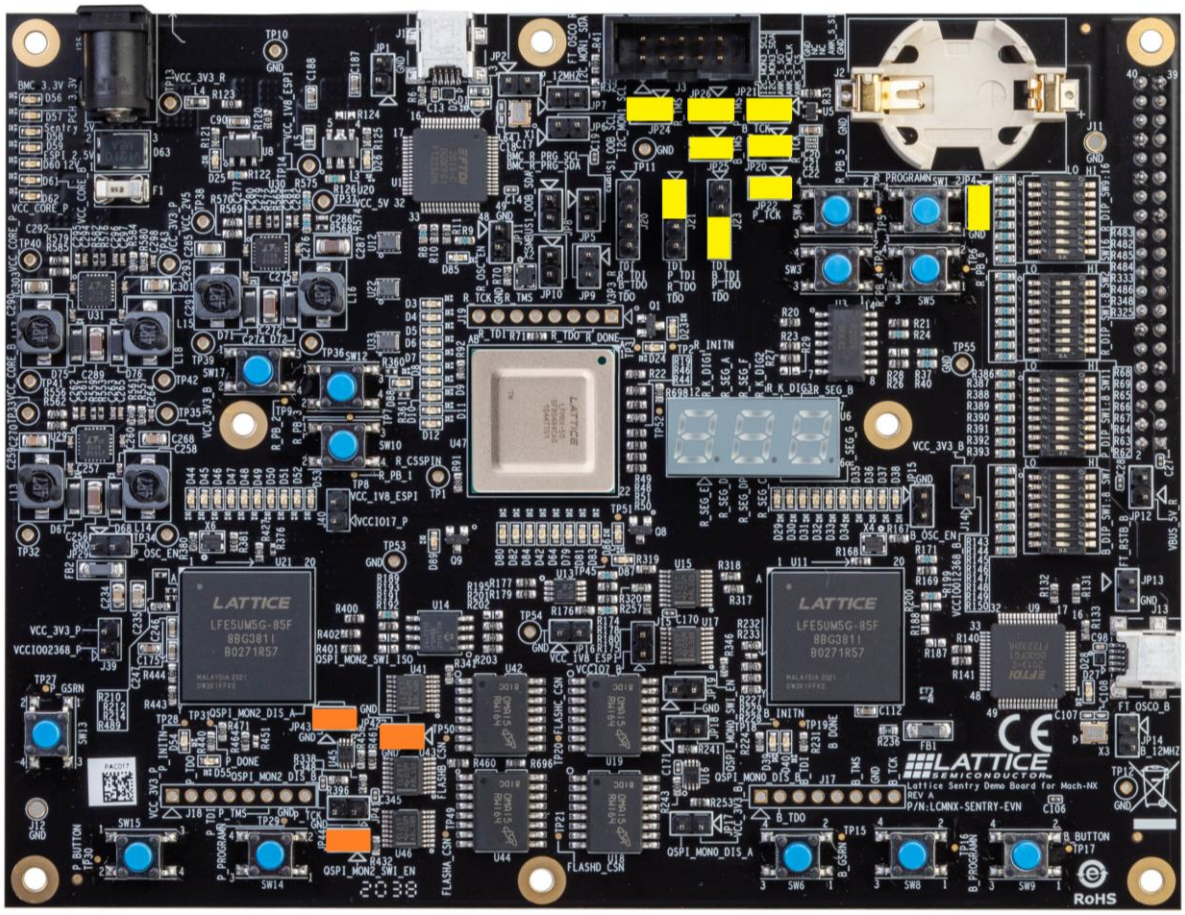

Figure 4.15. PCH Jumper Settings

2. Scan the board in Diamond Programmer. The LFE5U-85F device is highlighted in yellow.
3. Select **Device** as LFE5U-85F if it is not already selected.
4. Double-click on the **Operation** field to open the **Device Properties** dialog box, as shown in Figure 4.16.

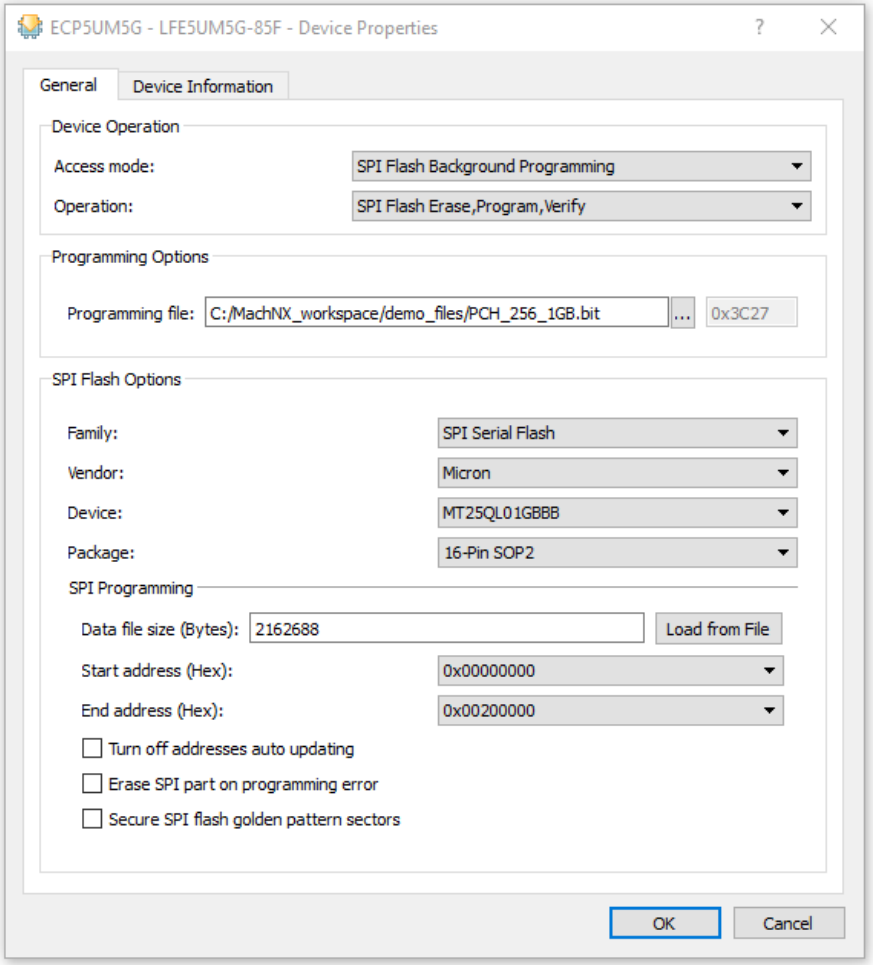

Figure 4.16. Device Properties for PCH

5. Select the following options under **Device Operation**.
   - **Access mode** – **SPI Flash Background Programming**
   - **Operation** – **SPI Flash Erase, Program, Verify**
6. Under **Programming Options** in **Programming file**, enter or locate the PCH image file *PCH_256_1GB.bit* in the demo package folder.
7. Select the following options under **SPI Flash Options**.
   - **Family** – **SPI Serial Flash**
   - **Vendor** – **Micron**
   - **Device** – **MT25QL01GBBB**
   - **Package** – **16-Pin SOP2**
8. In **Start Address (Hex)**, enter **0x00000000**.
9. In **End Address (Hex)**, enter **0x00200000**.
10. Click **OK**.
11. In the main interface, click **Design > Program**.
12. Make sure the output log shows a success message.
13. Power OFF the board.
14. Remove jumpers from JP4, JP42, JP43, and JP44.

4.4. Programming the Mach-NX Public Key

The Mach-NX device must be programmed with the public key of the key pair that was used to sign the CFG0 and RISC-V image files. This key pair can be generated by the FST utility, as described in Section 3.4.

To program the Mach-NX Public key:

1. Follow steps 1-6 of Section 4.1.2 to set up the Mach-NX device for programming.
2. Double click on the **Operation** field to open the **Device Properties** dialog box.
3. In Device Properties, select **Advanced Security Keys Programming** under Access Mode. Select **Security Program Public Key** under Operation.

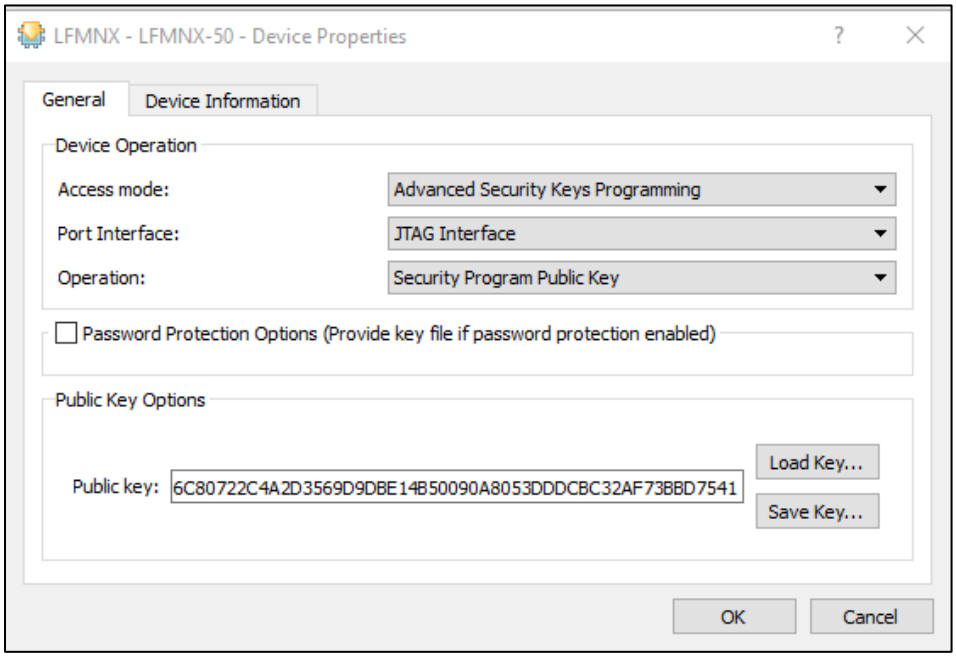

Figure 4.17. Programming Public Key

4. Click **Load Key** and select the **ECDSA_pub.pem** file in the keys directory of the FST utility.
5. Click **OK**.
6. In the main interface, click **Design -> Program.**
7. After the operation is completed, the log windows shows that the device has been programmed successfully.

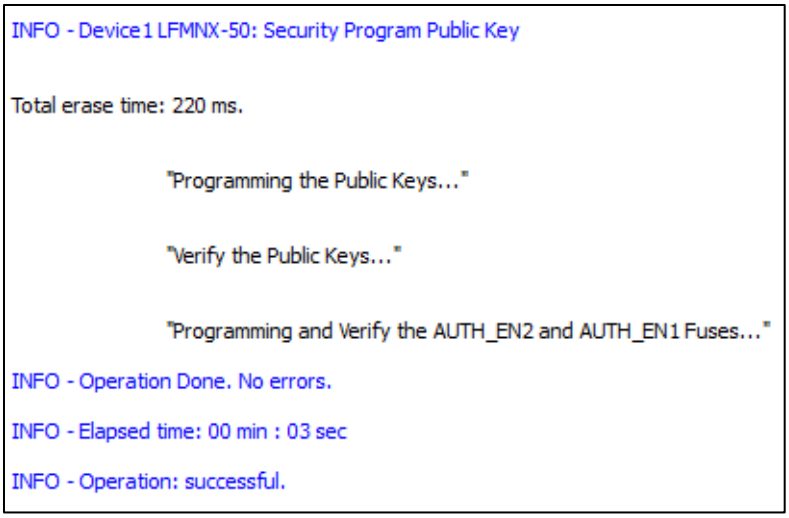

Figure 4.18. Log Output Success Message After Programming Public Key

8. Programming the public key will trigger a soft reboot of the device. After 20–30 seconds, the board LEDs should turn on or blink, according to the design.
9. If the soft reboot does not occur, power cycle the board.

5. Debugging Propel

Debugging in Propel with the GDB OpenOCD Debugger allows the designer to check and fine-tune the firmware. If the board already has firmware programmed onto it, running this debugger will temporarily replace this firmware with the current code in the software workspace.

To debug the firmware using Propel:

1. Open Propel, then double click **main.c** to open it.
2. Look for the following line and make sure it is **not** commented out. (Otherwise, uncomment it and rebuild the project.)

```
dev_jtag_cntl(pfr_inst, JTAG_GDB_EN);
```

3. Make sure the Mach-NX and BMC sections of the board are programmed, as described in Section 4.1 and Section 4.2.
   - If the project was rebuilt following Step 2, re-program the BMC Flash with the updated PFR firmware.
4. Ensure that the JTAG jumpers are set so that Mach-NX is in the JTAG chain. Refer to Table 4.2.
5. Ensure that the flash jumpers are set for the PFR Demo. Refer to Table 4.3.
6. Connect the machine running Propel to the board through the primary USB port (J1).
7. Create a debug configuration. Click **Run > Debug Configurations** from the Propel main interface toolbar. This opens the **Debug Configurations** window, as shown in Figure 5.1.

   Note: The fields are automatically populated. If they are blank, click **Browse** and select the current software project.

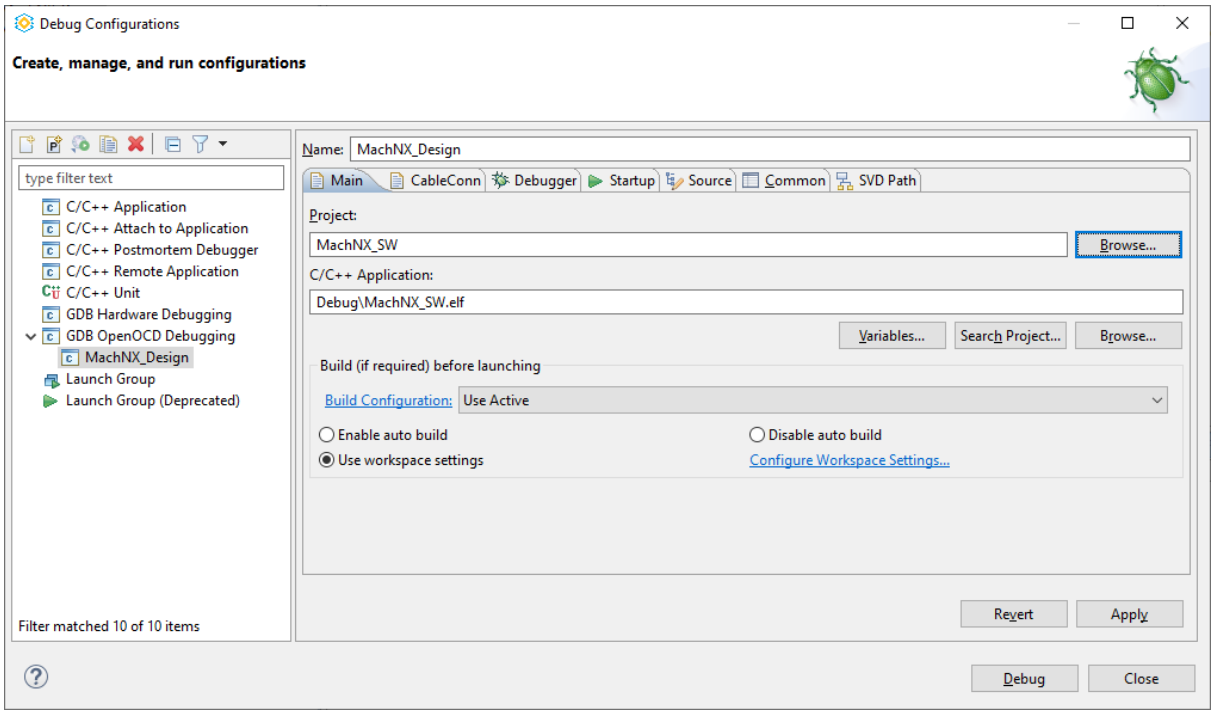

Figure 5.1. Debug Configurations

8. Select the **CableConn** tab as shown in Figure 5.2.
9. Click **Scan Device**.
10. Select **LFMNX-50** from the device list.
11. Click **Debug**.

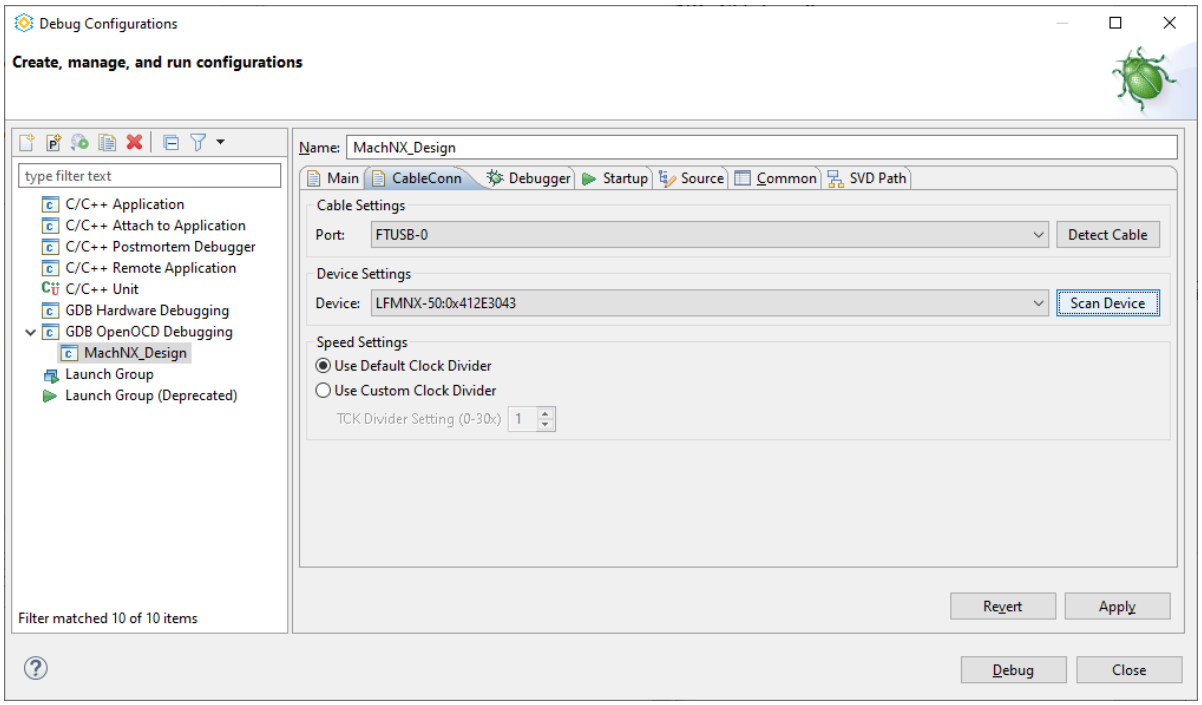

Figure 5.2. CableConn Settings

12. Switch perspective to debug when prompted. The software project is downloaded into the Mach-NX and halted at a breakpoint at the beginning of **main()**. At this point, the demonstration code can be stepped through with standard debugger controls (start, pause, step over/through, and others).

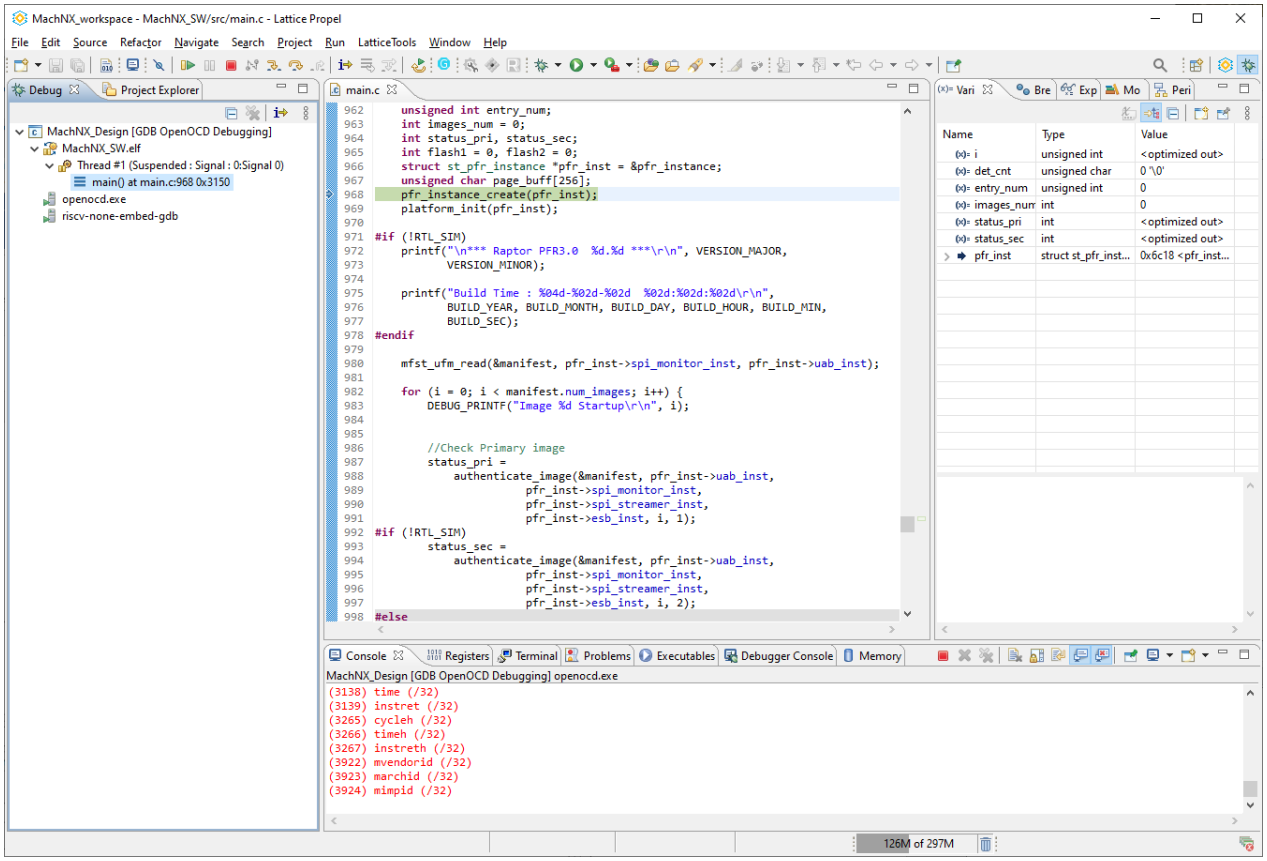

Figure 5.3. Propel Debugger View

6. Troubleshooting

This section covers troubleshooting procedures when programming a demo board that Diamond Programmer is unable to scan properly.

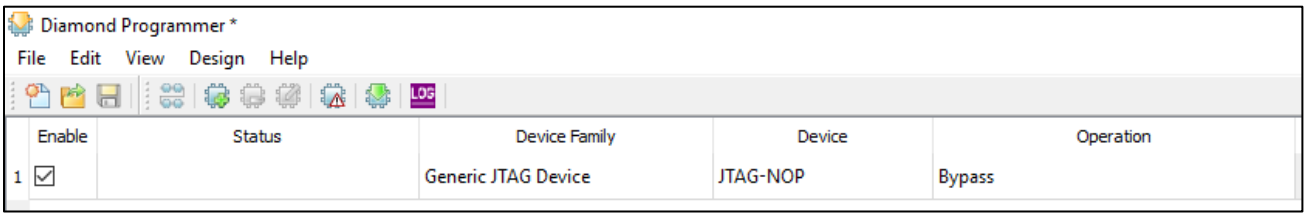

Figure 6.1. Generic JTAG Device

If scanning the board results in a generic JTAG device, as shown in Figure 6.1, you will need to reconnect the JTAG chain. There are two steps to reconnecting the JTAG chain: blind erasing the public key and programming a bypass pattern into the SRAM.

The bypass pattern provided with the demo package, *raptor_download_impl1_jtagenset_hi_new.bit,* will only work if the public key has already been erased, so the steps must be followed in the correct order. If the Mach-NX PUBKEY sector is not programmed with a key, then the PUBKEY does not need to be erased.

6.1. Blind Erasing the Public Key

The Lattice Sentry Demo Board for Mach-NX Devices has two FTDI chips (FT2232H) to communicate with the various FPGAs on the board over JTAG and UART. When programming the FPGAs, only the FTDI chip marked U1 (connected to USB port J1) should be connected. Make sure that no other FTDI chips are connected by USB to the computer performing the programming.

1. Connect the Mach-NX to the JTAG chain by setting the jumpers as shown in Figure 4.4.
2. The public key will be erased blindly from the Mach-NX. In **Device Family**, select **LFMNX**. In **Device**, select **LFMNX-50**.
3. In the Diamond Programmer main interface, click **Edit > Settings** to open the **Settings** dialog box, as shown in Figure 6.2.
4. In the **Programming** tab, select **Continue download on error**.
5. In the **Programming** tab, deselect **Check cable setup before programming.**
6. Click **OK**.

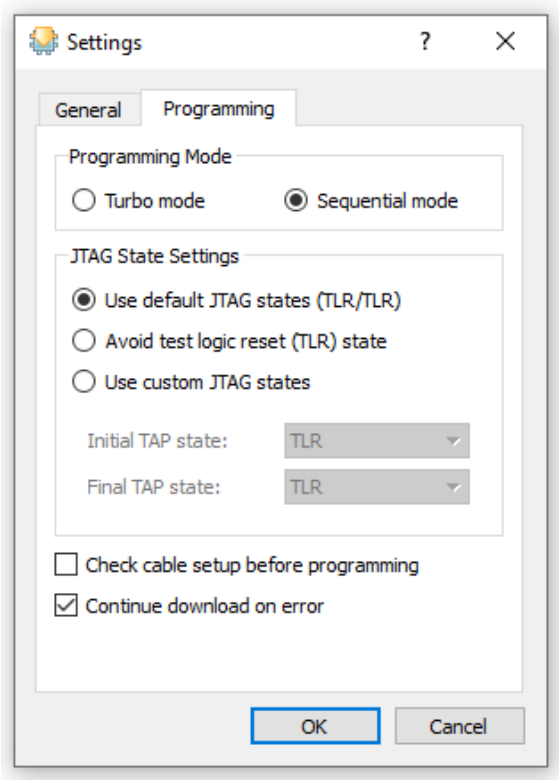

Figure 6.2. Diamond Programmer Settings for Blind Programming

7. Double-click on **Operation** to open the **Device Properties** dialog box, as shown in Figure 6.3.
8. In Device Properties, select **Advanced Security Keys Programming**, then **Security Erase Public Key.**

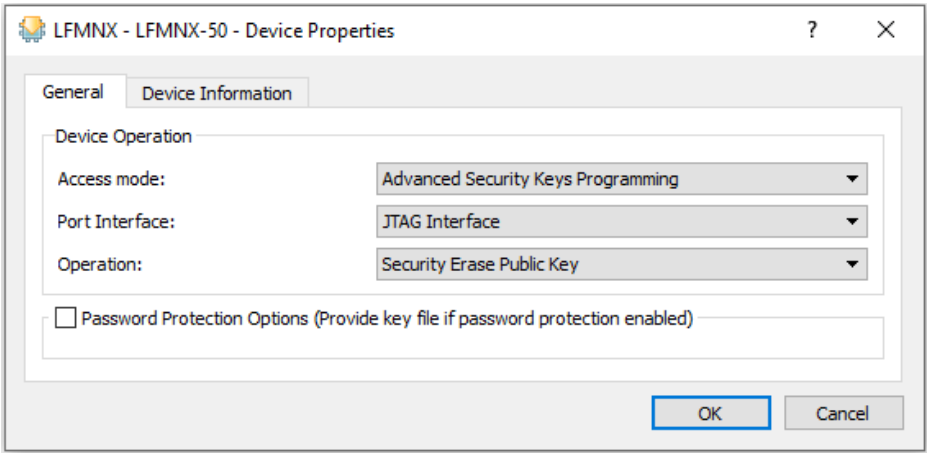

Figure 6.3. Mach-NX Erase Public Key

9. Click **OK.**
10. In the main interface, click **Design > Program**.
11. Once programming is completed, follow the procedure in Section 6.2 to program a bypass pattern into the SRAM.

6.2. Programming a bypass pattern into SRAM

The Lattice Sentry Demo Board for Mach-NX Devices has two FTDI chips (FT2232H) to communicate with the various FPGAs on the board over JTAG and UART. When programming the FPGAs, only the FTDI chip marked U1 (connected to USB port J1) should be connected. Make sure that no other FTDI chips are connected by USB to the computer performing the programming.

1. A recovery image will be programmed blindly into the Mach-NX. In **Device Family**, select **LFMNX**. In **Device**, select **LFMNX-50**.
2. In the Diamond Programmer main interface, click **Edit > Settings** to open the **Settings** dialog box, as shown in Figure 6.2.
3. In the **Programming** tab, select **Continue download on error**.
4. In the **Programming** tab, deselect **Check cable setup before programming**.
5. Click **OK**.

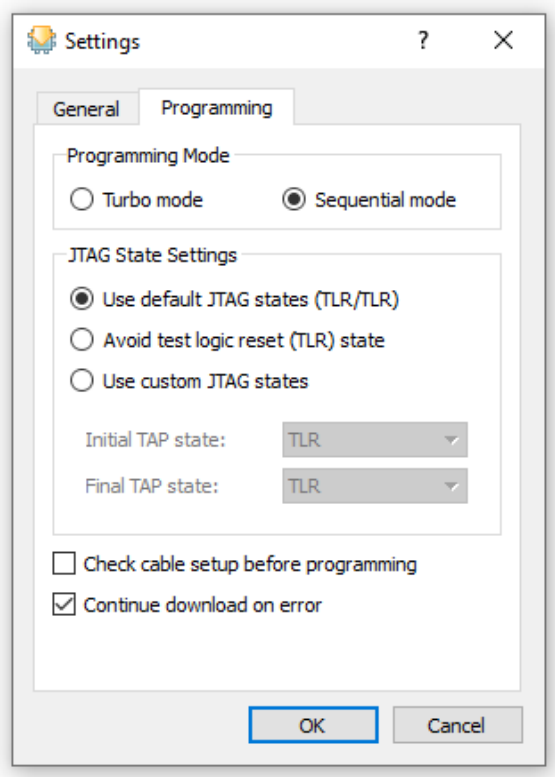

Figure 6.4. Diamond Programmer Settings for Blind Programming

6. Double-click on **Operation** to open the **Device Properties** dialog box, as shown in Figure 6.5.
7. Select the following options under **Device Operation**
   - **Access mode – Static RAM Cell Mode**
   - **Port Interface – JTAG Interface**
   - **Operation – SRAM Fast Configuration**
8. Under **Programming Options** in **Programming file**, enter the file *raptor_download_impl1_jtagenset_hi_new.bit* from the demo package folder.
9. Click **OK**.

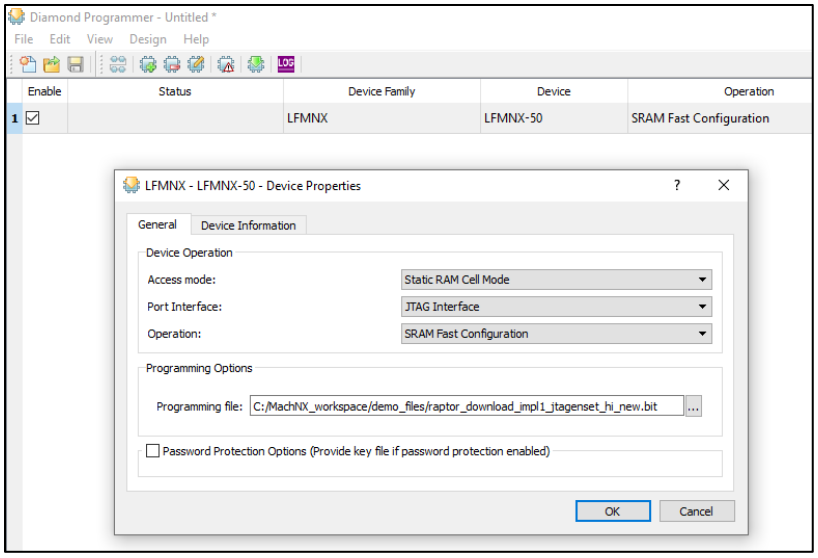

Figure 6.5. Programming Recovery File

10. In the main interface, click **Design > Program**.
11. Once programming is completed, rescan the board. The device is indicated under **Device Family** and **Device**, as shown in Figure 6.6.

Figure 6.6. Scanned Mach-NX Device

12. Revert the programming settings to default:
    a. In the Diamond Programmer main interface, click **Edit > Settings** to open the **Settings** dialog box, as shown in Figure 6.7.
    b. In the **Programming** tab, select **Check cable setup before programming**.
    c. In the **Programming** tab, deselect **Continue download on error.**
    d. Click **OK**.

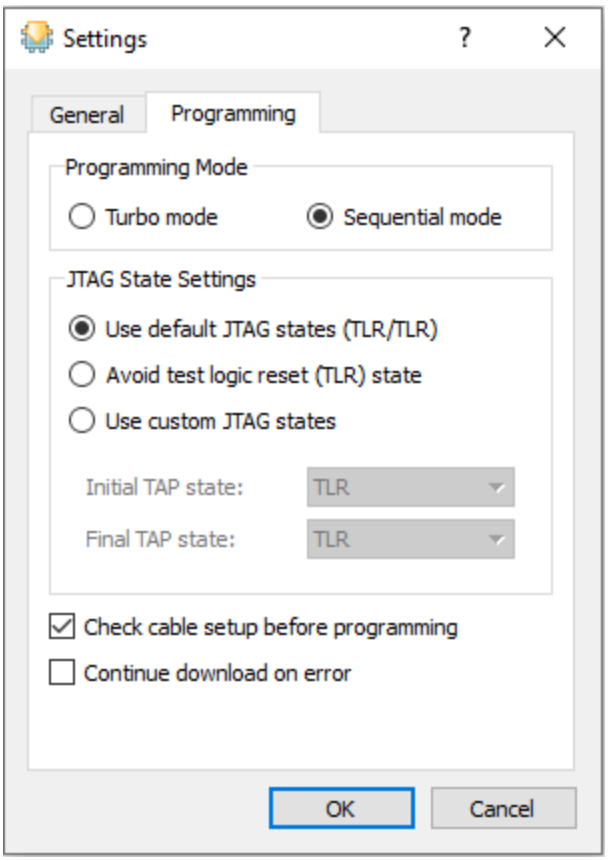

Figure 6.7. Programming Settings Restored to Default Settings

13. To complete the recovery, program a valid image into the CFG0 flash sector, as described in Section 4.1.4, and program the public key into the PUBKEY sector of the Mach-NX device, as described in Section 4.4.

6.3. Unlocking UFM2

If the system boots and finds a valid FAM, but the SFB is invalid, corrupted, or missing, it will lock UFM2. There will be an error message in Diamond Programmer saying UFM2 is erase locked, as shown in Figure 6.8.

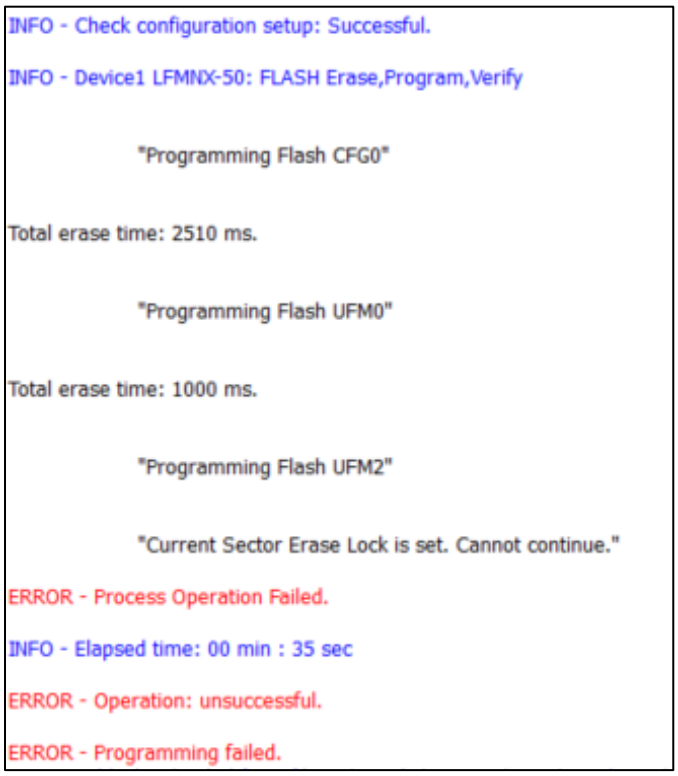

Figure 6.8. Programming Settings Restored to Default Settings

6.3.1. Erase and Restore the SFB image

To unlock the UFM2 Sector Erase Lock, first restore the SFB image.

1. Connect to the BMC Flash by following Step 1 of Section 4.2.
2. Scan the board in Diamond Programmer. The LFE5U-85F device is highlighted in yellow.
3. Select **Device** as **LFE5U-85F** if it is not already selected.
4. Double-click on the **Operation** field to open the **Device Properties** dialog box, as shown in Figure 6.9.
5. Switch to **SPI Flash Background Programming mode**; **SPI Flash Erase All**. Click **OK** and then use the configuration button to program.

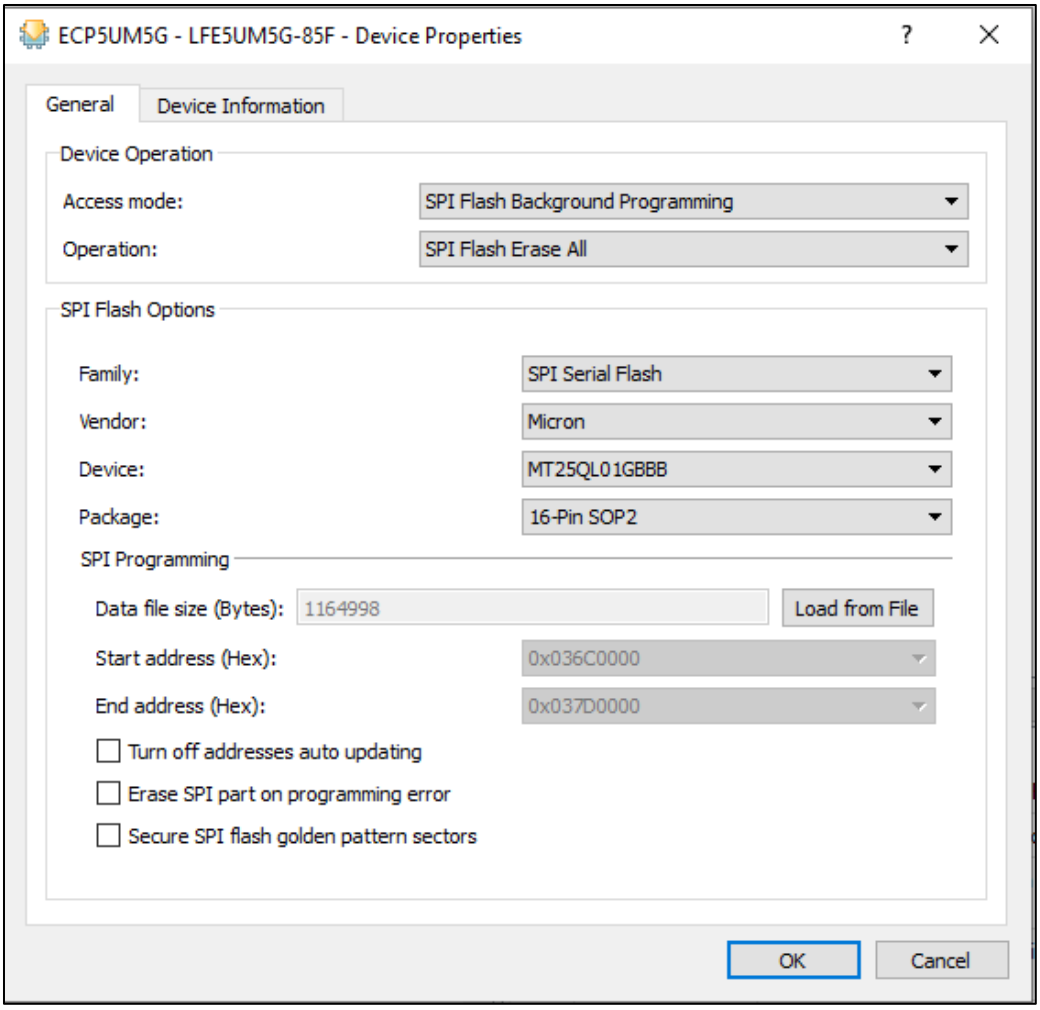

Figure 6.9 BMC Flash Erase All

6. After a successful erase, reprogram the Flash by following Steps 4-25 in Section 4.2.

6.3.2. Erase and Restore CFG0, UFM2, and FAM

If following the steps in Section 6.3.1 was unsuccessful and the UFM2 sector is still locked, follow these steps to restore the UFM and FAM:

1. Set up the jumpers to connect the JTAG chain to Mach-NX, as shown in Figure 4.4.
2. Scan the board. If LFMNX does not appear, follow the steps in Section 6.1 and Section 6.2 to reconnect the JTAG chain. Note that if the public key is erased, it will need to be reprogrammed onto the board at the end of these steps.
3. Double click Device Properties and choose Flash Background Mode, XFLASH Erase, Program, Verify. Click only the checkbox next to CFG0. Click the three dots to browse for the file and choose Sentry22_484_impl1_a_factory.jed. Click OK and use the configuration button to program.

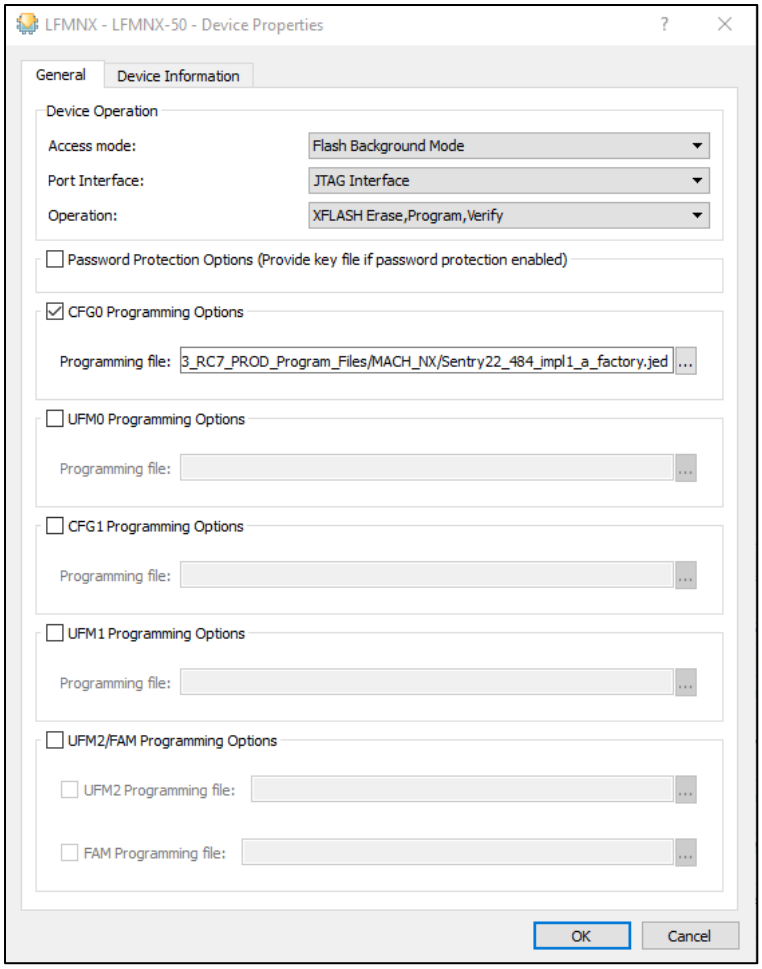

Figure 6.10. Programming Settings to Reprogram CFG0

4. Choose **Flash Background Mode**, then **XFLASH Erase Only**. Check the checkbox next to **UFM2/FAM Programming Options**, and then check only the checkbox next to **FAM**. Click **OK** and use the configuration button to program.

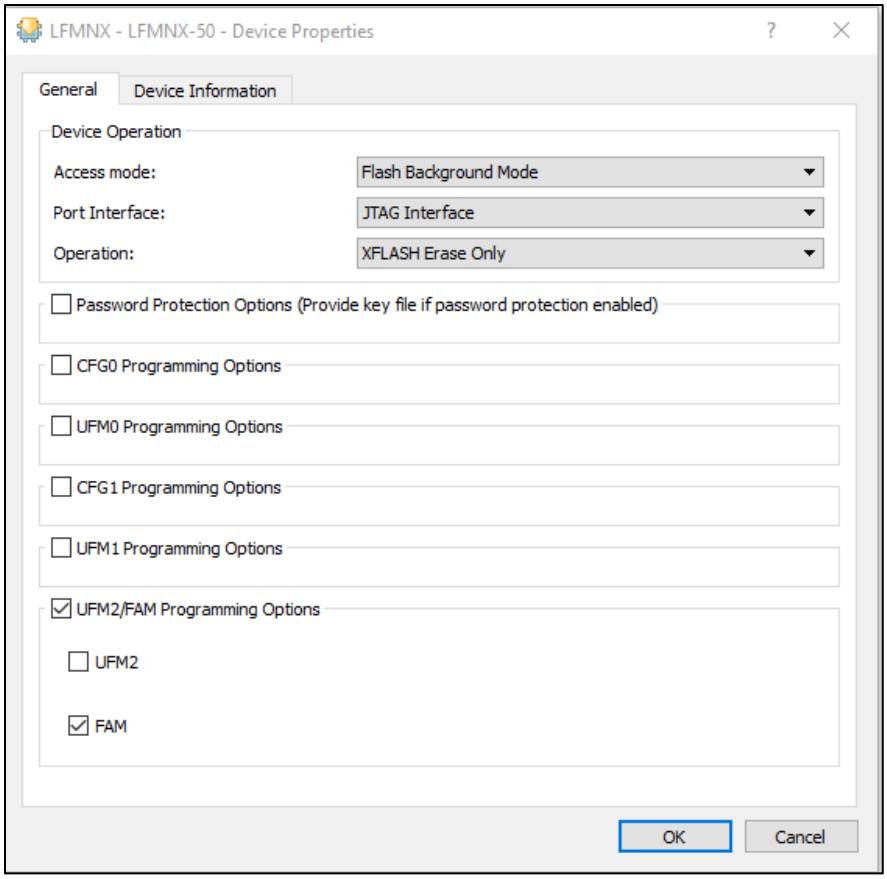

Figure 6.11. Programming Settings to Erase FAM Table

5. Power on and off the device to restart. UFM2 should now be unlocked, but the system will not have booted.
6. Scan the board. If **LFMNX** does not appear, follow the steps in Section 6.1 and Section 6.2 to reconnect the JTAG chain.
7. Choose **Flash Background Mode**, **XFLASH Erase, Program, Verify**. Check the checkbox next to **UFM2/FAM Programming Options**, and then check the checkboxes for both **UFM2 Programming file** and **FAM Programming file**. Choose **ufm2_w_cert.jed** for UFM2 and **fam_prod_new.jed** for FAM. Click **OK** and use the configuration button to program.

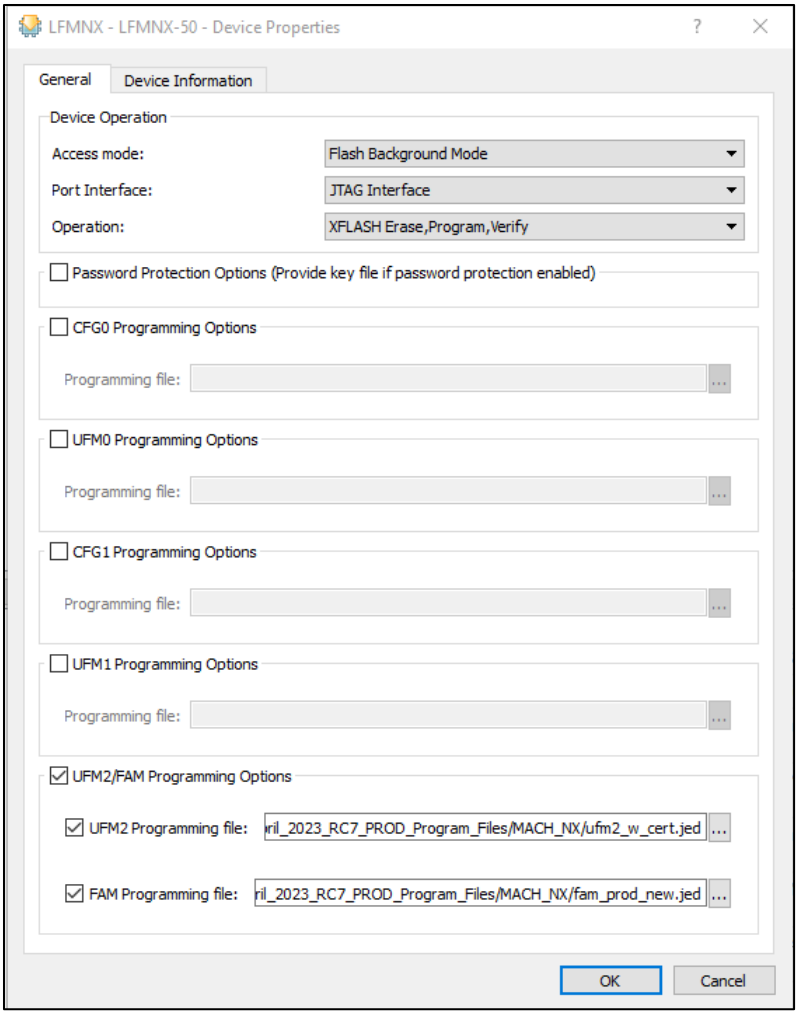

Figure 6.12. Reprogramming UFM2 and FAM

8. If the public key was erased to restore the JTAG chain, reprogram the public key as described in Section 4.4.
9. Power on and off the device to restart. The board should now boot successfully.

References

The following reference documents will provide further information about the Lattice Sentry solution and the Lattice Sentry Demo Board for Mach-NX Devices:

- Mach-NX web page
- Mach-NX Family datasheet (FPGA-DS-02084)
- Lattice Sentry Solution Stack web page
- Lattice Sentry Demo Board for Mach-NX User Guide (FPGA-EB-02045)
- Lattice Sentry Firmware Signing and Update Guide (FPGA-TN-02351)
- Embedded Security and Function Block User Guide for MachXO5-NX Devices (FPGA-TN-02320)
- Enhanced Embedded Security Block (FPGA-IPUG-02167)
- Lattice Propel 1.0 User Guide
- Lattice Proper 2023.1 Builder User Guide
- Lattice Diamond User Guide
- Lattice Propel FPGA design software
- Lattice Diamond 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 1.1, March 2024

Section	Change Summary
All	• Changed the title from *Lattice Sentry Demo Board for Mach-NX Walkthrough* to *Lattice Sentry Demo Board for Mach-NX Devices*. • Minor editorial fixes.
Disclaimer	Updated this section.

Demo Requirements	• Updated the list of hardware requirements for the Lattice Sentry Demo Board for Mach-NX Devices in Subsection 2.1.
	  • Updated main bullet no.6 to, *Four 1-Gb SPI/QSPI flash devices for device configuration and firmware image storage (two flash devices per ECP5 device)*.
	• Updated the Lattice Diamond Programmer checklist in Subsection 2.2.1.
	  • Updated the Lattice Diamond Programmer to *v.3.13.0.56*.
	  • Updated the Lattice Diamond Programmer Encryption Security patch to *v.3.13.0.56*.
	  • Added the Lattice Diamond patch v.3.13.05.56.2 131818.
	• Removed the statement, *Note that although Lattice Diamond Programmer is included with installation of Lattice Diamond, the standalone version of Lattice Diamond Programmer is required for programming Lattice Sentry*.

Setting up Projects	• Updated the steps to import an existing SoC project in Subsection 3.1.2 Importing an Existing Propel SoC Project. • Updated the steps to generate a Diamond project in Subsection 3.2 Generating Diamond Project. • Updated the steps to create a new software project from template in Subsection 3.3.1 Creating a New Software Project from Template. • Updated the steps to import existing software project in Subsection 3.3.2 Importing an Existing Software Project. • Added Subsection 3.3.3 Building and Updating the Software Project to this section. • Reworked contents of Subsection 3.4 Generating Board Files.
Setting up the Board	• Reworked contents of Table 4.1. Programming Files. • Reworked contents of Subsection 4.1 Programming the Mach-NX Files. • Reworked contents of Subsection 4.2 Programming the BMC Flash. • Reworked contents of Subsection 4.3 Programming the PCH Flash. • Added Subsection 4.4 Programming the Mach-NX Public Key.
Debugging Propel	Reworked section contents.
Troubleshooting	Reworked section contents.
References	Updated this section.
Technical Support Assistance	Added the FAQ link in this section.

Revision 1.0, October 2022

Section	Change Summary
All	Initial release.

www.latticesemi.com